UNIVERSIDADE FEDERAL DE MINAS GERAIS

ELT131 – REDES NEURAIS ARTIFICIAIS

RELATÓRIO EXERCÍCIO 6

Gabriel Saraiva Espeschit – 2015065541

15 de setembro de 2020

Utilizando o módulo *Pandas,* os dados do *Breast Cancer (diagnostic)* e *Statlog (Heart)* foram lidos e colocados no formato de *dataframes* do *Pandas*. Em seguida, os atributos foram normalizados utilizando a função dada na guia para restringi-los para valores entre 0 e 1:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

Visualizamos os dados do *dataframe* de câncer e coração (dispostos abaixo) para verificar que estavam corretamente normalizados e pronto para serem divididos em atributos e classes de teste e de treino.

	Diagnosis	1	2	3	4	5	6	7	8	9	...	21	22	23	24	25	26	27	28	29	30
842302	M	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731113	0.686364	...	0.620776	0.141525	0.668310	0.450698	0.601136	0.619292	0.568610	0.912027	0.598462	0.418864
842517	M	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348757	0.379798	...	0.606901	0.303571	0.539818	0.435214	0.347553	0.154563	0.192971	0.639175	0.233590	0.222878
84300903	M	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635686	0.509596	...	0.556386	0.360075	0.508442	0.374508	0.483590	0.385375	0.359744	0.835052	0.403706	0.213433
84348301	M	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522863	0.776263	...	0.248310	0.385928	0.241347	0.094008	0.915472	0.814012	0.548642	0.884880	1.000000	0.773711
84358402	M	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390	0.378283	...	0.519744	0.123934	0.506948	0.341575	0.437364	0.172415	0.319489	0.558419	0.157500	0.142595

5 rows × 31 columns

	1	2	3	4	5	6	7	8	9	10	11	12	13	Diagnosis
0	0.854167	1.0	1.000000	0.339623	0.447489	0.0	1.0	0.290076	0.0	0.387097	0.5	1.000000	0.0	2
1	0.791667	0.0	0.666667	0.198113	1.000000	0.0	1.0	0.679389	0.0	0.258065	0.5	0.000000	1.0	1
2	0.583333	1.0	0.333333	0.283019	0.308219	0.0	0.0	0.534351	0.0	0.048387	0.0	0.000000	1.0	2
3	0.729167	1.0	1.000000	0.320755	0.312785	0.0	0.0	0.259542	1.0	0.032258	0.5	0.333333	1.0	1
4	0.937500	0.0	0.333333	0.245283	0.326484	0.0	1.0	0.381679	1.0	0.032258	0.0	0.333333	0.0	1

Em seguida os *dataframes* de cada *dataset* foram convertidos em duas *arrays* de *Numpy*. Uma contendo somente os atributos e outra contendo somente as classes (diagnósticos) referente a cada atributo. Para cada *dataset* as classes foram convertidas para que ficasse inteligível para as funções de *Perceptreon* e *ELM* desenvolvidas para os exercícios passados. Isto é, as classificações ficaram expressas em -1/1 e 0/1 para serem usadas nos métodos de *ELM* e *Perceptron* respectivamente.

Utilizando a função *train_test_split* do módulo *SciKitLearn,* dividimos os datasets em instâncias para treino e instâncias para validação numa quantia de 70/30 para treinar as ELMs. Sendo assim, ficamos com vetores de treino e teste com as seguintes dimensões:

Dimensão atributos de treino *dataset* de câncer: (398, 30)
Dimensão do vetor de classes de treino *dataset* de câncer: (398, 1)
Dimensão atributos de teste *dataset* de câncer: (171, 30)

Dimensão do vetor de classes de teste *dataset* de câncer: (171, 1)
Dimensão atributos de treino *dataset* de coração: (189, 13)
Dimensão do vetor de classes de treino *dataset* de coração: (189, 1)
Dimensão atributos de teste *dataset* de coração: (81, 13)
Dimensão do vetor de classes de teste *dataset* de coração: (81, 1)

Em seguida, treinou a função ELM desenvolvida anteriormente com 5, 10, 20, 50, 100 e 300 neurônios. Calculamos a média da acurácia e o desvio padrão em cima dos dados de teste para 5 execuções de treinamento do algoritmo. Os resultados se encontram abaixo para cada *dataset*.

Acurácia câncer com 5 neurônios: 0.89 ± 0.03 %
Acurácia câncer com 10 neurônios: 0.93 ± 0.03 %
Acurácia câncer com 30 neurônios: 0.97 ± 0.01 %
Acurácia câncer com 50 neurônios: 0.98 ± 0.0 %
Acurácia câncer com 100 neurônios: 0.97 ± 0.01 %
Acurácia câncer com 300 neurônios: 0.86 ± 0.02 %

Acurácia coração com 5 neurônios: 0.67 ± 0.0 %
Acurácia coração com 10 neurônios: 0.77 ± 0.0 %
Acurácia coração com 30 neurônios: 0.8 ± 0.0 %
Acurácia coração com 50 neurônios: 0.75 ± 0.0 %
Acurácia coração com 100 neurônios: 0.72 ± 0.0 %
Acurácia coração com 300 neurônios: 0.62 ± 0.0 %

Podemos ver que, para a base de dados de câncer de mama o melhor número de neurônios é 50, pois apresenta melhor acurácia e menor desvio padrão, ou seja, os resultados são bem consistentes. Para a base de dados Heart, todos os resultados são consistentes entre si, porém apresentam uma acurácia mais baixa que para base de dados de câncer, o que é de se esperar, pois há um maior número de atributos e entidades na base de dados de câncer que na do coração.

Outro fenômeno que se pode observar é o *overfitting* que ocorre quando começa a se aumentar demais a quantidade de neurônios na modelo ELM.

Em seguida, se dividiu os dados para treinar o *perceptron* da mesma forma que fizemos com as ELMs resultando nas mesmas dimensionalidades que anteriormente. Calculamos a acurácia e o desvio padrão para 5 execuções de treino do *perceptron* tanto para o *dataset* de cancer, quanto para o do coração. Os resultados estão dispostos a seguir.

Acurácia câncer para perceptron: 0.98 ± 0.0 %
Acurácia coração para perceptron: 0.78 ± 0.02 %

Podemos concluir que ambas as formas de classificação produzem resultados semelhantes. Cada método tem suas vantagens e desvantagens: por um lado, as ELMs não precisam de um processo iterativo para chegar em uma solução, mas, por outro lado, para o caso dos *perceptrons* não é necessário ficar calibrando o número de neurônios a serem usados. Além disso, as operações feitas pelo *perceptron* são computacionalmente menos custosas que das ELMs, onde é necessário calcular a pseudo-inversa de uma matriz. O código utilizado para obter esse resultado, pode ser verificado no apêndice a seguir.

Apêndice

```
import pandas as pd
import numpy as np
import func_rna as func

# Utilizando o módulo Pandas, lemos os datasets de cancer e heart disponibilizados no relatório

colnames = list(map(str, range(1, 31)))
colnames.insert(0, 'Diagnosis')
colnames_heart = list(map(str, range(1, 14)))
colnames_heart.append('Diagnosis')
breast_cancer = pd.read_csv('wdbc.data', index_col=0, names = colnames, header = None)
heart = pd.read_csv('heart.dat', header = None, delim_whitespace=True, names = colnames_heart, index_col = False)

# Fazemos a normalização dos features dos datasets

breast_cancer_x = breast_cancer.iloc[:,1:31]
breast_cancer_x = (breast_cancer_x-breast_cancer_x.min())/(breast_cancer_x.max()-breast_cancer_x.min())
breast_cancer.iloc[:,1:31] = breast_cancer_x

heart_x = heart.iloc[:,0:13]
heart_x = (heart_x-heart_x.min())/(heart_x.max()-heart_x.min())
heart.iloc[:,0:13] = heart_x

# Visualizando o dataset

print(breast_cancer.head())

# Visualizando o dataset Heart

print(heart.head())

# Covertemos os dados do dataset de Cancer de pandas dataframes para arrays de Numpy

X_cancer = breast_cancer.iloc[:,1:31].to_numpy()
y_cancer = breast_cancer.Diagnosis.to_numpy().reshape((-1,1))

# Aqui nos vamos adequar o Y para que o algoritmo de perceptron e ELM consiga fazer a classificação corretamente

y_cancerP = np.where(y_cancer == 'M', 1, 0)
y_cancerELM = np.where(y_cancer == 'M', 1, -1)

# Fazemos o mesmo para o dataset heart

X_heart = heart.iloc[:,0:13].to_numpy()
y_heart = heart.Diagnosis.to_numpy().reshape((-1,1))

y_heartP = np.where(y_heart == 1, 1, 0)
y_heartELM = np.where(y_heart == 1, 1, -1)

print(f'X Cancer Shape: {X_cancer.shape}\nY Cancer shape: {y_cancerP.shape}')
print(f'X Heart Shape: {X_heart.shape}\nY Heart shape: {y_heartP.shape}')

# Dividimos os dados para treinar o ELM

from sklearn.model_selection import train_test_split as tts

X_cancer_train, X_cancer_test, y_cancerELM_train, y_cancerELM_test = tts(X_cancer, y_cancerELM, test_size=0.3)
X_heart_train, X_heart_test, y_heartELM_train, y_heartELM_test = tts(X_heart, y_heartELM, test_size=0.3)

print(f'X Cancer Training Shape: {X_cancer_train.shape}\nY Cancer Training shape: {y_cancerELM_train.shape}\nX Cancer Testing Shape: {X_cancer_test.shape}\nY Cancer Testing Shape: {y_cancerELM_test.shape}')
print(f'\n\nX Heart Training Shape: {X_heart_train.shape}\nY Heart Training shape: {y_heartELM_train.shape}\nX Heart
```

```
Testing Shape: {X_heart_test.shape}\nY Heart Testing Shape: {y_heartELM_test.shape}')


# Treinando a ELM para base de dados Cancer


num_neuronios = [5, 10, 30, 50, 100, 300]

for neuronio in num_neuronios:

    accuracy = []

    for i in range(5):

        W_cancer, H_cancer, Z_cancer = func.ELM_train(X_cancer_train, y_cancerELM_train, neuronio)

        y_h_cancer = func.ELM_y(X_cancer_test, W_cancer, Z_cancer)

        accuracy.append(((y_h_cancer - y_cancerELM_test)**2/4).mean())

    print(f'Accurácia CANCER com {neuronio} neuronios: {round(np.mean(accuracy), 2)} \u00B1 {round(np.std(accuracy), 2)} %')


# Treinando a ELM para base de dados Heart


for neuronio in num_neuronios:

    accuracy = []

    for i in range(5):

        W_heart, H_heart, Z_heart = func.ELM_train(X_heart_train, y_heartELM_train, neuronio)

        y_h_heart = func.ELM_y(X_heart_test, W_heart, Z_heart)

        accuracy = ((y_h_heart - y_heartELM_test)**2/4).mean()

    print(f'Accurácia HEART com {neuronio} neuronios: {round(np.mean(accuracy), 2)} \u00B1 {round(np.std(accuracy), 2)} %')


# Agora dividimos os dados para treinar o perceptron


X_cancer_train, X_cancer_test, y_cancerP_train, y_cancerP_test = tts(X_cancer, y_cancerP, test_size=0.3)

X_heart_train, X_heart_test, y_heartP_train, y_heartP_test = tts(X_heart, y_heartP, test_size=0.3)


print(f'X Cancer Training Shape: {X_cancer_train.shape}\nY Cancer Training shape: {y_cancerP_train.shape}\nX Cancer Testing Shape: {X_cancer_test.shape}\nY Cancer Testing Shape: {y_cancerP_test.shape}')

print(f'\n\nX Heart Training Shape: {X_heart_train.shape}\nY Heart Training shape: {y_heartP_train.shape}\nX Heart Testing Shape: {X_heart_test.shape}\nY Heart Testing Shape: {y_heartP_test.shape}')


# Treinando o perceptron para base de dados Cancer


accuracy = []

for i in range(5):

    wt_cancer, e_cancer = func.trainperceptron(X_cancer_train, y_cancerP_train, 0.0001, 0.0001, 100)

    y_cancer_pred = func.yperceptron(X_cancer_test, wt_cancer)

    accuracy = (y_cancer_pred == y_cancerP_test).mean()

print(f'Accurácia CANCER para perceptron: {round(np.mean(accuracy), 2)} \u00B1 {round(np.std(accuracy), 2)} %')


# Treinando o perceptron para base de dados Heart


accuracy = []

for i in range(5):

    wt_heart, e_heart = func.trainperceptron(X_heart_train, y_heartP_train, 0.0001, 0.0001, 100)

    y_heart_pred = func.yperceptron(X_heart_test, wt_heart)

    accuracy.append((y_heart_pred == y_heartP_test).mean())

print(f'Accurácia HEART para perceptron: {round(np.mean(accuracy), 2)} \u00B1 {round(np.std(accuracy), 2)} %')
```