Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Engenharia Eletrônica e de Computação
Engenharia Eletrônica e de Computação
Álgebra Linear 2

TRABALHO DE ÁLGEBRA LINEAR 2

Aluna: Karen dos Anjos Arcoverde

Professor: Marcello Luiz Rodrigues de Campos

Rio de Janeiro
2021

Sumário

0 Introdução

0.1 Conteúdo

O relatório contém a explicação do código, o código final, as funções utilizadas e os resultados encontrados para cada questão passada pelo professor.

0.2 Software e linguagem

O software usado para programação foi o Spyder e a linguagem foi o Python 3.8.5.

0.3 Biblioteca/Módulo

A biblioteca e o módulo utilizados para construir o código em Python foi:

- Numpy - É usada para fazer as operações com as matrizes
- Sys - É um módulo que faz parte da biblioteca padrão do Python e foi usado o "sys.exit (0)"para sair do programa quando o usuário pedir.

0.4 Base de dados

O conjunto de dados "Iris"selecionado para o trabalho foi:

ESPÉCIES	DE	PARA
Iris-setosa	25	39
Iris-versicolor	75	89
Iris-virginica	125	139

Tabela 1: Base de dados "Iris"selecionada

1 Explicação e Código

1.1 Explicação do código

1.2 Observação

Para um melhor entendimento, é necessário olhar o código.

1.2.1 Menu

O código contém um menu principal onde o usuário pode escolher qual questão deseja saber o resultado. Caso o usuário tenha selecionado as questões 1,2 e 3, aparecerá qual tipo de Iris (Iris-Setosa, Iris-Versicolor, Iris-Virginica) deseja ver o resultado. Se tiver selecionado a questão 4, aparecerá o resultado em seguida, sem a opção de escolher o tipo de Iris. Se o usário digitar 5 ("SAIR") no Menu Principal, o programa se encerra. Caso ele digite 4 ("VOLTAR AO MENU PRINCIPAL"), quando estiver no Menu Iris, o programa volta para o menu principal e o usuário poderá escolher novamente o número da questão ou poderá sair. Um exemplo da questão 1 de como aparece para o usuário:

```
##################### MENU PRINCIPAL ##############################
Digite somente o numero da questao que voce deseja ver o resultado:
1 = Questao 1
2 = Questao 2
3 = Questao 3
4 = Questao 4
5 = SAIR
###################################################################
```

```
##################### MENU IRIS ##########################
Digite qual especie voce deseja fazer a regressao linear:
1 = Iris-Setosa
2 = Iris-Versicolor
3 = Iris Virginica
4 = VOLTAR AO MENU PRINCIPAL
#########################################################
```

1.2.2 Funções construídas

O código também contém funções construídas e que são chamadas durante o menu principal, tais como: **pegarDados**, **construir_equacao_normal**, **PLU**, **decomposicao_espectral**, **construir_svd** e **estimar_amostras**.

1.2.3 pegarDados

A função **pegarDados** extrai os dados do arquivo ".csv", de acordo com o tipo de Iris selecionado pelo usuário, percorrendo todo o banco de dados e retorna os dados em forma de matriz para ser usada em outras funções. Nessa matriz, os valores estão em forma de matriz [15x4], onde cada linha é um ID, e cada coluna um dos dados das medições das flores. Por exemplo, se o usuário escolheu a Questão 1 e Iris-Versicolor, os dados guardados em uma matriz serão referentes à Iris-Versicolor.

1.2.4 construir_equacao_normal

A variável y da coluna PetalWidthCm foi escrita em função das outras variáveis x_1,x_2,x_3 das colunas SepalLengthCm,SepalWidthCm,PetalLengthCm, respectivamente. De forma que $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ sem o termo independente. Com o termo independente: $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3 + k$.

Além disso, a equação normal é: $x^T \cdot x \cdot w = x^T \cdot y$ ($R \cdot w = p$, $R = x^T \cdot x$ e $p = x^T \cdot y$), onde w é o vetor de coeficientes, o vetor y é a coluna PetalWidthCm e a matriz x é as colunas SepalLengthCm,SepalWidthCm,PetalLengthCm (se for com o termo independente, possui uma coluna a mais que só contém valores "1"na matriz x).

A função **construir_equacao_normal** utliza a equação normal $x^T \cdot x \cdot w = x^T \cdot y$ ($R \cdot w = p$, $R = x^T \cdot x$ e $p = x^T \cdot y$) e retorna os termos R e p com e sem o termo independente, considerando a construção de uma equação normal para cada tipo de Iris. Sem o termo independente, é construída uma matriz x deletando-se a última coluna da matriz de dados, de forma que a última coluna (PetalWidthCm) seja o nosso "target", formando uma matriz [15x3]. Caso seja com o termo independente, essa última coluna é substituída por uma coluna com valores 1, formando uma matriz [15x4]. Para construir a matriz y, são deletadas as três primeiras colunas (SepalLengthCm,SepalWidthCm,PetalLengthCm) da matriz de dados, sobrando só a última coluna (PetalWidthCm), formando uma matriz [15x1].

Por fim, para encontrar R, é feita a transposta de x multiplicada por x, formando uma matriz R [3x3] sem o termo independente e [4x4] com o termo independente. Também para encontrar p, é feita a transposta de x multiplicada por y, formando uma matriz [3x1] sem o termo independente e [4x1] com

o termo independente. Essa função retornará dois R e p diferentes, sem o termo independente e com o termo independente, que poderá ser utilizada posteriormente para as outras funções.

1.2.5 PLU

A função **PLU** tem como objetivo transformar R em uma matriz triangular superior. Primeiro é feito um loop para percorrer cada coluna da matriz R, depois um loop dentro desse para percorrer as linhas de R. Sendo assim, ele fixa uma coluna em R e percorre todas as linhas de R nessa mesma coluna, depois passa para próxima coluna e faz o mesmo loop das linhas até terminar na penúltima coluna. Então, durante o loop das colunas, é verificado sempre se o pivô, que é o elemento da diagonal principal, é igual a zero. Caso não seja zero, começa a construir a matriz L (triangular inferior). Caso seja igual a zero, vai para uma condição, para criar a matriz de permutação (P). Cada vez que é construída a matriz L ou P, R e p são multiplicadas por L ou P.

A matriz L é construída de forma que a cada coluna que o loop das colunas da matriz R percorre, é criada uma matriz L de zeros do mesmo tamanho que R. Desse modo, a matriz L será preenchida de forma que os elementos abaixo do pivô da coluna fixada se tornem zeros na matriz R. Assim, o loop das linhas de R começa sempre do número da coluna fixada de R+1 de modo que o loop das linhas comece sempre na posição abaixo do pivô. E o valor na linha e na coluna que estão sendo analisadas na matriz R será colocado na mesma posição na matriz L, só que dividido pelo valor do pivô da coluna fixada na matriz R e multiplicando essa razão por -1. Depois de percorrer todas as linhas da coluna fixada, a matriz L é preenchida com valor 1 na diagonal principal. Quando for para próxima coluna de R, é criada uma nova matriz L de zeros do mesmo tamanho que R e o loop das linhas se repete.

A matriz P é construída criando uma matriz de zeros do mesmo tamanho que R. Na matriz R, temos o objetivo de trocar a linha em que o pivô de R tem valor zero por uma linha em que o valor na mesma coluna do pivô não tenha zero. Desse modo, é feito um loop para percorrer todas as linhas abaixo da coluna do pivô de R e ver qual elemento é diferente de zero. Se for achado o elemento, é feita a construção da matriz P para fazer as trocas das duas linhas e o loop é terminado.

Depois é feito um loop para percorrer cada linha de P e ver se tem algum elemento 1 na linha, caso já tenha, não é colocado 1. Se não tiver o valor 1 em qualquer elemento da linha, é colocado 1 na posição com a linha e coluna iguais ao número da linha analisada.

Posteriormente, o loop das colunas de R volta para essa mesma coluna em que as linhas foram trocadas e ela é novamente analisada para zerar os elementos abaixo do pivô da coluna fixada. Caso não tenha um valor diferente de zero abaixo do pivô nessa coluna, é passada para a próxima coluna da matriz R e ela passará pelo mesmo processo de análise.

E assim o loop continua, indo para a próxima coluna da matriz R até ter percorrido a penúltima coluna. Quando tiver terminado de percorrer até a penúltima coluna, a matriz R já terá se tornado triangular superior, é feito o backsubstitution, em que é dado R e p e a função np.linalg.solve retorna w (os coeficientes).

1.2.6 decomposicao_espectral

A função **decomposicao_espectral** calcula os autovalores e autovetores de R. Posteriormente, retorna uma matriz diagonal de autovalores de R (Λ) e uma matriz de autovetores de R (matriz V). No menu principal, quando selecionada a opção Questão 2, essa parte chamará a função decomposicao_espectral duas vezes, uma para calcular as matrizes sem o termo independente e outra com o termo independente. Essa parte usará essas matrizes encontradas e imprimirá R, V, Λ e a transposta de V (V^T) com e sem o termo independente.

1.2.7 construir_svd

A função **construir_svd** foi considerada como: U os autovetores de $R \cdot R^T$, V os autovetores de $R^T \cdot R$ e Σ a matriz diagonal formada com os valores singulares (raiz quadrada dos autovalores) em comuns em $R^T \cdot R$ e $R \cdot R^T$. Assim: $R = U \cdot \Sigma \cdot V^T$.

Como R é a matriz da equação normal, formada assim: $R = x^T \cdot x$. Nesse sentido, R é simétrica, já que $R^T = R$, uma vez que $(x^T \cdot x)^T = x^T \cdot x$. Por consequência, $R^T \cdot R = R \cdot R^T$, $R^T \cdot R$ possuirá os mesmos autovalores e autovetores de $R \cdot R^T$ e U será igual a V. Dessa forma, só foi calculado os autovetores e autovalores de $R \cdot R^T$.

Assim, essa função calcula os autovalores e autovetores (matriz U) de $R \cdot R^T$, depois faz um loop, para descobrir os valores singulares, então percorre cada autovalor, tira a raiz quadrada e coloca em um vetor. Essa função retornará os autovetores de $R \cdot R^T$ (matriz U), um vetor com os valores singulares de $R \cdot R^T$ e a transposta de U (U^T). Já no menu principal, quando selecionada a opção Questão 3, essa parte chamará a função construir_svd duas vezes, uma para calcular as matrizes e os vetores sem o termo independente e outra com o termo independente. Essa parte usará essas matrizes e vetores encontrados e imprimirá R, U, uma matriz diagonal de valores singulares (Σ) e a transposta de U (Lembrando que a transposta de V é igual a transposta de U, já que $U = V$) com e sem o termo independente.

1.2.8 estimar_amostras

A função **estimar_amostras** terá o propósito de fazer o produto interno do vetor de amostra selecionada pelo professor, excluindo a coluna PetalWidthCm, com cada vetor de coeficientes da Iris-Setosa, Iris-Versicolor e Iris-Virginica. Caso seja com o termo independente, terá mais um valor no vetor de amostra selecionado, com o número 1. Uma vez que a coluna PetalWidthCm será a selecionada para estimar a classe da Iris.

Depois de fazer o produto interno, será calculada a diferença (erro) do resultado de cada produto interno com o valor da PetalWidthCm de da amostra e colocada numa lista que posteriormente será percorrida para cada erro ser transformado em seu módulo.

Por fim, é usada uma função que pega o menor valor dessa lista. Esse menor valor da lista é comparado com o módulo de cada erro da Iris-Setosa, Iris-Versicolor e Iris-Virginica, quando a comparação der igual, é retornada uma string com o tipo de Iris que deu igual. Por exemplo, se a comparação deu igual na Iris-Virginica, retornará uma string "Iris-Virginica".

No Menu Principal, a função estimar_amostras é chamada duas vezes para cada amostra, uma para calcular sem o termo independente e imprimir a classe estimada e outra com o termo independente. Por isso existe uma variável booleana que é True quando quer calcular com o termo independente e False quando é sem o termo independente. De forma que se essa variável for True, ela adiciona o valor 1 nos vetores de amostras.

1.3 Código

```
1  # Programa codigo.py
2  # Autora: Karen dos Anjos Arcoverde
3  # Data: 06/02/2021
4  #
5
6
7  import numpy as np
8  import sys
9
10
```

```
11
12 ############################## Funcoes ##############################
13 def pegarDados(tipo_iris):
14
15     # tipo_iris = 1 Setosa  , tipo_iris = 2  Versicolor,
16     # tipo_iris = 3 Virginica
17     dados = []
18     IDs = []
19
20     ## definicao dos ids das especies selecionadas para o trabalho
21     Setosa = range(25,39+1)
22     Versicolor = range(75,89+1)
23     Virginica = range (125,139+1)
24
25     arquivo= open("dados_13.csv",'r')
26     arquivo.readline() # ignora a primeira linha
27
28     if (tipo_iris == 1):
29         IDs = Setosa
30     if (tipo_iris == 2):
31         IDs = Versicolor
32     if (tipo_iris == 3):
33         IDs = Virginica
34
35     for i in range(1,46): # percorre todo o banco de dados 1-45
36         linha = (arquivo.readline()).split(',') #separa os dados por virgula
37
38         if  (int(linha[0]) in IDs):# percorre os ids selecionados
39             linha.pop(0) # retira o ID dos dados
40             linha.pop(-1) # retira a especie dos dados
41             for j in range(4): #para cada dado
42                 linha[j] = float(linha[j]) #transforma em numero
43
44             dados.append(linha) #adiciona na lista de dados
45
46     arquivo.close()
47
48     return dados
49
50 # ------------------------------------------------------------
51 def construir_equacao_normal (dados):
52
53     #equacao normal - minimos quadrados:
54     # (x^T).x.w = (x^T).y
55     #(x_transposta).x.w =(x_transposta).y
56     # R.w = p
57     # R = (x^T).x
58     # p = (x^T).y
59
60     ###### sem termo independente #####
61     # y = a*x1 +b*x2 +c*x3
62     x = []
```

```
    y = []

    #achar (x^T) e x
    # x - colunas SepalLengthCm,SepalWidthCm,PetalLengthCm
    x = np.array(dados)
    x = np.delete(x.reshape(15,4),3,1) #deleta a ultima coluna de x
    x_transposta = np.transpose(x) #faz a transposta de x: (x^T)

    #achar R
    R = np.dot(x_transposta,x) #multiplica (x^T) por x

    #achar y - coluna PetalWidthCm
    y = np.array(dados)
    y = np.delete(y.reshape(15,4),0,1) #deleta a primeira coluna de y
    y = np.delete(y.reshape(15,3),0,1) #deleta a segunda coluna de y
    y = np.delete(y.reshape(15,2),0,1) #deleta a terceira coluna de y

    #achar p
    p = np.dot(x_transposta,y)

    ###### com termo independente #####
    # y = a*x1 +b*x2 +c*x3 +k

    for i in range (0,15):
        dados[i][3] = 1

    #achar (x^T) e x
    x = np.array(dados)
    x_transposta = np.transpose(x) #faz a transposta de x: (x^T)
    #achar R
    R1= np.dot(x_transposta,x) #multiplica (x^T) por x
    #achar p
    p1 = np.dot(x_transposta,y)

    return R,p,R1,p1

# ------------------------------------------------------------
def PLU(R,p):
    indice_coluna = 0
    indice_L_1s = 0
    indice_escolher_linha = 0
    tamanho_R = len(R)

    while (indice_coluna < (tamanho_R-1)):
        #constroi a matriz L, triangular inferior
        if (R[indice_coluna][indice_coluna] != 0):
            L = np.zeros((tamanho_R, tamanho_R))

            indice_coluna_aux = indice_coluna + 1
            while (indice_coluna_aux < tamanho_R):
                L[indice_coluna_aux][indice_coluna] = - R[indice_coluna_aux][
                    indice_coluna]/R[indice_coluna][indice_coluna]
```

```
                indice_coluna_aux += 1

            while (indice_L_1s < tamanho_R):
                L[indice_L_1s][indice_L_1s] = 1
                indice_L_1s += 1

            indice_L_1s = 0

            #multiplica a matriz R por L
            R = np.dot (L,R)
            p = np.dot (L,p)


        #se o pivo for zero, necessario multiplicar por uma matriz P de
           permutacao
        if (R[indice_coluna][indice_coluna] == 0):
            P = np.zeros((tamanho_R, tamanho_R))
            indice_escolher_linha = indice_coluna + 1
            while (indice_escolher_linha < tamanho_R):
                if (R[indice_escolher_linha][indice_coluna] != 0):
                    P[indice_coluna][indice_escolher_linha] = 1
                    P[indice_escolher_linha][indice_coluna] = 1

                    indice_coluna -= 1
                    break

                indice_escolher_linha += 1

            i = 0
            j = 0
            guarda_1 = False
            while (i < tamanho_R):
                while (j < tamanho_R):
                    if (P[i][j] == 1):
                        guarda_1 = True
                        j += 1
                if (guarda_1 == False):
                    P[i][i] = 1
                i += 1

            R = np.dot(P,R)
            p = np.dot (P,p)


        indice_coluna += 1

    #backsubstitution
    w = np.linalg.solve(R, p)

    return w

# ------------------------------------------------------------
```

```
165 def decomposicao_espectral(R):
166
167     # R = VDV^(T)
168     #determinando autovalores e autovetores
169     autovalores, autovetores = np.linalg.eig(R)
170     # matriz diagonal de autovalores
171     matrizDiagonal = np.diag(autovalores)
172
173     return  autovetores, matrizDiagonal
174
175 # ------------------------------------------------------------
176 def contruir_svd (R):
177     s = []
178     i = 0
179
180     # R = U.s.VT
181     #determinando autovalores e autovetores
182     ### U
183     ## autovetores de R.(R^T)
184     autovaloresU, autovetoresU = np.linalg.eig(np.dot(R,np.transpose(R)))
185
186     # como R eh simetrica e quadrada: (R^T).R = R.(R^T)
187     # entao autovetores e autovalores de U sao iguais aos de V
188
189     tamanho_autovalores_U = autovaloresU.shape
190
191     while (i < tamanho_autovalores_U [0]):
192         s.append(np.sqrt(autovaloresU [i]))
193         i += 1
194
195     return  autovetoresU, np.array(s),np.transpose(autovetoresU)
196
197 # ------------------------------------------------------------
198 def estimar_amostras(amostra, w_setosa,w_versicolor,w_virginica,c_independente
    ):
199
200     lista_erros = []
201     estimativa = ""
202     indice = 0
203     amostra_x = []
204     indice_erros_modulo = 0
205
206     while (indice < 3):
207         amostra_x.append (amostra[indice])
208         indice += 1
209
210     if (c_independente == True):
211         amostra_x.append(1)
212
213     amostra_x = np.array(amostra_x)
214
215     #produto interno <x,y> = (x^T).y
```

```
216
217     estimativa_setosa = np.dot(amostra_x,w_setosa)
218     estimativa_versicolor = np.dot(amostra_x,w_versicolor)
219     estimativa_virginica = np.dot(amostra_x,w_virginica)
220
221     erro_setosa = estimativa_setosa[0] - amostra[3]
222     lista_erros.append(erro_setosa)
223
224     erro_versicolor = estimativa_versicolor[0] - amostra[3]
225     lista_erros.append(erro_versicolor)
226
227     erro_virginica = estimativa_virginica[0] - amostra[3]
228     lista_erros.append(erro_virginica)
229
230     while (indice_erros_modulo < len(lista_erros)):
231         lista_erros [indice_erros_modulo] = abs(lista_erros[indice_erros_modulo
              ])
232         indice_erros_modulo += 1
233
234     if (min(lista_erros) == abs(erro_setosa)):
235         estimativa = "Iris-Setosa"
236
237         return estimativa
238
239     if (min(lista_erros) == abs(erro_versicolor)):
240         estimativa = "Iris-Versicolor"
241
242         return estimativa
243
244     if (min(lista_erros) == abs(erro_virginica)):
245         estimativa = "Iris-Virginica"
246
247         return estimativa
248
249 ################### Programa Principal ###################
250 def menu():
251     resultado = 0
252     tipo_iris = 0
253     coeficientes_sem_aux = []
254     coeficientes_com_aux = []
255     indice = 0
256
257
258     while (resultado != 5):
259         print()
260         print('###################### MENU PRINCIPAL
              ##############################')
261         print("Digite somente o numero da questao que voce deseja ver o
              resultado: ")
262         print("1 = Questao 1")
263         print("2 = Questao 2")
264         print("3 = Questao 3")
```

```
265         print("4 = Questao 4")
266         print("5 = SAIR")
267         print('
               ##################################################################
               ')
268         print()
269 
270         resultado = int(input())
271 
272         if (resultado == 5):
273             sys.exit(0)
274 
275         if (resultado == 1):
276 
277             while (tipo_iris != 4):
278                 print('###################### MENU IRIS
                       ##########################')
279                 print("Digite qual especie voce deseja fazer a regressao
                       linear: ")
280                 print("1 = Iris-Setosa")
281                 print("2 = Iris-Versicolor")
282                 print("3 = Iris Virginica")
283                 print("4 = VOLTAR AO MENU PRINCIPAL")
284                 print('
                       ##########################################################')
285                 print()
286 
287                 tipo_iris = int(input())
288 
289                 if (tipo_iris == 4):
290                     menu()
291 
292                 else:
293                     dados = pegarDados (tipo_iris)
294                     R,p,R1,p1 = construir_equacao_normal(dados)
295                     w = PLU(R,p)
296                     w1 = PLU(R1,p1)
297 
298                     print()
299                     if  (tipo_iris == 1):
300                         print("Iris-Setosa\n")
301                     elif (tipo_iris == 2):
302                         print("Iris-Versicolor\n")
303                     elif (tipo_iris == 3):
304                         print("Iris-Virginica\n")
305 
306                     print("SEM O TERMO INDEPENDENTE: ")
307                     print("y = a*x1 + b*x2 + c*x3")
308                     print("[a b c] = ",end="")
309 
310 
311                     while (indice < len(w)):
```

```
                    coeficientes_sem_aux.append(w[indice][0])
                    coeficientes_sem = np.array(coeficientes_sem_aux)
                    indice += 1

                print(coeficientes_sem)
                coeficientes_sem_aux = []


                print()

                print("COM O TERMO INDEPENDENTE: ")
                print("y = a*x1 + b*x2 + c*x3 + k")
                print("[a b c k] = ",end="")

                indice = 0
                while (indice < len(w1)):
                    coeficientes_com_aux.append(w1[indice][0])
                    coeficientes_com = np.array(coeficientes_com_aux)
                    indice += 1

                print(coeficientes_com)
                coeficientes_com_aux = []
                indice = 0

                print()

    if (resultado == 2):

        while (tipo_iris != 4):
            print('###################### MENU IRIS
               ##############################')
            print("Digite qual especie voce deseja fazer a decomposicao
               espectral: ")
            print("1 = Iris-Setosa")
            print("2 = Iris-Versicolor")
            print("3 = Iris Virginica")
            print("4 = VOLTAR AO MENU PRINCIPAL")
            print('
               ############################################################
               ')
            print()


            tipo_iris = int(input())

            if (tipo_iris == 4):
               menu()

            else:
                dados = pegarDados (tipo_iris)
                R,p,R1,p1 = construir_equacao_normal(dados)
                autovetores, matrizDiagonal = decomposicao_espectral(R)
```

```
                    autovetores1, matrizDiagonal1 = decomposicao_espectral(R1)

                    if  (tipo_iris == 1):
                        print("Iris-Setosa\n")
                    elif (tipo_iris == 2):
                        print("Iris-Versicolor\n")
                    elif (tipo_iris == 3):
                        print("Iris-Virginica\n")

                    print("R = V\u039BV^(T)")
                    print()


                    print("SEM O TERMO INDEPENDENTE: ")
                    print("R = ", R)
                    print()
                    print("V = ",autovetores)
                    print()
                    print('\u039B = ',matrizDiagonal)
                    print()
                    print("V^T = ",np.transpose(autovetores))

                    print()
                    print()

                    print("COM O TERMO INDEPENDENTE: ")
                    print("R = ", R1)
                    print()
                    print("V = ", autovetores1)
                    print()
                    print('\u039B = ', matrizDiagonal1)
                    print()
                    print("V^T = ", np.transpose(autovetores1))

                    print()

        if (resultado == 3):

            while (tipo_iris != 4):
                print('###################### MENU IRIS #############')
                print("Digite qual especie voce deseja fazer o SVD: ")
                print("1 = Iris-Setosa")
                print("2 = Iris-Versicolor")
                print("3 = Iris Virginica")
                print("4 = VOLTAR AO MENU PRINCIPAL")
                print('##############################################')
                print()

                tipo_iris = int(input())

                if (tipo_iris == 4):
                    menu()
```

```
412
413             else:
414
415                 dados = pegarDados (tipo_iris)
416                 R,p,R1,p1 = construir_equacao_normal(dados)
417                 U, s, VT = contruir_svd(R)
418                 U1, s1, VT1 = contruir_svd(R1)
419
420                 if  (tipo_iris == 1):
421                      print("Iris-Setosa\n")
422                 elif (tipo_iris == 2):
423                      print("Iris-Versicolor\n")
424                 elif (tipo_iris == 3):
425                      print("Iris-Virginica\n")
426
427                 print("R = U\u03A3V^(T)")
428                 print()
429
430                 print("SEM O TERMO INDEPENDENTE: ")
431                 print("R = ", R)
432                 print()
433                 print("U = ", U)
434                 print()
435                 print('\u03A3 = ', np.diag(s))
436                 print()
437                 print("V^T = ", VT)
438                 print()
439                 print()
440
441                 print("COM O TERMO INDEPENDENTE: ")
442                 print("R = ", R1)
443                 print()
444                 print("U = ", U1)
445                 print()
446                 print('\u03A3 = ', np.diag(s1))
447                 print()
448                 print("V^T = ", VT1)
449
450                 print()
451
452
453
454         if (resultado == 4):
455             A = [5.0,2.3,3.3,1.0]
456             B = [4.6,3.2,1.4,0.2]
457             C = [5.0,3.3,1.4,0.2]
458             D = [6.1,3.0,4.6,1.4]
459             E = [5.9,3.0,5.1,1.8]
460
461             print("SEM O TERMO INDEPENDENTE: ")
462             dados = pegarDados (1)
463             R,p,R1,p1 = construir_equacao_normal(dados)
```

```
464             w_setosa = PLU(R,p)
465
466             dados = pegarDados (2)
467             R,p,R1,p1 = construir_equacao_normal(dados)
468             w_versicolor = PLU(R,p)
469
470             dados = pegarDados (3)
471             R,p,R1,p1 = construir_equacao_normal(dados)
472             w_virginica = PLU(R,p)
473
474             c_independente = False
475             estimativa = estimar_amostras(A,w_setosa,w_versicolor,w_virginica,
                    c_independente)
476             print("A = ", estimativa)
477             estimativa = estimar_amostras(B,w_setosa,w_versicolor,w_virginica,
                    c_independente)
478             print("B = ", estimativa)
479             estimativa = estimar_amostras(C,w_setosa,w_versicolor,w_virginica,
                    c_independente)
480             print("C = ", estimativa)
481             estimativa = estimar_amostras(D,w_setosa,w_versicolor,w_virginica,
                    c_independente)
482             print("D = ", estimativa)
483             estimativa = estimar_amostras(E,w_setosa,w_versicolor,w_virginica,
                    c_independente)
484             print("E = ", estimativa)
485
486             print()
487             print("COM O TERMO INDEPENDENTE: ")
488             dados = pegarDados (1)
489             R,p,R1,p1 = construir_equacao_normal(dados)
490             w1_setosa = PLU(R1,p1)
491
492             dados = pegarDados (2)
493             R,p,R1,p1 = construir_equacao_normal(dados)
494             w1_versicolor = PLU(R1,p1)
495
496             dados = pegarDados (3)
497             R,p,R1,p1 = construir_equacao_normal(dados)
498             w1_virginica = PLU(R1,p1)
499
500             c_independente = True
501             estimativa1 = estimar_amostras(A,w1_setosa,w1_versicolor,
                    w1_virginica,c_independente)
502             print("A = ", estimativa1)
503             estimativa1 = estimar_amostras(B,w1_setosa,w1_versicolor,
                    w1_virginica,c_independente)
504             print("B = ", estimativa1)
505             estimativa1 = estimar_amostras(C,w1_setosa,w1_versicolor,
                    w1_virginica,c_independente)
506             print("C = ", estimativa1)
507             estimativa1 = estimar_amostras(D,w1_setosa,w1_versicolor,
```

```
                w1_virginica,c_independente)
            print("D = ", estimativa1)
            estimativa1 = estimar_amostras(E,w1_setosa,w1_versicolor,
                w1_virginica,c_independente)
            print("E = ", estimativa1)

            menu()

######## chamada ao menu
menu()
```

2 Questão 1

2.1 Funções utilizadas

Para essa parte foram utilizadas as funções: menu, pegarDados, construir_equacao_normal e PLU.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal e com o R e p retornados da equação normal, aplicar PLU e backsubstitution para descobrir os coeficientes da equação normal.

2.2 Resultados

2.2.1 Iris-Setosa

```
Iris-Setosa

SEM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3
[a b c] = [ 0.07455282 -0.06602361  0.03673264]

COM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3 + k
[a b c k] = [ 0.13497209 -0.07886596  0.10365958 -0.36144997]
```

2.2.2 Iris-Versicolor

```
Iris-Versicolor

SEM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3
[a b c] = [-0.14382683  0.17051618  0.40397714]

COM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3 + k
[a b c k] = [-0.06826027  0.24149193  0.39877688 -0.63863516]
```

2.2.3 Iris-Virginica

```
Iris-Virginica

SEM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3
[a b c] = [-0.11329721  0.3990124   0.25976859]

COM O TERMO INDEPENDENTE:
y = a*x1 + b*x2 + c*x3 + k
[a b c k] = [-0.11686468  0.35346044  0.22325742  0.36734017]
```

3 Questão 2

3.1 Funções utilizadas

Para essa parte foram utilizadas as funções: menu, pegarDados, construir_equacao_normal e decomposicao_espectral.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal e utilizar o R retornado da equação normal para fazer a decomposição espectral.

3.2 Resultados

3.2.1 Iris-Setosa

```
Iris-Setosa

R = VΛV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57]
 [255.73 172.5   75.51]
 [112.57  75.51  33.84]]

V =  [[-0.80618188 -0.45661854  0.37625828]
 [-0.54157827  0.31342349 -0.78003762]
 [-0.23825146  0.8326255   0.49997102]]

Λ =  [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.29776824e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.47589571e-01]]

V^T =  [[-0.80618188 -0.54157827 -0.23825146]
 [-0.45661854  0.31342349  0.8326255 ]
 [ 0.37625828 -0.78003762  0.49997102]]


COM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57  75.5 ]
 [255.73 172.5   75.51  50.6 ]
 [112.57  75.51  33.84  22.4 ]
 [ 75.5   50.6   22.4   15.  ]]

```

```
V =  [[-0.79610972  0.1628056  -0.47792329 -0.33360603]
 [-0.53478061 -0.03864172  0.33762985  0.77364243]
 [-0.23529777  0.18572124  0.80860497 -0.50626137]
 [-0.15765144 -0.96825037  0.06126507 -0.18407563]]

Λ =  [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 2.79352229e-02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 7.73257486e-01]]

V^T =  [[-0.79610972 -0.53478061 -0.23529777 -0.15765144]
 [ 0.1628056  -0.03864172  0.18572124 -0.96825037]
 [-0.47792329  0.33762985  0.80860497  0.06126507]
 [-0.33360603  0.77364243 -0.50626137 -0.18407563]]
```

3.2.2 Iris-Versicolor

```
Iris-Versicolor

R = VΛV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59]
 [256.64 119.98 184.35]
 [397.59 184.35 286.5 ]]

V =  [[ 0.76039163  0.64759888 -0.0491961 ]
 [ 0.35223975 -0.47485674 -0.80649751]
 [ 0.54564799 -0.59592513  0.58918716]]

Λ =  [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.33162776e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 9.57976568e-01]]

V^T =  [[ 0.76039163  0.35223975  0.54564799]
 [ 0.64759888 -0.47485674 -0.59592513]
 [-0.0491961  -0.80649751  0.58918716]]


COM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59  91.  ]
 [256.64 119.98 184.35  42.2 ]
 [397.59 184.35 286.5   65.2 ]
 [ 91.    42.2   65.2   15.  ]]

V =  [[-0.7545525   0.11812875  0.64474223  0.03167953]
 [-0.34954066  0.11526607 -0.46962259  0.80248968]
 [-0.54144512 -0.01367345 -0.60234554 -0.58637025]
 [-0.1237297  -0.98619084  0.030691    0.1057197 ]]

Λ =  [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 7.14132919e-02 0.00000000e+00 0.00000000e+00]
```

```
 [0.00000000e+00 0.00000000e+00 1.33280153e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]

V^T =  [[-0.7545525  -0.34954066 -0.54144512 -0.1237297 ]
 [ 0.11812875  0.11526607 -0.01367345 -0.98619084]
 [ 0.64474223 -0.46962259 -0.60234554  0.030691  ]
 [ 0.03167953  0.80248968 -0.58637025  0.1057197 ]]
```

3.2.3 Iris-Virginica

```
Iris-Virginica

R = VΛV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72]
 [304.85 138.7  253.94]
 [562.72 253.94 469.3 ]]

V =  [[-0.72592866 -0.63483621  0.2645951 ]
 [-0.32772296 -0.01894783 -0.94458385]
 [-0.60466953  0.77241438  0.19429563]]

Λ =  [[1.28249881e+03 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.79253032e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.07193335e+00]]

V^T =  [[-0.72592866 -0.32772296 -0.60466953]
 [-0.63483621 -0.01894783  0.77241438]
 [ 0.2645951  -0.94458385  0.19429563]]


COM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72 100.3 ]
 [304.85 138.7  253.94  45.4 ]
 [562.72 253.94 469.3   83.6 ]
 [100.3   45.4   83.6   15.  ]]

V =  [[-7.21743001e-01 -2.76297507e-01  6.34623278e-01 -1.50933108e-04]
 [-3.25843186e-01  9.35769444e-01  3.68031853e-02 -1.29642937e-01]
 [-6.01187515e-01 -1.93722166e-01 -7.68083597e-01 -1.05322749e-01]
 [-1.07176625e-01  1.02308151e-01 -7.71129602e-02  9.85951218e-01]]

Λ =  [[1.29740071e+03 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.08243546e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.82311582e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 8.45463191e-02]]

V^T =  [[-7.21743001e-01 -3.25843186e-01 -6.01187515e-01 -1.07176625e-01]
 [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
 [ 6.34623278e-01  3.68031853e-02 -7.68083597e-01 -7.71129602e-02]
 [-1.50933108e-04 -1.29642937e-01 -1.05322749e-01  9.85951218e-01]]
```

4 Questão 3

4.1 Funções utilizadas

Para essa parte foram utilizadas as funções: menu, PegarDados, construir_equacao_normal e construir_svd.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal e utilizar o R retornado da equação normal para fazer o SVD.

4.2 Resultados

4.2.1 Iris-Setosa

```
Iris-Setosa

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57]
 [255.73 172.5   75.51]
 [112.57  75.51  33.84]]

U =  [[ 0.80618188 -0.45661854 -0.37625828]
 [ 0.54157827  0.31342349  0.78003762]
 [ 0.23825146  0.8326255  -0.49997102]]

Σ =  [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.29776824e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.47589571e-01]]

V^T =  [[ 0.80618188  0.54157827  0.23825146]
 [-0.45661854  0.31342349  0.8326255 ]
 [-0.37625828  0.78003762 -0.49997102]]


COM O TERMO INDEPENDENTE:
R =  [[381.33 255.73 112.57  75.5 ]
 [255.73 172.5   75.51  50.6 ]
 [112.57  75.51  33.84  22.4 ]
 [ 75.5   50.6   22.4   15.  ]]

U =  [[ 0.79610972 -0.33360603 -0.47792329 -0.1628056 ]
 [ 0.53478061  0.77364243  0.33762985  0.03864172]
 [ 0.23529777 -0.50626137  0.80860497 -0.18572124]
 [ 0.15765144 -0.18407563  0.06126507  0.96825037]]

Σ =  [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 7.73257486e-01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 2.79352230e-02]]

V^T =  [[ 0.79610972  0.53478061  0.23529777  0.15765144]
```

```
 [-0.33360603  0.77364243 -0.50626137 -0.18407563]
 [-0.47792329  0.33762985  0.80860497  0.06126507]
 [-0.1628056   0.03864172 -0.18572124  0.96825037]]
```

4.2.2 Iris-Versicolor

```
Iris-Versicolor

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59]
 [256.64 119.98 184.35]
 [397.59 184.35 286.5 ]]

U =  [[ 0.76039163  0.64759888 -0.0491961 ]
 [ 0.35223975 -0.47485674 -0.80649751]
 [ 0.54564799 -0.59592513  0.58918716]]

Σ =  [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.33162776e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 9.57976568e-01]]

V^T =  [[ 0.76039163  0.35223975  0.54564799]
 [ 0.64759888 -0.47485674 -0.59592513]
 [-0.0491961  -0.80649751  0.58918716]]


COM O TERMO INDEPENDENTE:
R =  [[555.38 256.64 397.59  91.  ]
 [256.64 119.98 184.35  42.2 ]
 [397.59 184.35 286.5   65.2 ]
 [ 91.    42.2   65.2   15.  ]]

U =  [[ 0.7545525  -0.64474223 -0.11812875  0.03167953]
 [ 0.34954066  0.46962259 -0.11526607  0.80248968]
 [ 0.54144512  0.60234554  0.01367345 -0.58637025]
 [ 0.1237297  -0.030691    0.98619084  0.1057197 ]]

Σ =  [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.33280153e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.14132919e-02 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]

V^T =  [[ 0.7545525   0.34954066  0.54144512  0.1237297 ]
 [-0.64474223  0.46962259  0.60234554 -0.030691  ]
 [-0.11812875 -0.11526607  0.01367345  0.98619084]
 [ 0.03167953  0.80248968 -0.58637025  0.1057197 ]]
```

4.2.3 Iris-Virginica

```
Iris-Virginica

R = UΣV^(T)

SEM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72]
 [304.85 138.7  253.94]
 [562.72 253.94 469.3 ]]

U =  [[ 0.72592866 -0.63483621 -0.2645951 ]
 [ 0.32772296 -0.01894783  0.94458385]
 [ 0.60466953  0.77241438 -0.19429563]]

Σ =  [[1.28249881e+03 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 5.79253032e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.07193335e+00]]

V^T =  [[ 0.72592866  0.32772296  0.60466953]
 [-0.63483621 -0.01894783  0.77241438]
 [-0.2645951   0.94458385 -0.19429563]]


COM O TERMO INDEPENDENTE:
R =  [[676.15 304.85 562.72 100.3 ]
 [304.85 138.7  253.94  45.4 ]
 [562.72 253.94 469.3   83.6 ]
 [100.3   45.4   83.6   15.  ]]

U =  [[ 7.21743001e-01 -2.76297507e-01 -6.34623278e-01 -1.50933128e-04]
 [ 3.25843186e-01  9.35769444e-01 -3.68031855e-02 -1.29642937e-01]
 [ 6.01187515e-01 -1.93722166e-01  7.68083597e-01 -1.05322749e-01]
 [ 1.07176625e-01  1.02308151e-01  7.71129601e-02  9.85951218e-01]]

Σ =  [[1.29740071e+03 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.08243546e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.82311582e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 8.45463191e-02]]

V^T =  [[ 7.21743001e-01  3.25843186e-01  6.01187515e-01  1.07176625e-01]
 [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
 [-6.34623278e-01 -3.68031855e-02  7.68083597e-01  7.71129601e-02]
 [-1.50933128e-04 -1.29642937e-01 -1.05322749e-01  9.85951218e-01]]
```

5 Questão 4

5.1 Funções utilizadas

Para essa parte foram utilizadas as funções: menu, pegarDados, construir_equacao_normal, PLU e estimar_amostras.

Foi preciso pegar os dados do arquivo fornecido, depois descobrir a equação normal, utilizar o R e p retornados da equação normal para aplicar PLU e backsubstitution para descobrir os coeficientes da

equação normal e estimar as amostras fornecidas utilizando os coeficientes achados.

5.2 Resultados

```
SEM O TERMO INDEPENDENTE:
A =  Iris-Versicolor
B =  Iris-Setosa
C =  Iris-Setosa
D =  Iris-Versicolor
E =  Iris-Virginica

COM O TERMO INDEPENDENTE:
A =  Iris-Versicolor
B =  Iris-Setosa
C =  Iris-Setosa
D =  Iris-Versicolor
E =  Iris-Virginica
```

6 Bibliografia

- https://algebralinearufcg.github.io/jup-not/prog02-learning-numpy.html
- https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/
- https://pt.coredump.biz/questions/34007632/how-to-remove-a-column-in-a-numpy-array
- https://pythonforundergradengineers.com/unicode-characters-in-python.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.svd.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.diagsvd.html#scipy.linalg.diagsvd
- https://www.geeksforgeeks.org/python-exit-commands-quit-exit-sys-exit-and-os-_exit/