UNIVERSIDADE CATÓLICA DE PELOTAS
Mestrado em Engenharia Eletrônica e Computação - PGEEC

LISTA DE EXERCÍCIOS 1

PEDRO A. TAVARES

DISCIPLINA: Reconhecimento de Padrões
PROFESSOR: Jose C. M. Bermudez

Pelotas, outubro de 2019

EXERCÍCIO 1

CÁLCULO MATEMÁTICO:

(1) PDF GAUSSIANA $N(\mu, \Sigma)$ p/ $x_1 = [0.2, 1.3]^T$ e $x_2 = [2.2, -1.3]^T$

SENDO: $\mu = [0,1]^T$, $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

PDF normal l-dimensional $\rightarrow p(x) = \frac{1}{(2\pi)^{l/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$

$$p(x_1) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}\left(\begin{bmatrix}0.2\\1.3\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right)^T \cdot \begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}0.2\\1.3\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right]$$

$$p(x_1) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}[0.2, 0.3]\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}0.2\\0.3\end{bmatrix}\right]$$

$$p(x_1) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}[0.2\ 0.3]\begin{bmatrix}0.2\\0.3\end{bmatrix}\right]$$

$$p(x_1) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}[0,04\ 0,09]\right]$$

$$\boxed{p(x_1) = 0,1491}$$

$$p(x_2) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}\left(\begin{bmatrix}2.2\\-1.3\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right)^T \cdot \begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}2.2\\-1.3\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right]$$

$$p(x_2) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}[2.2, -2.3]\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}2.2\\-2.3\end{bmatrix}\right]$$

$$p(x_2) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}[2.2, -2.3]\begin{bmatrix}2.2\\-2.3\end{bmatrix}\right]$$

$$p(x_2) = \frac{1}{2\pi} \exp\left[-\frac{1}{2}[4.84\ 5.29]\right]$$

$$p(x_2) = \frac{1}{2\pi} \exp[-5,065]$$

$$\boxed{p(x_2) = 0,001}$$

CÓDIGO EM PYTHON:

```python
# -*- coding: utf-8 -*-
"""
Exercício 1
"""
import numpy as np
from scipy.stats import multivariate_normal

x1 = np.array([0.2, 1.3])
x2 = np.array([2.2, -1.3])

print("PDF(x1) =", multivariate_normal(mean=[0, 1], cov=[[1, 0],[0, 1]]).pdf(x1))
print("PDF(x2) =", multivariate_normal(mean=[0, 1], cov=[[1, 0],[0, 1]]).pdf(x2))
```

SAÍDA NO CONSOLE DO SPYDER:

```
In [26]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Pedro/Exercicio1.py', wdir='C:/Users/
Pedro/Desktop/Reconhecimento de Padrões/Pedro')
PDF(x1) = 0.1491389188070974
PDF(x2) = 0.0010048901304231335
```

EXERCÍCIO 2

CÁLCULO MATEMÁTICO:

(2) Duas classes (w_1 e w_2) em espaço bidimensional:

$\mu_1 = [1,1]^T$, $\mu_2 = [3,3]^T$, $\Sigma_1 = \Sigma_2 = \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$P(w_1) = P(w_2) = 1/2$, classificar o vetor $x = [1.8, 1.8]^T$

$$w_i = \frac{\mu_i}{\sigma^2} \Rightarrow w_1 = \frac{\mu_1}{\sigma^2} = \frac{[1,1]^T}{1} \therefore w_1 = [1,1]^T$$

$$w_2 = \frac{\mu_2}{\sigma^2} = \frac{[3,3]^T}{1} \therefore w_2 = [3,3]^T$$

$$w_{i0} = \ln P(w_i) - \frac{1}{2\sigma^2} \cdot \mu_i^T \cdot \mu_i$$

$$w_{10} = \ln P(w_1) - \frac{1}{2} \cdot \mu_1^T \cdot \mu_1 = \ln\left(\frac{1}{2}\right) - \frac{1}{2}\begin{bmatrix}1\\1\end{bmatrix}^T \begin{bmatrix}1\\1\end{bmatrix}$$

$$w_{10} = -0{,}693147 - \frac{2}{2} \therefore w_{10} = -1{,}6931$$

$$w_{20} = \ln P(w_2) - \frac{1}{2} \cdot \mu_2^T \cdot \mu_2 = \ln\left(\frac{1}{2}\right) - \frac{1}{2}\begin{bmatrix}3\\3\end{bmatrix}^T \begin{bmatrix}3\\3\end{bmatrix}$$

$$w_{20} = -0{,}693147 - 9 \therefore w_{20} = -9{,}6931$$

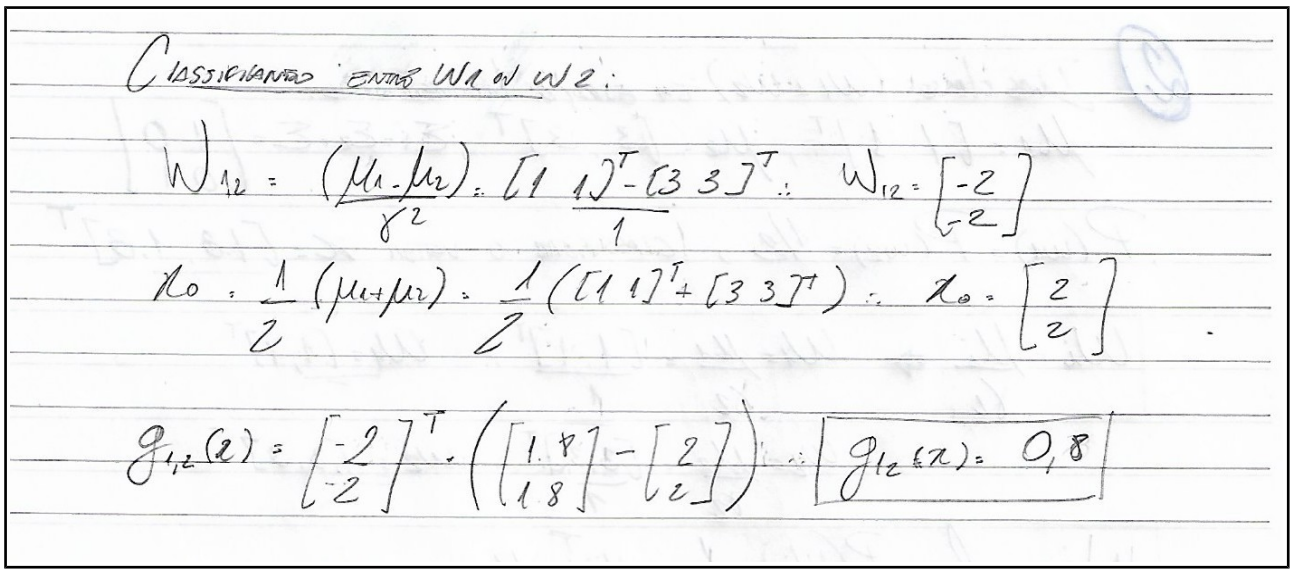
$$g_i(x) = (w_i)^T x + (w_{i0})$$

$$g_1(x) = (w_1)^T x + (w_{10}) = [1\ 1] \cdot \begin{bmatrix}1.8\\1.8\end{bmatrix} + (-1{,}6931)$$

$$g_1(x) = [1.8\ 1.8] - 1{,}6931 = 3{,}6 - 1{,}6931 \therefore g_1(x) = 1{,}9069$$

$$g_2(x) = (w_2)^T x + (w_{10}) = [3\ 3] \begin{bmatrix}1.8\\1.8\end{bmatrix} + (-9{,}6931)$$

$$g_2(x) = [5.4\ 5.4] - 9{,}6931 = 10{,}8 - 9{,}6931 \therefore g_2(x) = 1{,}1069$$

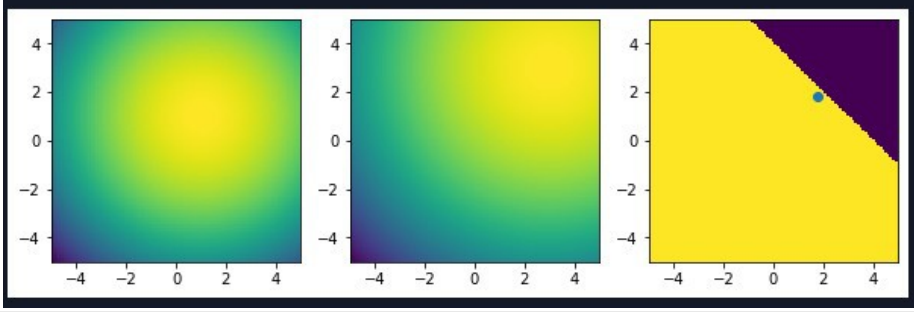


CÓDIGO EM PYTHON:

```
# -*- coding: utf-8 -*-
"""
Exerício 2
"""
import pylab as pl
import numpy as np
import math

m1 = np.array([1.0, 1.0])
m2 = np.array([3.0, 3.0])

cov1 = np.array([[1.0, 0.0], [0.0, 1.0]])

pw1 = (1/2);
pw2 = (1/2);

X, Y = np.mgrid[-5:5:100j, -5:5:100j]
xy = np.c_[X.ravel(), Y.ravel()]

def discriminante(cov, xy, m, pw):
    invC = np.linalg.inv(cov)
    v = xy - m
    g = -0.5*np.sum(np.dot(v, invC) * v, axis=1) - 2*0.5*np.log(2*np.pi) -
0.5*np.log(np.linalg.det(cov)) + math.log(pw);
    g.shape = 100, 100
    return g

g1 = discriminante(cov1, xy, m1, pw1)
g2 = discriminante(cov1, xy, m2, pw2)

fig, axes = pl.subplots(1, 3, figsize=(10, 10))
ax1, ax2, ax3 = axes.ravel()
for ax in axes.ravel():
    ax.set_aspect("equal")

ax1.pcolormesh(X, Y, g1)
ax2.pcolormesh(X, Y, g2)

ax3.pcolormesh(X, Y, g1 > g2)
ax3.scatter([1.8], [1.8])
```

SAÍDA NO CONSOLE DO SPYDER:

```
In [28]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Pedro/Exercicio2.py', wdir='C:/Users/
Pedro/Desktop/Reconhecimento de Padrões/Pedro')
```



EXERCÍCIO 3

CÁLCULO MATEMÁTICO:

(3) a) $P(w_1) = 1/6$ & $P(w_2) = 5/6$

$P(w_1) = 0,1667$ $P(w_2) = 0,8333$

$$w_{10} = \ln P(w_1) - \frac{1}{2\delta^2} \mu_1^T \cdot \mu_1$$

$$w_{10} = \ln(0,1667) - \frac{1}{2}[1\ 1]\begin{bmatrix}1\\1\end{bmatrix}$$

$w_{10} = -1,7915 - 1 \therefore \boxed{w_{10} = -2,7915}$

$$w_{20} = \ln P(w_2) - \frac{1}{2\delta^2} \mu_2^T \mu_2$$

$$w_{20} = \ln(0,8333) - \frac{1}{2}[3\ 3]\begin{bmatrix}3\\3\end{bmatrix}$$

$w_{20} = -0,1823 - 9 \therefore \boxed{w_{20} = -9,1823}$

$g_i(x) = (w_i)^T x + (w_{i0})$

$g_1(x) = (w_1)^T x + (w_{10})$

$$g_1(x) = \begin{bmatrix}1\\1\end{bmatrix}^T \cdot \begin{bmatrix}1,8\\1,8\end{bmatrix} + (-2,7915)$$

$g_1(x) = 3,6 - 2,7915 \therefore \boxed{g_1(x) = 0,8085}$

$g_2(x) = (w_2)^T x + (w_{20})$

$$g_2(x) = \begin{bmatrix}3\\3\end{bmatrix}^T \cdot \begin{bmatrix}1,8\\1,8\end{bmatrix} + (-9,1823)$$

$g_2(x) = 10,8 - 9,1823 \therefore \boxed{g_2(x) = 1,6177}$

* CLASSIFICANDO ENTRE w_1 e w_2:

$$w_{12} = \frac{(\mu_1 - \mu_2)}{\delta^2} = [1\ 1]^T - [3\ 3]^T \therefore w_{12} = \begin{bmatrix}-2\\-2\end{bmatrix}$$

$$x_0 = \frac{1}{2}(\mu_1+\mu_2) = \frac{1}{2}\left(\begin{bmatrix}1\\1\end{bmatrix}-\begin{bmatrix}3\\3\end{bmatrix}\right) \therefore x_0 = \begin{bmatrix}-1\\-1\end{bmatrix}$$

$$g_{1,2}(x) = \begin{bmatrix}-2\\-2\end{bmatrix}^T \cdot \left(\begin{bmatrix}1.8\\1.8\end{bmatrix}-\begin{bmatrix}-1\\-1\end{bmatrix}\right)$$

$$g_{1,2}(x) = [-2\ -2] \cdot \begin{bmatrix}2.8\\2.8\end{bmatrix} \therefore \boxed{g_{1,2}(x) = -11,2}$$

b) $P(w_1) = 5/6$ & $P(w_2) = 1/6$

$P(w_1) = 0,8333$ $\quad$ $P(w_2) = 0,1667$

$$W_{10} = \ln(0,8333) - \frac{1}{2}[1\ 1]\begin{bmatrix}1\\1\end{bmatrix}$$

$$W_{10} = -0,1823 - 1 \therefore \boxed{W_{10} = -1,1823}$$

$$W_{20} = \ln(0,1667) - \frac{1}{2}[3\ 3]\begin{bmatrix}3\\3\end{bmatrix}$$

$$W_{20} = -1,7915 - 9 \therefore \boxed{W_{20} = -10,7915}$$

$W_{12} = [-2\ -2]^T$ PREVIAMENTE CALCULADO

$$x_0 = \frac{1}{2}\left(\begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}3\\3\end{bmatrix}\right) - 1 \cdot \ln\left(\frac{5/6}{1/6}\right) \cdot \left(\frac{\begin{bmatrix}1\\1\end{bmatrix}-\begin{bmatrix}3\\3\end{bmatrix}}{\left(\begin{bmatrix}1\\1\end{bmatrix}-\begin{bmatrix}3\\3\end{bmatrix}\right)^2}\right)$$

$$x_0 = \begin{bmatrix}2\\2\end{bmatrix} - \begin{bmatrix}-0.4\\-0.4\end{bmatrix} \therefore x_0 = \begin{bmatrix}2.4\\2.4\end{bmatrix}$$

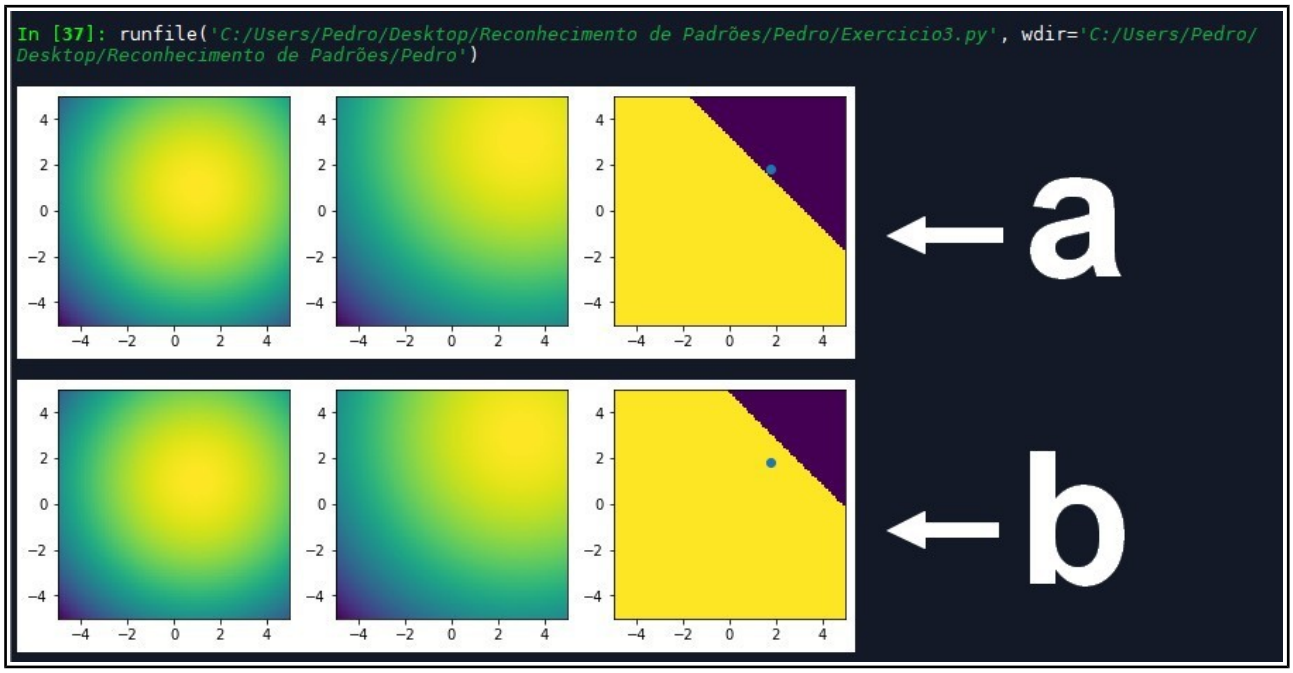
$$g_{1,2}(x) = \begin{bmatrix}-2\\-2\end{bmatrix}^T \cdot \left(\begin{bmatrix}1.8\\1.8\end{bmatrix}-\begin{bmatrix}2.4\\2.4\end{bmatrix}\right) = [-2\ -2] \cdot \begin{bmatrix}-0.6\\-0.6\end{bmatrix}$$

$$g_{1,2}(x) = \begin{bmatrix}1.2\\1.2\end{bmatrix} \therefore \boxed{g_{1,2}(x) = 1.2}$$

CÓDIGO EM PYTHON:

```
# -*- coding: utf-8 -*-
"""
Exercício 3
"""
import pylab as pl
import numpy as np
import math

m1 = np.array([1.0, 1.0])
m2 = np.array([3.0, 3.0])

cov1 = np.array([[1.0, 0.0], [0.0, 1.0]])

X, Y = np.mgrid[-5:5:100j, -5:5:100j]
xy = np.c_[X.ravel(), Y.ravel()]

def discriminante(cov, xy, m, pw):
    invC = np.linalg.inv(cov)
    v = xy - m
    g = -0.5*np.sum(np.dot(v, invC) * v, axis=1) - 2*0.5*np.log(2*np.pi) -
0.5*np.log(np.linalg.det(cov)) + math.log(pw);
    g.shape = 100, 100
    return g

#Item 3a)
pw1 = (1/6);
pw2 = (5/6);

g1 = discriminante(cov1, xy, m1, pw1)
g2 = discriminante(cov1, xy, m2, pw2)

fig, axes = pl.subplots(1, 3, figsize=(10, 10))
ax1, ax2, ax3 = axes.ravel()
for ax in axes.ravel():
    ax.set_aspect("equal")

ax1.pcolormesh(X, Y, g1)
ax2.pcolormesh(X, Y, g2)

ax3.pcolormesh(X, Y, g1 > g2)
ax3.scatter([1.8], [1.8])

#Item 3b)
pw1 = (5/6);
pw2 = (1/6);

g1 = discriminante(cov1, xy, m1, pw1)
g2 = discriminante(cov1, xy, m2, pw2)

fig, axes = pl.subplots(1, 3, figsize=(10, 10))
ax1, ax2, ax3 = axes.ravel()
for ax in axes.ravel():
    ax.set_aspect("equal")

ax1.pcolormesh(X, Y, g1)
ax2.pcolormesh(X, Y, g2)

ax3.pcolormesh(X, Y, g1 > g2)
ax3.scatter([1.8], [1.8])
```

SAÍDA NO CONSOLE DO SPYDER:



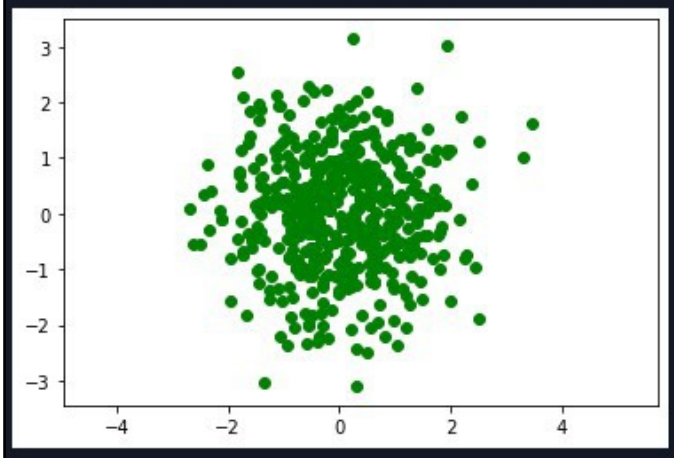
EXERCÍCIO 4

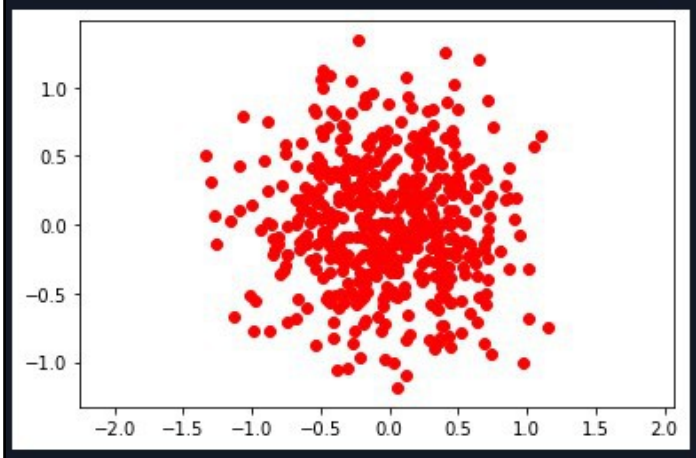
CÓDIGO EM PYTHON:

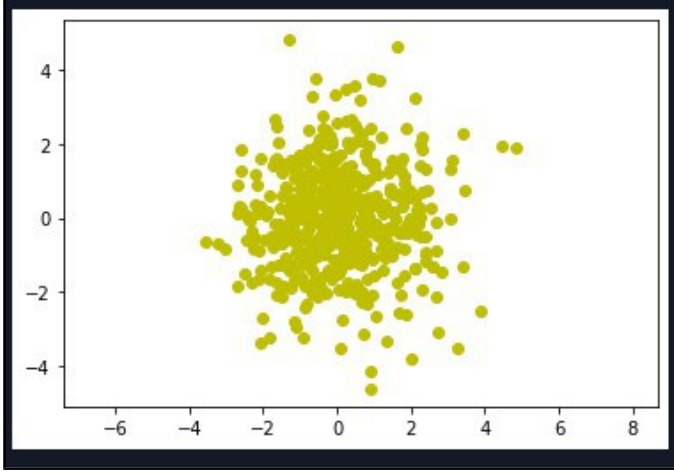
```
# -*- coding: utf-8 -*-
"""
Exercício 4
"""
import numpy as np
import matplotlib.pyplot as plt

N = 500

mi = [0, 0]
cova = [[1, 0], [0, 1]]
covb = [[0.2, 0], [0, 0.2]]
covc = [[2, 0], [0, 2]]
covd = [[0.2, 0], [0, 2]]
cove = [[1, 0.5], [0.5, 1]]
covf = [[0.3, 0.5], [0.5, 2]]
covg = [[0.3, -0.5], [-0.5, 2]]

def gaussiana(mi, cov, samples, color):
    x1, x2 = np.random.multivariate_normal(mi, cov, N).T
    plt.scatter(x1, x2, c=color)
    plt.axis('equal')
    return plt.show()

ga = gaussiana (mi, cova, N, 'g');
gb = gaussiana (mi, covb, N, 'r');
gc = gaussiana (mi, covc, N, 'y');
gd = gaussiana (mi, covd, N, 'k');
ge = gaussiana (mi, cove, N, 'b');
gf = gaussiana (mi, covf, N, 'm');
gg = gaussiana (mi, covg, N, 'c');
```

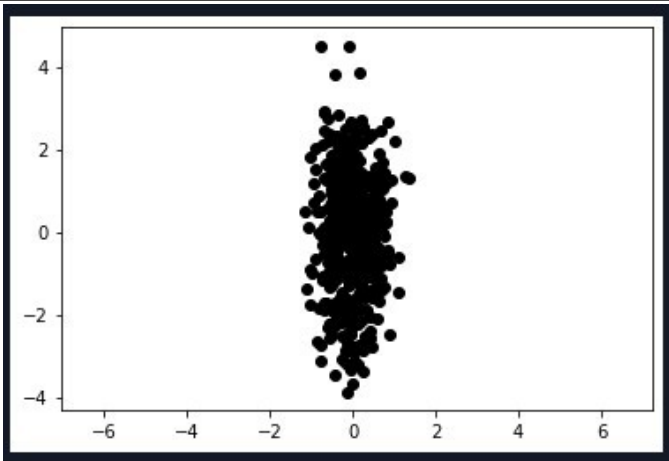
SAÍDA NO CONSOLE DO SPYDER:

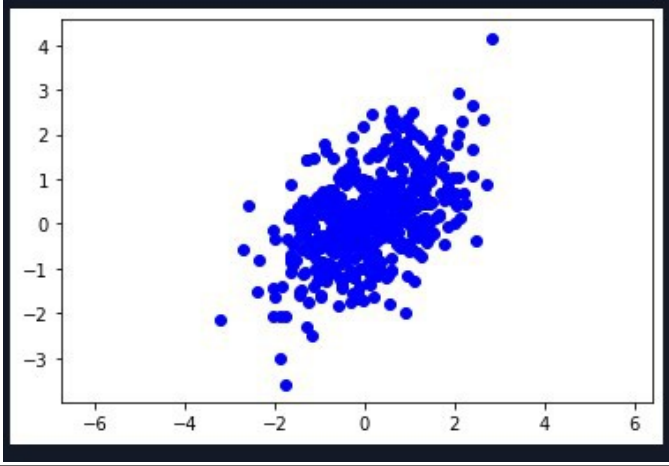
```
In [30]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Pedro/Exercicio4.py',
wdir='C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Pedro')
```

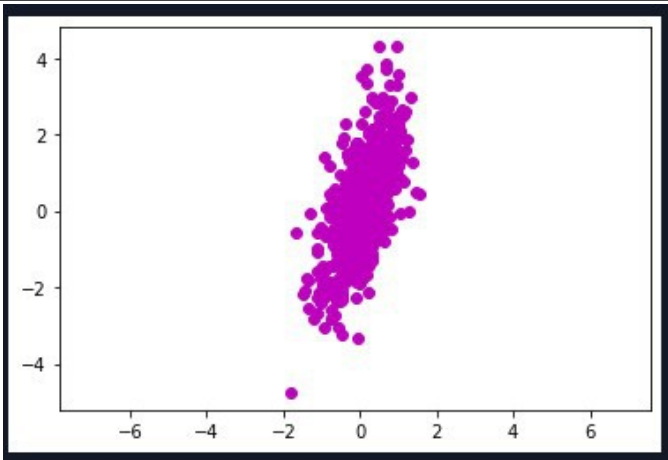


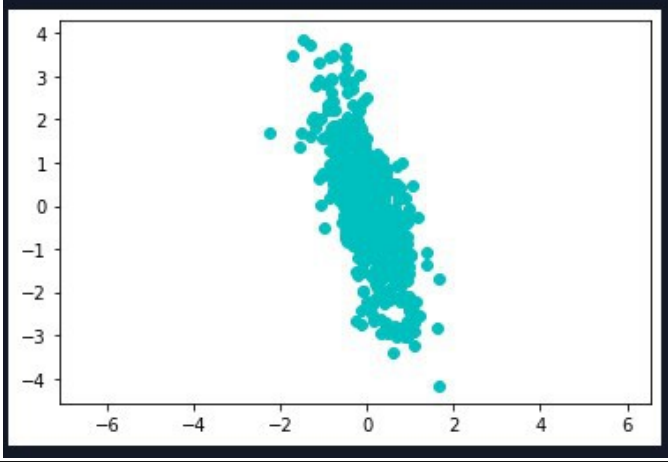





4
2
0
−2
−4
−6
−4
−2
0
2
4
6


4
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4
6


4
2
0
−2
−4
−6
−4
−2
0
2
4
6


4
3
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4
6

RESPOSTAS:

I) A forma do agrupamento irá rotacionar (formato de círculo).

II) Devido à diferença das variâncias, o agrupamento irá condensar (formato de elipse).

III) Afetará o sentido de rotação do agrupamento:

- Positivo (+): agrupamento rotaciona no sentido horário
- Negativo (-): agrupamento rotaciona no sentido anti-horário

EXERCÍCIO 5

CÁLCULO MATEMÁTICO:

(5) Duas classes ω_1 e ω_2

$m_1 = [0,0,0]^T$ e $m_2 = [0.5, 0.5, 0.5]^T$

$$\Sigma = \begin{bmatrix} 0.8 & 0.01 & 0.01 \\ 0.01 & 0.2 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$

Classificar o ponto: $x = [0.1, 0.5, 0.1]^T$

a) Classificador de mínima distância euclidiana

$$d^2_m(\mu_1, x) = (x-\mu_1)^T \Sigma^{-1} (x-\mu_1)$$

$$d^2_m(\mu_1, x) = \left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right)^T \left(\begin{bmatrix}0.8 & 0.01 & 0.01\\0.01 & 0.2 & 0.01\\0.01 & 0.01 & 0.2\end{bmatrix}\right)^{-1} \left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right)$$

$$\boxed{d^2_m(\mu_1, x) = 1,2846}$$

$$d^2_m(\mu_2, x) = (x-\mu_2)^T \Sigma^{-1} (x-\mu_2)$$

$$d^2_m(\mu_2, x) = \left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix} - \begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right)^T \left(\begin{bmatrix}0.8 & 0.01 & 0.01\\0.01 & 0.2 & 0.01\\0.01 & 0.01 & 0.2\end{bmatrix}\right)^{-1} \left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix} - \begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right)$$

$$\boxed{d^2_m(\mu_2, x) = 0,9836}$$

$$\boxed{d^2_m(\mu_2 2) < d^2_m(\mu_1 2)}$$

* Ponto $x\,[0.1\ 0.5\ 0.1]^T$ é classificado como ω_2 pois a distância entre ele e o ponto m_2 é menor do que a distância euclidiana até o ponto m_1.

b) Classificador de mínima distância de Mahalanobis

$$de: \|x-\mu_i\| = \sqrt{(x-\mu_1)^2 + (x-\mu_2)^2}$$

$$de = \sqrt{\left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right)^2 + \left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix} - \begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right)^2}$$

$$de = \sqrt{\left(\begin{bmatrix}0.1\\0.5\\0.1\end{bmatrix}\right)^2 + \left(\begin{bmatrix}-0.4\\0\\-0.4\end{bmatrix}\right)^2}$$

$$de = \sqrt{\begin{bmatrix}0.01\\0.25\\0.01\end{bmatrix} + \begin{bmatrix}0.16\\0\\0.16\end{bmatrix}}$$

$$de = \sqrt{\begin{bmatrix}0.17\\0.25\\0.17\end{bmatrix}}$$

$$de = \begin{bmatrix}0.4123\\0.5\\0.4123\end{bmatrix}$$ ⇨ MAIS PRÓXIMO DO PONTO M2 DO QUE DO PONTO M1, PORTANTO É CLASSIFICADO COMO $\underline{\underline{W_2}}$!

CÓDIGO EM PYTHON:

```python
# -*- coding: utf-8 -*-
"""
Exercício 5
"""
import numpy as np
from numpy.linalg import inv
from scipy.spatial.distance import euclidean

m1 = np.matrix([[0],[0],[0]])
m2 = np.matrix([[0.5],[0.5],[0.5]])

cov = np.matrix([[0.8, 0.01, 0.01],[0.01, 0.2, 0.01],[0.01, 0.01, 0.2]])

x = np.matrix([[0.1],[0.5],[0.1]])

def mahalanobis(X, m, sigma):
   return ((np.matrix.transpose(X-m))*(inv(sigma))*(X-m));

#MAHALANOBIS
mahal1 =  mahalanobis(x, m1, cov);
mahal2 =  mahalanobis(x, m2, cov);
print("MAHALANOBIS")
print("-----------")
print("Distancia de Mahalanobis 1 =", mahal1)
print("Distancia de Mahalanobis 2 =", mahal2)

if mahal1 > mahal2:
    print("Classificado em w2")
else:
    print("Classificado em w1")

print("\n")

#EUCLIDIANA
eucli1 = euclidean (x, m1)
eucli2 = euclidean (x, m2)
print("EUCLIDIANA")
print("-----------")
print("Distancia Euclidiana 1 =", eucli1)
print("Distancia Euclidiana 2 =", eucli2)

if eucli1 > eucli2:
    print("Classificado em w2")
else:
    print("Classificado em w1")
```

SAÍDA NO CONSOLE DO SPYDER:

```
In [29]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Pedro/Exercicio5.py', wdir='C:/Users/
Pedro/Desktop/Reconhecimento de Padrões/Pedro')
MAHALANOBIS
-----------
Distancia de Mahalanobis 1 = [[1.28458064]]
Distancia de Mahalanobis 2 = [[0.98362713]]
Classificado em w2


EUCLIDIANA
-----------
Distancia Euclidiana 1 = 0.5196152422706631
Distancia Euclidiana 2 = 0.565685424949238
Classificado em w1
```

EXERCÍCIO 6

CÓDIGO EM PYTHON:

```
# -*- coding: utf-8 -*-
"""
Exerício 6
"""
import matplotlib.pyplot as plt
import numpy as np
from scipy.spatial.distance import euclidean

def mahal(X, m, sigma):
   return (((X-m))*((sigma))*(X-m))

count = 0
count1 = 0
count2 = 0
count3 = 0
counte = 0
counte1 = 0
counte2 = 0
counte3 = 0

fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)

loc1 = 0
loc2 = 0.8

amostras = 500
amostras2 = 1000

i = 0

for i in range(0, amostras):
    if int(i) % 2 == 0:
        scale = 0.8;
        scaleg = 0.6325
    else:
        scale = 0.2
        scaleg = 0.3162

    xg1 =  np.random.normal(loc1, scale, amostras)
    xl1 = np.random.laplace(loc1, scale, amostras)
    pdf1 = np.exp(-abs(xl1-loc1)/scaleg)/(2.*scaleg)
    g1 = (1/(scale * np.sqrt(2 * np.pi)) * np.exp(-(xg1 - loc1)**2 / (2 * scale**2)))
    plt.subplot(411)
    plt.plot(xl1, pdf1)
    plt.subplot(412)
    plt.plot(xg1,g1)

    xg2 =  np.random.normal(loc2, scale, amostras)
    xl2 = np.random.laplace(loc2, scale, amostras)
    pdf2 = np.exp(-abs(xl2-loc2)/scaleg)/(2.*scaleg)
    g2 = (1/(scale * np.sqrt(2 * np.pi)) * np.exp(-(xg2 - loc2)**2 / (2 * scale**2)))
    plt.subplot(413)
    plt.plot(xl2, pdf2)
    plt.subplot(414)
    plt.plot(xg2,g2)

    #Mahalanobis
    mahAg1 =  mahal(xg1[i], loc1, scale)
    mahAg2 =  mahal(xg1[i], loc2, scale)
    if mahAg1 > mahAg2:
        print('X %d Gaussiano classificado em w2 por Mahalanobis' % (i))
        count = count + 1
    else:
        print('X %d Gaussiano classificado em w1 por Mahalanobis' % (i))

    mahBl =  mahal(xl1[i], loc1, scaleg)
    mahB2 =  mahal(xl1[i], loc2, scaleg)
    if mahBl > mahB2:
        print('X %d Laplaciano classificado em w2 por Mahalanobis' % (i))
        count1 = count1 + 1
    else:
```

```
        print('X %d Laplaciano classificado em w1 por Mahalanobis' % (i))

    mahCg1 =  mahal(xg2[i], loc1, scale)
    mahCg2 =  mahal(xg2[i], loc2, scale)
    if mahCg1 > mahCg2:
        print('X %d Gaussiano classificado em w2 por Mahalanobis' % (i))
    else:
        print('X %d Gaussiano classificado em w1 por Mahalanobis' % (i))
        count2 = count2 + 1

    mahDl =  mahal(xl2[i], loc1, scaleg)
    mahD2 =  mahal(xl2[i], loc2, scaleg)
    if mahDl > mahD2:
        print('X %d Laplaciano classificado em w2 por Mahalanobis' % (i))
    else:
        print('X %d Laplaciano classificado em w1 por Mahalanobis' % (i))
        count3 = count3 + 1

    #Euclidiana
    euA1 = euclidean (xg1[i], loc1)
    euA2 = euclidean (xg1[i], loc2)
    if euA1 > euA2:
        print('X %d Gaussiano classificado em w2 pela Euclidiana' % (i))
        counte = counte + 1
    else:
        print('X %d Gaussiano classificado em w1 pela Euclidiana' % (i))

    euB1 = euclidean (xl1[i], loc1)
    euB2 = euclidean (xl1[i], loc2)
    if euB1 > euB2:
        print('X %d Laplaciano classificado em w2 pela Euclidiana' % (i))
        counte1 = counte1 + 1
    else:
        print('X %d Laplaciano classificado em w1 pela Euclidiana' % (i))

    euC1 = euclidean (xg2[i], loc1)
    euC2 = euclidean (xg2[i], loc2)
    if euC1 > euC2:
        print('X %d Gaussiano classificado em w2 pela Euclidiana' % (i))
    else:
        print('X %d Gaussiano classificado em w1 pela Euclidiana' % (i))
        counte2 = counte2 + 1

    euD1 = euclidean (xl2[i], loc1)
    euD2 = euclidean (xl2[i], loc2)
    if euD1 > euD2:
        print('X %d Laplaciano classificado em w2 pela Euclidiana' % (i))
    else:
        print('X %d Laplaciano classificado em w1 pela Euclidiana' % (i))
        counte3 = counte3 + 1

def erro(cout, amostras):
    return (100 * (cout/amostras));

A = erro(count, amostras)
B = erro(count1, amostras)
C = erro(count2, amostras)
D = erro(count3, amostras)

Ae = erro(counte, amostras)
Be = erro(counte1, amostras)
Ce = erro(counte2, amostras)
De = erro(counte3, amostras)

E =  (count + count2)/amostras2;#gaussiana
F = (count1 + count3)/amostras2;#laplaciana

Ee =  (counte + counte2)/amostras2;#gaussiana
Fe = (counte1 + counte3)/amostras2;#laplaciana

print("Erro de classificacao com Mahalanobis classe w1 gaussiana", A , "%")
print("Erro de classificacao com Mahalanobis classe w1 laplaciana = ",B, "%")
print("Erro de classificacao com Mahalanobis classe w2 gaussiana", C, "%")
print("Erro de classificacao com Mahalanobis classe w2 laplaciana = ",D, "%")

print("\n")

print("Erro de classificacao pela Euclidiana classe w1 gaussiana", Ae , "%")
print("Erro de classificacao pela Euclidiana classe w1 laplaciana = ",Be, "%")
print("Erro de classificacao pela Euclidiana classe w2 gaussiana", Ce, "%")
```

```
print("Erro de classificacao pela Euclidiana classe w2 laplaciana = ",De, "%")

print("\n")

if (E < F):
    print("Distribuicao gaussiana apresenta o menor erro com base em Mahalanobis");
else:
    print("Distribuicao laplaciana apresenta o menor erro com base em Mahalanobis");
if (Ee < Fe):
    print("Distribuicao gaussiana apresenta o menor erro com base na Euclidiana");
else:
    print("Distribuicao laplaciana apresenta o menor erro com base na Euclidiana");

media1 = (A+B+C+D)/4;
media2 = (Ae+Be+Ce+De)/4;

if (media1 == media2):
    print("Para estes dados tanto Mahalanobis quanto a Euclidiana tiveram a mesma taxa de erro");
elif (media1 < media2):
    print("Mahalanobis apresenta maior precisão na classificacao");
else:
    print("Euclidiana apresenta maior precisão na classificacao");
```

SAÍDA NO CONSOLE DO SPYDER:

```
Erro de classificacao com Mahalanobis classe w1 gaussiana 17.8 %
Erro de classificacao com Mahalanobis classe w1 laplaciana =  19.400000000000002 %
Erro de classificacao com Mahalanobis classe w2 gaussiana 16.2 %
Erro de classificacao com Mahalanobis classe w2 laplaciana =  17.0 %


Erro de classificacao pela Euclidiana classe w1 gaussiana 17.8 %
Erro de classificacao pela Euclidiana classe w1 laplaciana =  19.400000000000002 %
Erro de classificacao pela Euclidiana classe w2 gaussiana 16.2 %
Erro de classificacao pela Euclidiana classe w2 laplaciana =  17.0 %


Distribuicao gaussiana apresenta o menor erro com base em Mahalanobis
Distribuicao gaussiana apresenta o menor erro com base na Euclidiana
Para estes dados tanto Mahalanobis quanto a Euclidiana tiveram a mesma taxa de erro
```