XMAC02
Métodos Matemáticos para Análise de Dados

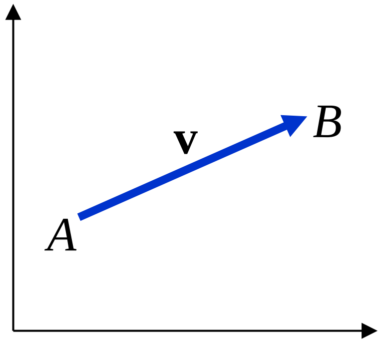
Aula 20 – Introdução à Álgebra Linear

Álgebra Linear

- Ramo da matemática que lida com equações lineares e suas representações utilizando vetores e matrizes
- Conhecimento importante para o estudo de Aprendizado de máquina
  - Classificação de dados
  - Regressão linear
  - Análise de componente principal
  - Sistemas de recomendação
  - Deep learning

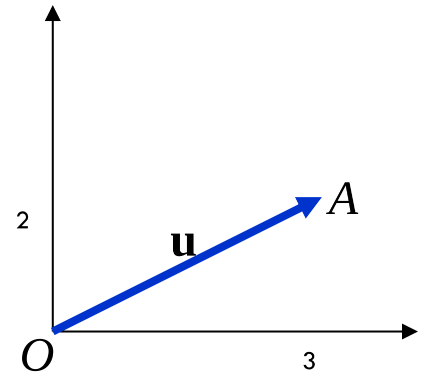
Vetores

- Pode-se representar um vetor **v** do ponto A ao ponto B como um segmento de reta de A a B, expresso da seguinte forma: $\overrightarrow{AB}$
  - Vetores devem ser nomeados com um único caracter em negrito: **v**
  - Ponto A é o ponto inicial ou cauda
  - Ponto B é o ponto terminal ou cabeça



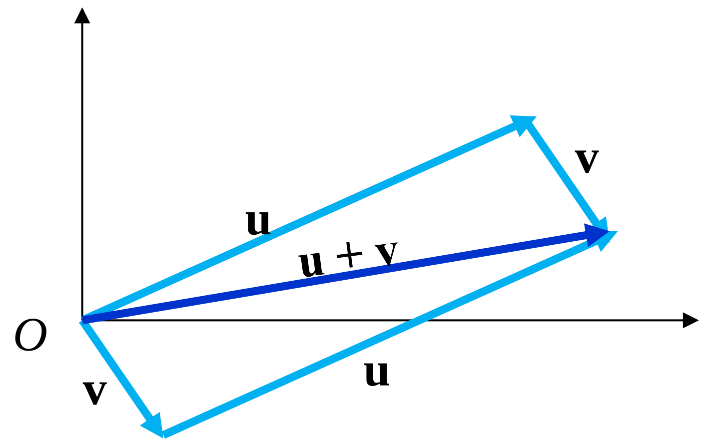
Vetores

- Todo ponto A no plano corresponde a um vetor cuja cauda está na origem 0 e a cabeça está no ponto A
  - Exemplo: se as coordenadas do ponto A são (3, 2), pode-se denotar o vetor $\mathbf{u} = \overrightarrow{OA} = [3, 2]$
  - É possível representar u como um vetor de linha $[3, 2]$ ou de coluna $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$



Aritmética de Vetores

- Suponha os vetores **u** = [3, 2] e **v** = [1, -1]
  - **u** + **v** = [3, 2] + [1, -1] = [3 + 1, 2 - 1] = [4, 1]



- Basta somar os componentes correspondentes
  - 3 + 1 = 4 e 2 - 1 = 1
- Note que **u** + **v** = **v** + **u**

Multiplicação escalar

- Para multiplicar um vetor por uma constante, basta multiplicar cada componente do vetor pela constante
  - Exemplo: se $\mathbf{u} = [3, 2]$, então $2\mathbf{u} = [6, 4]$

Combinação Linear

Um vetor $\mathbf{v}$ é uma combinação linear do vetores $\mathbf{v}_1$, $\mathbf{v}_2$, ..., $\mathbf{v}_k$ se houver constantes c_1, c_2, ..., c_k tais que $\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_k\mathbf{v}_k$.

Exemplo: O vetor $\begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix}$ é uma combinação linear dos

vetores $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$, $\begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}$, e $\begin{bmatrix} 5 \\ -4 \\ 0 \end{bmatrix}$ uma vez que

$$3\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + 2\begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} - \begin{bmatrix} 5 \\ -4 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix}$$

Produto Escalar

Sejam $\mathbf{u}$ e $\mathbf{v}$ tal que

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \text{ and } \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

então o produto escalar $\mathbf{u} \cdot \mathbf{v}$ é

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

Produto Escalar

- Exemplo: Sejam

$$\mathbf{u} = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix} \quad \text{e} \quad \mathbf{v} = \begin{bmatrix} -3 \\ 5 \\ 2 \end{bmatrix}$$

- Então $\mathbf{u} \cdot \mathbf{v} = 1 \cdot (-3) + 2 \cdot 5 + (-3) \cdot 2 = 1$

- Notem que um produto escalar é um escalar

Módulo ou Norma ou Comprimento

O **módulo** (ou **norma**) de um vetor $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$

é o escalar não negativo

$$\|\mathbf{v}\| = \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

- Exemplo: $\|[2, 3]\| = \sqrt{2^2 + 3^2} = \sqrt{13}$

Normalização de vetores

- Um vetor unitário é um vetor de comprimento 1
  - Exemplos: $\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- Se um vetor **v** é diferente de zero, é possível encontrar um vetor unitário na mesma direção de **v** dividindo **v** pelo valor de seu comprimento
- Encontrar um vetor unitário na mesma direção de um vetor **v** é chamado normalização de **v**

Normalização de vetores

- Exemplo: Seja $\mathbf{v} = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$ então $\|\mathbf{v}\| = \sqrt{14}$

O vetor unitário na mesma direção de v é

$$\mathbf{u} = \left(\frac{1}{\|\mathbf{v}\|}\right)\mathbf{v} = \left(1/\sqrt{14}\right)\begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2/\sqrt{14} \\ -1/\sqrt{14} \\ 3/\sqrt{14} \end{bmatrix}$$

Vetores no NumPy

- Usando Numpy arrays é possível realizar aritmética de vetores diretamente

```
import numpy as np

u = np.array([3, 2])
v = np.array([1, -1])

print(f'u     = {u}')
print(f'v     = {v}')
print(f'u + v = {u + v}')
```

```
u     = [3  2]
v     = [1 -1]
u + v = [4  1]
```

Vetores no NumPy

- Multiplicação escalar

```
u = np.array([3, 2])

print(f' u = {u}')
print(f'2u = {2*u}')
```

```
 u = [3 2]
2u = [6 4]
```

Vetores no NumPy

- Combinação linear

```
v1 = np.array([1,  0, -1])
v2 = np.array([2, -3,  1])
v3 = np.array([5, -4,  0])

print(f'v1 = {v1}')
print(f'v2 = {v2}')
print(f'v3 = {v3}')

print()
print(f'3v1 + 2v2 - v3 = {3*v1 + 2*v2 - v3}')
```

O vetor $\begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix}$ é uma combinação linear

dos vetores $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$, $\begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}$, e $\begin{bmatrix} 5 \\ -4 \\ 0 \end{bmatrix}$ uma vez que

$$3\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + 2\begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} - \begin{bmatrix} 5 \\ -4 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -1 \end{bmatrix}$$

```
v1 = [ 1  0 -1]
v2 = [ 2 -3  1]
v3 = [ 5 -4  0]

3v1 + 2v2 - v3 = [ 2 -2 -1]
```

Vetores no NumPy

- Produto escalar

```
u = np.array([ 1, 2, -3])
v = np.array([-3, 5,  2])

print(f'u = {u}')
print(f'v = {v}')

print()
print(f'u.dot(v) = {u.dot(v)}')
print(f'u@v = {u@v}')
```

Exemplo: Sejam

$$\mathbf{u} = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix} \quad \text{e} \quad \mathbf{v} = \begin{bmatrix} -3 \\ 5 \\ 2 \end{bmatrix}$$

Então $\mathbf{u} \cdot \mathbf{v} = 1 \cdot (-3) + 2 \cdot 5 + (-3) \cdot 2 = 1$

```
u = [ 1  2 -3]
v = [-3  5  2]

u.dot(v) = 1
u@v = 1
```

Vetores no NumPy

- Norma (módulo)

```
from numpy.linalg import norm

v = np.array([2, -1, 3])

print(f'v          = {v}')
print(f'norma(v) = {norm(v)}')
```

```
v          = [ 2 -1  3]
norma(v) = 3.7416573867739413
```

Vetores no NumPy

- Normalização de vetor

```
v = np.array([2, -1, 3])
u = (1/norm(v))*v

print(f'       v = {v}')
print(f'       u = {u}')
print(f'norma(u) = {norm(u)}')
```

```
       v = [ 2 -1  3]
       u = [ 0.53452248 -0.26726124  0.80178373]
norma(u) = 1.0
```

Matrizes

- Uma matriz é um vetor bidimensional
  - Matrizes são nomeadas com letras maiúsculas em itálico: *A*
  - Exemplos:

$$A = \begin{bmatrix} \sqrt{5} & -1 & 0 \\ 2 & \pi & 1/2 \end{bmatrix} \qquad C = [7]$$

$$E = [1 \quad 12 \quad 5]$$

$$B = \begin{bmatrix} 5.1 & 1.2 & -1 \\ 6.9 & 0 & 4.4 \\ -7.3 & 9 & 8.5 \end{bmatrix} \qquad F = \begin{bmatrix} -6 \\ 14 \end{bmatrix}$$

  - Numpy permite a realização de aritmética de matrizes

Aritmética de Matrizes

- Uma matriz nula possui todos os seus elementos iguais a zero.
  - Exemplo: $O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

- Uma matriz identidade é uma matriz quadrada que possui 1's na diagonal principal e 0's nas demais posições
  - Exemplo: $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Aritmética de Matrizes

- Adição
  - Só é possível somar matrizes que tem a mesma forma (mesmo número de linhas e colunas)
  - A matriz soma terá a mesma forma
  - Deve-se somar matrizes elemento a elemento correspondente
  - Exemplo:

$$A = \begin{bmatrix} 1 & 4 & 0 \\ -2 & 6 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} -3 & 1 & -1 \\ 3 & 0 & 2 \end{bmatrix}$$

$$A + B = \begin{bmatrix} -2 & 5 & -1 \\ 1 & 6 & 7 \end{bmatrix}$$

  - Para quaisquer matrizes, $A + B = B + A$

Aritmética de Matrizes

- Multiplicação escalar
  - Para multiplicar uma matriz por uma constante, basta multiplicar cada elemento da matriz pela constante
  - Exemplo:

$$A = \begin{bmatrix} 1 & 4 & 0 \\ -2 & 6 & 5 \end{bmatrix}$$

$$2A = \begin{bmatrix} 2 & 8 & 0 \\ -4 & 12 & 10 \end{bmatrix}$$

Multiplicação de Matrizes

Se A é uma matriz m x n e B é uma matriz n x r, então o produto $C = AB$ é uma matriz m x r.

O elemento (i, j) do produto c_{ij} é

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}$$

- Em outras palavras, o elemento c_{ij} do produto é o produto escalar da i-ésima linha de *A* e pela j-ésima coluna de *B*
- Se *A* é uma matriz quadrada n x n e i é a matriz identidade n x n, então $AI = IA = A$

Exemplo: Multiplicação de Matrizes

- João e Maria desejam comprar frutas, mas cada um tem uma lista com quantidades diferentes. Existem duas quitandas com preços diferentes. Quanto essas pessoas gastariam comprando em cada uma dessas quitandas?
  - Quantidade de frutas

	Maçã	Pera	Laranja
João	6	3	10
Maria	4	8	5

  - Preço das frutas

	Quit A	Quit B
Maçã	\$0.10	\$0.15
Pera	\$0.40	\$0.30
Laranja	\$0.10	\$0.20

Exemplo: Multiplicação de Matrizes

	Maçã	Pera	Laranja
João	6	3	10
Maria	4	8	5

	Quit A	Quit B
Maçã	\$0.10	\$0.15
Pera	\$0.40	\$0.30
Laranja	\$0.10	\$0.20

- João na Quitanda A: 6(0.10) + 3(0.40) + 10(0.10) = \$2.80
- João na Quitanda B: 6(0.15) + 3(0.30) + 10(0.20) = \$3.80
- Maria na Quitanda A: 4(0.10) + 8(0.40) + 5(0.10) = \$4.10
- Maria na Quitanda B: 4(0.15) + 8(0.30) + 5(0.20) = \$4.00
- Equivalente a: $\begin{bmatrix} 6 & 3 & 10 \\ 4 & 8 & 5 \end{bmatrix} \begin{bmatrix} 0.10 & 0.15 \\ 0.40 & 0.30 \\ 0.10 & 0.20 \end{bmatrix} = \begin{bmatrix} 2.80 & 3.80 \\ 4.10 & 4.00 \end{bmatrix}$

Matriz Inversa

- Uma matriz quadrada A n x n é inversível (ou não singular) se existir uma matriz B n x n tal que $AB = BA = I$. Neste caso, a matriz B é chamada matriz inversa de A
- Usaremos o Python para calcular a inversa de uma matriz, se ela existir

Matrizes no NumPy

- Zero

```
import numpy as np

print('np.zeros([3, 3]):')
print(np.zeros([3, 3]))
```

```
np.zeros([3, 3]):
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

- Identidade

```
print('np.identity(3):')
print(np.identity(3))
```

```
np.identity(3):
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Matrizes no NumPy

- Adição

```
A = np.array([[ 1, 4, 0],
              [-2, 6, 5]])
O = np.zeros([2, 3])

print('A:')
print(A)
print()
print('A + O:')
print(A + O)
print()
print('O + A:')
print(O + A)
```

```
A:
[[ 1  4  0]
 [-2  6  5]]

A + O:
[[ 1.  4.  0.]
 [-2.  6.  5.]]

O + A:
[[ 1.  4.  0.]
 [-2.  6.  5.]]
```

Matrizes no NumPy

- Adição

```
A = np.array([[ 1, 4, 0],
              [-2, 6, 5]])
B = np.array([[-3, 1, -1],
              [3, 0, 2]])
print('A:')
print(A)
print()
print('B:')
print(B)
print()
print('A + B:')
print(A + B)
print()
print('B + A:')
print(B + A)
```

```
A:
[[ 1  4  0]
 [-2  6  5]]

B:
[[-3  1 -1]
 [ 3  0  2]]

A + B:
[[-2  5 -1]
 [ 1  6  7]]

B + A:
[[-2  5 -1]
 [ 1  6  7]]
```

Matrizes no NumPy

- Multiplicação escalar

```
A = np.array([[ 1, 4, 0],
              [-2, 6, 5]])

print('A:')
print(A)
print()
print('2*A:')
print(2*A)
```

```
A:
[[ 1  4  0]
 [-2  6  5]]

2*A:
[[ 2  8  0]
 [-4 12 10]]
```

Matrizes no NumPy

- Multiplicação de matrizes

```
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
I = np.identity(3)

print('A:')
print(A)
print()
print('A@I:')
print(A@I)
print()
print('I@A:')
print(I@A)
```

```
A:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

A@I:
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]

I@A:
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

Matrizes no NumPy

- Multiplicação de matrizes

	Maçã	Pera	Laranja
João	6	3	10
Maria	4	8	5

	Quit A	Quit B
Maçã	$0.10	$0.15
Pera	$0.40	$0.30
Laranja	$0.10	$0.20

$$\begin{bmatrix} 6 & 3 & 10 \\ 4 & 8 & 5 \end{bmatrix} \begin{bmatrix} 0.10 & 0.15 \\ 0.40 & 0.30 \\ 0.10 & 0.20 \end{bmatrix} = \begin{bmatrix} 2.80 & 3.80 \\ 4.10 & 4.00 \end{bmatrix}$$

Matrizes no NumPy

$$\begin{bmatrix} 6 & 3 & 10 \\ 4 & 8 & 5 \end{bmatrix} \begin{bmatrix} 0.10 & 0.15 \\ 0.40 & 0.30 \\ 0.10 & 0.20 \end{bmatrix} = \begin{bmatrix} 2.80 & 3.80 \\ 4.10 & 4.00 \end{bmatrix}$$

```
F = np.array([[6, 3, 10],
              [4, 8,  5]])
P = np.array([[0.10, 0.15],
              [0.40, 0.30],
              [0.10, 0.20]])

print('Quant. frutas F:')
print(F)
print()
print('Preços P:')
print(P)
print()
print('Custo total F@S:')
print(F@P)
```

```
Quant. frutas F:
[[ 6  3 10]
 [ 4  8  5]]

Preços P:
[[0.1  0.15]
 [0.4  0.3 ]
 [0.1  0.2 ]]

Custo total F@P:
[[2.8 3.8]
 [4.1 4. ]]
```

Matrizes no NumPy

Matriz inversa

```
from numpy.linalg import inv
%precision 3

H = np.array([[1/(i + j - 1)
                for j in range(1, 6)]
                    for i in range(1, 6)])

print('H:')
print(H)
print()
print('inv(H)')
print(inv(H))
print()
print('H@inv(H):')
print(H@inv(H))
print()
print('inv(H)@H:')
print(inv(H)@H)
print()
print('inv(inv(H)):')
print(inv(inv(H)))
```

Hilbert matrix

$$H = \begin{pmatrix} 1 & 1/2 & 1/3 & \cdots & 1/n \\ 1/2 & 1/3 & 1/4 & \cdots & 1/n+1 \\ 1/3 & 1/4 & 1/5 & \cdots & 1/n+2 \\ \vdots & \vdots & \vdots & & \vdots \\ 1/n & 1/n+1 & 1/n+2 & \cdots & 1/2n-1 \end{pmatrix}$$

```
H =
[[1.    0.5   0.333 0.25  0.2  ]
 [0.5   0.333 0.25  0.2   0.167]
 [0.333 0.25  0.2   0.167 0.143]
 [0.25  0.2   0.167 0.143 0.125]
 [0.2   0.167 0.143 0.125 0.111]]
```

Matrizes no NumPy

- Matriz inversa

```
inv(H) =
[[ 2.500e+01 -3.000e+02  1.050e+03 -1.400e+03  6.300e+02]
 [-3.000e+02  4.800e+03 -1.890e+04  2.688e+04 -1.260e+04]
 [ 1.050e+03 -1.890e+04  7.938e+04 -1.176e+05  5.670e+04]
 [-1.400e+03  2.688e+04 -1.176e+05  1.792e+05 -8.820e+04]
 [ 6.300e+02 -1.260e+04  5.670e+04 -8.820e+04  4.410e+04]]

H@inv(H) =
[[ 1.000e+00 -5.037e-13  1.357e-12  4.760e-13 -2.380e-13]
 [ 1.312e-14  1.000e+00 -5.246e-13 -1.003e-12 -7.112e-13]
 [ 1.531e-14 -2.899e-13  1.000e+00  1.219e-12  4.298e-13]
 [ 0.000e+00 -2.274e-13  0.000e+00  1.000e+00  0.000e+00]
 [-7.278e-16  2.719e-14 -3.497e-13  1.150e-12  1.000e+00]]

inv(H)@H =
[[ 1.000e+00 -1.764e-13 -1.187e-13 -9.948e-14 -8.441e-14]
 [ 4.058e-13  1.000e+00  3.598e-13  4.547e-13  3.304e-13]
 [-4.464e-12 -3.556e-12  1.000e+00 -9.095e-13 -1.158e-12]
 [ 2.659e-12  1.422e-12  6.994e-13  1.000e+00  3.419e-13]
 [-2.380e-13  1.108e-12 -4.796e-13  0.000e+00  1.000e+00]]
```

Matrizes no NumPy

- Matriz inversa

```
inv(inv(H)) =
[[1.    0.5   0.333 0.25  0.2  ]
 [0.5   0.333 0.25  0.2   0.167]
 [0.333 0.25  0.2   0.167 0.143]
 [0.25  0.2   0.167 0.143 0.125]
 [0.2   0.167 0.143 0.125 0.111]]
```

```
H =
[[1.    0.5   0.333 0.25  0.2  ]
 [0.5   0.333 0.25  0.2   0.167]
 [0.333 0.25  0.2   0.167 0.143]
 [0.25  0.2   0.167 0.143 0.125]
 [0.2   0.167 0.143 0.125 0.111]]
```

Créditos

- Este conteúdo é uma tradução do original em inglês produzido pelo Prof. Ronald Mak (SJSU).