# exercicio6

May 15, 2022

## 1 Redes Neurais Artificiais

**Nome: João Pedro Miranda Marques**

**Matrícula: 2017050495**

## 2 Extreme Learning Machine - ELM

As imagens abaixo foram criadas utilizando o codigo em R dado pelo professor para as funcoes de trainELM e YELM.

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

Algoritmo de treinamento da Rede ELM traduzida de R para Python.

[ ]:
```
# Treinamento de uma rede ELM
def trainELM(xin, yin, nNeurons, par):
    xin = pd.DataFrame(xin)
    yin = pd.DataFrame(yin)

    nDimension = xin.shape[1]      # Dimensao de entrada.

    # Adiciona ou não um termo de polarização ao vetor de treinamento w.
    if par == 1:
        xin.insert(nDimension, nDimension, 1)
        # Z<-replicate(p, runif((n+1),-0.5,0.5))
        Z = [np.random.uniform(low=-0.5, high=0.5, size=nDimension+1) for _ in␣
 ↪range(nNeurons)]
    else:
        Z = [np.random.uniform(low=-0.5, high=0.5, size=nDimension) for _ in␣
 ↪range(nNeurons)]

    Z = pd.DataFrame(Z)
    Z = Z.T

    H = np.tanh(xin @ Z)
```

```
    W = ( np.linalg.pinv(H) @ yin)    #W<-pseudoinverse(H) %*% yin

    return [W,H,Z]
```

Algoritmo de Predição da ELM traduzido de R para Python

[ ]:
```
# Saída de uma rede ELM
def YELM(xin, Z, W, par):

    xin = pd.DataFrame(xin)
    Z = pd.DataFrame(Z)
    W = pd.DataFrame(W)

    nDimension = xin.shape[1]  # Dimensao de entrada.

    # Adiciona ou não termo de polarização
    if(par == 1):
        xin.insert(nDimension, nDimension, 1)
        # np.c_[ xin, np.ones(xin.shape[0]) ]

    # print("xin:", xin.shape)
    # print("Z:", Z.shape)
    H = np.tanh(xin @ Z)
    # print("H:", H.shape)
    # print("W:", W.shape)
    Yhat = np.sign(H @ W)

    return Yhat
```

## 2.1 Breast Cancer (diagnostic)

[ ]:
```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import accuracy_score

nNeurons = 100

def exercicio6(nNeurons):
    X, y = load_breast_cancer(return_X_y=True)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

    retlist = trainELM(X_train, y_train, nNeurons, 1)

    W = retlist[0]
    H = retlist[1]
```

```
    Z = retlist[2]

    # Make prediction from training process
    yhat = YELM(X_test, Z, W, 1)
    yhat = (yhat > 0.5).astype(int)
    yhat = pd.DataFrame(yhat).to_numpy()

    return accuracy_score(y_test, yhat)


def AcuraciaMedia(nNeurons):
    maxepocas = 100
    acuracia = []
    while(maxepocas > 0):
        acuracia = np.append(acuracia, exercicio6(nNeurons))
        maxepocas -= 1
    return acuracia

print('Acurácia média com 5 neuronios: ',np.average(AcuraciaMedia(5)), '+/-',␣
 ↪np.std(AcuraciaMedia(5)))
print('Acurácia média com 10 neuronios: ',np.average(AcuraciaMedia(10)), '+/-',␣
 ↪np.std(AcuraciaMedia(10)))
print('Acurácia média com 30 neuronios: ',np.average(AcuraciaMedia(30)), '+/-',␣
 ↪np.std(AcuraciaMedia(30)))
print('Acurácia média com 50 neuronios: ',np.average(AcuraciaMedia(50)), '+/-',␣
 ↪np.std(AcuraciaMedia(50)))
print('Acurácia média com 100 neuronios: ',np.average(AcuraciaMedia(100)), '+/
 ↪-', np.std(AcuraciaMedia(100)))
print('Acurácia média com 300 neuronios: ',np.average(AcuraciaMedia(300)), '+/
 ↪-', np.std(AcuraciaMedia(300)))
```

```
Acurácia média com 5 neuronios:  0.6430409356725147 +/- 0.047825246448636824
Acurácia média com 10 neuronios:  0.6502923976608188 +/- 0.052385748175901574
Acurácia média com 30 neuronios:  0.662982456140351 +/- 0.06394022186031889
Acurácia média com 50 neuronios:  0.692923976608187 +/- 0.06551273831113473
Acurácia média com 100 neuronios:  0.703216374269006 +/- 0.05584385031314485
Acurácia média com 300 neuronios:  0.7352631578947368 +/- 0.03868941498718851
```

### 2.2 Base Statlog (Heart)

[ ]:
```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

df = pd.read_csv('heart.csv')
```