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:

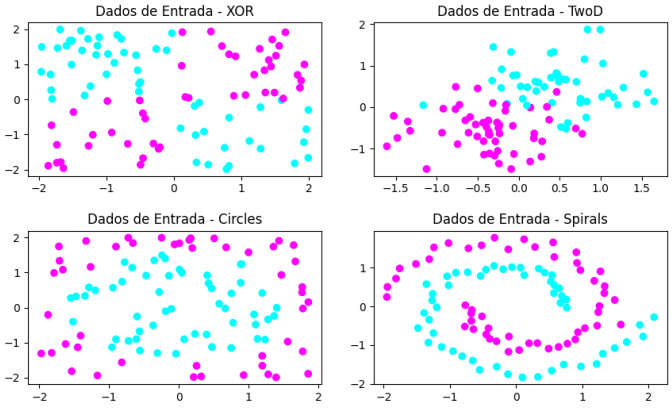

Imagem 1: Dados de entrada analisados

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:

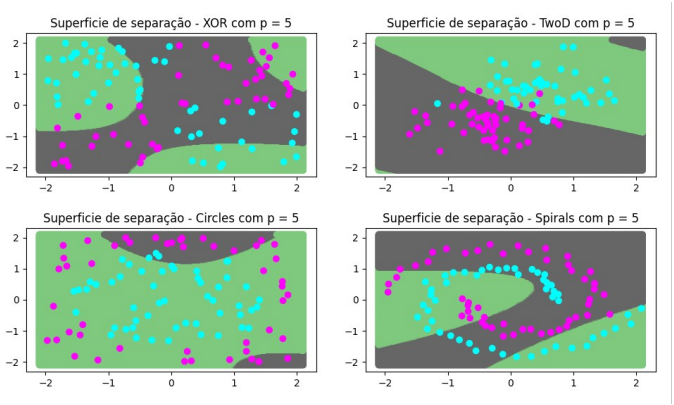

Imagem 2: Superfícies de separação para cada conjunto de dados com p=5

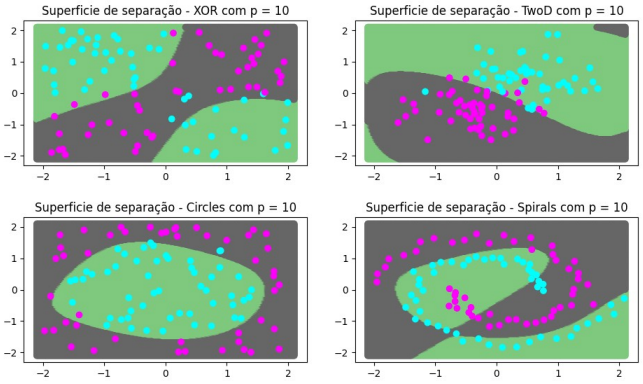

Imagem 3: Superfícies de separação para cada conjunto de dados com p=10

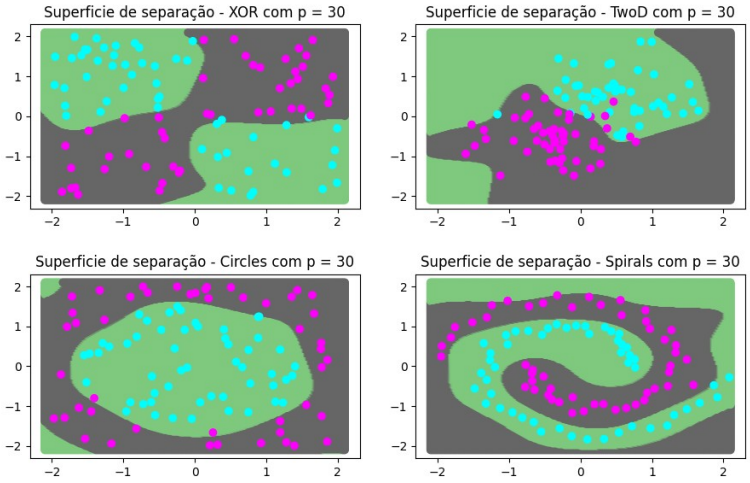

Imagem 4: Superfícies de separação para cada conjunto de dados com p=30

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.

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()
```