Relatório do exercício 1

Aluno: Pedro Ian Mota Moraes
Data de Entrega: 13/03/2020

Introdução:

Para o melhor entendimento da matéria de Reconhecimento de Padrões, foi elaborada essa atividade, constituída de gerar dados aleatórios e o estudo em cima deles, levando em consideração desvio padrão e média dos valores obtidos.

O presente exercício foi realizado na linguagem Python 3.7.

Discussão:

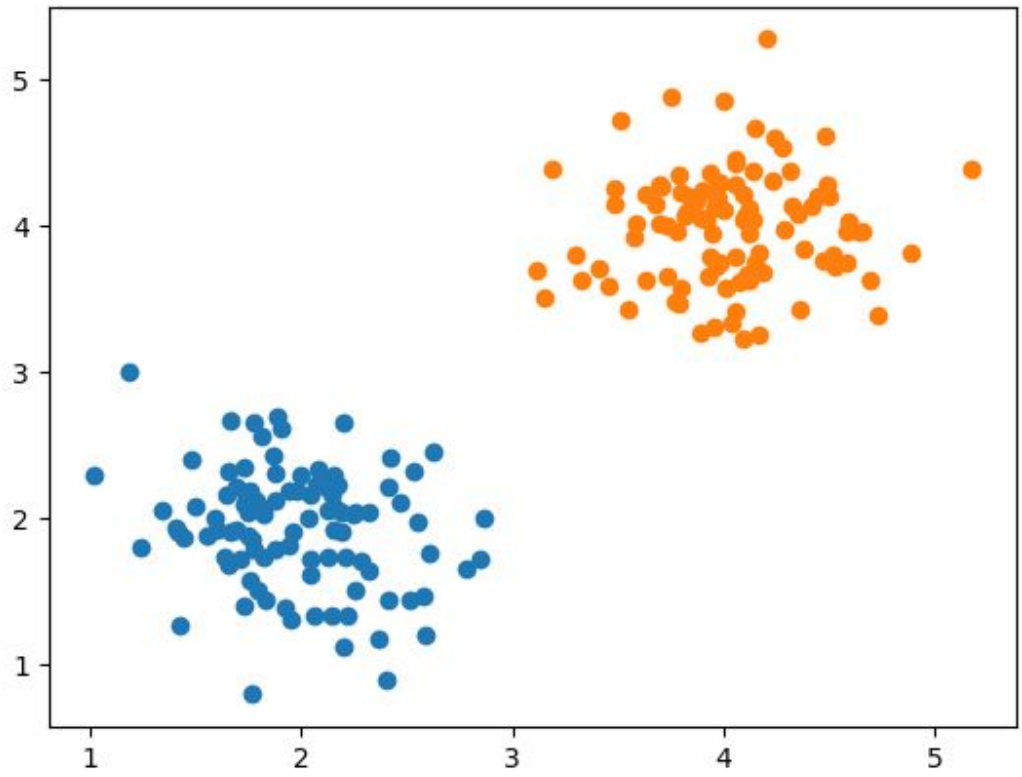
Os valores aleatórios dos pontos, seguindo a distribuição normal, foi realizada da seguinte maneira, onde as variáveis “origin” e “var” representam a origem e a variância, que são 0 e 0.6, respectivamente:

```
for i in range(100):
    x1 += [np.random.normal(origin, var) + 2]
    y1 += [np.random.normal(origin, var) + 2]
    x2 += [np.random.normal(origin, var) + 4]
    y2 += [np.random.normal(origin, var) + 4]
```

Para plotar o gráfico dos pontos distribuídos, utilizou-se as seguintes linhas de código:

```
plt.scatter(x1, y1)
plt.scatter(x2, y2)
plt.show()
```

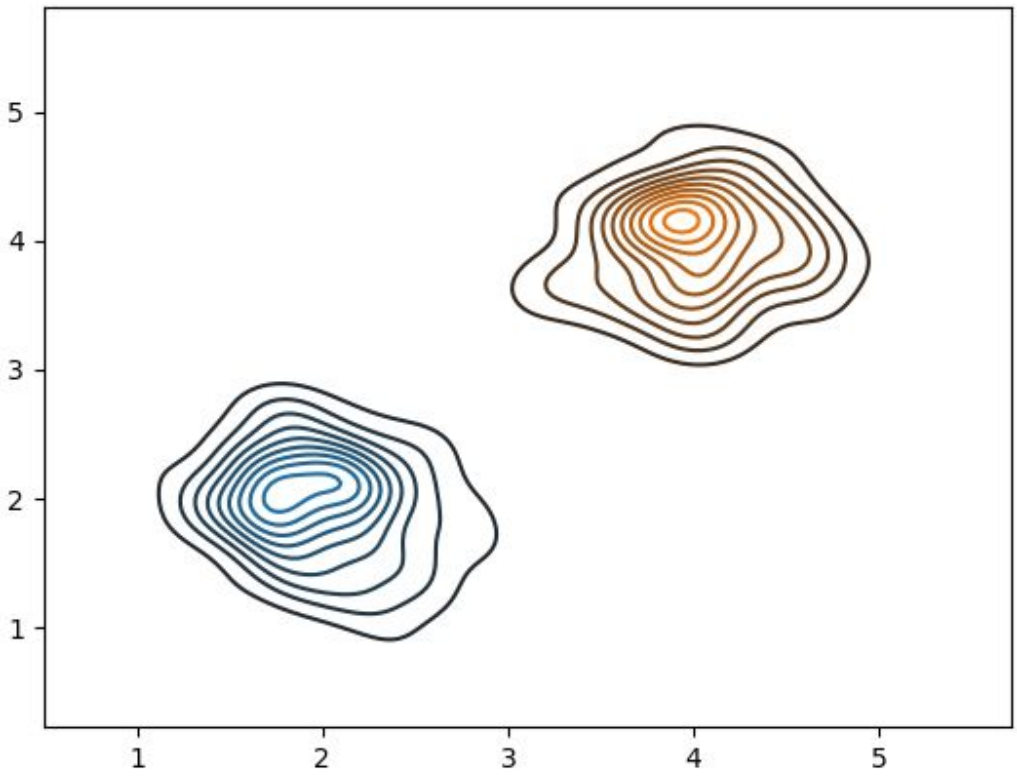
Obtendo-se o seguinte gráfico:



O contorno da densidade dos pontos foram obtidos com o auxílio da ferramenta de Kernel Distribution, assim como no código abaixo:

```
something = sns.kdeplot(x1, y1)
something2 = sns.kdeplot(x2, y2)
plt.show()
```

Obtendo-se o seguinte gráfico:



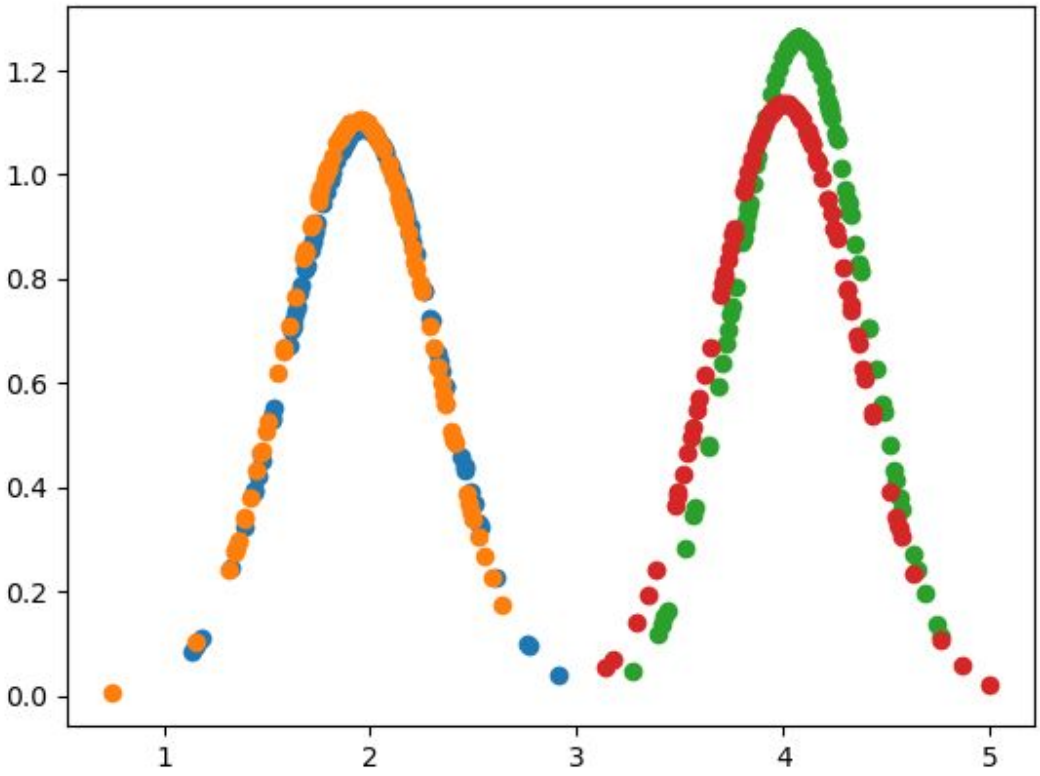
Infelizmente não foi possível implementar o código para a visualização 3D da junção destas duas informações, por motivo de incapacidade. Porém, nos relatórios em sequência, esta dificuldade já terá sido superada. Porém, os gráficos separados das distribuições de cada um dos pontos nos respectivos eixos (x e y), foram obtidos da seguinte maneira:

```
for i in range(len(x1)):
    distribuicaoX1 += [pdfUnivariada(desvioX1, mediaX1, x1[i])]
for i in range(len(y1)):
    distribuicaoY1 += [pdfUnivariada(desvioY1, mediaY1, y1[i])]
for i in range(len(x2)):
    distribuicaoX2 += [pdfUnivariada(desvioX2, mediaX2, x2[i])]
for i in range(len(y2)):
    distribuicaoY2 += [pdfUnivariada(desvioY2, mediaY2, y2[i])]

plt.scatter(x1, distribuicaoX1)
plt.scatter(y1, distribuicaoY1)
```

```
plt.scatter(x2, distribuicaoX2)
plt.scatter(y2, distribuicaoY2)
plt.show()
```

Obtendo-se o seguinte gráfico:



Código Completo Utilizado:

```
import numpy as np
import seaborn as sns
from scipy.stats import norm
from scipy.stats import kde
from matplotlib import pyplot as plt
```

```
def pdfUnivariada(sigma, media, inputX):
    expoente = - np.square(inputX - media)/(2 * np.square(sigma))
    return np.e ** expoente / (sigma * (np.sqrt(2 * np.pi)))

def calculoVerossimilhanca(sigma1, sigma2, media1, media2, inputX, inputY,
coeficienteCorrelacao=0):
    expoente = -(np.square((inputX - media1)/sigma1) + np.square((inputY -
media2)/sigma2) - 2*coeficienteCorrelacao *
                ((inputX - media1)/sigma1)*((inputY -
media2)/sigma2))/(2*(1 - np.square(coeficienteCorrelacao)))
    return (np.e**expoente)/(2*np.pi*sigma1*sigma2*np.sqrt(1 -
np.square(coeficienteCorrelacao)))


def media(input):
    return sum(input)/len(input)


def desvioPadrao(input):
    med = media(input)
    somatorio = 0
    for i in input:
        somatorio += np.square(i - med)
    return np.sqrt(somatorio/len(input))


origin = 0
var = 0.36  # Variancia da Distribuicao Normal
var2 = 0.6

# Vetores das posicoes das amostras 1 e 2
x1 = []
y1 = []
X1 = []

x2 = []
y2 = []
```

```
X2 = []

for i in range(100):
    x1 += [np.random.normal(origin, var) + 2]
    y1 += [np.random.normal(origin, var) + 2]
    x2 += [np.random.normal(origin, var) + 4]
    y2 += [np.random.normal(origin, var) + 4]

mediaX1 = media(x1)
mediaY1 = media(y1)
desvioX1 = desvioPadrao(x1)
desvioY1 = desvioPadrao(y1)

mediaX2 = media(x2)
mediaY2 = media(y2)
desvioX2 = desvioPadrao(x2)
desvioY2 = desvioPadrao(y2)

result = []

xAxesBase = np.linspace(0, 6, 100)

distribuicaoX1 = []
distribuicaoY1 = []
distribuicaoX2 = []
distribuicaoY2 = []

for i in range(len(x1)):
    distribuicaoX1 += [pdfUnivariada(desvioX1, mediaX1, x1[i])]
for i in range(len(y1)):
    distribuicaoY1 += [pdfUnivariada(desvioY1, mediaY1, y1[i])]
for i in range(len(x2)):
    distribuicaoX2 += [pdfUnivariada(desvioX2, mediaX2, x2[i])]
for i in range(len(y2)):
    distribuicaoY2 += [pdfUnivariada(desvioY2, mediaY2, y2[i])]

plt.scatter(x1, distribuicaoX1)
```

```
plt.scatter(y1, distribuicaoY1)
plt.scatter(x2, distribuicaoX2)
plt.scatter(y2, distribuicaoY2)
plt.show()

testeResult = []
for i in range(len(distribuicaoX1)):
    testeResult.append([distribuicaoX1[i], distribuicaoY1[i]])

for i in x1:
    temp = []
    for j in y1:
        temp += [calculoVerossimilhanca(float(desvioX1), float(desvioY1),
float(
            mediaX1), float(mediaY1), float(i), float(j), 0)]
    result += [temp]

fig = plt.figure()
ax = fig.gca(projection='3d')
xMesh, yMesh = np.meshgrid(distribuicaoX1, distribuicaoY1)
ax.plot_surface(x1, y1, np.asarray(result))
plt.show()

mediaX2 = media(x2)
mediaY2 = media(y2)
desvioX2 = desvioPadrao(x2)
desvioY2 = desvioPadrao(y2)

plt.scatter(x1, y1)
plt.scatter(x2, y2)
plt.show()



something = sns.kdeplot(x1, y1)
something2 = sns.kdeplot(x2, y2)
plt.show()
```