exercicio7

June 4, 2022

1 Redes Neurais Artificiais

Nome: João Pedro Miranda Marques

Matrícula: 2017050495

1.1 K-means RBF

[ ]:
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

1.1.1 Funções de treinamento e inferência da RBF

[ ]:
```python
import numpy as np
from sklearn.cluster import KMeans

def trainRBF(xin, yin, p):

    # Função Radial Gaussiana
    def pdfnvar(x, m, K, n):
        if(n == 1):
            r = np.sqrt(int(K))
            px = (1/(np.sqrt(2*np.pi*r*r)))*np.exp(-0.5 * ((x-m)/(r))**2)
        else:
            px = ((1/np.sqrt((2*np.pi)**n * (np.linalg.det(K))))) * np.exp(-0.
↪5*((x-m) @ (np.linalg.inv(K) @ np.transpose(x-m))))
        return px

    nSamples = xin.shape[0]      # Numero de amostras.
    nDimension = xin.shape[1]      # Dimensao de entrada.

    xin = np.matrix(xin)  # garante que xin seja matriz
    yin = np.matrix(yin)  # garante que yin seja matriz

    #clusteriza os dados de entrada por meio do algorítmo K-médias
    xclust = KMeans(n_clusters=p).fit(xin)
```

```
    # Armazena vetores de centros das funções
    m = np.matrix(xclust.cluster_centers_)
    covlist = []

    # Estima matrizes de covarância para todos os centros
    for i in range(p):
        ici = np.where(xclust.labels_ == i)
        xci = xin[ici,]
        if nDimension == 1:
            covi = np.var(xci)
        else:
            covi = np.cov(xci[0], rowvar=False)
        covlist.append(covi)

    H =  np.zeros((nSamples, p))

    #calcula matriz H
    for j in range(nSamples):
        for i in range(p):
            mi = m[i,]
            covi = np.transpose(covlist[i]) + 0.001*np.identity(nDimension)
            aux = pdfnvar(xin[j,], mi, covi, nSamples)
            aux = np.asarray(aux)
            H[j][i] = aux[0]

    Haug = pd.DataFrame(H)
    Haug.insert(H.shape[1], H.shape[1], 1)
    Haug.to_numpy()

    W = (np.linalg.inv(Haug.T @ Haug) @ Haug.T) @ yin.T     # W<-( solve(␣
↪t(Haug) %*% Haug) %*% t (Haug) ) %*% yin

    return [m, covlist, W, H]
```

[ ]:
```
# Calcula a saída da rede RBF
def YRBF(xin, modRBF):

    # Função Radial Gaussiana
    def pdfnvar(x, m, K, n):
        if(n == 1):
            r = np.sqrt(int(K))
            px = (1/(np.sqrt(2*np.pi*r*r)))*np.exp(-0.5 * ((x-m)/(r))**2)
        else:
            px = ((1/np.sqrt((2*np.pi)**n * (np.linalg.det(K))))) * np.exp(-0.
↪5*((x-m) @ (np.linalg.inv(K) @ np.transpose(x-m))))
        return px
```

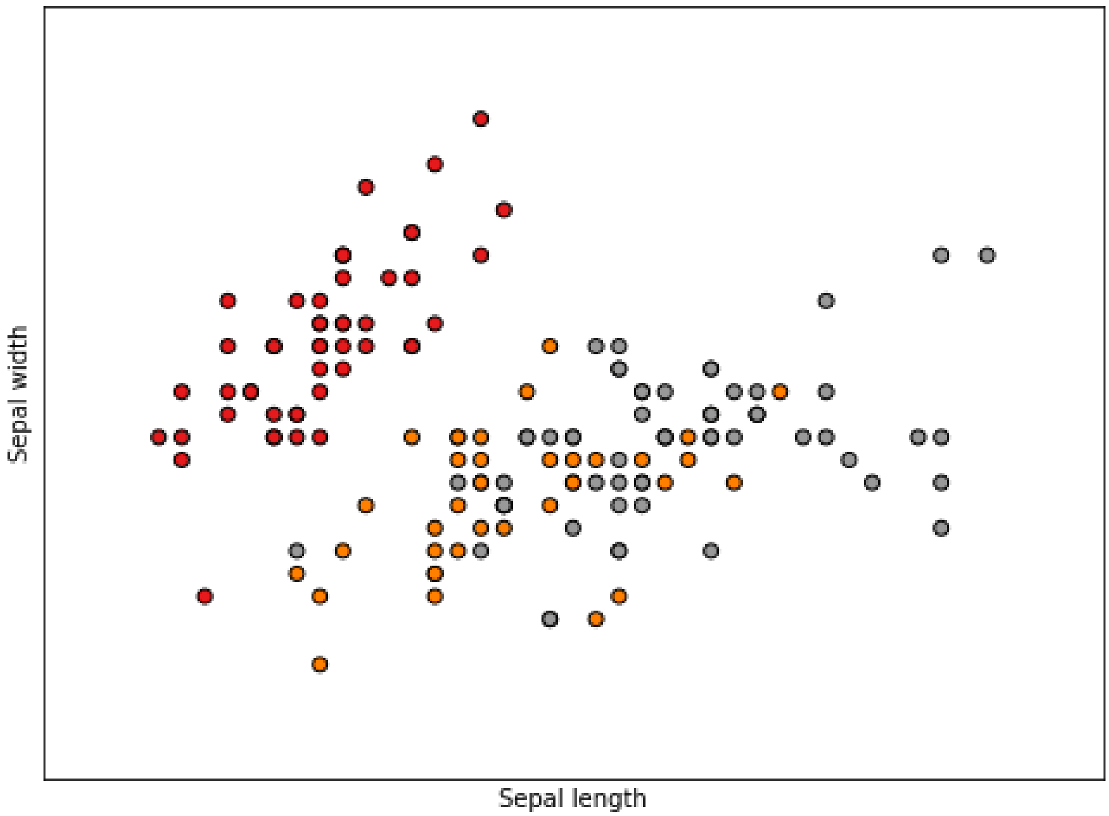
```python
    nSamples = xin.shape[0]      # Numero de amostras.
    nDimension = xin.shape[1]      # Dimensao de entrada.

    m = modRBF[0]
    covlist = modRBF[1]
    p = len(covlist) # Numero de funções radiais
    W = modRBF[2]

    H = np.zeros((nSamples, p))

    #calcula matriz H
    for j in range(nSamples):
        for i in range(p):
            mi = m[i,]
            covi = np.transpose(covlist[i]) + 0.001*np.identity(nDimension)
            aux = pdfnvar(xin[j,], mi, covi, nSamples)
            aux = np.asarray(aux)
            H[j][i] = aux[0]

    Haug = pd.DataFrame(H)
    Haug.insert(H.shape[1], H.shape[1], 1)
    Haug.to_numpy()

    Yhat = Haug @ W

    return Yhat
```

1.1.2 Teste das funções RBF com base de dados Iris

[ ]:
```python
from sklearn import datasets

# import
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
y = iris.target

xc1 = iris.data[:49,]
xc2 = iris.data[50:99,]

y1 = iris.target[:49,]
y2 = iris.target[50:99,]

# Plotting

x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
```

```python
plt.figure(2, figsize=(8, 6))
plt.clf()

# Plot the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor="k")
plt.xlabel("Sepal length")
plt.ylabel("Sepal width")

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

plt.show()

# Selecionando 30 valores para treinamento
xcTrain1 = xc1[:29,]
xcTrain2 = xc2[:29,]
yTrain1 = y1[:29,]
yTrain2= y2[:29,]
xcTest1 = xc1[30:49,]
xcTest2 = xc2[30:49,]
yTest1 = y1[30:49,]
yTest2 = y2[30:49,]
yTest1 = pd.Series(yTest1)
yTest2 = pd.Series(yTest2)

# Treinamento
xcTrain = np.concatenate((xcTrain1, xcTrain2), axis=0)
yTrain = np.concatenate((yTrain1, yTrain2), axis=0)

retlist = trainRBF(xcTrain, yTrain, 6)

m = retlist[0]
covlist = retlist[1]
W = retlist[2]
H = retlist[3]

print("m: ", m)
print("covlist: ", covlist)
print("W: ", W)
print("H: ", H)

print("Vetor de pesos do Perceptron")
print("w: \n", W)

Yhat = YRBF(xcTrain, retlist)
```

```
print("Inferências")
print("Yhat: \n", Yhat)
```




```
m:  [[6.65       3.04       4.66       1.47      ]
 [5.56666667 3.95       1.48333333 0.31666667]
 [5.45       2.4625     3.7875     1.1375    ]
 [4.64444444 3.14444444 1.33333333 0.17777778]
 [6.02727273 2.81818182 4.50909091 1.43636364]
 [5.06428571 3.45       1.55       0.26428571]]
covlist:  [array([[ 0.04722222, -0.00555556,  0.01666667, -0.00722222],
       [-0.00555556,  0.02933333, -0.00266667,  0.00577778],
       [ 0.01666667, -0.00266667,  0.04044444,  0.01311111],
       [-0.00722222,  0.00577778,  0.01311111,  0.01344444]]), array([[
0.03466667,  0.022     , -0.00866667, -0.00533333],
       [ 0.022     ,  0.059     , -0.009     ,  0.011     ],
       [-0.00866667, -0.009     ,  0.04166667,  0.00633333],
       [-0.00533333,  0.011     ,  0.00633333,  0.00966667]]), array([[
0.14857143,  0.02642857,  0.085     , -0.00357143],
       [ 0.02642857,  0.08839286,  0.01375   ,  0.02589286],
       [ 0.085     ,  0.01375   ,  0.08125   ,  0.00910714],
```

```
       [-0.00357143,  0.02589286,  0.00910714,  0.02839286]]), array([[
0.04277778,  0.00152778,  0.01708333, -0.00138889],
       [ 0.00152778,  0.05027778, -0.01916667,  0.00736111],
       [ 0.01708333, -0.01916667,  0.03      ,  0.00083333],
       [-0.00138889,  0.00736111,  0.00083333,  0.00444444]]), array([[
0.05818182, -0.03054545,  0.00272727, -0.00909091],
       [-0.03054545,  0.07163636, -0.00618182,  0.01027273],
       [ 0.00272727, -0.00618182,  0.07090909,  0.01563636],
       [-0.00909091,  0.01027273,  0.01563636,  0.02654545]]), array([[
0.02401099,  0.00423077, -0.00653846,  0.00093407],
       [ 0.00423077,  0.035     , -0.01038462,  0.00269231],
       [-0.00653846, -0.01038462,  0.02115385,  0.00192308],
       [ 0.00093407,  0.00269231,  0.00192308,  0.01016484]])]
W:                0
0  1.249814e+20
1 -1.640994e+20
2  3.676443e+20
3 -5.848053e+19
4  2.832649e+20
5 -6.344783e+19
6  5.508288e-01
H:  [[1.02558146e-081 8.01192416e-023 1.19826354e-058 5.82847209e-022
  1.28294588e-057 9.31233004e-021]
 [7.34524595e-082 3.98859382e-025 1.26441699e-052 2.46241760e-021
  7.67512488e-060 6.13076977e-023]
 [6.35033071e-087 5.68911269e-027 1.66226488e-054 1.68022660e-020
  3.08187159e-063 9.51882577e-024]
 [7.72647347e-085 6.09495736e-028 5.15752414e-049 8.16059247e-021
  1.01523395e-061 1.59735951e-023]
 [1.06397983e-083 3.54789278e-024 9.93565152e-059 3.28383396e-022
  1.43600726e-058 6.30307023e-021]
 [1.47718587e-072 5.56047336e-021 2.10112739e-056 2.68006632e-027
  1.18045915e-050 7.45780914e-024]
 [1.37026083e-084 1.26684984e-027 3.84218975e-052 3.09826996e-021
  3.28914245e-061 2.22979942e-023]
 [3.93333875e-080 1.55759242e-023 1.43083957e-054 1.47521908e-021
  8.16910442e-057 1.19965502e-020]
 [6.24329073e-090 4.80119976e-031 4.35112277e-048 4.45969367e-021
  3.77859466e-067 6.81042656e-028]
 [2.48894049e-083 1.57711245e-025 1.36196309e-052 3.83204292e-021
  1.62793028e-059 5.63218361e-022]
 [4.27269356e-079 2.75184269e-021 1.65225007e-061 1.12637522e-024
  4.79355204e-055 5.15623611e-022]
 [1.20956492e-081 4.29995395e-026 2.98212480e-051 1.76181741e-022
  2.46472080e-057 3.77197215e-021]
 [3.22480721e-086 1.13417470e-026 5.06449392e-053 9.18334602e-021
  1.97620363e-062 1.70504637e-023]
 [1.99632690e-100 4.21965061e-036 1.53500756e-055 1.94089006e-021
```

```
 1.29202747e-074 1.08819019e-032]
[2.06284471e-090 2.69637183e-021 1.45427270e-078 8.27014859e-031
 4.49562587e-061 1.34408511e-026]
[3.54517370e-085 2.07161287e-021 9.07116573e-071 6.24489278e-033
 1.59744824e-057 9.75522367e-030]
[2.80640312e-083 2.48374668e-021 2.26655111e-066 4.39117845e-027
 9.05018498e-057 1.11003054e-022]
[2.19438932e-079 2.66887143e-022 2.25773037e-057 2.14849612e-022
 2.45662228e-056 9.17655187e-021]
[3.59889253e-072 2.45764990e-021 6.09580187e-060 4.50855449e-028
 8.89856336e-051 3.48313590e-026]
[4.74377825e-080 1.03740973e-022 3.70697686e-058 6.42236110e-024
 1.35244600e-055 2.96806312e-021]
[2.45753462e-072 7.41236508e-022 4.49529914e-054 7.42890576e-024
 3.05749062e-051 2.79817435e-022]
[1.03027912e-076 5.24981362e-022 8.85550406e-056 5.66444749e-024
 4.51585760e-054 3.91582149e-021]
[2.48938033e-097 9.29604048e-032 2.29940262e-064 1.42597643e-021
 4.25204604e-069 3.30942017e-027]
[4.94191789e-068 9.03961103e-025 3.94018676e-047 1.24765796e-026
 1.39044307e-049 7.02614695e-022]
[3.14472370e-079 2.68672226e-027 8.93882753e-047 2.90862268e-027
 1.71310394e-053 3.11903209e-022]
[1.04586830e-076 2.78301593e-024 2.93884012e-049 2.61329084e-021
 9.44513980e-056 8.53272115e-022]
[4.32749365e-073 5.67670696e-023 5.95631868e-050 2.78840220e-023
 7.75747095e-053 6.31061207e-021]
[1.04471276e-078 4.34346591e-022 2.24713484e-057 1.45039190e-022
 2.01004115e-055 9.43701019e-021]
[4.07651358e-080 3.23455318e-022 8.45306818e-059 2.94160184e-022
 9.07372135e-057 6.23684369e-021]
[1.70848387e-021 3.28649456e-187 1.31168631e-025 2.85513507e-156
 7.67554381e-028 1.69267957e-209]
[4.86515440e-021 4.03908345e-156 6.30403844e-024 3.68716870e-153
 4.87660995e-023 1.31871410e-169]
[4.24264172e-021 1.59058206e-201 1.09464519e-025 4.95978948e-173
 1.33498229e-026 4.24097202e-222]
[8.16247629e-033 1.02602732e-121 6.64052614e-022 1.33538343e-118
 1.45526424e-025 9.51908204e-104]
[2.58612129e-021 1.95159885e-184 3.60730635e-024 1.00273355e-158
 2.33163368e-022 1.65862455e-177]
[8.52180525e-030 2.32176293e-116 6.52115346e-024 4.10929893e-135
 4.91637762e-022 8.99344323e-138]
[1.02444632e-021 1.15336637e-160 5.55962828e-025 1.26944562e-171
 3.13413092e-023 6.61273447e-185]
[1.17787121e-045 9.88706882e-066 4.46776632e-022 7.06372067e-075
 6.70202277e-035 2.86798855e-059]
[1.82612418e-021 1.25724251e-161 1.38762288e-023 1.05296909e-139
```

```
  4.00150378e-023 3.15089473e-177]
 [1.15102293e-032 1.71641696e-101 4.12059876e-022 2.59490647e-122
  4.78380432e-026 1.93706534e-096]
 [1.77432496e-048 1.35741479e-083 5.46749333e-022 3.04986912e-083
  5.14931090e-036 8.75173193e-070]
 [1.54736725e-023 3.29839236e-133 1.07387828e-022 1.19890921e-140
  1.04605718e-021 2.26003830e-131]
 [1.16425166e-031 1.05787828e-120 3.71980035e-022 6.05551427e-095
  2.45857558e-024 2.22119370e-109]
 [2.05802325e-024 5.93581898e-145 4.13807723e-024 1.65371421e-151
  1.70924592e-021 2.34644003e-167]
 [4.16805492e-029 6.38926109e-097 1.95395390e-022 2.80271657e-103
  1.88936449e-024 3.18512248e-086]
 [3.38399103e-021 1.38511502e-166 1.50991591e-024 3.65686457e-142
  1.96105081e-024 3.40828452e-171]
 [2.27451284e-027 2.35163772e-123 1.36947141e-024 3.86656693e-153
  6.57235200e-022 1.66170898e-142]
 [5.39280177e-032 8.40251836e-093 2.75798458e-022 4.75152717e-097
  2.13282250e-023 6.18289329e-109]
 [6.38162549e-028 9.13246701e-198 3.76889232e-025 3.30198741e-160
  1.14876226e-022 1.90454951e-158]
 [6.12863476e-032 2.91578135e-097 2.26111769e-021 2.10437390e-096
  1.20932196e-024 5.92794086e-095]
 [3.47669323e-024 1.69623083e-168 5.14983766e-027 2.33187045e-198
  1.47418247e-022 6.70802174e-185]
 [4.98905326e-024 1.91991507e-128 1.58824351e-022 1.36897627e-116
  4.76506856e-022 1.15687960e-118]
 [1.38184684e-024 2.23913715e-192 1.92114115e-025 1.48171719e-170
  4.82506703e-022 3.03029409e-190]
 [3.84339653e-029 9.02541418e-130 2.33253267e-024 2.12567532e-136
  4.28591628e-022 5.25039615e-162]
 [1.09600064e-021 1.67769160e-144 5.88209111e-023 3.15829266e-127
  3.43805479e-022 7.20637126e-148]
 [4.79547193e-021 4.73861891e-165 5.11670207e-024 1.92860263e-141
  2.42518239e-023 5.28135803e-166]
 [3.41919008e-021 5.66106063e-196 8.86766559e-025 2.42039759e-158
  3.11688928e-024 1.86298672e-203]
 [1.19331450e-021 2.32529160e-221 8.72946101e-027 8.28609555e-198
  3.96651472e-025 1.07811395e-225]
 [5.41933247e-023 4.26595307e-147 2.59607750e-023 4.71244963e-152
  2.78525494e-021 7.58455050e-153]]
Vetor de pesos do Perceptron
w:
               0
0  1.249814e+20
1 -1.640994e+20
2  3.676443e+20
3 -5.848053e+19
```

```
4  2.832649e+20
5 -6.344783e+19
6  5.508288e-01
Inferências
Yhat:
            0
0  -0.087251
1   0.402870
2  -0.432382
3   0.072579
4   0.131126
5  -0.362115
6   0.368225
7  -0.299154
8   0.290024
9   0.290968
10  0.066472
11  0.301195
12  0.012698
13  0.437325
14  0.108355
15  0.210878
16  0.136204
17 -0.087764
18  0.147528
19  0.345112
20  0.411004
21  0.215898
22  0.467437
23  0.506100
24  0.531039
25  0.343407
26  0.139488
27 -0.127687
28  0.084833
29  0.764406
30  1.175014
31  1.081124
32  0.795005
33  0.941419
34  0.692490
35  0.687948
36  0.715084
37  0.795497
38  0.702334
39  0.751838
40  0.888555
41  0.688282
```

```
42  1.036777
43  0.623200
44  0.974875
45  0.737504
46  0.658266
47  0.583508
48  1.382458
49  0.593024
50  0.744821
51  0.687749
52  0.673091
53  0.806822
54  1.158924
55  0.979373
56  0.700087
57  1.356111
```

```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```

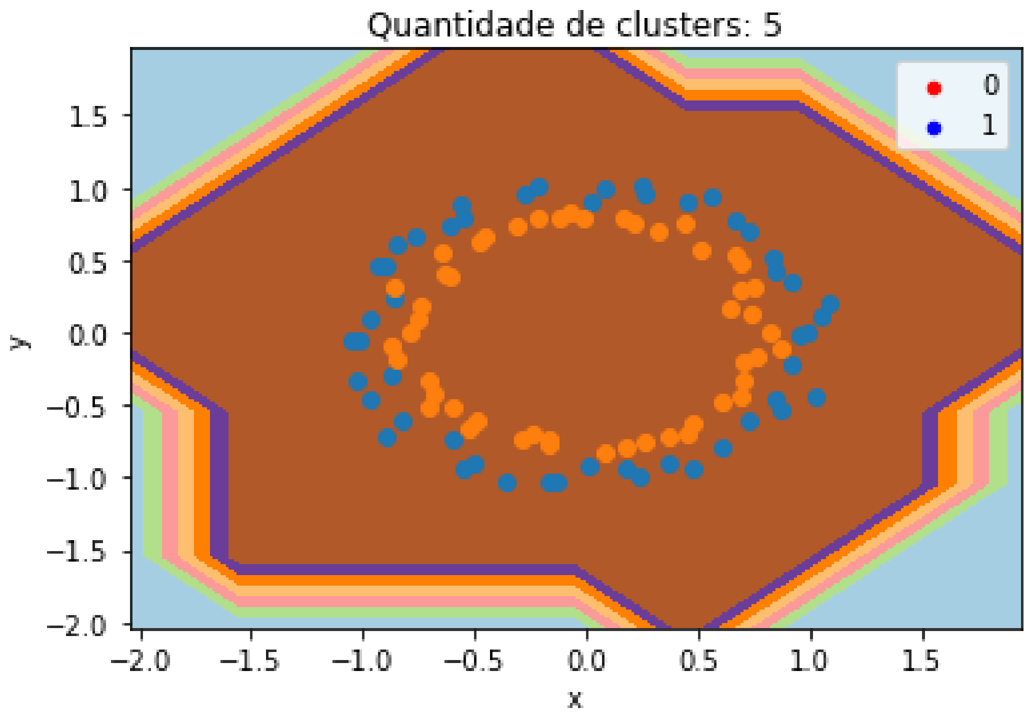
1.1.3 Exercicio

[ ]:
```python
from sklearn.datasets import make_circles, make_moons, make_blobs
from matplotlib import pyplot
from pandas import DataFrame
from numpy import where
from numpy import meshgrid
from numpy import arange
from numpy import hstack

def plotContour(format, nClusters):
    if(format == 1):
        X, y = make_circles(n_samples=100, noise=0.05)
    elif(format == 2):
        X, y = make_moons(n_samples=100, noise=0.05)
    # scatter plot, dots colored by class value

    # define bounds of the domain
    min1, max1 = X[:, 0].min()-1, X[:, 0].max()+1
    min2, max2 = X[:, 1].min()-1, X[:, 1].max()+1
    # define the x and y scale
    x1grid = arange(min1, max1, 0.5)
    x2grid = arange(min2, max2, 0.5)
    # create all of the lines and rows of the grid
    xx, yy = meshgrid(x1grid, x2grid)
```

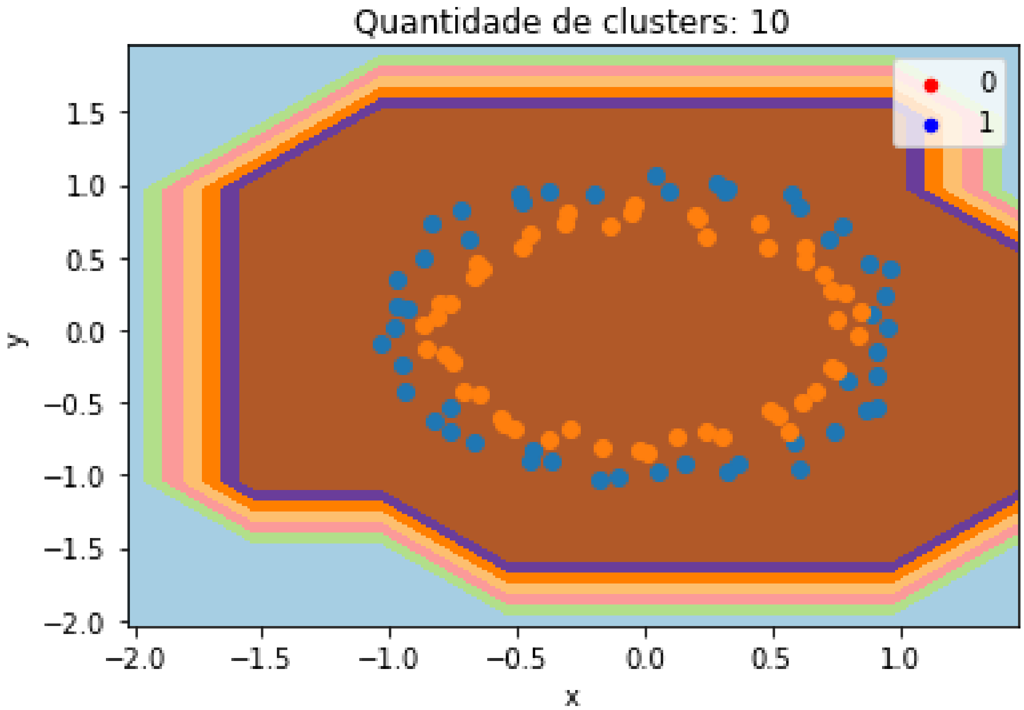
```python
    # flatten each grid to a vector
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    # horizontal stack vectors to create x1,x2 input for the model
    grid = hstack((r1,r2))

    df = DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
    colors = {0:'red', 1:'blue'}
    fig, ax = pyplot.subplots()
    grouped = df.groupby('label')
    for key, group in grouped:
        group.plot(ax=ax, kind='scatter', x='x', y='y', label=key,␣
 ↪color=colors[key])

    retlist = trainRBF(X[:,:2], y, nClusters)

    m = retlist[0]
    covlist = retlist[1]
    W = retlist[2]
    H = retlist[3]

    #plotting contours
    # Make prediction from training process
    yhat = YRBF(grid, retlist)
    yhat = (yhat > 0.5).astype(int)
    yhat = pd.DataFrame(yhat).to_numpy()

    # reshape the predictions back into a grid
    zz = yhat.reshape(xx.shape)
    # plot the grid of x, y and z values as a surface
    pyplot.contourf(xx, yy, zz, cmap='Paired')
    # create scatter plot for samples from each class
    for class_value in range(2):
        # get row indexes for samples with this class
        row_ix = where(y == class_value)
        # create scatter of these samples
        pyplot.scatter(X[row_ix, 0], X[row_ix, 1], cmap='Paired')
    # show the plot
    if(nClusters == 5):
        plt.title('Quantidade de clusters: 5')
    elif(nClusters == 10):
        plt.title('Quantidade de clusters: 10')
    elif(nClusters == 30):
        plt.title('Quantidade de clusters: 30')
    pyplot.show()

plotContour(1,5)
```

```
plotContour(1,10)
plotContour(1,30)
plotContour(2,5)
plotContour(2,10)
plotContour(2,30)
```

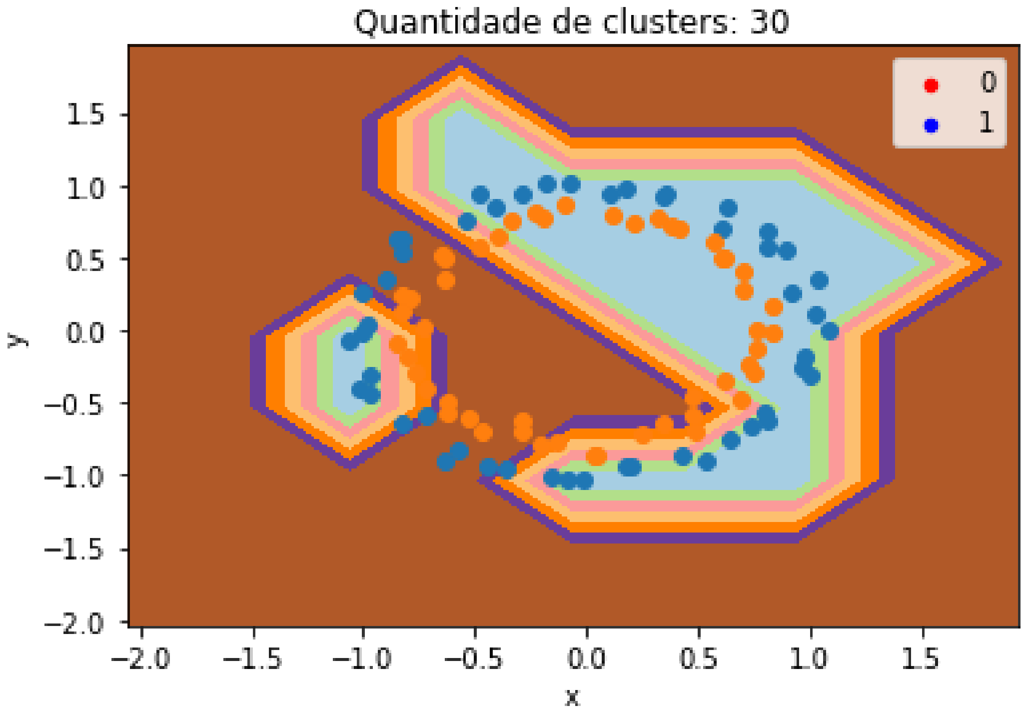
```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```



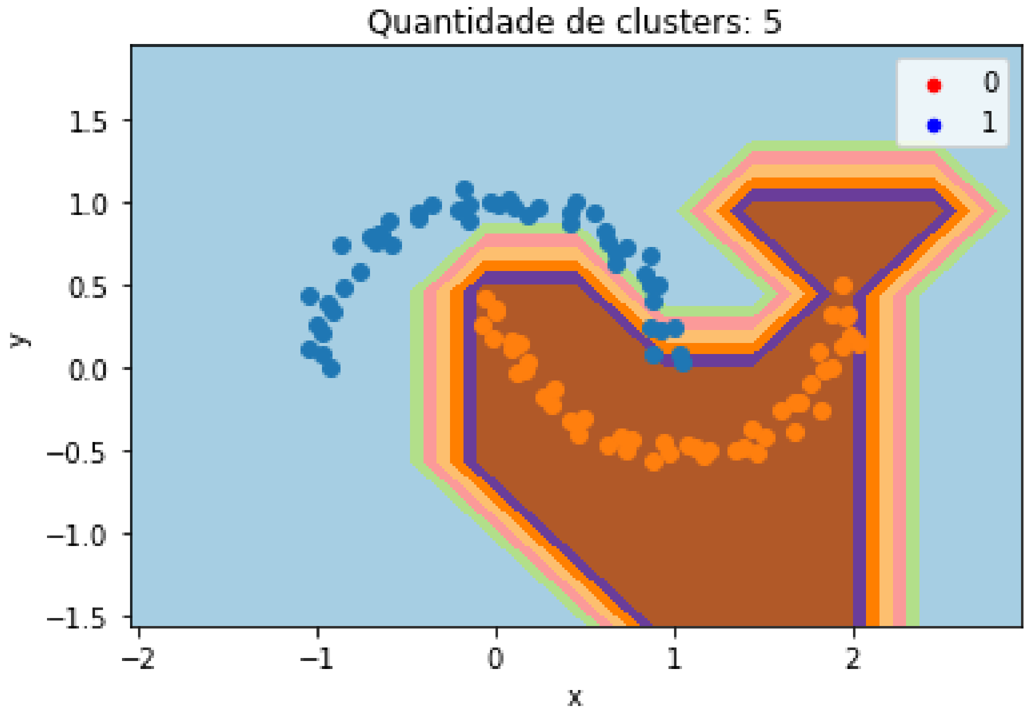

```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```



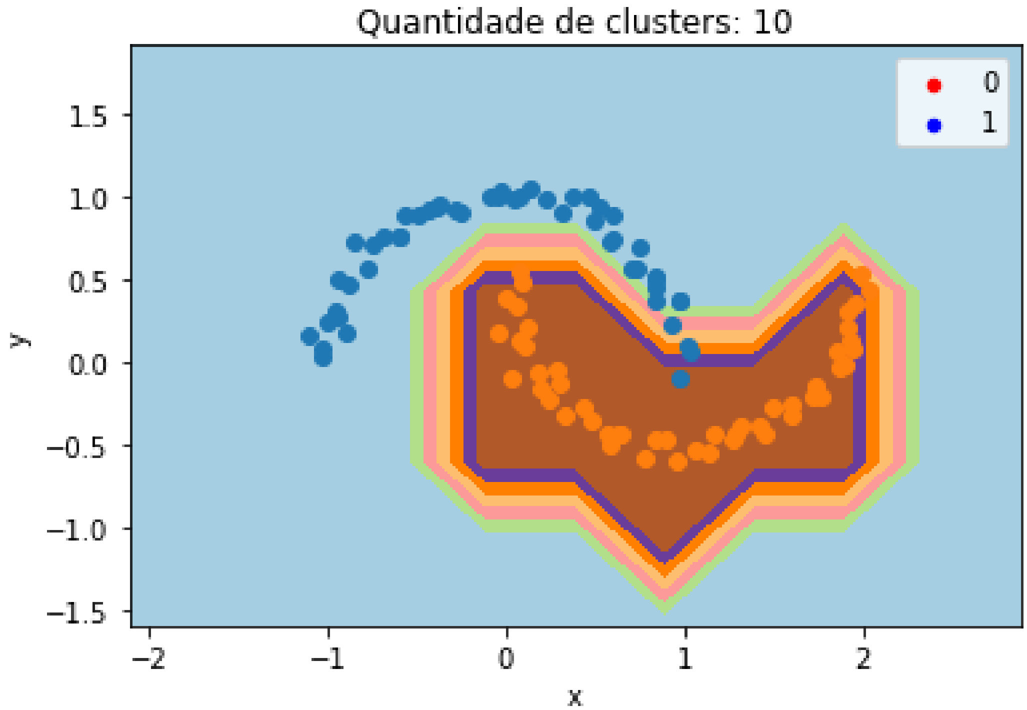

```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```



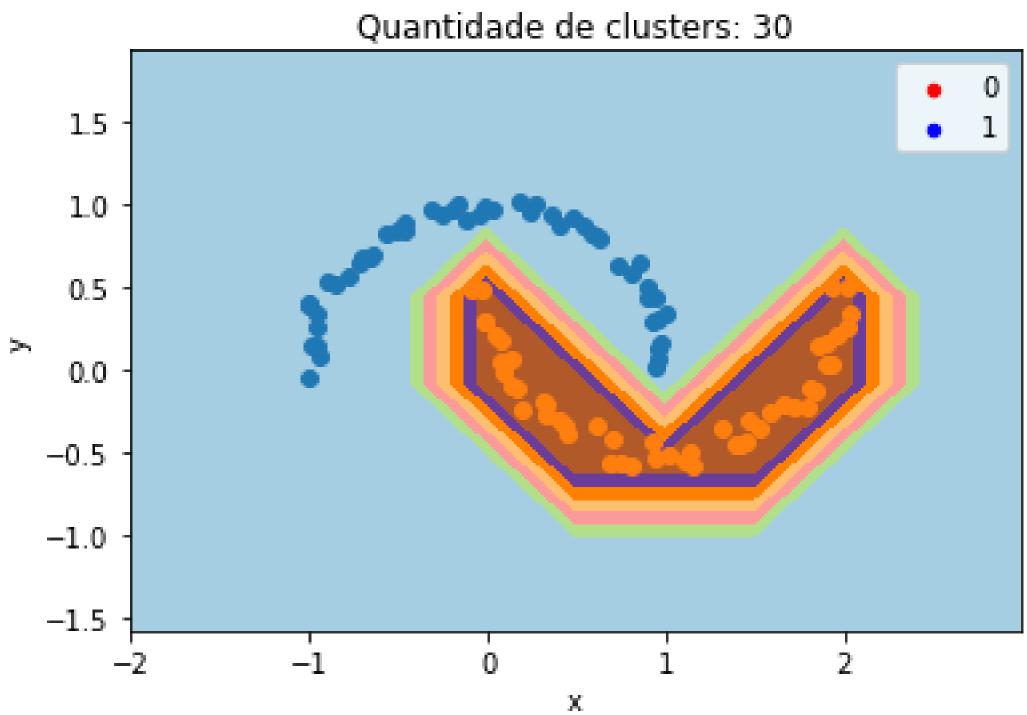

```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```




```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```




```
/opt/homebrew/lib/python3.9/site-packages/sklearn/utils/validation.py:593:
FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeError
in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```



1.1.4 Aproximação da função Sinc

[ ]:
```
xin = np.arange(0, 2*np.pi, 0.1*np.pi)

yin = pd.Series(np.sinc(xin))

retlist = trainRBF(xin, yin, 3)

yhat = YRBF(xin, retlist)

plt.plot(range(1, len(yhat) + 1), yhat)

plt.show()
```

```
␣
↪---------------------------------------------------------------------------

    IndexError                                Traceback (most recent call␣
↪last)

    /Users/jota/Documents/RNA/Artificial Neural Networks/RBF/part1/
↪exercicio7.ipynb Cell 12' in <cell line: 5>()
```

```
        <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000012?line=0'>1</
↪a> xin = np.arange(0, 2*np.pi, 0.1*np.pi)
        <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000012?line=2'>3</
↪a> yin = pd.Series(np.sinc(xin))
  ----> <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000012?line=4'>5</
↪a> retlist = trainRBF(xin, yin, 3)
        <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000012?line=6'>7</
↪a> yhat = YRBF(xin, retlist)
        <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000012?line=8'>9</
↪a> plt.plot(range(1, len(yhat) + 1), yhat)


      /Users/jota/Documents/RNA/Artificial Neural Networks/RBF/part1/
↪exercicio7.ipynb Cell 5' in trainRBF(xin, yin, p)
       <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000003?
↪line=12'>13</a>     return px
       <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000003?
↪line=14'>15</a> nSamples = xin.shape[0]     # Numero de amostras.
  ---> <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000003?
↪line=15'>16</a> nDimension = xin.shape[1]     # Dimensao de entrada.
       <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000003?
↪line=17'>18</a> xin = np.matrix(xin)  # garante que xin seja matriz
       <a href='vscode-notebook-cell:/Users/jota/Documents/RNA/
↪Artificial%20Neural%20Networks/RBF/part1/exercicio7.ipynb#ch0000003?
↪line=18'>19</a> yin = np.matrix(yin)  # garante que yin seja matriz


      IndexError: tuple index out of range
```