UNIVERSIDADE FEDERAL DE MINAS GERAIS

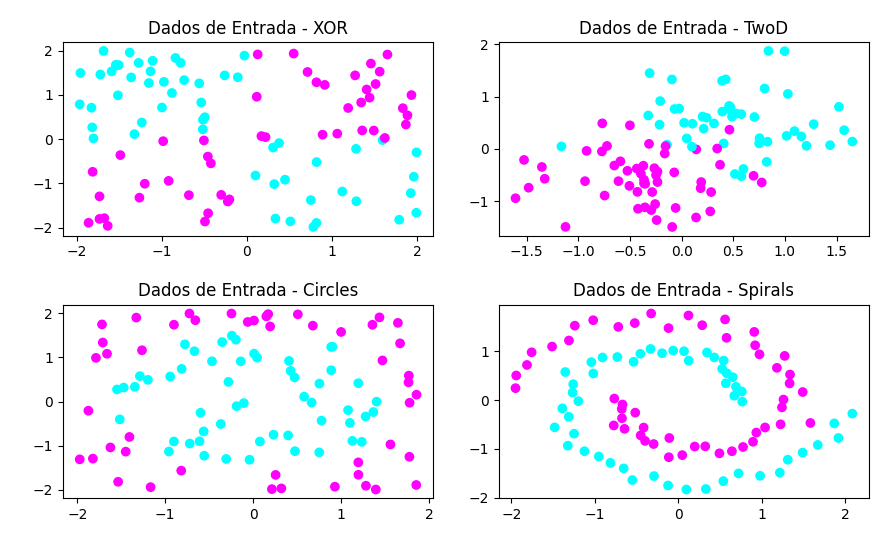
ELT131– REDES NEURAIS ARTIFICIAIS

EXERCÍCIO 5 – ELMS

Gabriel Saraiva Espeschit – 2015065541

8 de setembro de 2020

Para a realização do exercício 5, primeiro foi necessário importar os dados. Fez-se isso utilizando o módulo *rpy2* para Python que permite a chamada de funções e pacotes de R em Python. Sendo assim, criou-se 4 *datasets* usando o pacote *mlbench* de R conforme especificado na guia do relatório:

Imagem 1: Dados de entrada analisados

Vale ressaltar que, para que a metodologia de ELM será corretamente implementada, foi necessário classificar as classes em -1 e 1. Em seguida, criou-se duas funções: *ELM\_train* e *ELM\_y.* A função *ELM\_train* implementa o treinamento de ELMs:

1. Criar a matriz de pesos Z de tamanho ((n+1), p), onde n é a dimensão dos dados de entrada e p é o número de neurônios desejados;
2. Calcular a matriz H por meio da multiplicação matricial entre Xaug e Z, onde Xaug são os dados de entrada com uma coluna de uns para representar os valores de bias;
3. Calcular a matriz W por meio da pseudo-inversa de H multiplicada por Y.
4. A função retorna os parâmetros W, H e Z.

Para cada conjunto de dados de entrada obtemos os parâmetros W, H e Z considerando 5, 10 e 30 neurônios.

Em seguida criou-se um espaço de dados teste, que vai de -2,1 até 2,1, tanto em X1 quanto em X2. Utilizando a função *ELM\_y*, alimentou os dados teste para descobrir quais são as superfícies de separação para cada conjunto de dados. A função *ELM\_y* retorna a classe que cada ponto do *X\_data* dado os pesos W e Z.

Os resultados obtidos podem ser vistos abaixo:

Os resultados mostram que quanto maior a quantidade de neurônios, melhor foi a separação entre as classes, visto que para p = 5 e 10 podemos ver que ocorreu um *underfitting* em alguns dos conjuntos de dados. O código desenvolvido pode ser conferido no apêndice desse documento.

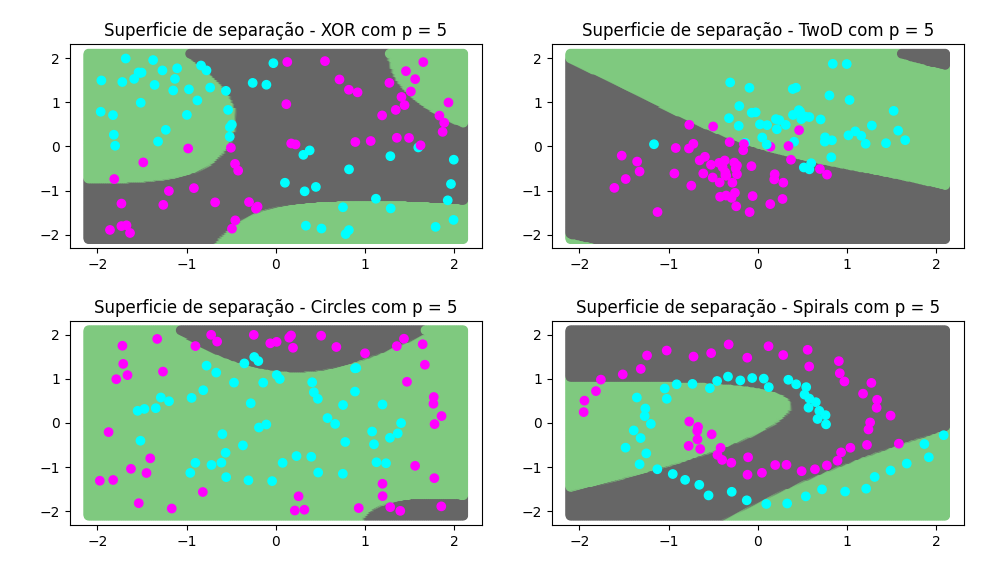
Imagem 2: Superfícies de separação para cada conjunto de dados com p=5

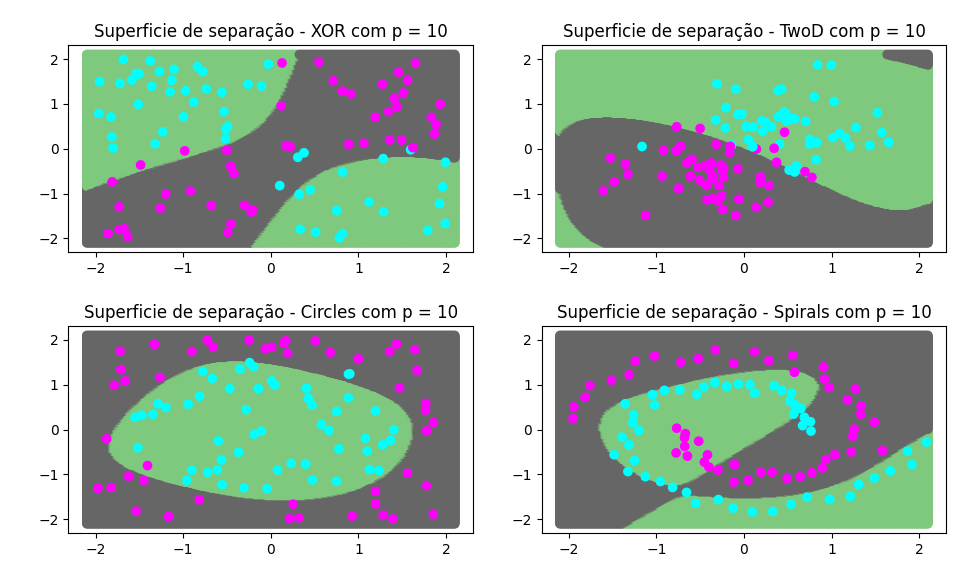
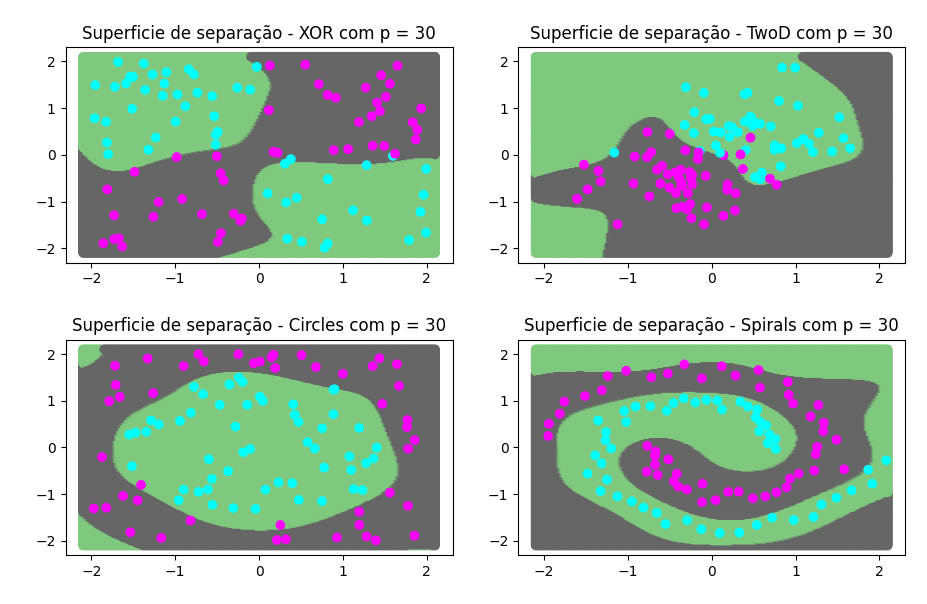
Imagem 3: Superfícies de separação para cada conjunto de dados com p=10

Imagem 4: Superfícies de separação para cada conjunto de dados com p=30

# Apêndice

# Importando Bibliotecas

from rpy2.robjects.packages import importr

import matplotlib.pyplot as plt

import numpy as np

import perceptron

mlbench = importr('mlbench')

#Gerando dados

xor = mlbench.mlbench\_xor(100)

twod = mlbench.mlbench\_2dnormals(100)

circles = mlbench.mlbench\_circle(100)

spirals = mlbench.mlbench\_spirals(100, sd=0.05)

# Transformando dados em matrizes

xor\_data, xor\_class = np.array(xor[0])\*2, np.array(xor[1])

twod\_data,twod\_class = np.array(twod[0])/2, np.array(twod[1])

circles\_data, circles\_class = np.array(circles[0])\*2, np.array(circles[1])

spirals\_data, spirals\_class = np.array(spirals[0])\*2, np.array(spirals[1])

# Plotando dados de entrada

f, (ax) = plt.subplots(2,2)

ax[0, 0].set\_title('Dados de Entrada - XOR')

ax[0, 0].scatter(xor\_data[:, 0], xor\_data[:,1], c = xor\_class, cmap = 'cool')

ax[0, 1].set\_title('Dados de Entrada - TwoD')

ax[0, 1].scatter(twod\_data[:, 0], twod\_data[:,1], c = twod\_class, cmap = 'cool')

ax[1, 0].set\_title('Dados de Entrada - Circles')

ax[1, 0].scatter(circles\_data[:, 0], circles\_data[:,1], c = circles\_class, cmap = 'cool')

ax[1, 1].set\_title('Dados de Entrada - Spirals')

ax[1, 1].scatter(spirals\_data[:, 0], spirals\_data[:,1], c = spirals\_class, cmap = 'cool')

f.tight\_layout()

plt.show()

# Função ELM\_Train

def ELM\_train(X\_data, Y\_data, num\_neuronios):

p = num\_neuronios

X = X\_data

Y = np.where(Y\_data == 2, -1, 1)

n = X.shape[1]

Z = np.random.uniform(low = -0.5, high = 0.5, size = (n+1, p))

Xaug = np.append(X, np.ones((X.shape[0], 1)), 1)

H = np.tanh(np.matmul(Xaug, Z))

W = np.matmul(np.linalg.pinv(H), Y)

return W, H, Z

# Função ELM\_y

def ELM\_y(X\_data, W, Z):

X = X\_data

Xaug\_t = np.append(X, np.ones((X.shape[0], 1)), 1)

H\_t = np.tanh(np.matmul(Xaug\_t, Z))

Y\_hat = np.matmul(H\_t, W)

return np.where(Y\_hat < 0, 1, -1)

# Cirando espaço teste

x1\_lin = np.linspace(-2.1, 2.1, 200)

x2\_lin = np.linspace(-2.1, 2.1, 200)

X1, X2 = np.meshgrid(x1\_lin, x2\_lin)

X\_lin = np.column\_stack((X1.ravel(),X2.ravel()))

num\_layers = 1

# Achando os parametros W, H e Z

W\_xor, H\_xor, Z\_xor = ELM\_train(xor\_data, xor\_class, num\_layers)

Y\_xor = ELM\_y(X\_lin, W\_xor, Z\_xor)

W\_twod, H\_twod, Z\_twod = ELM\_train(twod\_data, twod\_class, num\_layers)

Y\_twod = ELM\_y(X\_lin, W\_twod, Z\_twod)

W\_circles, H\_circles, Z\_circles = ELM\_train(circles\_data, circles\_class, num\_layers)

Y\_circles = ELM\_y(X\_lin, W\_circles, Z\_circles)

W\_spirals, H\_spirals, Z\_spirals = ELM\_train(spirals\_data, spirals\_class, num\_layers)

Y\_spirals = ELM\_y(X\_lin, W\_spirals, Z\_spirals)

# Plotando Resultados

f, (ax) = plt.subplots(2,2)

ax[0, 0].scatter(X\_lin[:, 0], X\_lin[:,1], c = Y\_xor, cmap = 'Accent')

ax[0, 1].scatter(X\_lin[:, 0], X\_lin[:,1], c = Y\_twod, cmap = 'Accent')

ax[1,0].scatter(X\_lin[:, 0], X\_lin[:,1], c = Y\_circles, cmap = 'Accent')

ax[1,1].scatter(X\_lin[:, 0], X\_lin[:,1], c = Y\_spirals, cmap = 'Accent')

ax[0, 0].set\_title(f'Superficie de separação - XOR com p = {num\_layers}')

ax[0, 1].set\_title(f'Superficie de separação - TwoD com p = {num\_layers}')

ax[1, 0].set\_title(f'Superficie de separação - Circles com p = {num\_layers}')

ax[1, 1].set\_title(f'Superficie de separação - Spirals com p = {num\_layers}')

ax[0, 0].scatter(xor\_data[:, 0], xor\_data[:,1], c = xor\_class, cmap = 'cool')

ax[0, 1].scatter(twod\_data[:, 0], twod\_data[:,1], c = twod\_class, cmap = 'cool')

ax[1,0].scatter(circles\_data[:, 0], circles\_data[:,1], c = circles\_class, cmap = 'cool')

ax[1,1].scatter(spirals\_data[:, 0], spirals\_data[:,1], c = spirals\_class, cmap = 'cool')

f.tight\_layout()

plt.show()