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:

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)} %')