CS509 Pattern Recognition Assignment 1

Wentao Lu (002276355), Yi Ren (002269013)

1 Operations with events

Let A, B and C be three independent events.

1. $P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B \cap C)$

$= P(A) + P(B) + P(C) - P(A)P(B)P(C)$

$= 1 - 0.2 \times 0.5 \times 0.3 = 0.97$

2. $P(A \cap B) = 0.2 = P(A)P(B) \Rightarrow P(A) = \frac{0.2}{P(B)} = \frac{0.2}{0.8} = \frac{1}{4}$

$\Rightarrow P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A)P(B)P(C)$

$= 0.25 + 0.8 + 0.1 - 0.25 \times 0.8 \times 0.1$

$= 1.15 - 0.02 = 1.13$

3. $P(A) = 1 - P(\bar{A}) = 1 - 0.2 = 0.8$

$P(C) = 1 - P(\bar{C}) = 1 - 0.4 = 0.6$

$P(A \cap B) = P(A)P(B) = 0.8P(B) = 0.16 \Rightarrow P(B) = 0.2$

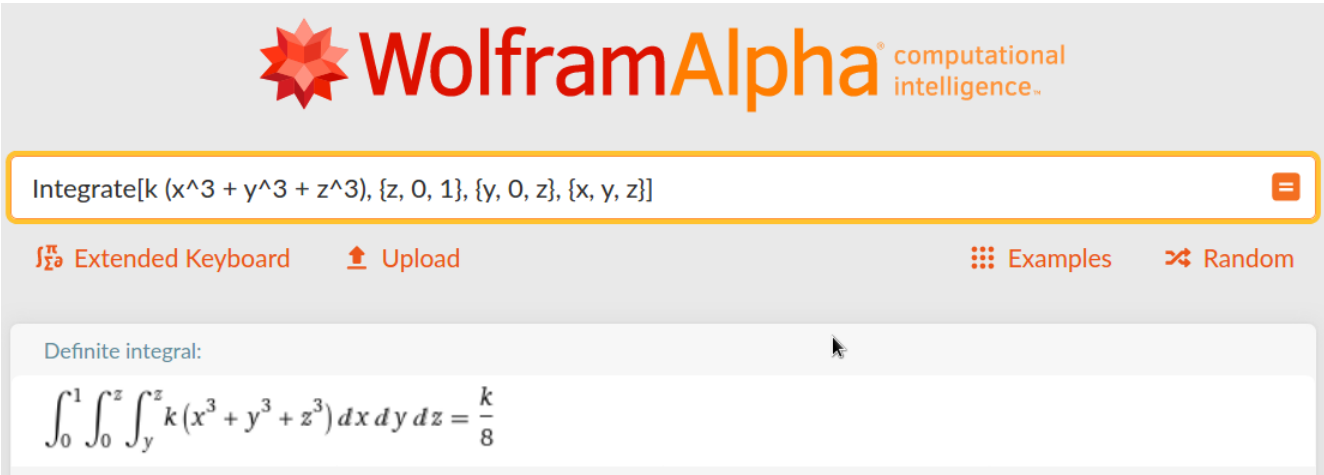
$\Rightarrow P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A)P(B)P(C)$

$= 0.8 + 0.2 + 0.6 - 0.8 \times 0.2 \times 0.6$

$= 1.504$

2 Law of probability of a triplet of continuous random variables

1. Using mathematical programming languages such as WOLFRAM MATHEMATICA or SAGE, we can easily obtain that:



Since the integral of $f(x, y, z)$ must sum up to 1, we have $k/8 = 1 \Rightarrow k = 8$. Alternatively, we can also compute integrals the hard way:

$$\iiint_D (x^3+y^3+z^3)\,dx\,dy\,dz$$

$$= \iint \left[\int_y^z (x^3+y^3+z^3)\,dx\right] dy\,dz = \iint \left[\left(\frac{1}{4}x^4 + y^3 x + z^3 x\right)\Big|_{x=y}^{x=z}\right] dy\,dz$$

$$= \iint \left(\frac{5}{4}z^4 - \frac{5}{4}y^4 + y^3 z - z^3 y\right) dy\,dz$$

$$= \int \left[\int_0^z \left(\frac{5}{4}z^4 - \frac{5}{4}y^4 + y^3 z - z^3 y\right) dy\right] dz$$

$$= \int \left[\left(\frac{5}{4}z^4 y - \frac{y^5}{4} + \frac{z}{4}y^4 - \frac{z^3}{2}y^2\right)\Big|_{y=0}^{y=z}\right] dz$$

$$= \int_0^1 \left(\frac{5}{4}z^5 - \frac{1}{4}z^5 + \frac{z^5}{4} - \frac{z^5}{2}\right) dz = \int_0^1 \frac{3}{4}z^5\,dz$$

$$= \frac{3}{4}\cdot\frac{z^6}{6}\Big|_{z=0}^{z=1} = \frac{1}{8} = \frac{1}{k} \Longrightarrow k = 8$$

2. The marginal pdfs of X, Y, Z can be computed manually:

$$\iint 8(x^3+y^3+z^3)\,dy\,dz$$

$$= \int \left[\int_0^z 8(x^3+y^3+z^3)\,dy\right] dz = 8\int \left[(x^3y+\tfrac{1}{4}y^4+z^3y)\Big|_{y=0}^{y=x}\right] dz$$

$$= 8\int (\tfrac{5}{4}x^4+z^3x)\,dz = 8(\tfrac{5}{4}x^4z+x\,\tfrac{z^4}{4})\Big|_{z=x}^{z=1}$$

$$= 8(\tfrac{5}{4}x^4+\tfrac{x}{4}-\tfrac{5}{4}x^5-\tfrac{x^5}{4})$$

$$= -12x^5+10x^4+2x$$

$$\Rightarrow \text{margin}(x) = \begin{cases} -12x^5+10x^4+2x, & 0 \le x \le 1 \\ 0 & \text{o.w.} \end{cases}$$

$$\iint 8(x^3+y^3+z^3)\,dx\,dz = 8\int \left[\int (x^3+y^3+z^3)\,dx\right] dz$$

$$= 8\int \left[(\tfrac{x^4}{4}+y^3x+z^3x)\Big|_{x=y}^{x=z}\right] dz$$

$$= 8\cdot\int_{z=y}^{z=1} (\tfrac{5}{4}z^4+y^3z-\tfrac{5}{4}y^4-z^3y)\,dz$$

$$= 8(\tfrac{z^5}{4}+y^3\cdot\tfrac{z^2}{2}-\tfrac{5}{4}y^4z-y\cdot\tfrac{z^4}{4})\Big|_{z=y}^{z=1}$$

$$= 6y^5-10y^4+4y^3-2y+2$$

$$\Rightarrow \text{margin}(y) = \begin{cases} 6y^5-10y^4+4y^3-2y+2, & 0 \le y \le 1 \\ 0 & \text{; o.w.} \end{cases}$$

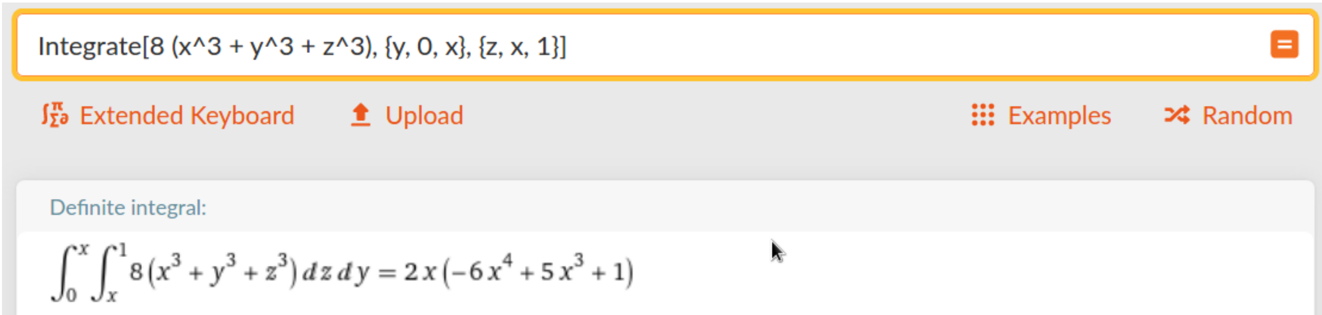
$$\iint 8(x^3+y^3+z^3)\,dx\,dy = 8\int \left[(\tfrac{x^4}{4}+y^3x+z^3x)\Big|_{x=y}^{x=z}\right] dy$$

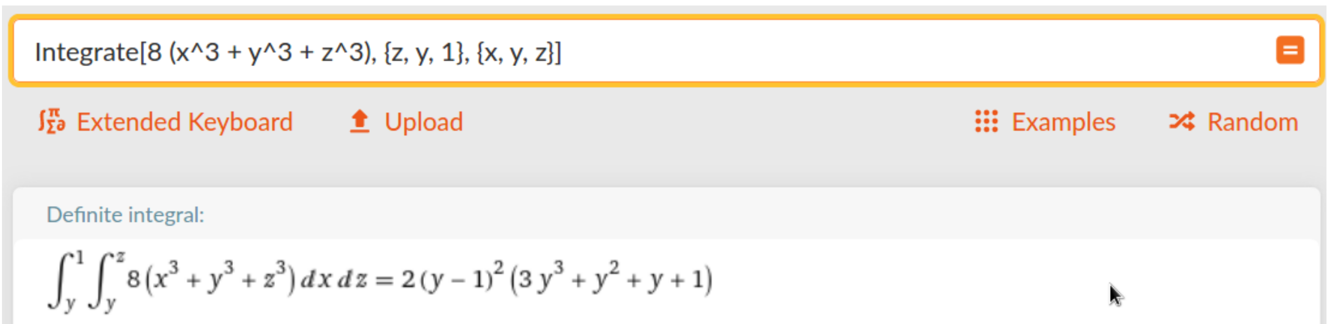
$$= 8\int (\tfrac{5}{4}z^4+y^3z-\tfrac{5}{4}y^4-z^3y)\,dy$$

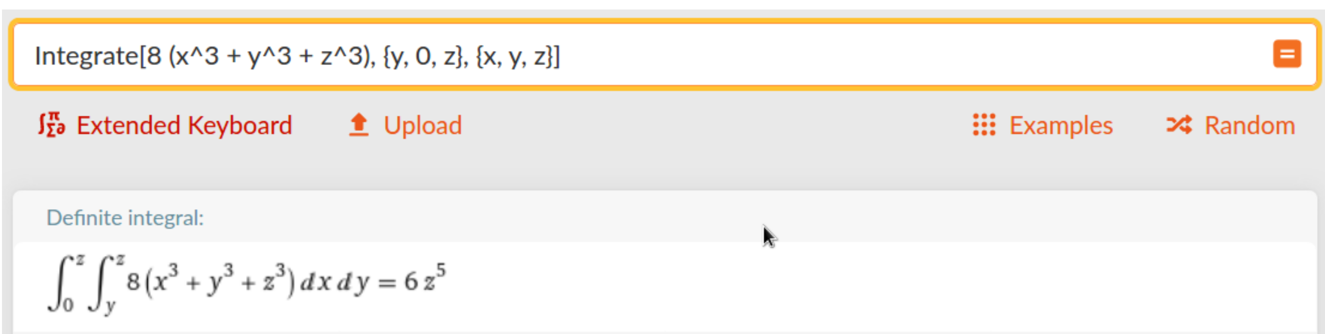
$$= 8(\tfrac{5}{4}z^4y+z\cdot\tfrac{y^4}{4}-\tfrac{y^5}{4}-z^3\cdot\tfrac{y^2}{2})\Big|_{y=0}^{y=z} = 6z^5$$

$$\Rightarrow \text{margin}(z) = \begin{cases} 6z^5, & 0 \le z \le 1 \\ 0 & \text{. o.w.} \end{cases}$$

However, the computation is tedious and error-prone, it'd be better to program the solution out:







As we can see, the automatic function gives the same results as we computed before.

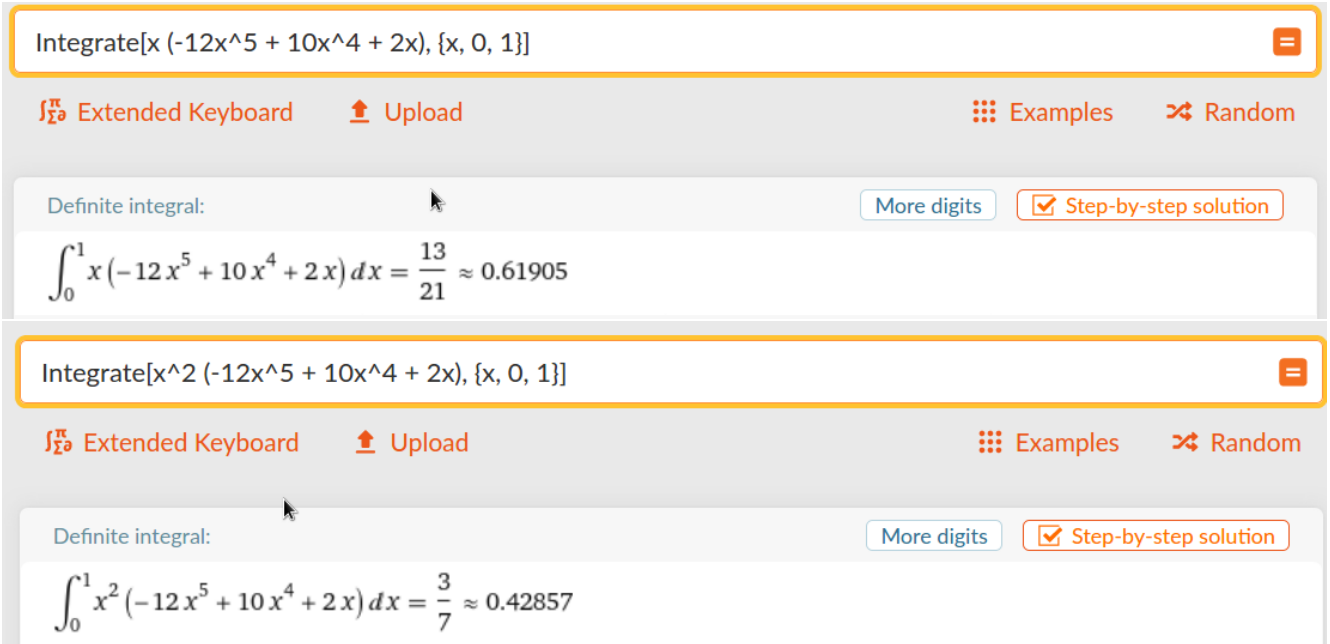
3. Since $margin(x) * margin(y) * margin(z)$ is a very high-order polynomial expression, which apparently does not equal to $f(x, y, z)$, so the three variables are not independent.

4. We can compute the expectation and variance using the following equations:

$$E(X) = \int_0^1 xmargin(x)dx \tag{1}$$

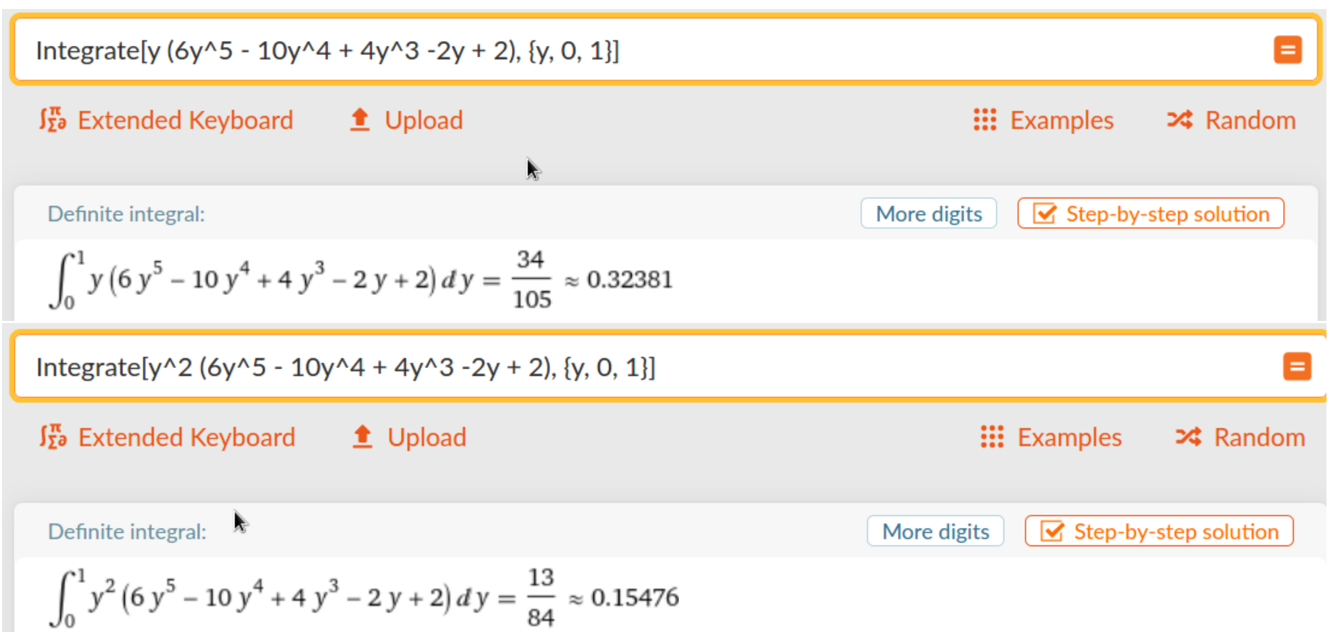
$$E(X^2) = \int_0^1 x^2 margin(x)dx \tag{2}$$

$$Var(X) = E(X^2) - E(X)^2 \tag{3}$$



Hence, $E(X) = 0.61905$, $Var(X) = 0.42857 - 0.61905^2 = 0.045347097$.

5. ...



Hence, $E(Y) = 0.32381$, $Var(Y) = 0.15476 - 0.32381^2 = 0.049907084$.

6. ...

Integrate[z (6z^5), {z, 0, 1}]

Extended Keyboard Upload Examples Random

Definite integral: More digits Step-by-step solution

$$\int_0^1 z\,(6\,z^5)\,dz = \frac{6}{7} \approx 0.85714$$

Integrate[z^2 (6z^5), {z, 0, 1}]

Extended Keyboard Upload Examples Random

Definite integral: Step-by-step solution

$$\int_0^1 z^2\,(6\,z^5)\,dz = \frac{3}{4} = 0.75$$

Hence, $E(Z) = 0.85714$, $Var(Z) = 0.75 - 0.85714^2 = 0.01531102$.

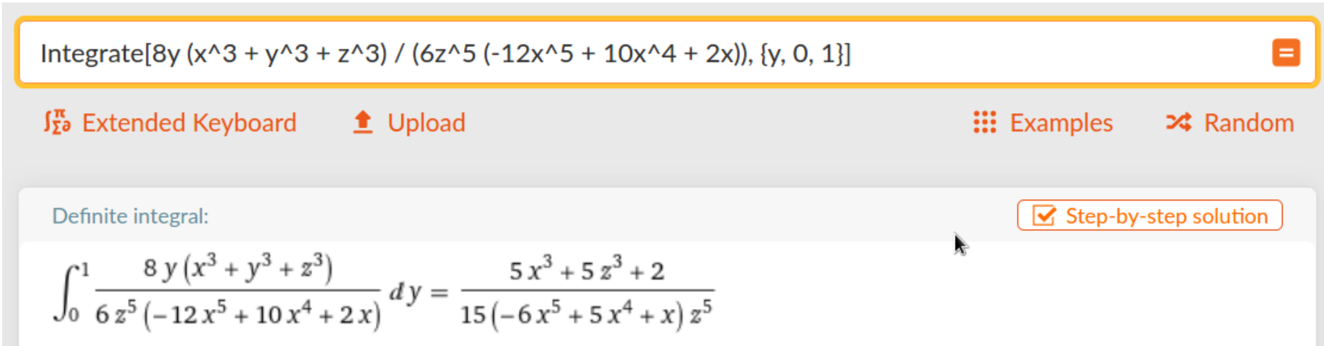
7. Dividing $f(x, y, z)$ by the marginal pdf of X and Z yields the conditional density of Y. We can then derive the expectation by definition.

$$f_{Y|X,Z}(y|x,z) = \frac{f(x,y,z)}{\text{margin}(x)\cdot\text{margin}(z)}$$

$$= \frac{8(x^3+y^3+z^3)}{(-12x^5+10x^4+2x)\cdot(6z^5)}$$

$$E(Y|X=x, Z=z) = \int_0^1 y\cdot f_{Y|X,Z}(y|x,z)\,dy$$

$$= \int_0^1 \frac{8(x^3+y^3+z^3)\cdot y}{(-12x^5+10x^4+2x)\cdot 6z^5}\,dy$$



3 Bayesian decision theory

1. ω_1 has 6 data points, ω_2 has 8 data points, so the prior probability of each class is:
   $P(\omega_1) = 6/14 = 3/7 = 0.428571429, P(\omega_1) = 8/14 = 4/7 = 0.571428571$

2. By definition, we can calculate that:

```python
import numpy as np

w1 = np.array([[0, 0], [1, 1], [2, 2], [3, 2], [4, 2], [5, 3]])
w2 = np.array([[6, 9], [8, 10], [9, 10], [9, 11], [10, 10], [11, 12], [12, 12], [12,
    13]])

def mean(w):
    return np.array([np.sum(w[:,0]) / w.shape[0], np.sum(w[:,1]) / w.shape[0]])

def sigma(w):
    dof = w.shape[0] - 1  # degree of freedom
    deviation = (w - mean(w))
    return 1 / dof * np.dot(deviation.T, deviation)

if __name__ == '__main__':
    for idx, w in enumerate([w1, w2]):
        with np.printoptions(precision=3, suppress=True):
            print(f'mu_{idx + 1} = {mean(w)}')
            print(f'sigma_{idx + 1} = {sigma(w)}')

    '''results'''
    # mu_1    = [2.5, 1.667]
    # sigma_1 = [3.5   1.8]
    #           [1.8   1.067]
    #
    # mu_2    = [9.625, 10.875]
    # sigma_2 = [4.268  2.518]
    #           [2.518  1.839]
```

test.py

However, if the data size is large, the functions are very computationally expensive, we should use vectorized functions to boost performance.

```python
        mu = np.mean(w, axis=0)
        sigma = np.cov(w.T)
```

3. Since the covariance matrices for the two classes are neither identical nor $\sigma^2 I$, this problem falls into the case where sigma is arbitrary. In particular, the prior probabilities are not uniform, so our decision is based on both priors and likelihood. As we did in the exercise, the discriminant function can be formulated as:

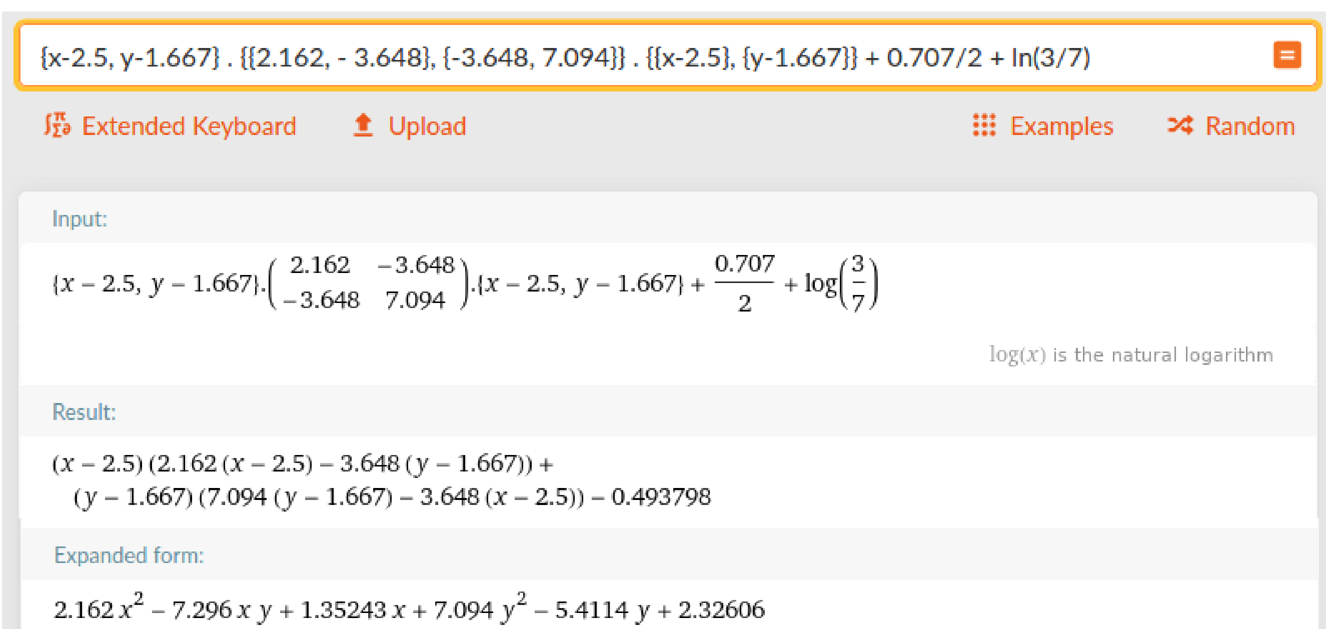
$$g_i(x) = \ln p(x|\omega_i) + \ln P(\omega_i)$$
$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

In Python, we then computate the inverse and determinant of the covariance matrix:

```
import numpy as np

w1 = np.array([[0, 0], [1, 1], [2, 2], [3, 2], [4, 2], [5, 3]])
w2 = np.array([[6, 9], [8, 10], [9, 10], [9, 11], [10, 10], [11, 12], [12, 12], [12,
    13]])

if __name__ == '__main__':
    for w in [w1, w2]:
        sigma = np.cov(w.T)
        sigma_inv = np.linalg.inv(sigma)
        print('inverse matrix = ', np.linalg.inv(sigma))
        print('determinant = ', np.log(np.linalg.det(sigma)))

        '''result'''
        # inverse matrix = [2.16216216 - 3.64864865]
        #                  [-3.64864865  7.09459459]
        # determinant = -0.7065702008920836
        # inverse matrix = [1.21790541 - 1.66722973]
        #                  [-1.66722973  2.82601351]
        # determinant = 0.4122447950935426
```

test1.py

Therefore, we have:


{x-2.5, y-1.667} . {{2.162, - 3.648}, {-3.648, 7.094}} . {{x-2.5}, {y-1.667}} + 0.707/2 + ln(3/7)

Extended Keyboard Upload Examples Random

Input:

$\{x-2.5,\ y-1.667\}.\begin{pmatrix} 2.162 & -3.648 \\ -3.648 & 7.094 \end{pmatrix}.\{x-2.5,\ y-1.667\} + \frac{0.707}{2} + \log\left(\frac{3}{7}\right)$

$\log(x)$ is the natural logarithm

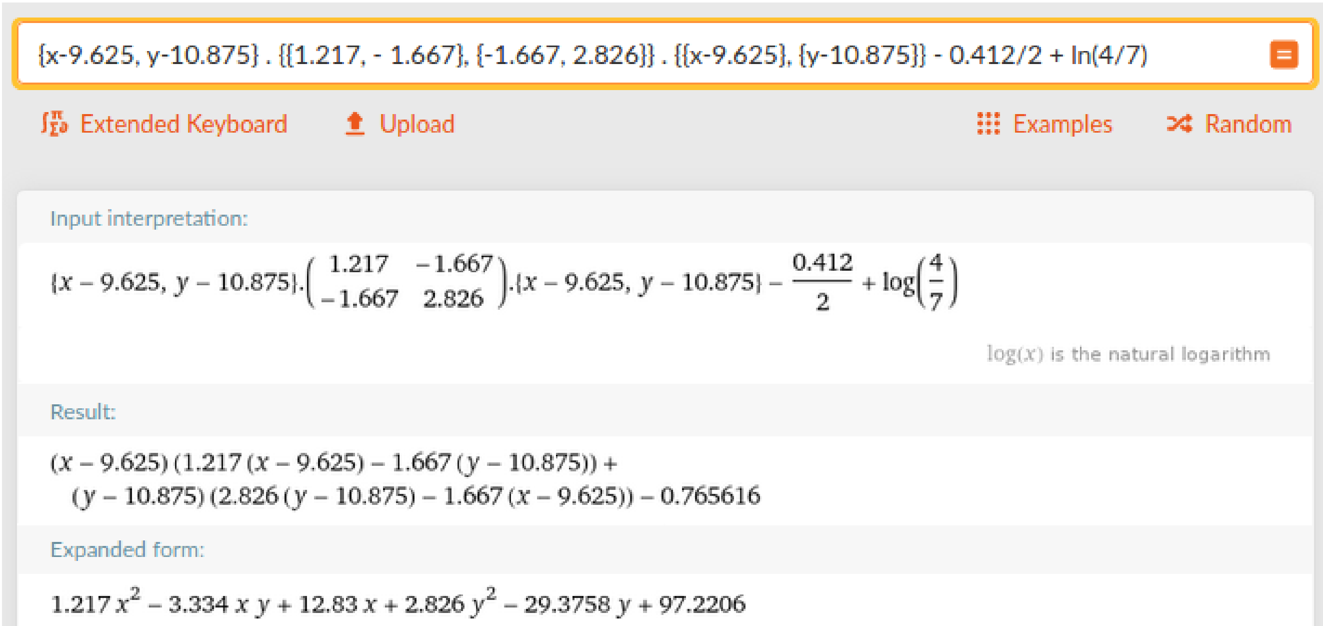
Result:

$(x-2.5)(2.162(x-2.5) - 3.648(y-1.667)) + (y-1.667)(7.094(y-1.667) - 3.648(x-2.5)) - 0.493798$

Expanded form:

$2.162x^2 - 7.296xy + 1.35243x + 7.094y^2 - 5.4114y + 2.32606$

$g_1(x)$



$g_2(x)$

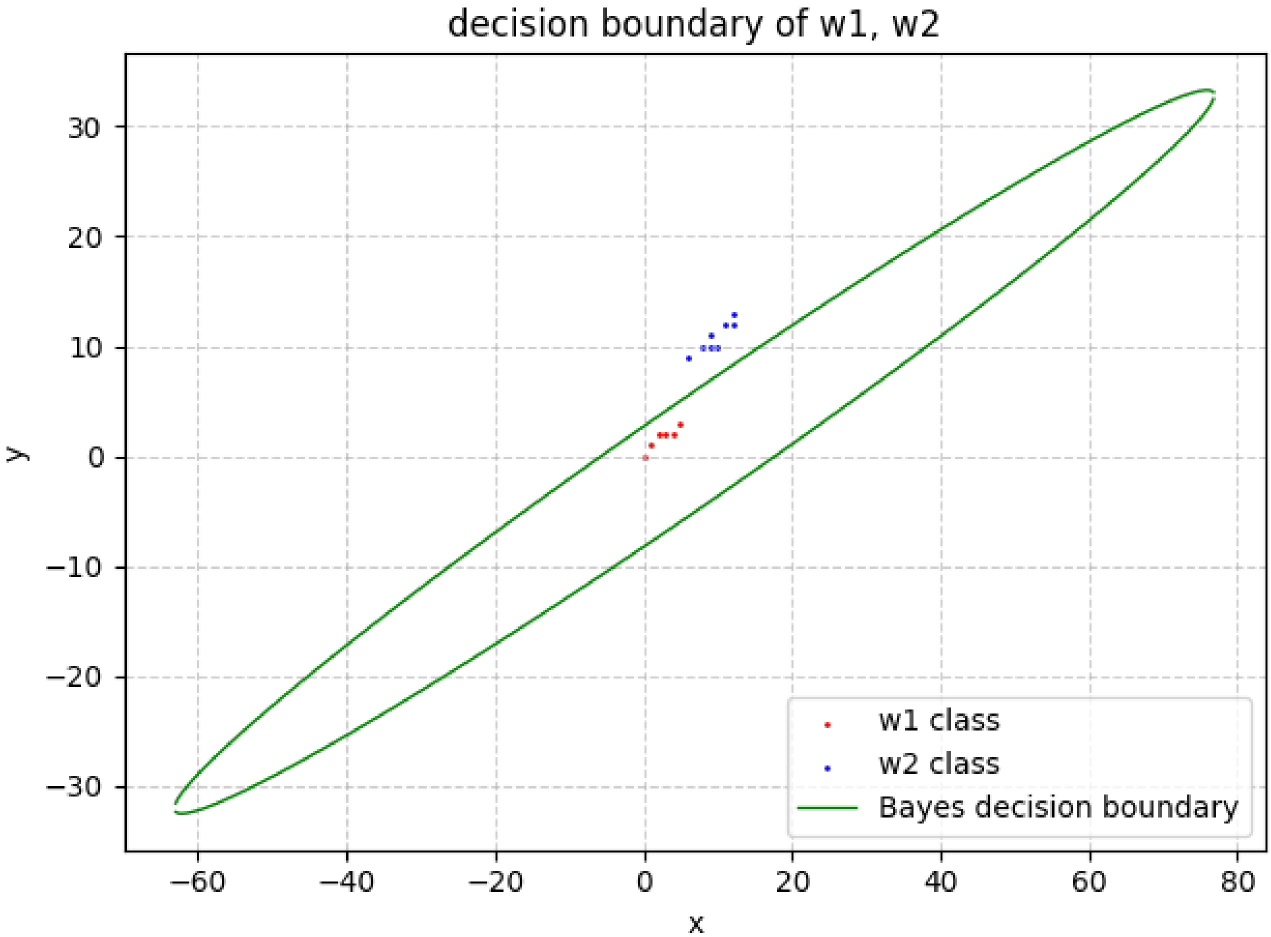
The decision boundary is derived by setting: $g_1(x) = g_2(x)$, which is equivalent to the ellipse function computed by Mathematica:

$$0.945x^2 - 3.96xy - 11.478x + 4.268y^2 + 23.96y - 94.89 = 0 \Rightarrow$$
$$y = \frac{990x - 5990 \pm \sqrt{-28215x^2 + 386826x + 137127730}}{2134}$$

Finally, we visualize the scatter plot of the two classes and draw the decision boundary in 2D.

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

w1 = np.array([[0, 0], [1, 1], [2, 2], [3, 2], [4, 2], [5, 3]])
w2 = np.array([[6, 9], [8, 10], [9, 10], [9, 11], [10, 10], [11, 12], [12, 12], [12,
    13]])

if __name__ == '__main__':
    scatter1 = plt.scatter(w1[:, 0], w1[:, 1], c='red', s=1)
    scatter2 = plt.scatter(w2[:, 0], w2[:, 1], c='blue', s=1)

    x = np.linspace(-80, 80, 500)
    y1 = (990 * x - 5990 + np.sqrt(-28215 * (x ** 2) + 386826 * x + 137127730)) / 2134
    y2 = (990 * x - 5990 - np.sqrt(-28215 * (x ** 2) + 386826 * x + 137127730)) / 2134
    plt.plot(x, y1, color='green', lw=1)
    boundary, = plt.plot(x, y2, color='green', lw=1)

    plt.title('decision boundary of w1, w2')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(alpha=.6, linestyle='--')
    plt.legend([scatter1, scatter2, boundary],
               ['w1 class', 'w2 class', 'Bayes decision boundary'],
               loc='lower right')
    plt.show()
```

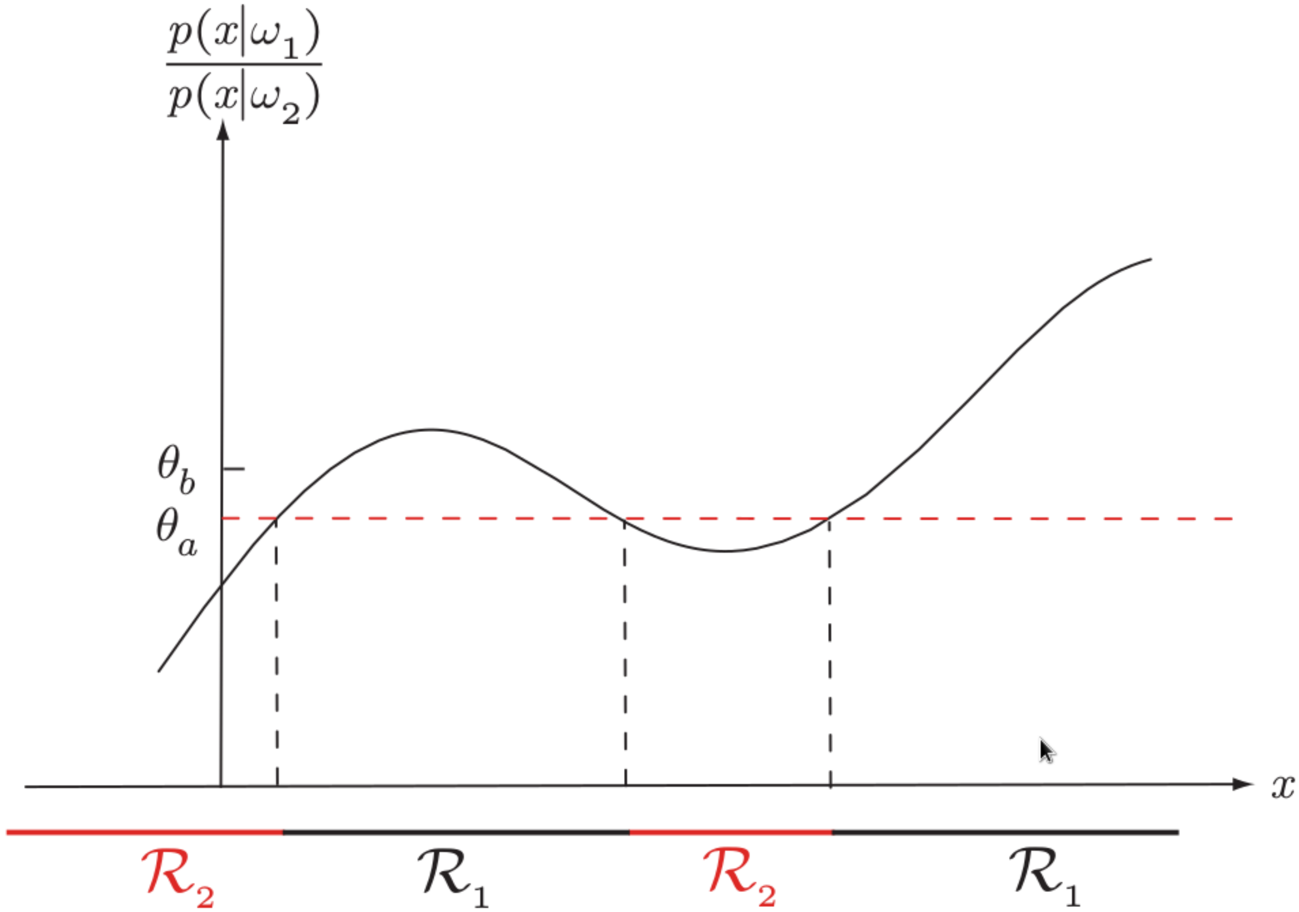
test2.py



4. If the penalties for misclassification are different for the two classes, it will definitely affect the decision boundary. In the case of 2-category classification, the Bayes decision rule is based on the likelihood ratio, so we would decide ω_1 if

$$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(\omega_2)}{P(\omega_2)}$$

The left hand side is the likelihood ratio, the right hand side is the threshold value, which can vary drastically depending on how we define the loss function $\lambda()$. For instance, if our loss function penalizes miscategorizing ω_2 as ω_1 more than the converse ($\lambda_{12} > \lambda_{21}$), we end up with a larger threshold, and hence the decision regions change as well, as shown in the figure below.


$\frac{p(x|\omega_1)}{p(x|\omega_2)}$
θ_b
θ_a
x
$\mathcal{R}_2$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$