UNIVERSIDADE FEDERAL DE MINAS GERAIS

REDES NEURAIS ARTIFICIAIS

EXERCÍCIO 8

GABRIEL SARAIVA ESPESCHIT – 2015065541

07 de out de 2020

Assim como foi feito no exercício 6, usando 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:

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

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 a função *treinaRBF* desenvolvidas para os exercícios passados. Isto é, as classificações ficaram expressas em -1/1 para serem usadas no método RBF. A função *treinaRBF* foi desenvolvida de modo a utilizar centros de raios aleatórios ou o método de k-médias.

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 RBFs. 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 RBF desenvolvida anteriormente com 2, 5 e 10 neurônios. Calculamos a média da acurácia e o desvio padrão em cima dos dados de teste para 5 execuções de treinamento do algoritmo utilzando o k-médias. Os resultados se encontram abaixo para cada *dataset*.

Acurácia câncer com 2 neurônios: 0.91 ± 0.0 %
Acurácia câncer com 5 neurônios: 0.87 ± 0.01 %
Acurácia câncer com 10 neurônios: 0.91 ± 0.05 %
Acurácia coração com 2 neurônios: 0.69 ± 0.0 %
Acurácia coração com 5 neurônios: 0.66 ± 0.01 %
Acurácia coração com 10 neurônios: 0.76 ± 0.07 %

Após isso, fez-se o treinamento da função RBF para 2, 5 e 10 neurônios. Calculamos a média da acurácia e o desvio padrão em cima dos dados de teste para 5 execuções de treinamento do algoritmo utilzando raios aleatórios. Os resultados se encontram abaixo para cada *dataset*.

Acurácia câncer com 2 neurônios: 0.63 ± 0.1 %
Acurácia câncer com 5 neurônios: 0.75 ± 0.14 %
Acurácia câncer com 10 neurônios: 0.78 ± 0.08 %
Acurácia coração com 2 neurônios: 0.57 ± 0.03 %
Acurácia coração com 5 neurônios: 0.58 ± 0.03 %
Acurácia coração com 10 neurônios: 0.58 ± 0.06 %

Ao comparar com os resultados obtidos no exercício 6 podemos concluir que ambos os métodos nos dão acurácias semelhantes. O melhor resultado obtido para o *dataset* de câncer no exercício 6 foi de 98%, usando ELM com 50 neurônios. Para o método de RBF, obtivemos um resultado de 91%. Já para o *dataset* de coração, o melhor resultado no exercício 6 foi de 78%. Para o RBF, tivemos 76% de acurácia.

Ao comparar os resultados das RBFs entre si, podemos observar que usando o método de K-médias obtemos resultados melhores que usando os raios aleatórios. O código usado nesse exercício está disposto no apêndice abaixo.

Apêndice

```
import pandas as pd
import numpy as np
import funcoes_aux as rna
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)
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
X_cancer = breast_cancer.iloc[:,1:31].to_numpy()
y_cancer = breast_cancer.Diagnosis.to_numpy().reshape((-1,1))
y_cancerELM = np.where(y_cancer == 'M', 1, -1)
X_heart = heart.iloc[:,0:13].to_numpy()
y_heart = heart.Diagnosis.to_numpy().reshape((-1,1))
y_heartELM = np.where(y_heart == 1, 1, -1)
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)
mean_acc_cancer = []
std_cancer = []
num_centros = [2, 5, 10]
for num in num_centros:
    accuracy = []
    print(num)
    for i in range(5):
        modRBF = rna.treinaRBF(X_cancer_train, y_cancerELM_train, num)
        y_h_cancer = rna.YRBF(X_cancer_test, modRBF)
        y_h_cancer = np.where(y_h_cancer < 0, 1, -1)
        accuracy.append(((y_h_cancer - y_cancerELM_test)**2/4).mean())
    mean_acc_cancer.append(np.mean(accuracy))
```

```
    std_cancer.append(np.std(accuracy))
mean_acc_heart = []
std_heart = []
for num in num_centros:
    accuracy = []
    for i in range(5):
        modRBF = rna.treinaRBF(X_heart_train, y_heartELM_train, num)
        y_h_heart = rna.YRBF(X_heart_test, modRBF)
        y_h_heart = np.where(y_h_heart < 0, 1, -1)
        accuracy.append(((y_h_heart - y_heartELM_test)**2/4).mean())
    mean_acc_heart.append(np.mean(accuracy))
    std_heart.append(np.std(accuracy))
mean_acc_cancer_rand = []
std_cancer_rand = []
num_centros = [2, 5, 10]
for num in num_centros:
    accuracy = []
    print(num)
    for i in range(5):
        modRBF = rna.treinaRBF(X_cancer_train, y_cancerELM_train, num, 'rand')
        y_h_cancer = rna.YRBF(X_cancer_test, modRBF)
        y_h_cancer = np.where(y_h_cancer < 0, 1, -1)
        accuracy.append(((y_h_cancer - y_cancerELM_test)**2/4).mean())
    mean_acc_cancer_rand.append(np.mean(accuracy))
    std_cancer_rand.append(np.std(accuracy))
mean_acc_heart_rand = []
std_heart_rand = []
for num in num_centros:
    accuracy = []
    for i in range(5):
        modRBF = rna.treinaRBF(X_heart_train, y_heartELM_train, num, 'rand')
        y_h_heart = rna.YRBF(X_heart_test, modRBF)
        y_h_heart = np.where(y_h_heart < 0, 1, -1)
        accuracy.append(((y_h_heart - y_heartELM_test)**2/4).mean())
    mean_acc_heart_rand.append(np.mean(accuracy))
    std_heart_rand.append(np.std(accuracy))
for num, mean_cancer, mean_heart, std_heart_k, std_cancer_k in zip(num_centros, mean_acc_cancer, mean_acc_heart, std_heart,
std_cancer):
    print(f'Accurácia câncer com {num} neuronios: {round(mean_cancer, 2)} \u00B1 {round(std_cancer_k, 2)} %')
    print(f'Accurácia coração com {num} neuronios: {round(mean_heart, 2)} \u00B1 {round(std_heart_k, 2)} %')
```

```
for num, mean_cancer, mean_heart, std_heart_r, std_cancer_r in zip(num_centros, mean_acc_cancer_rand, mean_acc_heart_rand, std_heart_rand, std_cancer_rand):
    print(f'Accurácia CANCER com {num} neuronios: {round(mean_cancer, 2)} \u00B1 {round(std_cancer_r, 2)} %')
    print(f'Accurácia HEART com {num} neuronios: {round(mean_heart, 2)} \u00B1 {round(std_heart_r, 2)} %')
```