

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	3
0.1	Conteúdo	3
0.2	Software e linguagem	3
0.3	Biblioteca/Módulo	3
0.4	Base de dados	3
1	Explicação e Código	3
1.1	Explicação do código	3
1.1.1	Menu	3
1.1.2	Funções construídas	4
1.1.3	pegarDados	4
1.1.4	construir_equacao_normal	4
1.1.5	PLU	4
1.1.6	decomposicao_espectral	5
1.1.7	construir_svd	5
1.1.8	estimar_amstras	6
1.2	Código	6
2	Questão 1	16
2.1	Funções utilizadas	16
2.2	Resultados	17
2.2.1	Iris-Setosa	17
2.2.2	Iris-Versicolor	17
2.2.3	Iris-Virginica	17
3	Questão 2	17
3.1	Funções utilizadas	17
3.2	Resultados	18
3.2.1	Iris-Setosa	18
3.2.2	Iris-Versicolor	18
3.2.3	Iris-Virginica	19
4	Questão 3	20
4.1	Funções utilizadas	20
4.2	Resultados	20
4.2.1	Iris-Setosa	20
4.2.2	Iris-Versicolor	21
4.2.3	Iris-Virginica	22
5	Questão 4	23
5.1	Funções utilizadas	23
5.2	Resultados	23
6	Bibliografia	24

0 Introdução

0.1 Conteúdo

O relatório contém a explicação do código e os resultados encontrados para cada questão passada pelo professor e o código final em linguagem Python com uma explicação de como o programa funciona.

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

1.1.1 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 usuário 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:

```
1 ##### MENU PRINCIPAL #####
2 Digite somente o numero da questao que voce deseja ver o resultado:
3 1 = Questao 1
4 2 = Questao 2
5 3 = Questao 3
6 4 = Questao 4
7 5 = SAIR
8 #####
```

```

1 ##### MENU IRIS #####
2 Digite qual especie voce deseja fazer a regressao linear:
3 1 = Iris-Setosa
4 2 = Iris-Versicolor
5 3 = Iris Virginica
6 4 = VOLTAR AO MENU PRINCIPAL
7 #####

```

1.1.2 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_amstras**.

1.1.3 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.

1.1.4 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").

A função **construir_equacao_normal** utiliza 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 a nossa estimativa posteriormente, 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.

1.1.5 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 até terminar as colunas. Durante o loop das colunas, é verificado sempre se o pivô 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 coluna por coluna, criando uma matriz de zeros chamada L do mesmo tamanho que R. Tomando como pivô os elementos na diagonal principal de R, um loop que percorre as linhas de R e partindo da informação do pivô da coluna atual, coloca-se na matriz L na posição com número da linha igual ao número da coluna do pivô de R+1 e número da coluna do pivô de R: o negativo do valor atual da linha e coluna que está percorrendo o loop em R dividido pelo pivô da coluna analisada em R. Soma-se 1 no loop que percorre as linhas de R e faz o mesmo procedimento até finalizar todas as linhas de R. Depois completa a diagonal principal de L todas com valor 1. Cada vez que o pivô de R é diferente de zero, a matriz L é criada com elementos zeros do mesmo tamanho que a matriz R, até ter percorrido todas as colunas de R. Para a próxima coluna construída de L ou a próxima matriz L, o loop das linhas de R começa no número da linha de R igual ao número da coluna do pivô de R+1 e segue-se o mesmo procedimento para percorrer as linhas de R, somando-se 1 a esse valor de linha de R já estabelecido.

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 pela linha abaixo dela e a linha abaixo do pivô de R pela linha do pivô de R. Dessa forma, na matriz P, o número da linha do pivô de R e da coluna do pivô de R+1 recebe o valor 1. E o número da linha abaixo do pivô de R e com o mesmo número da coluna do pivô de R recebe o valor 1. 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 linha e coluna iguais ao número da linha analisada, de forma a ter 1 na diagonal principal nas linhas que não foram trocadas.

E assim o loop continua, indo para a próxima coluna da matriz R até ter percorrido todas as colunas. Quando tiver terminado de percorrer todas as colunas, já que 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).

1.1.6 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.

1.1.7 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.

1.1.8 estimar_amstras

A função `estimar_amstras` terá o propósito de fazer o produto interno de cada vetor de amostras selecionadas 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 nos vetores de amostras selecionados, 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 do produto interno com o valor da `PetalWidthCm` de cada 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_amstras` é chamada duas vezes, 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.

1.2 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     #  $(x^T) \cdot x \cdot w = (x^T) \cdot y$ 
55     #  $(x\_transposta) \cdot x \cdot w = (x\_transposta) \cdot y$ 
56     #  $R \cdot w = p$ 
57     #  $R = (x^T) \cdot x$ 
58     #  $p = (x^T) \cdot y$ 
59
60     ##### sem termo independente #####
61     #  $y = a \cdot x_1 + b \cdot x_2 + c \cdot x_3$ 
62     x = []
63     y = []
64
65     #achar  $(x^T)$  e x
66     # x - colunas SepalLengthCm, SepalWidthCm, PetalLengthCm
67     x = np.array(dados)
68     x = np.delete(x.reshape(15,4),3,1) #deleta a ultima coluna de x
69     x_transposta = np.transpose(x) #faz a transposta de x:  $(x^T)$ 
70
71     #achar R
72     R = np.dot(x_transposta,x) #multiplica  $(x^T)$  por x
73
74     #achar y - coluna PetalWidthCm
75     y = np.array(dados)
76     y = np.delete(y.reshape(15,4),0,1) #deleta a primeira coluna de y
77     y = np.delete(y.reshape(15,3),0,1) #deleta a segunda coluna de y

```

```

78     y = np.delete(y.reshape(15,2),0,1) #deleta a terceira coluna de y
79
80     #achar p
81     p = np.dot(x_transposta,y)
82
83     ##### com termo independente #####
84     # y = a*x1 +b*x2 +c*x3 +k
85
86     for i in range (0,15):
87         dados[i][3] = 1
88
89     #achar (x^T) e x
90     x = np.array(dados)
91     x_transposta = np.transpose(x) #faz a transposta de x: (x^T)
92     #achar R
93     R1= np.dot(x_transposta,x) #multiplica (x^T) por x
94     #achar p
95     p1 = np.dot(x_transposta,y)
96
97     return R,p,R1,p1
98
99 # -----
100 def PLU(R,p):
101     indice_coluna = 0
102     indice_linha = 1
103     indice_L_1s = 0
104     tamanho_R = len(R)
105
106     while (indice_coluna < tamanho_R):
107         #constroi a matriz L, triangular inferior
108         if (R[indice_coluna][indice_coluna] != 0):
109             L = np.zeros((tamanho_R, tamanho_R))
110
111             indice_coluna_aux = indice_coluna + 1
112             while (indice_coluna_aux < tamanho_R):
113                 L[indice_coluna_aux][indice_coluna] = - R[indice_coluna_aux][
114                     indice_coluna]/R[indice_coluna][indice_coluna]
115                 indice_linha += 1
116                 indice_coluna_aux += 1
117
118             while (indice_L_1s < tamanho_R):
119                 L[indice_L_1s][indice_L_1s] = 1
120                 indice_L_1s += 1
121
122             indice_L_1s = 0
123             indice_linha = 1
124
125             #multiplica a matriz R por L
126             R = np.dot (L,R)
127             p = np.dot (L,p)
128

```



```

129     #se o pivo for zero, necessario multiplicar por uma matriz P de
        permutacao
130     if (R[indice_coluna][indice_coluna] == 0):
131         P = np.zeros((tamanho_R, tamanho_R))
132         P[indice_coluna][indice_coluna + 1] = 1
133         P[indice_coluna + 1][indice_coluna] = 1
134
135         i = 0
136         j = 0
137         guarda_1 = False
138         while (i < tamanho_R):
139             while (j < tamanho_R):
140                 if (P[i][j] == 1):
141                     guarda_1 = True
142                     j += 1
143                 if (guarda_1 == False):
144                     P[i][i] = 1
145                 i += 1
146
147         R = np.dot(P,R)
148         p = np.dot (P,p)
149
150         indice_coluna += 1
151
152     #backsubstitution
153     w = np.linalg.solve(R, p)
154
155     return w
156
157 # -----
158 def decomposicao_espectral(R):
159
160     # R = VDV^(T)
161     #determinando autovalores e autovetores
162     autovalores, autovetores = np.linalg.eig(R)
163     # matriz diagonal de autovalores
164     matrizDiagonal = np.diag(autovalores)
165
166     return autovetores, matrizDiagonal
167
168 # -----
169 def contruir_svd (R):
170     s = []
171     i = 0
172
173     # R = U.s.VT
174     #determinando autovalores e autovetores
175     ### U
176     ## autovetores de R.(R^T)
177     autovaloresU, autovetoresU = np.linalg.eig(np.dot(R,np.transpose(R)))
178
179     # como R eh simetrica e quadrada: (R^T).R = R.(R^T)

```

```

180     # entao autovetores e autovalores de U sao iguais aos de V
181
182     tamanho_autovalores_U = autovaloresU.shape
183
184     while (i < tamanho_autovalores_U [0]):
185         s.append(np.sqrt(autovaloresU [i]))
186         i += 1
187
188     return autovetoresU, np.array(s),np.transpose(autovetoresU)
189
190 # -----
191 def estimar_amostras(amostra, w_setosa,w_versicolor,w_virginica,c_independente
192 ):
193
194     lista_erros = []
195     estimativa = ""
196     indice = 0
197     amostra_x = []
198     indice_erros_modulo = 0
199
200     while (indice < 3):
201         amostra_x.append (amostra[indice])
202         indice += 1
203
204     if (c_independente == True):
205         amostra_x.append(1)
206
207     amostra_x = np.array(amostra_x)
208
209     #produto interno <x,y> = (x^T).y
210
211     estimativa_setosa = np.dot(amostra_x,w_setosa)
212     estimativa_versicolor = np.dot(amostra_x,w_versicolor)
213     estimativa_virginica = np.dot(amostra_x,w_virginica)
214
215     erro_setosa = estimativa_setosa[0] - amostra[3]
216     lista_erros.append(erro_setosa)
217
218     erro_versicolor = estimativa_versicolor[0] - amostra[3]
219     lista_erros.append(erro_versicolor)
220
221     erro_virginica = estimativa_virginica[0] - amostra[3]
222     lista_erros.append(erro_virginica)
223
224     while (indice_erros_modulo < len(lista_erros)):
225         lista_erros [indice_erros_modulo] = abs(lista_erros[indice_erros_modulo
226 ])
227         indice_erros_modulo += 1
228
229     if (min(lista_erros) == abs(erro_setosa)):
230         estimativa = "Iris-Setosa"

```

```

230     return estimativa
231
232     if (min(lista_erros) == abs(erro_versicolor)):
233         estimativa = "Iris-Versicolor"
234
235     return estimativa
236
237     if (min(lista_erros) == abs(erro_virginica)):
238         estimativa = "Iris-Virginica"
239
240     return estimativa
241
242 ##### Programa Principal #####
243 def menu():
244     resultado = 0
245     tipo_iris = 0
246     coeficientes_sem_aux = []
247     coeficientes_com_aux = []
248     indice = 0
249
250
251     while (resultado != 5):
252         print()
253         print('##### MENU PRINCIPAL
254             #####')
255         print("Digite somente o numero da questao que voce deseja ver o
256             resultado: ")
257         print("1 = Questao 1")
258         print("2 = Questao 2")
259         print("3 = Questao 3")
260         print("4 = Questao 4")
261         print("5 = SAIR")
262         print('
263             #####
264             ')
265         print()
266
267         resultado = int(input())
268
269         if (resultado == 5):
270             sys.exit(0)
271
272         if (resultado == 1):
273
274             while (tipo_iris != 4):
275                 print('##### MENU IRIS

```

```

276 print("4 = VOLTAR AO MENU PRINCIPAL")
277 print('
      #####')
278 print()
279
280 tipo_iris = int(input())
281
282 if (tipo_iris == 4):
283     menu()
284
285 else:
286     dados = pegarDados (tipo_iris)
287     R,p,R1,p1 = construir_equacao_normal(dados)
288     w = PLU(R,p)
289     w1 = PLU(R1,p1)
290
291     print()
292     if (tipo_iris == 1):
293         print("Iris-Setosa\n")
294     elif (tipo_iris == 2):
295         print("Iris-Versicolor\n")
296     elif (tipo_iris == 3):
297         print("Iris-Virginica\n")
298
299     print("SEM O TERMO INDEPENDENTE: ")
300     print("y = a*x1 + b*x2 + c*x3")
301     print("[a b c] = ",end="")
302
303
304     while (indice < len(w)):
305         coeficientes_sem_aux.append(w[indice][0])
306         coeficientes_sem = np.array(coeficientes_sem_aux)
307         indice += 1
308
309     print(coeficientes_sem)
310     coeficientes_sem_aux = []
311
312
313     print()
314
315     print("COM O TERMO INDEPENDENTE: ")
316     print("y = a*x1 + b*x2 + c*x3 + k")
317     print("[a b c k] = ",end="")
318
319     indice = 0
320     while (indice < len(w1)):
321         coeficientes_com_aux.append(w1[indice][0])
322         coeficientes_com = np.array(coeficientes_com_aux)
323         indice += 1
324
325     print(coeficientes_com)
326     coeficientes_com_aux = []

```

```

327         indice = 0
328
329         print()
330
331     if (resultado == 2):
332
333         while (tipo_iris != 4):
334             print('##### MENU IRIS
335                   #####')
336             print("Digite qual especie voce deseja fazer a decomposicao
337                   espectral: ")
338             print("1 = Iris-Setosa")
339             print("2 = Iris-Versicolor")
340             print("3 = Iris Virginica")
341             print("4 = VOLTAR AO MENU PRINCIPAL")
342             print('
343                   #####
344                   ')
345             print()
346
347             tipo_iris = int(input())
348
349             if (tipo_iris == 4):
350                 menu()
351
352             else:
353                 dados = pegarDados (tipo_iris)
354                 R,p,R1,p1 = construir_equacao_normal(dados)
355                 autovetores, matrizDiagonal = decomposicao_espectral(R)
356                 autovetores1, matrizDiagonal1 = decomposicao_espectral(R1)
357
358                 if (tipo_iris == 1):
359                     print("Iris-Setosa\n")
360                 elif (tipo_iris == 2):
361                     print("Iris-Versicolor\n")
362                 elif (tipo_iris == 3):
363                     print("Iris-Virginica\n")
364
365                 print("R =  $V \lambda V^T$ ")
366                 print()
367
368                 print("SEM O TERMO INDEPENDENTE: ")
369                 print("R = ", R)
370                 print()
371                 print("V = ", autovetores)
372                 print()
373                 print("\u039B = ", matrizDiagonal)
374                 print()
375                 print("V^T = ", np.transpose(autovetores))

```

```

375         print()
376         print()
377
378         print("COM O TERMO INDEPENDENTE: ")
379         print("R = ", R1)
380         print()
381         print("V = ", autovetores1)
382         print()
383         print('\u039B = ', matrizDiagonal1)
384         print()
385         print("V^T = ", np.transpose(autovetores1))
386
387         print()
388
389     if (resultado == 3):
390
391         while (tipo_iris != 4):
392             print('##### MENU IRIS #####')
393             print("Digite qual especie voce deseja fazer o SVD: ")
394             print("1 = Iris-Setosa")
395             print("2 = Iris-Versicolor")
396             print("3 = Iris Virginica")
397             print("4 = VOLTAR AO MENU PRINCIPAL")
398             print('#####')
399             print()
400
401             tipo_iris = int(input())
402
403             if (tipo_iris == 4):
404                 menu()
405
406             else:
407
408                 dados = pegarDados (tipo_iris)
409                 R,p,R1,p1 = construir_equacao_normal(dados)
410                 U, s, VT = contruir_svd(R)
411                 U1, s1, VT1 = contruir_svd(R1)
412
413                 if (tipo_iris == 1):
414                     print("Iris-Setosa\n")
415                 elif (tipo_iris == 2):
416                     print("Iris-Versicolor\n")
417                 elif (tipo_iris == 3):
418                     print("Iris-Virginica\n")
419
420                 print("R = U\u03A3V^(T)")
421                 print()
422
423                 print("SEM O TERMO INDEPENDENTE: ")
424                 print("R = ", R)
425                 print()
426                 print("U = ", U)

```

```

427         print()
428         print('\u03A3 = ', np.diag(s))
429         print()
430         print("V^T = ", VT)
431         print()
432         print()
433
434         print("COM O TERMO INDEPENDENTE: ")
435         print("R = ", R1)
436         print()
437         print("U = ", U1)
438         print()
439         print('\u03A3 = ', np.diag(s1))
440         print()
441         print("V^T = ", VT1)
442
443         print()
444
445
446
447     if (resultado == 4):
448         A = [5.0,2.3,3.3,1.0]
449         B = [4.6,3.2,1.4,0.2]
450         C = [5.0,3.3,1.4,0.2]
451         D = [6.1,3.0,4.6,1.4]
452         E = [5.9,3.0,5.1,1.8]
453
454         print("SEM O TERMO INDEPENDENTE: ")
455         dados = pegarDados (1)
456         R,p,R1,p1 = construir_equacao_normal(dados)
457         w_setosa = PLU(R,p)
458
459         dados = pegarDados (2)
460         R,p,R1,p1 = construir_equacao_normal(dados)
461         w_versicolor = PLU(R,p)
462
463         dados = pegarDados (3)
464         R,p,R1,p1 = construir_equacao_normal(dados)
465         w_virginica = PLU(R,p)
466
467         c_independente = False
468         estimativa = estimar_amstras(A,w_setosa,w_versicolor,w_virginica,
469                                     c_independente)
470         print("A = ", estimativa)
471         estimativa = estimar_amstras(B,w_setosa,w_versicolor,w_virginica,
472                                     c_independente)
473         print("B = ", estimativa)
474         estimativa = estimar_amstras(C,w_setosa,w_versicolor,w_virginica,
475                                     c_independente)
476         print("C = ", estimativa)
477         estimativa = estimar_amstras(D,w_setosa,w_versicolor,w_virginica,
478                                     c_independente)

```

```

475     print("D = ", estimativa)
476     estimativa = estimar_amstras(E,w_setosa,w_versicolor,w_virginica,
        c_independente)
477     print("E = ", estimativa)
478
479     print()
480     print("COM O TERMO INDEPENDENTE: ")
481     dados = pegarDados (1)
482     R,p,R1,p1 = construir_equacao_normal(dados)
483     w1_setosa = PLU(R1,p1)
484
485     dados = pegarDados (2)
486     R,p,R1,p1 = construir_equacao_normal(dados)
487     w1_versicolor = PLU(R1,p1)
488
489     dados = pegarDados (3)
490     R,p,R1,p1 = construir_equacao_normal(dados)
491     w1_virginica = PLU(R1,p1)
492
493     c_independente = True
494     estimativa1 = estimar_amstras(A,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
495     print("A = ", estimativa1)
496     estimativa1 = estimar_amstras(B,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
497     print("B = ", estimativa1)
498     estimativa1 = estimar_amstras(C,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
499     print("C = ", estimativa1)
500     estimativa1 = estimar_amstras(D,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
501     print("D = ", estimativa1)
502     estimativa1 = estimar_amstras(E,w1_setosa,w1_versicolor,
        w1_virginica,c_independente)
503     print("E = ", estimativa1)
504
505     menu()
506
507     ##### chamada ao menu
508     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

```
1 Iris-Setosa
2
3 SEM O TERMO INDEPENDENTE:
4 y = a*x1 + b*x2 + c*x3
5 [a b c] = [ 0.07455282 -0.06602361  0.03673264]
6
7 COM O TERMO INDEPENDENTE:
8 y = a*x1 + b*x2 + c*x3 + k
9 [a b c k] = [ 0.13497209 -0.07886596  0.10365958 -0.36144997]
```

2.2.2 Iris-Versicolor

```
1 Iris-Versicolor
2
3 SEM O TERMO INDEPENDENTE:
4 y = a*x1 + b*x2 + c*x3
5 [a b c] = [-0.14382683  0.17051618  0.40397714]
6
7 COM O TERMO INDEPENDENTE:
8 y = a*x1 + b*x2 + c*x3 + k
9 [a b c k] = [-0.06826027  0.24149193  0.39877688 -0.63863516]
```

2.2.3 Iris-Virginica

```
1 Iris-Virginica
2
3 SEM O TERMO INDEPENDENTE:
4 y = a*x1 + b*x2 + c*x3
5 [a b c] = [-0.11329721  0.3990124  0.25976859]
6
7 COM O TERMO INDEPENDENTE:
8 y = a*x1 + b*x2 + c*x3 + k
9 [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, utilizar o R retornado da equação normal para fazer a decomposição espectral.

3.2 Resultados

3.2.1 Iris-Setosa

```
1 Iris-Setosa
2
3 R = V\A\^(T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[381.33 255.73 112.57]
7      [255.73 172.5   75.51]
8      [112.57  75.51  33.84]]
9
10 V = [[-0.80618188 -0.45661854  0.37625828]
11      [-0.54157827  0.31342349 -0.78003762]
12      [-0.23825146  0.8326255   0.49997102]]
13
14 Λ = [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
15      [0.00000000e+00 5.29776824e-01 0.00000000e+00]
16      [0.00000000e+00 0.00000000e+00 7.47589571e-01]]
17
18 V^T = [[-0.80618188 -0.54157827 -0.23825146]
19         [-0.45661854  0.31342349  0.8326255 ]
20         [ 0.37625828 -0.78003762  0.49997102]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[381.33 255.73 112.57 75.5 ]
25      [255.73 172.5   75.51 50.6 ]
26      [112.57  75.51  33.84 22.4 ]
27      [ 75.5   50.6   22.4  15.  ]]
28
29 V = [[-0.79610972  0.1628056  -0.47792329 -0.33360603]
30      [-0.53478061 -0.03864172  0.33762985  0.77364243]
31      [-0.23529777  0.18572124  0.80860497 -0.50626137]
32      [-0.15765144 -0.96825037  0.06126507 -0.18407563]]
33
34 Λ = [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35      [0.00000000e+00 2.79352229e-02 0.00000000e+00 0.00000000e+00]
36      [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
37      [0.00000000e+00 0.00000000e+00 0.00000000e+00 7.73257486e-01]]
38
39 V^T = [[-0.79610972 -0.53478061 -0.23529777 -0.15765144]
40         [ 0.1628056  -0.03864172  0.18572124 -0.96825037]
41         [-0.47792329  0.33762985  0.80860497  0.06126507]
42         [-0.33360603  0.77364243 -0.50626137 -0.18407563]]
```

3.2.2 Iris-Versicolor

```
1 Iris-Versicolor
2
3 R = V\A\^(T)
```

```

4
5 SEM O TERMO INDEPENDENTE:
6 R = [[555.38 256.64 397.59]
7       [256.64 119.98 184.35]
8       [397.59 184.35 286.5 ]]
9
10 V = [[ 0.76039163  0.64759888 -0.0491961 ]
11       [ 0.35223975 -0.47485674 -0.80649751]
12       [ 0.54564799 -0.59592513  0.58918716]]
13
14 Λ = [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
15       [0.00000000e+00 1.33162776e+00 0.00000000e+00]
16       [0.00000000e+00 0.00000000e+00 9.57976568e-01]]
17
18 V^T = [[ 0.76039163  0.35223975  0.54564799]
19         [ 0.64759888 -0.47485674 -0.59592513]
20         [-0.0491961  -0.80649751  0.58918716]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[555.38 256.64 397.59  91.  ]
25       [256.64 119.98 184.35  42.2 ]
26       [397.59 184.35 286.5   65.2 ]
27       [ 91.    42.2   65.2   15.  ]]
28
29 V = [[-0.7545525  0.11812875  0.64474223  0.03167953]
30       [-0.34954066  0.11526607 -0.46962259  0.80248968]
31       [-0.54144512 -0.01367345 -0.60234554 -0.58637025]
32       [-0.1237297  -0.98619084  0.030691    0.1057197 ]]
33
34 Λ = [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35       [0.00000000e+00 7.14132919e-02 0.00000000e+00 0.00000000e+00]
36       [0.00000000e+00 0.00000000e+00 1.33280153e+00 0.00000000e+00]
37       [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]
38
39 V^T = [[-0.7545525  -0.34954066 -0.54144512 -0.1237297 ]
40         [ 0.11812875  0.11526607 -0.01367345 -0.98619084]
41         [ 0.64474223 -0.46962259 -0.60234554  0.030691  ]
42         [ 0.03167953  0.80248968 -0.58637025  0.1057197 ]]

```

3.2.3 Iris-Virginica

```

1 Iris-Virginica
2
3 R = VΛV^T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[676.15 304.85 562.72]
7       [304.85 138.7  253.94]
8       [562.72 253.94 469.3 ]]
9
10 V = [[-0.72592866 -0.63483621  0.2645951 ]

```

```

11 [-0.32772296 -0.01894783 -0.94458385]
12 [-0.60466953  0.77241438  0.19429563]]
13
14  $\Lambda$  = [[1.28249881e+03  0.00000000e+00  0.00000000e+00]
15 [0.00000000e+00  5.79253032e-01  0.00000000e+00]
16 [0.00000000e+00  0.00000000e+00  1.07193335e+00]]
17
18  $V^T$  = [[-0.72592866 -0.32772296 -0.60466953]
19 [-0.63483621 -0.01894783  0.77241438]
20 [ 0.2645951 -0.94458385  0.19429563]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[676.15  304.85  562.72  100.3 ]
25 [304.85  138.7  253.94  45.4 ]
26 [562.72  253.94  469.3  83.6 ]
27 [100.3  45.4  83.6  15.  ]]
28
29 V = [[-7.21743001e-01 -2.76297507e-01  6.34623278e-01 -1.50933108e-04]
30 [-3.25843186e-01  9.35769444e-01  3.68031853e-02 -1.29642937e-01]
31 [-6.01187515e-01 -1.93722166e-01 -7.68083597e-01 -1.05322749e-01]
32 [-1.07176625e-01  1.02308151e-01 -7.71129602e-02  9.85951218e-01]]
33
34  $\Lambda$  = [[1.29740071e+03  0.00000000e+00  0.00000000e+00  0.00000000e+00]
35 [0.00000000e+00  1.08243546e+00  0.00000000e+00  0.00000000e+00]
36 [0.00000000e+00  0.00000000e+00  5.82311582e-01  0.00000000e+00]
37 [0.00000000e+00  0.00000000e+00  0.00000000e+00  8.45463191e-02]]
38
39  $V^T$  = [[-7.21743001e-01 -3.25843186e-01 -6.01187515e-01 -1.07176625e-01]
40 [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
41 [ 6.34623278e-01  3.68031853e-02 -7.68083597e-01 -7.71129602e-02]
42 [-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, utilizar o R retornado da equação normal para fazer o SVD.

4.2 Resultados

4.2.1 Iris-Setosa

```

1 Iris-Setosa
2
3 R = UΣVT(T)
4
5 SEM O TERMO INDEPENDENTE:

```

```

6 R = [[381.33 255.73 112.57]
7      [255.73 172.5 75.51]
8      [112.57 75.51 33.84]]
9
10 U = [[ 0.80618188 -0.45661854 -0.37625828]
11       [ 0.54157827 0.31342349 0.78003762]
12       [ 0.23825146 0.8326255 -0.49997102]]
13
14 Σ = [[5.86392634e+02 0.00000000e+00 0.00000000e+00]
15       [0.00000000e+00 5.29776824e-01 0.00000000e+00]
16       [0.00000000e+00 0.00000000e+00 7.47589571e-01]]
17
18 V^T = [[ 0.80618188 0.54157827 0.23825146]
19         [-0.45661854 0.31342349 0.8326255 ]
20         [-0.37625828 0.78003762 -0.49997102]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[381.33 255.73 112.57 75.5 ]
25       [255.73 172.5 75.51 50.6 ]
26       [112.57 75.51 33.84 22.4 ]
27       [ 75.5 50.6 22.4 15. ]]
28
29 U = [[ 0.79610972 -0.33360603 -0.47792329 -0.1628056 ]
30       [ 0.53478061 0.77364243 0.33762985 0.03864172]
31       [ 0.23529777 -0.50626137 0.80860497 -0.18572124]
32       [ 0.15765144 -0.18407563 0.06126507 0.96825037]]
33
34 Σ = [[6.01336860e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35       [0.00000000e+00 7.73257486e-01 0.00000000e+00 0.00000000e+00]
36       [0.00000000e+00 0.00000000e+00 5.31947766e-01 0.00000000e+00]
37       [0.00000000e+00 0.00000000e+00 0.00000000e+00 2.79352230e-02]]
38
39 V^T = [[ 0.79610972 0.53478061 0.23529777 0.15765144]
40         [-0.33360603 0.77364243 -0.50626137 -0.18407563]
41         [-0.47792329 0.33762985 0.80860497 0.06126507]
42         [-0.1628056 0.03864172 -0.18572124 0.96825037]]

```

4.2.2 Iris-Versicolor

```

1 Iris-Versicolor
2
3 R = UΣV^(T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[555.38 256.64 397.59]
7       [256.64 119.98 184.35]
8       [397.59 184.35 286.5 ]]
9
10 U = [[ 0.76039163 0.64759888 -0.0491961 ]
11       [ 0.35223975 -0.47485674 -0.80649751]
12       [ 0.54564799 -0.59592513 0.58918716]]

```

```

13
14 Σ = [[9.59570396e+02 0.00000000e+00 0.00000000e+00]
15 [0.00000000e+00 1.33162776e+00 0.00000000e+00]
16 [0.00000000e+00 0.00000000e+00 9.57976568e-01]]
17
18 V^T = [[ 0.76039163  0.35223975  0.54564799]
19 [ 0.64759888 -0.47485674 -0.59592513]
20 [-0.0491961 -0.80649751  0.58918716]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[555.38 256.64 397.59 91. ]
25 [256.64 119.98 184.35 42.2 ]
26 [397.59 184.35 286.5 65.2 ]
27 [ 91. 42.2 65.2 15. ]]
28
29 U = [[ 0.7545525 -0.64474223 -0.11812875 0.03167953]
30 [ 0.34954066 0.46962259 -0.11526607 0.80248968]
31 [ 0.54144512 0.60234554 0.01367345 -0.58637025]
32 [ 0.1237297 -0.030691 0.98619084 0.1057197 ]]
33
34 Σ = [[9.74487597e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35 [0.00000000e+00 1.33280153e+00 0.00000000e+00 0.00000000e+00]
36 [0.00000000e+00 0.00000000e+00 7.14132919e-02 0.00000000e+00]
37 [0.00000000e+00 0.00000000e+00 0.00000000e+00 9.68188359e-01]]
38
39 V^T = [[ 0.7545525 0.34954066 0.54144512 0.1237297 ]
40 [-0.64474223 0.46962259 0.60234554 -0.030691 ]
41 [-0.11812875 -0.11526607 0.01367345 0.98619084]
42 [ 0.03167953 0.80248968 -0.58637025 0.1057197 ]]

```

4.2.3 Iris-Virginica

```

1 Iris-Virginica
2
3 R = UΣV^T)
4
5 SEM O TERMO INDEPENDENTE:
6 R = [[676.15 304.85 562.72]
7 [304.85 138.7 253.94]
8 [562.72 253.94 469.3 ]]
9
10 U = [[ 0.72592866 -0.63483621 -0.2645951 ]
11 [ 0.32772296 -0.01894783 0.94458385]
12 [ 0.60466953 0.77241438 -0.19429563]]
13
14 Σ = [[1.28249881e+03 0.00000000e+00 0.00000000e+00]
15 [0.00000000e+00 5.79253032e-01 0.00000000e+00]
16 [0.00000000e+00 0.00000000e+00 1.07193335e+00]]
17
18 V^T = [[ 0.72592866 0.32772296 0.60466953]
19 [-0.63483621 -0.01894783 0.77241438]

```

```

20 [-0.2645951    0.94458385 -0.19429563]]
21
22
23 COM O TERMO INDEPENDENTE:
24 R = [[676.15 304.85 562.72 100.3 ]
25      [304.85 138.7  253.94  45.4 ]
26      [562.72 253.94 469.3   83.6 ]
27      [100.3  45.4   83.6   15.  ]]
28
29 U = [[ 7.21743001e-01 -2.76297507e-01 -6.34623278e-01 -1.50933128e-04]
30      [ 3.25843186e-01  9.35769444e-01 -3.68031855e-02 -1.29642937e-01]
31      [ 6.01187515e-01 -1.93722166e-01  7.68083597e-01 -1.05322749e-01]
32      [ 1.07176625e-01  1.02308151e-01  7.71129601e-02  9.85951218e-01]]
33
34 Σ = [[1.29740071e+03 0.00000000e+00 0.00000000e+00 0.00000000e+00]
35      [0.00000000e+00 1.08243546e+00 0.00000000e+00 0.00000000e+00]
36      [0.00000000e+00 0.00000000e+00 5.82311582e-01 0.00000000e+00]
37      [0.00000000e+00 0.00000000e+00 0.00000000e+00 8.45463191e-02]]
38
39 V^T = [[ 7.21743001e-01  3.25843186e-01  6.01187515e-01  1.07176625e-01]
40        [-2.76297507e-01  9.35769444e-01 -1.93722166e-01  1.02308151e-01]
41        [-6.34623278e-01 -3.68031855e-02  7.68083597e-01  7.71129601e-02]
42        [-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_amstras.

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

```

1 SEM O TERMO INDEPENDENTE:
2 A = Iris-Versicolor
3 B = Iris-Setosa
4 C = Iris-Setosa
5 D = Iris-Versicolor
6 E = Iris-Virginica
7
8 COM O TERMO INDEPENDENTE:
9 A = Iris-Versicolor
10 B = Iris-Setosa
11 C = Iris-Setosa
12 D = Iris-Versicolor
13 E = Iris-Virginica

```

6 Bibliografia

- <https://algebrailinearufcg.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/