Exercício de Programação 1

Prazo de submissão: 23:55 do dia 04.01.2021

2020.2 Álgebra Linear Computacional - DCC - UFMG

Erickson - Fabricio

Instruções:

- Antes de submeter suas soluções, certifique-se de que tudo roda como esperado. Primeiro, **reinicie o kernel** no menu, selecione Kernel$\rightarrow$Restart e então execute **todas as células** (no menu, Cell$\rightarrow$Run All)
- Apenas o arquivo .ipynb deve ser submetido. Ele não deve ser compactado.
- Não deixe de preencher seu nome e número de matrícula na célula a seguir

Nome do aluno: INSIRA SEU NOME AQUI

Matricula: INSIRA SUA MATRICULA AQUI

Questão 1

Dadas as matrizes A, B e o vetor C:

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 1 & 4 \\ 5 & 7 & 12 \end{bmatrix}_{3\times 3} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3\times 3} \qquad C = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}_{3\times 1}$$

Gere as matrizes D, E e o vetor F tal que:

$$D = AB$$
$$E = A^\top + B$$
$$F = AC$$

Dicas:

- *Imprima* `A.shape, B.shape, C.shape` *e confira se as dimensões de suas matrizes batem com a descrição do enunciado.*
- As operações de produto e transposição estão definidas na documentação da biblioteca `numpy`.

```
# Código para Exercício 1 (alt)

import numpy as np

A = np.array([[2,1,3],[3,1,4],[5,7,12]])
B = np.eye(3)
C = np.array([[3],[1],[2]])
```

```
print(A)
print(B)
print(C)

D = np.dot(A,B)
E = A.T + B
F = np.dot(A,C)

print(D)
print(E)
print(F)
```

```
[[ 2  1  3]
 [ 3  1  4]
 [ 5  7 12]]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[3]
 [1]
 [2]]
[[ 2.   1.   3.]
 [ 3.   1.   4.]
 [ 5.   7. 12.]]
[[ 3.   3.   5.]
 [ 1.   2.   7.]
 [ 3.   4. 13.]]
[[13]
 [18]
 [46]]
```

Questão 2

Questão 2.1

Uma forma de representar um vetor no espaço é através de um ponto localizado na extremidade da "seta" do vetor. Essa representação é especialmente útil quando queremos vizualizar uma grande quantidade de vetores. Crie um array chamado `dados`, de tamanho $n \times 2$, em que cada linha represente um dos seguintes vetores de tamanho 2:

- $(\,0.7,\ 0.7\,)$
- $(\,0.0,\ 1.0\,)$
- $(\,-0.7,\ 0.7\,)$
- $(\,-1.0,\ 0.0\,)$
- $(\,-0.7,\ -0.7\,)$
- $(\,0.0,\ -1.0\,)$
- $(\,0.7,\ -0.7\,)$
- $(\,1.0,\ 0.0\,)$

Você pode achar a documentação do [`np.array`](np.array) útil.

```
# Código para Exercício 2.1

dados = np.array(
  [[0.7,   0.7],
   [0.0,   1.0],
   [-0.7,  0.7],
   [-1.0,    0],
   [-0.7, -0.7],]
   [0,    -1.0],
   [0.7,  -0.7],
   [1.0,     0]])

print(dados, dados.shape)
```

```
[[ 0.7  0.7]
 [ 0.   1. ]
 [-0.7  0.7]
 [-1.   0. ]
 [-0.7 -0.7]
 [ 0.  -1. ]
 [ 0.7 -0.7]
 [ 1.   0. ]] (8, 2)
```

Questão 2.2

Vamos agora visualizar esses pontos no espaço 2D. Para isso, podemos usar a biblioteca `matplotlib`. Agora, com as coordenadas x e y dos pontos do exercício anterior, crie um gráfico de dispersão que mostre cada ponto no plano. Voce pode achar a documentação de `plt.scatter` útil, além da dica que para escolher a coluna i de um array bi-dimensional usamos `x[:, i]` (consulte essa página sobre indexing em `numpy.arrays` para mais detalhes).

```
# Código para Exercício 2.2

import matplotlib.pyplot as plt

x = dados[:, 0]
y = dados[:, 1]

plt.scatter(x, y)
plt.title('scatter-plot dos pontos criados')
plt.show()
```

Questão 2.3

Agora que temos como visualizar os vetores no plano, vamos fazer operação de adição de vetores. Crie um array que represente um vetor $\mathbf{a} = (\ 6,\ 9\ )$ e adicione-o a todos os vetores no nosso array, criando um novo array chamado `novos_dados`.

Dica: Quando estamos tratando de matrizes, não podemos simplesmente adicionar uma matriz de tamanho $n \times 2$ por um vetor de tamanho 2, ou 1×2. Porém, o `numpy` tem uma funcionalidade que é muito útil quando queremos fazer operações entre arrays que não possuem o mesmo tamanho, como é o nosso caso (podemos ver isso usando: `print(dados.shape, a.shape)`). Essa funcionalidade é o [broadcasting](), e ela nos ajuda a fazer operações entre arrays que não possuem o mesmo tamanho, mas algumas dimensões são compatíveis.

```
# Código para Exercício 3
a = np.array([6, 9])
novos_dados = dados + a
print(novos_dados, novos_dados.shape)
```

```
[[ 6.7  9.7]
 [ 6.  10. ]
 [ 5.3  9.7]
 [ 5.   9. ]
 [ 5.3  8.3]
 [ 6.   8. ]
 [ 6.7  8.3]
 [ 7.   9. ]] (8, 2)
```

Questão 2.4

Note que para somar arrays de dimensões diferentes (nesse caso, dados é 2D e a é 1D), o broadcasting primeiro adiciona dimensões de tamanho 1 ao início do array com menos dimensões. Só depois ele expande as dimensões de tamanho 1 para que casem com os tamanhos das dimensões do outro array.

Portanto, devemos pensar no array `a` como um vetor coluna ou como um vetor linha?

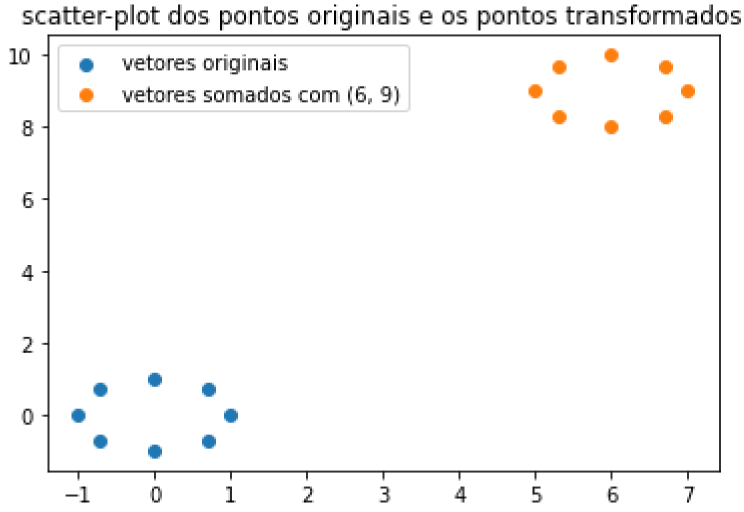
Resposta: Vetor linha, pois o broadcasting irá fazer com que o seu shape seja (1,2), antes de replicá-lo como muitas linhas de forma a casar com a dimensão do array dados.

Questão 2.5

Agora podemos ver no espaço 2D nossos vetores originais e os vetores resultantes da soma. Para isso, podemos usar a mesma função que usamos para criar o gráfico de dispersão na

Questão 2.2, porém agora queremos mostrar os pontos de 2 arrays, e não de apenas 1.

```
plt.scatter(dados[:, 0], dados[:, 1], label='vetores originais')
plt.scatter(novos_dados[:, 0], novos_dados[:, 1], label='vetores somados com (6, 9)')
plt.title('scatter-plot dos pontos originais e os pontos transformados')
plt.legend()
plt.show()
```



Questão 3

Como visto em aula, a multiplicação de uma matriz G por um vetor x pode ser vista como uma combinação linear das colunas de G.

Questão 3.1

Dado a matriz G e o vetor x:

$$G = \begin{bmatrix} 3 & 6 & 9 \\ 5 & 10 & 15 \end{bmatrix}_{2\times 3} \qquad \mathrm{x} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}_{3\times 1}$$

Gere o vetor y tal que:

$$\mathrm{y} = G\mathrm{x}$$

```
# Código para Exercício 3

# 3.1
G = np.array([[3,6,9], [5,10,15]])
x = np.array([[2],[1],[1]])
y = np.dot(G,x)

print(y)
```

```
[[21]
 [35]]
```

Questão 3.2

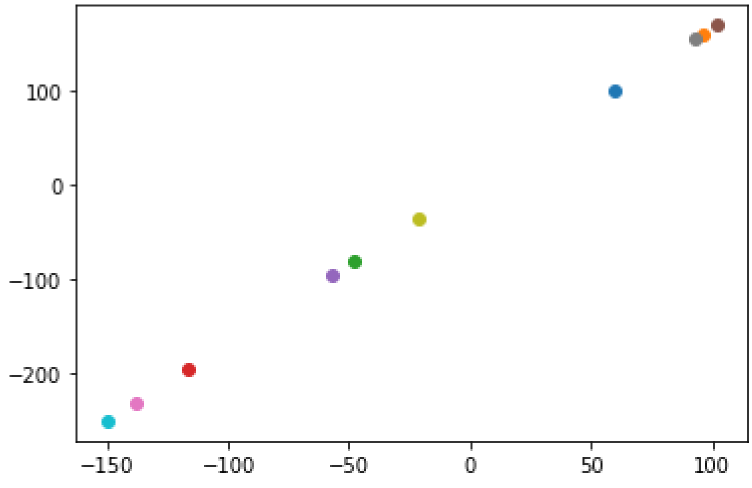
O espaço de colunas de uma matriz pode ser interpretado como o espaço formado por todas as combinações lineares das colunas da matriz. Então, vamos simular a representação do espaço de colunas da matriz G criada anteriormente fazendo várias combinações lineares de suas colunas. Para isso, podemos criar vários vetores-coluna x aleatórios, calcular a operação $y = Gx$ para cada um deles, e mostrar onde cada vetor y está localizado no espaço. Portanto, faça os seguintes passos:

1. Crie um vetor-coluna $x \in \mathbb{R}^{3\times1}$ com valores aleatórios entre -10 e 10.
2. Calcule $y = Gx$. Esse passo pode ser feito da mesma forma que a **Questão 3.1**.
3. Plote no plano 2D um ponto com as coordenadas do y resultante.

Repita esses passos 10 vezes. **Dica:** para a criação dos valores x aleatórios, voce pode escolher os números você mesmo ou usar a função `np.random.randint`. Para a visualização, utilize a biblioteca [`matplotlib`], e deixe para usar `plt.show()` apenas depois de ter plotado todos os vetores y, para que todos apareçam no mesmo gráfico.

```
# 3.2
for _ in range(10):
    x = np.random.randint(low=-10, high=10 + 1, size=[3, 1])
    y = np.dot(G, x)
    plt.scatter(y[0], y[1])

plt.show()
```



Questão 3.3

No item anterior, você deve ter obtido uma reta ao visualizar o gráfico resultante. Por que isso acontece no caso dessa matriz G em específico?

Resposta: *insira sua resposta aqui*

Questão 3.4

Agora que sabemos que o espaço coluna da matriz G é uma reta, encontre dois pontos no espaço coluna $C(G)$ e use o comando `plt.plot()` para traçá-la.

```
# 3.4

X = np.random.randint(low=-10, high=10 + 1, size=[3, 2])
Y = G @ X
plt.plot(Y[:,0],Y[:,1])
```

```
[<matplotlib.lines.Line2D at 0x7f3cbdfeba58>]
```