Project 2:

سوال ۱: (۱۵ نمره)

برای یک مسئله‌ی طبقه‌بندی دو کلاسه، برای متغییر ورودی تک بعدی **x**، با در نظر گرفتن:

$$P(x|y=0) = e^{-2x} \qquad x \geq 0$$

$$P(x|y=1) = \frac{2.25}{\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2}} \qquad x \geq 0$$

$$ln\frac{2.25}{\sqrt{2\pi}} \cong -1$$

و فرض برابر بودن احتمال پسین هر دو کلاس، بازه‌ی ناحیه‌ی کلاس اول و دوم را بدست آورید.

To determine the decision boundary between the two classes in this binary classification problem, we start by comparing the class-conditional probability densities P(x|y=0) and P(x | y=1). Since the prior probabilities for both classes are equal (P(y=1)=0.5), the decision boundary occurs where these densities are equal:

$P(x \mid y=0) = P(x \mid y=1) P(x|y=0) = P(x|y=1)$

Given:

$P(x|y=0) = e^{-2x}, \quad x \geq 0$

$P(x|y=1) = \frac{2.25}{\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2}}, \quad x \geq 0$

Taking the natural logarithm of both sides:

$\ln P(x|y=0) = \ln P(x|y=1)$

Substitute the given expressions:

$-2x = \ln\left(\frac{2.25}{\sqrt{2\pi}}\right) - \frac{(x-1)^2}{2}$

Given that $\ln\left(\frac{2.25}{\sqrt{2\pi}}\right) \approx -1$:

$-2x = -1 - \frac{(x-1)^2}{2}$

Multiply both sides by 2 to eliminate the fraction:

$-4x = -2 - (x-1)^2$

Expand $(x-1)^2$:

$$-4x = -2 - (x^2 - 2x + 1)$$

Simplify:

$$-4x = -2 - x^2 + 2x - 1$$

$$-4x = -x^2 + 2x - 3$$

Bring all terms to one side:

$$x^2 - 6x + 3 = 0$$

Solve the quadratic equation:

$$x = \frac{6 \pm \sqrt{(6)^2 - 4(1)(3)}}{2(1)} = \frac{6 \pm \sqrt{36 - 12}}{2} = \frac{6 \pm 2\sqrt{6}}{2}$$

Simplify:

$$x = 3 \pm \sqrt{6}$$

Approximate the roots:

$$\sqrt{6} \approx 2.449$$

$$x_1 = 3 + 2.449 \approx 5.449$$
$$x_2 = 3 - 2.449 \approx 0.551$$

These two values represent the decision boundaries. Next, we determine which regions correspond to each class by testing sample points in each interval:

For x<0.551x < 0.551:

Let x=0.3x = 0.3:

$$P(x|y = 0) = e^{-2(0.3)} \approx 0.5488$$
$$P(x|y = 1) = \frac{2.25}{\sqrt{2\pi}} e^{-\frac{(0.3-1)^2}{2}} \approx 0.2871$$

Since P(x ∣ y=0)>P(x ∣ y=1)P(x|y=0) > P(x|y=1), Class 0 is more probable.

For 0.551<x<5.449 0.551 < x < 5.449:

Let x=1x = 1:

$$P(x|y = 0) = e^{-2(1)} \approx 0.1353$$

$$P(x|y = 1) = \frac{2.25}{\sqrt{2\pi}} e^{-\frac{(1-1)^2}{2}} \approx 0.3679$$

Since P(x ∣ y=1)>P(x ∣ y=0), Class 1 is more probable.

For x>5.449x > 5.449:

Let x=6x = 6:

$$P(x|y = 0) = e^{-2(6)} \approx e^{-12} \approx 6.14 \times 10^{-6}$$
$$P(x|y = 1) = \frac{2.25}{\sqrt{2\pi}} e^{-\frac{(6-1)^2}{2}} \approx e^{-13.5} \approx 1.50 \times 10^{-6}$$

Since P(x|y=0) > P(x|y=1), Class 0 is more probable.

Decision Regions:

- **Class 0**:

  $$x \in [0, 0.551) \cup [5.449, \infty)$$

- **Class 1**:

$x \in [0.551, 5.449) x \in [0.551, 5.449)$

Answer:

The decision boundaries are at $x = 3 \pm \sqrt{6}$ (approximately 0.551 and 5.449). The regions are:

- **Class 0:**

  $x \leq 0.551 x \leq 0.551$

  $x \geq 5.449 x \geq 5.449$

- **Class 1**:

  $0.551 \leq x \leq 5.449 0.551 \leq x \leq 5.449$

سوال ۲: (۲۰ نمره)

مجموعه داده‌ی جدول ۱ را در نظر بگیرید که در آن متغیر y برچسب و متغییرهای A,B,C ویژگی‌های باینری هستند.

جدول ۱

A	B	C	Y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	0
1	1	1	1
1	0	0	1
1	1	0	1

طبقه‌بند Naive Bayes هر کدام از نمونه‌های زیر را چگونه دسته‌بندی می‌کند.

A	B	C	$\hat{Y}$
0	0	0	?
1	1	1	?
X	1	0	?
X	0	1	?

Naive Bayes Classification Process: Explanation and Solution

Naive Bayes is a probabilistic classifier that uses Bayes' theorem, assuming the features are conditionally independent given the label. Here's how we proceed:

Step 1: Dataset and Definitions

- **Dataset:**

A	B	C	Y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	0
1	1	1	1
1	0	0	1
1	1	0	1

- Features: , Label:

  A,B,C

  Y

- Prior probabilities and .

  P(Y=1)

  P(Y=0)

- Conditional probabilities for each label.

  P(A ∣ Y),P(B ∣ Y),P(C ∣ Y)P(A|Y), P(B|Y), P(C|Y)

Step 2: Prior Probabilities

The prior probabilities are based on the label distribution:

- Total rows =

  7

- Rows with :

  Y=1

  3

- Rows with :

  Y=0

  4

$P(Y = 1) = \frac{3}{7}, \quad P(Y = 0) = \frac{4}{7}$

Step 3: Conditional Probabilities

For Y=1Y=1:

- Subset:

$\{(1,1,1),(1,0,0),(1,1,0)\}$

A	B	C	Count
1	1	1	11
1	0	0	11
1	1	0	11

- $P(A=1|Y=1)=\frac{3}{3}$, $P(A=0|Y=1)=\frac{0}{3}$
- $P(B=1|Y=1)=\frac{2}{3}$, $P(B=0|Y=1)=\frac{1}{3}$
- $P(C=1|Y=1)=\frac{1}{3}$, $P(C=0|Y=1)=\frac{2}{3}$

For Y=0Y=0:

- Subset:

  {(0,0,1),(0,1,0),(1,1,0),(0,0,1)}\{(0,0,1), (0,1,0), (1,1,0), (0,0,1)\}

A	B	C	Count
0	0	1	22
0	1	0	11
1	1	0	11

- $P(A=0|Y=0)=\frac{3}{4}$, $P(A=1|Y=0)=\frac{1}{4}$
- $P(B=1|Y=0)=\frac{2}{4}$, $P(B=0|Y=0)=\frac{2}{4}$
- $P(C=1|Y=0)=\frac{2}{4}$, $P(C=0|Y=0)=\frac{2}{4}$

Step 4: Posterior Probability and Classification

Using Bayes' theorem:

$P(Y|A,B,C) \propto P(A|Y)P(B|Y)P(C|Y)P(Y)$

For each sample:

1. Calculate

   $P(Y=1|A,B,C)$

2. Calculate

   $P(Y=0|A,B,C)$

3. Assign if , else .

   $\hat{Y}=1$

   $P(Y=1|A,B,C) > P(Y=0|A,B,C)$

   $\hat{Y}=0$

Sample Classification

Sample 1: A=0,B=0,C=0

$P(Y=1|A=0,B=0,C=0) \propto P(A=0|Y=1)P(B=0|Y=1)P(C=0|Y=1)P(Y=1)$

- $P(A=0|Y=1)=0, P(B=0|Y=1)=\frac{1}{3}, P(C=0|Y=1)=\frac{2}{3}, P(Y=1)=\frac{3}{7}$

$P(Y=1|A=0,B=0,C=0)=0$

- $P(A=0|Y=0)=\frac{3}{4}, P(B=0|Y=0)=\frac{2}{4}, P(C=0|Y=0)=\frac{2}{4}, P(Y=0)=\frac{4}{7}$

$(Y=0|A=0,B=0,C=0)=\frac{3}{4}\cdot\frac{2}{4}\cdot\frac{2}{4}\cdot\frac{4}{7}=\frac{24}{112}$

- **Class:**

  $\hat{Y}=0$

Sample 3: (X,1,0)

Case 1: X=0

For (A=0,B=1,C=0):

1. $P(Y=1|A=0,B=1,C=0)$:

   $P(Y=1|A=0,B=1,C=0)\propto P(A=0|Y=1)P(B=1|Y=1)P(C=0|Y=1)P(Y=1)$

   Using earlier values:

   - $P(B=1|Y=1)=\frac{2}{3}$, $P(C=0|Y=1)=\frac{2}{3}$, $P(Y=1)=\frac{3}{7}$

   $P(Y=1|A=0,B=1,C=0)=0$
2. $P(Y=0|A=0,B=1,C=0)$:

   $P(Y=0|A=0,B=1,C=0)\propto P(A=0|Y=0)P(B=1|Y=0)P(C=0|Y=0)P(Y=0)$

   Using earlier values:

   - $P(A=0|Y=0)=\frac{3}{4}$, $P(B=1|Y=0)=\frac{1}{2}$, $P(C=0|Y=0)=\frac{1}{2}$, $P(Y=0)=\frac{4}{7}$

   $P(Y=0|A=0,B=1,C=0)=\frac{3}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4}{7}=\frac{3}{28}$

Class: $\hat{Y}=0$.

Case 2: X=1X = 1

For (A=1,B=1,C=0):

1. $P(Y=1 \mid A=1,B=1,C=0)$

   $P(Y=1|A=1,B=1,C=0)\propto P(A=1|Y=1)P(B=1|Y=1)P(C=0|Y=1)P(Y=1)$

   Using earlier values:

   - $P(A=1|Y=1)=1$, $P(B=1|Y=1)=\frac{2}{3}$, $P(C=0|Y=1)=\frac{2}{3}$, $P(Y=1)=\frac{3}{7}$

   $P(Y=1|A=1,B=1,C=0)=1\cdot\frac{2}{3}\cdot\frac{2}{3}\cdot\frac{3}{7}=\frac{4}{21}$
2. $P(Y=0 \mid A=1,B=1,C=0)$:

   $P(Y=0|A=1,B=1,C=0)\propto P(A=1|Y=0)P(B=1|Y=0)P(C=0|Y=0)P(Y=0)$

   Using earlier values:

   - $P(A=1|Y=0)=\frac{1}{4}$, $P(B=1|Y=0)=\frac{1}{2}$, $P(C=0|Y=0)=\frac{1}{2}$, $P(Y=0)=\frac{4}{7}$

   $P(Y=0|A=1,B=1,C=0)=\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4}{7}=\frac{1}{28}$

Class: Y^=1\hat{Y} = 1.

Sample 4: (X, 0, 1)

Case 1: X=0

For (A=0,B=0,C=1):

1. **P(Y=1 | A=0,B=0,C=1):**

   $P(Y=1|A=0,B=0,C=1) \propto P(A=0|Y=1)P(B=0|Y=1)P(C=1|Y=1)P(Y=1)$

   Using earlier values:

   - $P(A=0|Y=1)=0$, $P(B=0|Y=1)=\frac{1}{3}$, $P(C=1|Y=1)=\frac{1}{3}$, $P(Y=1)=\frac{3}{7}$

   $P(Y=1|A=0,B=0,C=1)=0$

2. **P(Y=0|A=0, B=0, C=1):**

   $P(Y=0|A=0,B=0,C=1) \propto P(A=0|Y=0)P(B=0|Y=0)P(C=1|Y=0)P(Y=0)$

   Using earlier values:

   - $P(A=0|Y=0)=\frac{3}{4}$, $P(B=0|Y=0)=\frac{1}{2}$, $P(C=1|Y=0)=\frac{1}{2}$, $P(Y=0)=\frac{4}{7}$

   $P(Y=0|A=0,B=0,C=1)=\frac{3}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4}{7}=\frac{3}{28}$

Class: Y^=0\hat{Y} = 0.

Case 2: X=1

For (A=1,B=0,C=1):

1. **P(Y=1 | A=1,B=0,C=1):**

   $P(Y=1|A=1,B=0,C=1) \propto P(A=1|Y=1)P(B=0|Y=1)P(C=1|Y=1)P(Y=1)$

   Using earlier values:

   - $P(A=1|Y=1)=1$, $P(B=0|Y=1)=\frac{1}{3}$, $P(C=1|Y=1)=\frac{1}{3}$, $P(Y=1)=\frac{3}{7}$

   $P(Y=1|A=1,B=0,C=1)=1\cdot\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{3}{7}=\frac{1}{21}$

2. **P(Y=0|A=1, B=0, C=1):**

   $P(Y=0|A=1,B=0,C=1) \propto P(A=1|Y=0)P(B=0|Y=0)P(C=1|Y=0)P(Y=0)$

   Using earlier values:

   - $P(A=1|Y=0)=\frac{1}{4}$, $P(B=0|Y=0)=\frac{1}{2}$, $P(C=1|Y=0)=\frac{1}{2}$, $P(Y=0)=\frac{4}{7}$

   $P(Y=0|A=1,B=0,C=1)=\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4}{7}=\frac{1}{28}$

Class: ($\hat{Y}$ = 1).

Final Results:

A	B	C	Y^\hat{Y}
X=0, 1, 0	?	Y^=0	
X=1, 1, 0	?	Y^=1	
X=0, 0, 1	?	Y^=0	
X=1, 0, 1	?	Y^=1	

سوال ۳: (۱۵ نمره)

برای یک مسئله‌ی طبقه‌بندی دو کلاسه، برای متغییر ورودی تک بعدی **x**، با در نظر گرفتن:

$$P(x|y=0) = N(0, \sigma^2)$$

$$P(x|y=1) = N(2, \sigma^2)$$

و فرض برابر بودن احتمال پسین هر دو کلاس، آستانه‌ی به حداقل رسیدن ریسک با فرض اینکه:

$$\begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix} = \begin{pmatrix} 0 & a \\ a^2 & 0 \end{pmatrix}$$

را بدست آوردید. تاثیر افزایش **a** را در آستانه‌ی بدست آمده تفسیر کنید.

Problem Breakdown

We are tasked with determining the decision threshold in a binary classification problem where:

- $P(x|y=0) = N(0, \sigma^2)$: Gaussian distribution with mean and variance .
- $P(x|y=1) = N(2, \sigma^2)$: Gaussian distribution with mean and variance .
- Prior probabilities for and are equal: .

  $y = 0$

  $y = 1$

  $P(y=0) = P(y=1) = 0.5$
- Loss matrix:

  $\Lambda = \begin{pmatrix} 0 & a \\ a^2 & 0 \end{pmatrix}$

The task is to derive the decision threshold xtx_t for minimizing risk and to explain how xtx_t changes with aa.

Step 1: Bayes Risk and Decision Rule

Bayes risk is minimized by assigning y=1 if the expected loss for y=1 is less than or equal to that for y=0. The decision rule is based on comparing posterior probabilities weighted by the loss matrix:

$a^2 P(y=0|x) \leq a P(y=1|x)$

Using Bayes' theorem:

$P(y=0|x) = \frac{P(x|y=0)P(y=0)}{P(x)}, \quad P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x)}$

The threshold is determined by comparing likelihoods because P(x) cancels:

$a^2 P(x|y=0)P(y=0) \leq a P(x|y=1)P(y=1)$

Since P(y=0) = P(y=1) = 0.5, this simplifies to:

$P(x|y=0) \leq aP(x|y=1)$

Dividing through by a>0a > 0 (assuming a≠0a \neq 0):

$aP(x|y=0) \leq P(x|y=1)$

Step 2: Likelihood Ratio and Threshold

The likelihoods are:

$P(x|y=0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right), \quad P(x|y=1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-2)^2}{2\sigma^2}\right)$

Taking the ratio $P(x|y=1)/P(x|y=0)$:

$P(x|y=0) = \exp\left(-\frac{(x-2)^2}{2\sigma^2} + \frac{x^2}{2\sigma^2}\right)$

Simplify the exponent:

$-2\sigma 2 = 4x - 42\sigma 2 = 2x - 2\sigma 2 - \frac{(x-2)^2}{2\sigma^2} + \frac{x^2}{2\sigma^2} = -\frac{x^2-4x+4}{2\sigma^2} + \frac{x^2}{2\sigma^2} = \frac{4x-4}{2\sigma^2} = \frac{2x-2}{\sigma^2}$

The decision rule becomes:

$a \leq \exp\left(\frac{2x-2}{\sigma^2}\right)$

Take the natural logarithm:

$\ln(a) \leq \frac{2x-2}{\sigma^2}$

Solve for xx:

$x \geq \frac{\sigma^2}{2}\ln(a) + 1$

Thus, the decision threshold is:

$x_t = \frac{\sigma^2}{2}\ln(a) + 1$

Step 3: Effect of Increasing aa

As aa increases:

1. ln(a) increases.
2. The threshold shifts to the right.

   x_t

This means that the classifier becomes more conservative about assigning $y = 1$, requiring a larger xx value before doing so.

Final Results

- **Decision Threshold:**

$x_t = \frac{\sigma^2}{2}\ln(a) + 1$

- **Effect of aa:**
  - Increasing raises , making the decision boundary favor .

سوال ۴: (۱۵ نمره)

یک مجموعه داده شامل دو کلاس داده‌ی دوبعدی C_1 و C_2 داریم. داده‌های هر کلاس از یک توزیع گاوسی با پارامترهای زیر تولید شده‌اند:

$$C_1 : N(\mu_1, \Sigma_1), \quad \mu_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C_2 : N(\mu_2, \Sigma_2), \quad \mu_2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

احتمال پیشین کلاس‌ها نیز به صورت زیر است:

$$P(C_1) = 0.6, \quad P(C_2) = 0.4$$

مرز تصمیم بیزی بین C_1 و C_2 را بدست آورید.

Answer:

To determine the **Bayes decision boundary** between two classes C1 and C2 with given Gaussian distributions, we follow a systematic approach. This involves comparing the posterior probabilities of each class and deriving the equation that delineates the regions where one class is more probable than the other.

Problem Statement

Given:

- **Class C1C_1:**

  $$N(\mu_1, \Sigma_1), \quad \mu_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} 4 & amp;0 \\ 0 & amp;1 \end{bmatrix}$$

- **Class C2C_2:**

  $$N(\mu_2, \Sigma_2), \quad \mu_2 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 2 & amp;0 \\ 0 & amp;3 \end{bmatrix}$$

- **Prior Probabilities:**

  $$P(C_1) = 0.6, \quad P(C_2) = 0.4$$

Objective:

Find the Bayes decision boundary between C1 and C2.

Step 1: Understanding the Bayes Decision Boundary

The **Bayes decision boundary** is the set of points x\mathbf{x} where the posterior probabilities of the two classes are equal:

$$P(C_1|\mathbf{x}) = P(C_2|\mathbf{x})$$

Using **Bayes' Theorem**, the posterior probability is given by:

$$P(C_i|\mathbf{x}) = \frac{P(\mathbf{x}|C_i)P(C_i)}{P(\mathbf{x})}$$

where:

- $P(\mathbf{x}|C_i)$is the **likelihood** of given class .

$\mathbf{x}$

- P(C_i) is the **prior** probability of class .
- $P(\mathbf{x})$is the **evidence** or marginal likelihood.

Setting the posterior probabilities equal:

$\frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|C_2)P(C_2)}{P(\mathbf{x})}$

$\Rightarrow P(\mathbf{x}|C_1)P(C_1) = P(\mathbf{x}|C_2)P(C_2)$

Step 2: Expressing the Likelihoods

Both classes follow a **multivariate Gaussian distribution**, so their likelihoods are:

$P(\mathbf{x}|C_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right)$

where:

- d is the dimensionality of the data (here).
- $|\Sigma_i|$is the determinant of the covariance matrix .
- $\Sigma_i^{-1}$is the inverse of .

Step 3: Deriving the Decision Boundary Equation

Starting from:

$P(\mathbf{x}|C_1)P(C_1) = P(\mathbf{x}|C_2)P(C_2)$

Taking the natural logarithm on both sides:

$\ln[P(\mathbf{x}|C_1)P(C_1)] = \ln[P(\mathbf{x}|C_2)P(C_2)]$

Expanding the logarithms:

$\ln P(\mathbf{x}|C_1) + \ln P(C_1) = \ln P(\mathbf{x}|C_2) + \ln P(C_2)$

Substituting the Gaussian likelihood expressions:

$$amp; -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_1| - \frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1) + \ln P(C_1)$$

$$amp; = -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_2| - \frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2) + \ln P(C_2)$$

Simplifying by canceling common terms:

$-\frac{1}{2}\ln|\Sigma_1| - \frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1) + \ln P(C_1) = -\frac{1}{2}\ln|\Sigma_2| - \frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2) + \ln P(C_2)$

Rearranging:

$(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2) - (\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1) + \ln\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) + 2\ln\left(\frac{P(C_1)}{P(C_2)}\right) = 0$

Step 4: Plugging in the Given Parameters

Given:

$$\mu_1 = \begin{bmatrix}2\\3\end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix}4 & amp;0\\0 & amp;1\end{bmatrix}$$

$$\mu_2 = \begin{bmatrix}5\\1\end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix}2 & amp;0\\0 & amp;3\end{bmatrix}$$

$P(C_1) = 0.6, \quad P(C_2) = 0.4$

First, compute the determinants and inverses of the covariance matrices:

1. **Determinants:**

   $|\Sigma_1| = 4 \times 1 - 0 \times 0 = 4$

   $|\Sigma_2| = 2 \times 3 - 0 \times 0 = 6$

2. **Inverses:**
   Since Σ1\Sigma_1 and Σ2\Sigma_2 are diagonal, their inverses are simply the reciprocals of the diagonal elements:

   $\Sigma_1^{-1} = \begin{bmatrix} \frac{1}{4} & amp; 0 \\ 0 & amp; 1 \end{bmatrix}, \quad \Sigma_2^{-1} = \begin{bmatrix} \frac{1}{2} & amp; 0 \\ 0 & amp; \frac{1}{3} \end{bmatrix}$

Now, substitute these into the decision boundary equation:

$(\mathbf{x} - \mu_2)^T \Sigma_2^{-1} (\mathbf{x} - \mu_2) - (\mathbf{x} - \mu_1)^T \Sigma_1^{-1} (\mathbf{x} - \mu_1) + \ln\left(\frac{6}{4}\right) + 2\ln\left(\frac{0.6}{0.4}\right) = 0$

Simplify the logarithmic terms:

$\ln\left(\frac{6}{4}\right) = \ln\left(\frac{3}{2}\right) \approx 0.4055$

$2\ln\left(\frac{0.6}{0.4}\right) = 2\ln(1.5) \approx 2 \times 0.4055 = 0.811$

$\text{Total constant term} = 0.4055 + 0.811 = 1.2165$

Thus, the equation becomes:

$(\mathbf{x} - \mu_2)^T \Sigma_2^{-1} (\mathbf{x} - \mu_2) - (\mathbf{x} - \mu_1)^T \Sigma_1^{-1} (\mathbf{x} - \mu_1) + 1.2165 = 0$

Step 5: Expanding the Quadratic Terms

Let $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Expand each quadratic form:

1. **For C2C_2:**

   $(\mathbf{x} - \mu_2)^T \Sigma_2^{-1} (\mathbf{x} - \mu_2) = \frac{(x_1 - 5)^2}{2} + \frac{(x_2 - 1)^2}{3}$

2. **For C1C_1:**

   $(\mathbf{x} - \mu_1)^T \Sigma_1^{-1} (\mathbf{x} - \mu_1) = \frac{(x_1 - 2)^2}{4} + (x_2 - 3)^2$

Substitute these into the boundary equation:

$\frac{(x_1 - 5)^2}{2} + \frac{(x_2 - 1)^2}{3} - \left(\frac{(x_1 - 2)^2}{4} + (x_2 - 3)^2\right) + 1.2165 = 0$

Expand each term:

$$\frac{x_1^2 - 10x_1 + 25}{2} + \frac{x_2^2 - 2x_2 + 1}{3}$$
$$- \left(\frac{x_1^2 - 4x_1 + 4}{4} + x_2^2 - 6x_2 + 9\right) + 1.2165 = 0$$

Simplify term by term:

1. **First Term:**

   $\frac{x_1^2 - 10x_1 + 25}{2} = 0.5x_1^2 - 5x_1 + 12.5$

2. **Second Term:**

   $\frac{x_2^2 - 2x_2 + 1}{3} \approx 0.333x_2^2 - 0.666x_2 + 0.333$

3. **Third Term:**

$\frac{x_1^2-4x_1+4}{4} = 0.25x_1^2 - x_1 + 1$

4. **Fourth Term:**

   $x_2^2 - 6x_2 + 9$

Substituting back:

$0.5x_1^2 - 5x_1 + 12.5 + 0.333x_2^2 - 0.666x_2 + 0.333 - (0.25x1^2 - x1 + 1 + x2^2 - 6x2 + 9) + 1.2165 = 0$

Distribute the negative sign:

$0.5x_1^2 - 5x_1 + 12.5 + 0.333x_2^2 - 0.666x_2 + 0.333 - 0.25x1^2 + x1 - 1 - x2^2 + 6x2 - 9 + 1.2165 = 0$

Combine like terms:

$(0.5x_1^2 - 0.25x_1^2) + (0.333x_2^2 - x2^2) + (-5x1 + x1) + (-0.666x2 + 6x2) + (12.5 + 0.333 - 1 - 9 + 1.2165) = 0$

$0.25x1^2 - 0.667x2^2 - 4x1 + 5.334x2 + 3.0495 = 0$

Step 6: Simplifying the Equation

The equation derived is:

$0.25x_1^2 - 0.667x_2^2 - 4x_1 + 5.334x_2 + 3.0495 = 0$

To eliminate decimals and make the equation more manageable, multiply both sides by **12** (the least common multiple of the denominators 4 and 3):

$12 \times 0.25x_1^2 - 12 \times 0.667x_2^2 - 12 \times 4x_1 + 12 \times 5.334x_2 + 12 \times 3.0495 = 0$

$3x_1^2 - 8x_2^2 - 48x_1 + 64x_2 + 36.594 = 0$

For exactness, note that:

$0.667 \approx \frac{2}{3}, \quad 5.334 \approx \frac{16}{3}, \quad 3.0495 \approx \frac{49}{16}$

But for practical purposes, we'll proceed with the decimal approximations.

Thus, the equation becomes:

$3x_1^2 - 8x_2^2 - 48x_1 + 64x_2 + 36.594 = 0$

Step 7: Converting to Standard Conic Form

To identify the type of conic section and its properties, we convert the equation into the **standard form** by completing the squares for both x1 and x2.

Group the x1x_1 and x2x_2 terms:

$3x_1^2 - 48x_1 - 8x_2^2 + 64x_2 = -36.594$

Factor out the coefficients of the squared terms:

$3(x_1^2 - 16x_1) - 8(x_2^2 - 8x_2) = -36.594$

Complete the square for each variable:

1. **For x1x_1:**

   $x_1^2 - 16x_1 = (x_1 - 8)^2 - 64$

2. **For x2x_2:**

   $x_2^2 - 8x_2 = (x_2 - 4)^2 - 16$

Substitute back into the equation:

$$3[(x_1 - 8)^2 - 64] - 8[(x_2 - 4)^2 - 16] = -36.594$$

Expand and simplify:

$$3(x_1 - 8)^2 - 192 - 8(x_2 - 4)^2 + 128 = -36.594$$

$$3(x_1 - 8)^2 - 8(x_2 - 4)^2 - 64 = -36.594$$

Bring constants to the right side:

$$3(x_1 - 8)^2 - 8(x_2 - 4)^2 = 27.406$$

Divide both sides to normalize:

$$\frac{(x_1-8)^2}{\frac{27.406}{3}} - \frac{(x_2-4)^2}{\frac{27.406}{8}} = 1$$

$$\frac{(x_1-8)^2}{9.135} - \frac{(x_2-4)^2}{3.42575} = 1$$

Step 8: Final Form of the Decision Boundary

The equation:

$$\frac{(x_1-8)^2}{9.135} - \frac{(x_2-4)^2}{3.42575} = 1$$

is the **standard form of a hyperbola** centered at (8,4), with:

- **Transverse Axis** along the axis.

  x1

- **Conjugate Axis** along the axis.

  x2

Parameters of the Hyperbola:

- **Center:**

  $(h, k) = (8, 4)$

- **Semi-transverse axis length (aa):**

  $\sqrt{9.135} \approx 3.022$

- **Semi-conjugate axis length (bb):**

  $\sqrt{3.42575} \approx 1.851$

Graphical Interpretation

The **Bayes decision boundary** is a hyperbola defined by the equation above. This hyperbola separates the feature space into two regions:

- **One side** where (i.e., the point is classified as).

  $P(C_1|\mathbf{x}) > P(C_2|\mathbf{x})$

- **The other side** where (i.e., the point is classified as).

  $P(C_2|\mathbf{x}) > P(C_1|\mathbf{x})$

Summary

The **Bayes decision boundary** between classes C1 and C2 with the given Gaussian distributions and prior probabilities is a **hyperbola** defined by the equation:

This boundary effectively partitions the feature space, ensuring that any new data point $\mathbf{x}$ is classified into the class with the higher posterior probability based on its position relative to this hyperbola.

Verification and Insights

- **Covariance Matrices:** The differing covariance matrices Σ1 and Σ2 lead to a quadratic (hyperbolic) decision boundary. If the covariance matrices were identical, the boundary would have been linear.
  $\frac{(x_1-8)^2}{9.135} - \frac{(x_2-4)^2}{3.42575} = 1$
- **Prior Probabilities:** The higher prior probability for $C_1 (P(C_1) = 0.6)$ slightly shifts the decision boundary towards C_2, reflecting the higher likelihood of data points belonging to C_1.
- **Interpretation:** Points closer to $\mu_1 = (2, 3)$ are more likely to belong to C_1, while points closer to $\mu_2 = (5, 1)$ are more likely to belong to C2. The hyperbolic boundary precisely captures this probabilistic separation.

سوال ۵ (شبیه سازی، ۲۰ نمره):

در یک مسئله طبقه‌بندی ۲ کلاسه، داده‌ها از ۲ توزیع گاوسی متمایز هستند:

$$C_1: p(x|C_1) = N(\mu_1, \sigma_1)$$
$$C_2: p(x|C_2) = N(\mu_2, \sigma_2)$$

احتمال پیشین کلاس‌ها نیز به صورت زیر است:

$$P(C_1) = \pi_1, \quad P(C_2) = \pi_2, \quad \pi_1 + \pi_2 = 1$$

الف) به ازای هر کدام از کلاس‌ها discriminant function را محاسبه کنید. سپس با کم کردن این دو تابع از هم، به یک discriminant function برای مسئله برسید.

$$g(x) = g_1(x) - g_2(x)$$

ب) با استفاده از $g(x)$ ، مرز تصمیم را برای حالت حداقل خطا بدست آورید.

ج) با افزایش احتمال پیشین π_1 و کاهش همزمان π_2 ، مرز تصمیم چگونه تغییر می‌کند؟ به طور شهودی توضیح دهید.

د) ۱۰۰ نمونه داده را از کلاس‌های C_1, C_2 نمونه‌برداری کنید.

$$(\mu_1 = 2, \sigma_1 = 1, \mu_2 = 4, \sigma_2 = 1, \pi_1 = 0.5, \pi_2 = 0.5)$$

و در یک نمودار scatter plot ، داده‌ها را نمایش دهید. مرز تصمیمی را نیز در نمودار رسم کنید.

ه) بر اساس مرز تصمیم، لیبل داده‌ها را مشخص کنید. سپس confusion matrix, accuracy, precision, recall, F1-score را برای این طبقه‌بند محاسبه و گزارش کنید.

و) نمودار ROC را محاسبه و رسم کنید. مساحت زیر این نمودار(AUROC) را نیز محاسبه و گذارش کنید. برای به دست آوردن مقادیر لازم برای رسم این نمودار، می‌توانید مرز تصمیم را به صورت $g(x) = \tau$ تعریف کنید و با تغییر دادن مقدار آستانه τ از $-\infty$ تا $+\infty$ به ازای τ های مختلف true positive rate و false positive rate را محاسبه کنید.

و) موارد "د" و "ه" و "و" را به ازای $\pi_1 = 0.9, \pi_2 = 0.1$ تکرار کنید و تغییرات معیارهای ارزیابی را تحلیل کنید. کدام معیارها در این شرایط مناسب‌تر هستند؟

(a) Derivation of Discriminant Functions

In a Bayesian classifier, the discriminant function for class CiC_iCi is defined as:

$g_i(x) = \ln p(x|C_i) + \ln P(C_i)$

Since the class-conditional densities are Gaussian distributions, we have:

$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$

Taking the natural logarithm of the likelihood:

$\ln p(x|C_i) = -\frac{(x-\mu_i)^2}{2\sigma_i^2} - \ln(\sqrt{2\pi}\sigma_i)$

Therefore, the discriminant function for each class is:

$\ln p(x|C_i) = -\frac{(x-\mu_i)^2}{2\sigma_i^2} - \ln(\sqrt{2\pi}\sigma_i)$

The single discriminant function is the difference between the two class discriminant functions:

$g_i(x) = -\frac{(x-\mu_i)^2}{2\sigma_i^2} - \ln(\sqrt{2\pi}\sigma_i) + \ln \pi_i$

Substituting the expressions for $g_1(x)$ and $g_2(x)$:

$$
\begin{aligned}
g(x) &= \left(-\frac{(x-\mu_1)^2}{2\sigma_1^2} - \ln(\sqrt{2\pi}\sigma_1) + \ln \pi_1\right) \\
&\quad - \left(-\frac{(x-\mu_2)^2}{2\sigma_2^2} - \ln(\sqrt{2\pi}\sigma_2) + \ln \pi_2\right) \\
&= -\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2} - \ln\left(\frac{\sigma_1}{\sigma_2}\right) + \ln\left(\frac{\pi_1}{\pi_2}\right)
\end{aligned}
$$

This is the discriminant function $g(x)$ for the problem.

(b) Decision Boundary for Minimizing Classification Error

To find the decision boundary, set g(x)=0:

$-\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2} - \ln\left(\frac{\sigma_1}{\sigma_2}\right) + \ln\left(\frac{\pi_1}{\pi_2}\right) = 0$

Assuming equal variances (σ1=σ2=σ\sigma_1 = \sigma_2 = \sigmaσ1=σ2=σ):

$-\frac{(x-\mu_1)^2}{2\sigma^2} + \frac{(x-\mu_2)^2}{2\sigma^2} + \ln\left(\frac{\pi_1}{\pi_2}\right) = 0$

Simplify:

$(x-\mu_2)^2 - (x-\mu_1)^2 = 2\sigma^2 \ln\left(\frac{\pi_1}{\pi_2}\right)$

Expand and rearrange:

$2x(\mu_1 - \mu_2) + (\mu_2^2 - \mu_1^2) = 2\sigma^2 \ln\left(\frac{\pi_1}{\pi_2}\right)$

Solve for xxx:

$x = \frac{\sigma^2 \ln\left(\frac{\pi_1}{\pi_2}\right) - \frac{1}{2}(\mu_2^2 - \mu_1^2)}{\mu_1 - \mu_2}$

(c) Effect of Changing Prior Probabilities on the Decision Boundary

When π_2 increases and π_1 decreases, the term $\ln\left(\frac{\pi_1}{\pi_2}\right)$ decreases (becomes more negative). This shifts the decision boundary toward μ1\mu_1μ1, meaning the classifier becomes more likely to assign samples to class C2

Intuitively, increasing the prior probability of class C2 influences the classifier to favor C2, thus moving the boundary closer to C1 's mean.

Classification Metrics: Manual Computation Without `scikit-learn`

In this section, we will manually compute the confusion matrix and key performance metrics (`Accuracy`, `Precision`, `Recall`, and `F1-Score`) without relying on external libraries like `scikit-learn`. Understanding these computations helps in grasping the foundational concepts of classification evaluation.

Overview

1. **Confusion Matrix**: A table that describes the performance of a classification model by comparing actual and predicted labels.
2. **Performance Metrics**:
   - **Accuracy**: Overall correctness of the model.
   - **Precision**: Correctness of positive predictions.
   - **Recall (Sensitivity)**: Ability of the model to find all positive instances.
   - **F1-Score**: Harmonic mean of Precision and Recall.

Setting Up

Assuming you have already generated your data, computed predictions, and have the true and predicted labels (`y_true_new` and `y_pred_new` respectively), we will proceed with manual computations.

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

# Assuming the following variables are already defined:
# y_true_new: Array of true class labels (0 or 1)
# y_pred_new: Array of predicted class labels (0 or 1)
```

1. Manually Computing the Confusion Matrix

The confusion matrix summarizes the counts of correct and incorrect predictions categorized by each class.

Definitions

- **True Positive (TP)**: Correctly predicted positive instances.
- **False Positive (FP)**: Incorrectly predicted positive instances (Type I error).
- **True Negative (TN)**: Correctly predicted negative instances.
- **False Negative (FN)**: Incorrectly predicted negative instances (Type II error).

Implementation

```
# Initialize counts
TP = FP = TN = FN = 0
```

```
# Iterate through true and predicted labels to compute counts
for true_label, pred_label in zip(y_true_new, y_pred_new):
    if true_label == 1 and pred_label == 1:
        TP += 1  # True Positive
    elif true_label == 0 and pred_label == 1:
        FP += 1  # False Positive
    elif true_label == 0 and pred_label == 0:
        TN += 1  # True Negative
    elif true_label == 1 and pred_label == 0:
        FN += 1  # False Negative

# Construct the confusion matrix as a 2x2 NumPy array
confusion_mat_manual = np.array([[TN, FP],
                                 [FN, TP]])

# Display the confusion matrix
print("Confusion Matrix (Manual Computation):")
print(confusion_mat_manual)
```

Example Output

```
Confusion Matrix (Manual Computation):
[[TN FP]
 [FN TP]]
```

Replace `TN`, `FP`, `FN`, and `TP` with their computed values.

2. Manually Computing Performance Metrics

2.1 Accuracy

Definition: The proportion of correct predictions out of all predictions.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Implementation

```
# Total number of samples
total = TP + TN + FP + FN

# Compute Accuracy
accuracy_manual = (TP + TN) / total if total > 0 else 0

# Display Accuracy
print(f"Accuracy (Manual Computation): {accuracy_manual:.2f}")
```

2.2 Precision

Definition: The proportion of positive predictions that are actually correct.

$\text{Precision} = \frac{TP}{TP+FP}$

Implementation

```
# Compute Precision
precision_manual = TP / (TP + FP) if (TP + FP) > 0 else 0

# Display Precision
print(f"Precision (Positive Class, Manual Computation): {precision_manual:.2f}")
```

2.3 Recall (Sensitivity)

Definition: The proportion of actual positive instances that were correctly identified.

$\text{Recall} = \frac{TP}{TP+FN}$

Implementation

```
# Compute Recall
recall_manual = TP / (TP + FN) if (TP + FN) > 0 else 0

# Display Recall
print(f"Recall (Positive Class, Manual Computation): {recall_manual:.2f}")
```

2.4 F1-Score

Definition: The harmonic mean of Precision and Recall, providing a balance between them.

$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Implementation

```
# Compute F1-Score
f1_score_manual = 2 * (precision_manual * recall_manual) / (precision_manual + recall_m
anual) if (precision_manual + recall_manual) > 0 else 0

# Display F1-Score
print(f"F1-Score (Positive Class, Manual Computation): {f1_score_manual:.2f}")
```

3. Combining All Metrics

For convenience, here's a consolidated code block that computes all metrics together:

```
# Initialize counts
TP = FP = TN = FN = 0

# Compute counts
for true_label, pred_label in zip(y_true_new, y_pred_new):
    if true_label == 1 and pred_label == 1:
        TP += 1  # True Positive
    elif true_label == 0 and pred_label == 1:
```

```
        FP += 1  # False Positive
    elif true_label == 0 and pred_label == 0:
        TN += 1  # True Negative
    elif true_label == 1 and pred_label == 0:
        FN += 1  # False Negative

# Construct confusion matrix
confusion_mat_manual = np.array([[TN, FP],
                                 [FN, TP]])
print("Confusion Matrix (Manual Computation):")
print(confusion_mat_manual)

# Compute Accuracy
total = TP + TN + FP + FN
accuracy_manual = (TP + TN) / total if total > 0 else 0

# Compute Precision
precision_manual = TP / (TP + FP) if (TP + FP) > 0 else 0

# Compute Recall
recall_manual = TP / (TP + FN) if (TP + FN) > 0 else 0

# Compute F1-Score
f1_score_manual = 2 * (precision_manual * recall_manual) / (precision_manual + recall_m
anual) if (precision_manual + recall_manual) > 0 else 0

# Display metrics
print(f"Accuracy (Manual Computation): {accuracy_manual:.2f}")
print(f"Precision (Positive Class, Manual Computation): {precision_manual:.2f}")
print(f"Recall (Positive Class, Manual Computation): {recall_manual:.2f}")
print(f"F1-Score (Positive Class, Manual Computation): {f1_score_manual:.2f}")
```

4. Verification: Comparing Manual Computations with `scikit-learn`

To ensure the correctness of our manual computations, we can compare them with the results from `scikit-learn`. Although you mentioned not wanting to use libraries for computations, this step is optional and serves as a sanity check.

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_s
core, f1_score

# Using scikit-learn for comparison
confusion_mat_sklearn = confusion_matrix(y_true_new, y_pred_new)
accuracy_sklearn = accuracy_score(y_true_new, y_pred_new)
precision_sklearn = precision_score(y_true_new, y_pred_new)
recall_sklearn = recall_score(y_true_new, y_pred_new)
f1_score_sklearn = f1_score(y_true_new, y_pred_new)

print("\nComparison with scikit-learn:")
```

```
print("Confusion Matrix (sklearn):")
print(confusion_mat_sklearn)
print(f"Accuracy (sklearn): {accuracy_sklearn:.2f}")
print(f"Precision (sklearn): {precision_sklearn:.2f}")
print(f"Recall (sklearn): {recall_sklearn:.2f}")
print(f"F1-Score (sklearn): {f1_score_sklearn:.2f}")
```

Note: This step is optional and primarily for validation purposes.

Summary

In this section, we manually computed the confusion matrix and key performance metrics (`Accuracy`, `Precision`, `Recall`, and `F1-Score`) without relying on external libraries. This approach provides a deeper understanding of how these metrics are derived and their significance in evaluating classification models.

By iterating through the true and predicted labels, we counted the occurrences of True Positives, False Positives, True Negatives, and False Negatives. Using these counts, we calculated the performance metrics using their respective formulas. This manual computation is crucial for educational purposes and situations where you might need to implement custom evaluation metrics.

Note: While manual computations are valuable for understanding, leveraging optimized library functions (like those from `scikit-learn`) is recommended for efficiency and reliability in production environments.

Classification with Updated Prior Probabilities and Manual ROC Curve Computation

In this notebook, we will:

1. **Generate new samples** from two classes with updated prior probabilities.
2. **Compute the decision boundary** based on the new priors.
3. **Classify the samples** using the new decision boundary.
4. **Manually compute** the confusion matrix and performance metrics.
5. **Manually compute and plot** the ROC curve and calculate the AUROC.

Importing Necessary Libraries

We will use `numpy` for numerical computations and `matplotlib` for plotting.

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

Setting Parameters

We define the means, standard deviations, and new prior probabilities for the two classes.

```
# Class parameters
mu1 = 2       # Mean of Class 1
sigma1 = 1    # Standard deviation of Class 1
mu2 = 4       # Mean of Class 2
```

```
sigma2 = 1    # Standard deviation of Class 2

# Updated prior probabilities
pi1_new = 0.9  # Prior probability of Class 1
pi2_new = 0.1  # Prior probability of Class 2

# Number of samples
n_samples = 100

# Set random seed for reproducibility
np.random.seed(10)
```

Generating Samples with New Prior Probabilities

We generate class labels according to the new prior probabilities and then generate samples from the corresponding normal distributions.

```
# Generate class labels according to new mixing probabilities
classes_new = np.random.choice([0, 1], size=n_samples, p=[pi1_new, pi2_new])

# Define means and standard deviations for each class
mu = np.array([mu1, mu2])
sigma = np.array([sigma1, sigma2])

# Generate samples based on new class labels
X_new = np.random.normal(mu[classes_new], sigma[classes_new])

# True class labels
y_true_new = classes_new
```

Visualizing the Samples

We separate the samples based on their class labels for visualization.

```
# Separate samples for plotting
x1_new = X_new[classes_new == 0]  # Samples from Class 1
x2_new = X_new[classes_new == 1]  # Samples from Class 2

# Plot the samples
plt.figure(figsize=(10, 6))
plt.scatter(x1_new, np.zeros_like(x1_new), color='blue', alpha=0.6, label='Class 1')
plt.scatter(x2_new, np.zeros_like(x2_new), color='red', alpha=0.6, label='Class 2')
plt.title('Scatter Plot of Samples from Class 1 and Class 2 (New Priors)')
plt.xlabel('Feature x')
plt.yticks([])
plt.legend()
plt.grid(True)
plt.show()
```

Explanation:

- We plot the samples on a one-dimensional scatter plot since our feature `x` is one-dimensional.
- The `np.zeros_like` function is used to place all points along the x-axis (at `y = 0`).

Calculating the New Decision Boundary

We compute the new decision boundary (threshold) based on the updated prior probabilities.

```
# Assuming equal variances
sigma = sigma1

# Calculate the new decision boundary
threshold_new = ((mu1 + mu2) / 2) + (sigma**2 / (mu2 - mu1)) * np.log(pi1_new / pi2_ne
w)
print(f"New Decision Boundary (Threshold): x = {threshold_new:.2f}")
```

Explanation:

- The decision boundary is derived from the equality of the discriminant functions of the two classes.
- The formula adjusts the threshold based on the log ratio of the new prior probabilities.

Visualizing the New Decision Boundary

We plot the samples again and add the new decision boundary.

```
# Plot with new decision boundary
plt.figure(figsize=(10, 6))
plt.scatter(x1_new, np.zeros_like(x1_new), color='blue', alpha=0.6, label='Class 1')
plt.scatter(x2_new, np.zeros_like(x2_new), color='red', alpha=0.6, label='Class 2')
plt.axvline(x=threshold_new, color='purple', linestyle='--',
            label=f'New Decision Boundary (x={threshold_new:.2f})')
plt.title('Scatter Plot with New Decision Boundary')
plt.xlabel('Feature x')
plt.yticks([])
plt.legend()
plt.grid(True)
plt.show()
```

Explanation:

- The vertical dashed line represents the new decision boundary.
- Samples to the left of the threshold are classified as Class 1, and those to the right as Class 2.

Defining the Classification Function

We define a function to classify samples based on the threshold.

```
def classify(x, threshold):
    return np.where(x >= threshold, 1, 0)
```

Explanation:

- The function returns `1` if `x` is greater than or equal to the threshold (Class 2) and `0` otherwise (Class 1).

Predicting Class Labels

We apply the classification function to our samples.

```
# Predicted class labels using the new threshold
y_pred_new = classify(X_new, threshold_new)
```

Manually Computing the Confusion Matrix

We manually compute the confusion matrix by counting true positives, false positives, true negatives, and false negatives.

```
# Initialize counts
TP = FP = TN = FN = 0

# Compute counts
for true_label, pred_label in zip(y_true_new, y_pred_new):
    if true_label == 1 and pred_label == 1:
        TP += 1  # True Positive
    elif true_label == 0 and pred_label == 1:
        FP += 1  # False Positive
    elif true_label == 0 and pred_label == 0:
        TN += 1  # True Negative
    elif true_label == 1 and pred_label == 0:
        FN += 1  # False Negative

# Construct the confusion matrix
confusion_mat_new = np.array([[TN, FP],
                              [FN, TP]])

print("Confusion Matrix with New Priors:")
print(confusion_mat_new)
```

Explanation:

- **True Positive (TP):** Correctly predicted Class 2 samples.
- **False Positive (FP):** Samples from Class 1 incorrectly predicted as Class 2.
- **True Negative (TN):** Correctly predicted Class 1 samples.
- **False Negative (FN):** Samples from Class 2 incorrectly predicted as Class 1.

Computing Performance Metrics Manually

We calculate accuracy, precision, recall, and F1-score without using external libraries.

```
# Total number of samples
total = TP + TN + FP + FN

# Compute performance metrics
```

```
accuracy_new = (TP + TN) / total if total > 0 else 0
precision_new = TP / (TP + FP) if (TP + FP) > 0 else 0
recall_new = TP / (TP + FN) if (TP + FN) > 0 else 0
f1_score_new = 2 * (precision_new * recall_new) / (precision_new + recall_new) if (prec
ision_new + recall_new) > 0 else 0

# Display performance metrics
print(f"Accuracy: {accuracy_new:.2f}")
print(f"Precision (Positive Class): {precision_new:.2f}")
print(f"Recall (Positive Class): {recall_new:.2f}")
print(f"F1-Score (Positive Class): {f1_score_new:.2f}")
```

Explanation:

- **Accuracy:** Proportion of correct predictions.
- **Precision:** Proportion of positive predictions that are correct.
- **Recall (Sensitivity):** Proportion of actual positives correctly identified.
- **F1-Score:** Harmonic mean of precision and recall.

Defining the Discriminant Score Function

We define the discriminant score function based on the Gaussian distributions and prior probabilities.

```
def discriminant_score_new(x):
    term1 = -((x - mu1)**2) / (2 * sigma1**2) + np.log(pi1_new)
    term2 = -((x - mu2)**2) / (2 * sigma2**2) + np.log(pi2_new)
    return term1 - term2
```

Explanation:

- The discriminant score is the difference between the log-likelihoods of the two classes for each sample.
- It incorporates both the class conditional probabilities and the prior probabilities.

Computing Discriminant Scores

We compute the discriminant scores for all samples.

```
# Compute the discriminant scores
scores_new = discriminant_score_new(X_new)
```

Manually Computing the ROC Curve

We compute the True Positive Rate (TPR) and False Positive Rate (FPR) at various thresholds to construct the ROC curve.

Generating Thresholds

```
# Generate a list of thresholds
thresholds = np.sort(np.unique(scores_new))
```

```
thresholds = np.concatenate(([scores_new.min() - 1], thresholds, [scores_new.max() +
1]))
```

Explanation:

- We create a range of thresholds by sorting the unique discriminant scores.
- We extend the thresholds beyond the minimum and maximum scores to cover all possible cases.

Computing TPR and FPR for Each Threshold

```
# Initialize lists to store TPR and FPR
tpr_list = []
fpr_list = []

# Loop over thresholds
for thresh in thresholds:
    # Classify samples using the current threshold
    y_pred_thresh = np.where(scores_new >= thresh, 1, 0)

    # Compute TP, FP, TN, FN
    TP = np.sum((y_true_new == 1) & (y_pred_thresh == 1))
    FP = np.sum((y_true_new == 0) & (y_pred_thresh == 1))
    TN = np.sum((y_true_new == 0) & (y_pred_thresh == 0))
    FN = np.sum((y_true_new == 1) & (y_pred_thresh == 0))

    # Compute TPR and FPR
    TPR = TP / (TP + FN) if (TP + FN) > 0 else 0
    FPR = FP / (FP + TN) if (FP + TN) > 0 else 0

    # Append TPR and FPR to lists
    tpr_list.append(TPR)
    fpr_list.append(FPR)
```

Explanation:

- For each threshold, we classify the samples and compute TP, FP, TN, and FN.
- **True Positive Rate (TPR):** Sensitivity or recall.
- **False Positive Rate (FPR):** Proportion of negatives incorrectly classified as positives.

Preparing Data for Plotting

```
# Convert lists to arrays
tpr_array = np.array(tpr_list)
fpr_array = np.array(fpr_list)

# Sort the arrays for plotting
sorted_indices = np.argsort(fpr_array)
fpr_array = fpr_array[sorted_indices]
tpr_array = tpr_array[sorted_indices]
```

Explanation:

- Sorting ensures that the FPR values are in ascending order, which is necessary for correctly plotting the ROC curve.

Computing the Area Under the ROC Curve (AUROC)

We calculate the AUROC using the trapezoidal rule.

```
# Compute AUROC
auroc_new = np.trapz(tpr_array, fpr_array)
print(f"Area Under the ROC Curve (AUROC) with New Priors: {auroc_new:.2f}")
```

Explanation:

- The trapezoidal rule approximates the integral of TPR with respect to FPR, giving us the AUROC.
- An AUROC of 1 represents perfect classification, while 0.5 represents random guessing.

Plotting the ROC Curve

We visualize the ROC curve using `matplotlib`.

```
# Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr_array, tpr_array, color='darkorange', lw=2,
         label=f'ROC curve (AUROC = {auroc_new:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.title('ROC Curve with New Priors')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

Explanation:

- The ROC curve plots TPR (y-axis) against FPR (x-axis).
- The dashed diagonal line represents the performance of a random classifier.
- The area under the curve (AUROC) provides a single scalar value to assess the overall performance of the classifier.

Summary

In this notebook, we:

- Generated samples from two classes with updated prior probabilities.
- Calculated the new decision boundary based on the updated priors.
- Classified the samples using the new decision boundary.
- Manually computed the confusion matrix and performance metrics without using external libraries.
- Manually computed and plotted the ROC curve, and calculated the AUROC.

This process provided a comprehensive understanding of how prior probabilities affect the decision boundary and the classifier's performance. By manually computing the performance metrics and ROC curve, we gained deeper insights into the mechanics of classification and evaluation metrics.

سوال ۶: (۱۵ نمره)

مسئله بهینه‌سازی زیر را در نظر بگیرید:

$$\min_{x\in\mathbb{R}^2} f(x) = \frac{1}{2}x^TQx + c^Tx$$
$$subject\ to\ Ax \leq b$$
$$Q \in \mathbb{R}^{2\times2}, c \in \mathbb{R}^2, A \in \mathbb{R}^{3\times2}, b \in \mathbb{R}^3$$

با استفاده از ضرایب لاگرانژ، مسئله دوگان این مسئله را بدست آورید و آن را بر حسب ضرایب لاگرانژ بنویسید.

Complete Solution Using KKT Conditions

To fully solve the given optimization problem using the Karush-Kuhn-Tucker (KKT) conditions, we'll follow a systematic approach. This involves formulating the primal and dual problems, deriving the KKT conditions, and solving them to find the optimal solution.

Problem Statement

We are given the following quadratic optimization problem:

$$\begin{aligned} \underset{x\in\mathbb{R}^2}{\text{minimize}} \quad & f(x) = \frac{1}{2}x^TQx + c^Tx \\ \text{subject to} \quad & Ax \leq b \end{aligned}$$

where:

- $Q \in \mathbb{R}^{2\times2}$ is a symmetric positive definite matrix.
- $c \in \mathbb{R}^2$ is a coefficient vector.
- $A \in \mathbb{R}^{3\times2}$ is the constraint matrix.
- $b \in \mathbb{R}^3$ is the constraint vector.

Step 1: Formulating the Dual Problem

To derive the dual problem, we'll use the method of Lagrange multipliers.

1.1. Lagrangian Function

The Lagrangian $L(x, \lambda)$ for the primal problem is defined as:

$L(x, \lambda) = \frac{1}{2}x^T Qx + c^T x + \lambda^T (Ax - b)$

where$\lambda \in \mathbb{R}^3$ are the Lagrange multipliers associated with the inequality constraints $Ax \leq b, and \lambda \geq 0$.

1.2. Dual Function

The dual function g(λ)g(\lambda) is the infimum of the Lagrangian over xx:

$g(\lambda) = \inf_x L(x, \lambda)$

To find $g(\lambda)$, we minimize $L(x, \lambda)$with respect to xx.

1.3. Minimizing the Lagrangian

To minimize$L(x, \lambda)$ with respect to xx, we take the gradient of L with respect to xx and set it to zero:

$\nabla_x L(x, \lambda) = Qx + c + A^T \lambda = 0$

Solving for xx:

$Qx = -(c + A^T \lambda) \quad \Rightarrow \quad x = -Q^{-1}(c + A^T \lambda)$

Note: We assume QQ is invertible, which holds if QQ is positive definite.

1.4. Substituting xx Back into the Lagrangian

Substitute x=−Q−1(c+ATλ)x = -Q^{-1} (c + A^T \lambda) into L(x,λ)L(x, \lambda):

$$
\begin{aligned}
g(\lambda) amp; = \frac{1}{2}x^T Qx + c^T x + \lambda^T (Ax - b) \\
amp; = \frac{1}{2}(-Q^{-1}(c + A^T \lambda))^T Q(-Q^{-1}(c + A^T \lambda)) + c^T(-Q^{-1}(c + A^T \lambda)) + \lambda^T (A(-Q^{-1}(c + A^T \lambda)) - b) \\
amp; = \frac{1}{2}(c + A^T \lambda)^T Q^{-1}(c + A^T \lambda) - (c + A^T \lambda)^T Q^{-1} c - \lambda^T b \\
amp; = -\frac{1}{2}(c + A^T \lambda)^T Q^{-1}(c + A^T \lambda) - \lambda^T b
\end{aligned}
$$

1.5. Dual Problem

The dual problem is to maximize g(λ)g(\lambda) subject to λ≥0:

$\underset{\lambda \geq 0}{\text{maximize}} g(\lambda) = -\frac{1}{2}(c + A^T \lambda)^T Q^{-1}(c + A^T \lambda) - \lambda^T b$

Summary of Dual Problem:

$\text{Dual Problem:} \quad \underset{\lambda \geq 0}{\text{maximize}} -\frac{1}{2}(c + A^T \lambda)^T Q^{-1}(c + A^T \lambda) - \lambda^T b$

Step 2: Deriving and Solving KKT Conditions

The KKT conditions provide necessary (and under convexity, sufficient) conditions for optimality. We'll outline these conditions and solve them for our problem.

2.1. KKT Conditions

For our optimization problem, the KKT conditions are:

1. **Primal Feasibility:**

   $Ax \leq b$

2. **Dual Feasibility:**

$\lambda \geq 0$

3. **Stationarity:**

   $Qx + c + A^T\lambda = 0$

4. **Complementary Slackness:**

   $\lambda_i(A_ix - b_i) = 0 \quad \forall i = 1, 2, 3$

2.2. Solving the KKT Conditions

We'll solve the KKT conditions step-by-step.

Step 2.2.1: Express xx in Terms of λ\lambda

From the **Stationarity** condition:

$Qx + c + A^T\lambda = 0 \quad \Rightarrow \quad x = -Q^{-1}(c + A^T\lambda)$

Step 2.2.2: Substitute xx into the Primal Feasibility Condition

$Ax \leq b \quad \Rightarrow \quad A(-Q^{-1}(c + A^T\lambda)) \leq b \quad \Rightarrow \quad -AQ^{-1}A^T\lambda \leq b + AQ^{-1}c$

Step 2.2.3: Incorporate Complementary Slackness

For each constraint i=1,2,3i = 1, 2, 3:

$\lambda_i(A_ix - b_i) = 0$

Substituting $x = -Q^{-1}(c + A^T\lambda)$:

$\lambda_i(A_i(-Q^{-1}(c + A^T\lambda)) - b_i) = 0 \quad \Rightarrow \quad \lambda_i(-A_iQ^{-1}c - A_iQ^{-1}A^T\lambda - b_i) = 0$

Simplifying:

$\lambda_i(-A_iQ^{-1}A^T\lambda - A_iQ^{-1}c - b_i) = 0 \quad \forall i$

This leads to:

$\lambda_i(-(A_iQ^{-1}A^T)\lambda - A_iQ^{-1}c - b_i) = 0 \quad \forall i$

Step 2.2.4: Analyzing the Conditions

The Complementary Slackness condition implies that for each ii:

- If , then the corresponding constraint is active: .

  $\lambda_i > 0$

  $A_ix = b_i$

- If , then .

  $A_ix < b_i$

  $\lambda_i = 0$

Given the complexity of the general case, we'll outline the procedure to solve these conditions:

1. **Identify Active Constraints:**

   Determine which constraints are active$(A_ix = b_i)$ and which are inactive $(A_ix < b_i)$. This may involve considering different cases where different combinations of constraints are active.

2. **Solve for λ:**

   For active constraints, set $\lambda_i \geq 0$and solve the resulting system of equations.

3. **Ensure Feasibility:**

Verify that the solutions satisfy all primal and dual feasibility conditions.
