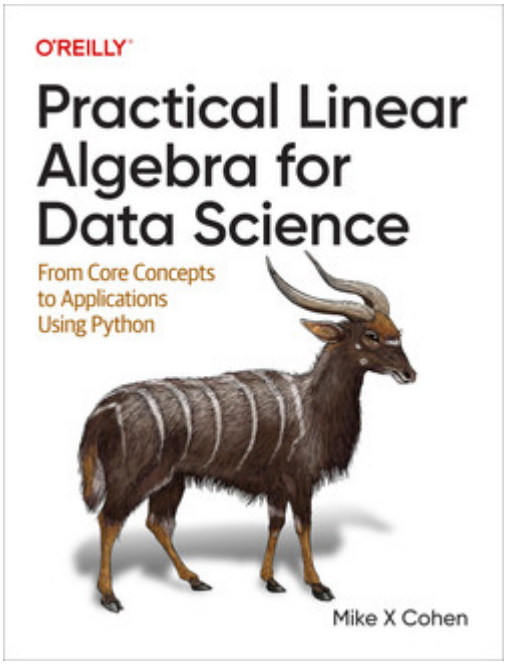
Matrizes - Parte 2



Bibliotecas

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy
```

Normas Matriciais

Existem multiplas normas distintas que podem ser calculadas de uma matriz. Normas matriciais são de alguma forma similares a normas vetoriais em que cada norma fornece um número qeu caracteriza a matriz.

A norma é indicada usando duas linhas verticais, por exemplo, a norma da matriz A é indicada por $\|\boldsymbol{A}\|$.

In [3]:
```
# Exemplo de norma vetorial
vetor = np.array([3, 4])
norma_vetorial = np.linalg.norm(vetor)
print(f"Norma vetorial: {norma_vetorial}")

# Exemplo de norma matricial
matriz = np.array([[1, 2], [3, 4]])
norma_matricial = np.linalg.norm(matriz, ord='fro')  # Norma de Frobenius
print(f"Norma matricial (Frobenius): {norma_matricial}")
```

```
Norma vetorial: 5.0
Norma matricial (Frobenius): 5.477225575051661
```

As normas matriciais podem ser amplamente divididas em duas famílias: **normas elementares** (ou de entrada) e **normas induzidas**.

As **normas elementares** são calculadas com base nos elementos individuais da matriz. Por isso, essas normas podem ser interpretadas como uma medida das magnitudes dos

elementos da matriz.

Por outro lado, as **normas induzidas** podem ser entendidas da seguinte forma: uma das funções de uma matriz é representar a transformação de um vetor. A norma induzida de uma matriz mede o quanto essa transformação escala (alongando ou contraindo) o vetor.

Este capítulo abordará as normas elementares. A **norma Euclidiana**, também chamada de **norma de Frobenius**, é uma extensão direta da norma vetorial. Ela é calculada como a raiz quadrada da soma de todos os elementos da matriz elevados ao quadrado:

$$\|\bold{A}\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} a_{i,j}^2}$$

Os índices i e j correspondem às M linhas e N colunas da matriz. A anotação subscrita $_F$ indica a norma de Frobenius.

Essa norma também é chamada de **norma** ℓ_2, pois é um caso especial da fórmula geral para normas-p elementares:

$$\|\bold{A}\|_p = (\sum_{i=1}^{M}\sum_{j=1}^{N} |a_{i,j}|^p)^{\frac{1}{p}}$$

Perceba que você obtém a norma de Frobenius quando $p = 2$

Uma das aplicações mais importantes das normas matriciais na análise estatística e no machine learning é sua utilização na regularização, cujo objetivo é melhorar o ajuste do modelo e aumentar sua capacidade de generalização para novos dados. A ideia básica da regularização é adicionar uma norma matricial como uma função de custo a um algoritmo de minimização. Essa norma ajuda a:

- **Evitar que os parâmetros do modelo se tornem muito grandes** (a regularização ℓ_2, também conhecida como regressão Ridge).
- **Incentivar soluções esparsas** (a regularização ℓ_1, também conhecida como regressão Lasso).

As arquiteturas modernas de **deep learning** dependem amplamente das normas matriciais para alcançar altos desempenhos na resolução de problemas, como os de visão computacional.

Outra aplicação da norma de Frobenius é medir a **distância entre matrizes**. A distância matricial de uma matriz consigo mesma é 0, e a distância entre duas matrizes distintas aumenta à medida que os valores numéricos dessas matrizes se tornam mais diferentes.

Essa métrica pode ser usada como critério de otimização em **machine learning**, por exemplo, para reduzir o tamanho de armazenamento de dados de uma imagem enquanto minimiza a distância de Frobenius entre a matriz reduzida e a matriz original.

In [4]:
```
# Definindo duas matrizes
matriz1 = np.array([[1, 2], [3, 4]])
```

```
matriz2 = np.array([[2, 3], [4, 5]])

# Calculando a distância de Frobenius
distancia = np.linalg.norm(matriz1 - matriz2, ord='fro')
print(f"Distância de Frobenius entre as matrizes: {distancia}")
```

```
Distância de Frobenius entre as matrizes: 2.0
```

Traço de Matriz e Norma de Frobenius

O **traço** de uma matriz é a soma de seus elementos diagonais, indicado por $tr(\boldA)$, e só existe para matrizes quadradas. Por exemplo, as duas matrizes a seguir possuem o mesmo traço (14):

$$\begin{bmatrix} 4 & 5 & 6 \\ 0 & 1 & 4 \\ 9 & 9 & 9 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 8 & 0 \\ 1 & 2 & 6 \end{bmatrix}$$

O traço possui algumas propriedades interessantes. Por exemplo:

- O traço de uma matriz é igual à soma de seus autovalores, sendo, portanto, uma medida do volume de seu autoespaço.
- Ele é invariante sob transformações de similaridade, ou seja, $tr(\boldA) = tr(\boldP^{-1}\boldA\boldP)$ para qualquer matriz invertível $\boldP$.

Outra propriedade importante relaciona o traço com a norma de Frobenius:

$$\|\boldA\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} a_{i,j}^2} = \sqrt{tr(A^T A)}$$

A norma de Frobenius pode ser calculada como a raiz quadrada do traço do produto da matriz transposta por ela mesma. Isso ocorre porque cada elemento diagonal da matriz $\boldA^T\boldA$ é definido pelo produto escalar de cada linha da matriz $\boldA$ com ela mesma.

In [5]:
```
# Exemplo de cálculo do traço
matriz = np.array([[4, 5, 6], [0, 1, 4], [9, 9, 9]])
traco = np.trace(matriz)
print(f"Traço da matriz: {traco}")
```

```
Traço da matriz: 14
```

Espaços Matriciais (colunas, linhas e nulos)

Os **espaços matriciais** são conceitos fundamentais em álgebra linear e podem ser entendidos como combinações lineares ponderadas das características de uma matriz. Esses espaços incluem:

- **Espaço das colunas**: O conjunto de todas as combinações lineares das colunas de uma matriz. Ele representa o subespaço gerado pelos vetores coluna.
- **Espaço das linhas**: O conjunto de todas as combinações lineares das linhas de uma matriz. Ele representa o subespaço gerado pelos vetores linha.
- **Espaço nulo (ou núcleo)**: O conjunto de todos os vetores que, quando

multiplicados pela matriz, resultam no vetor nulo. Ele descreve as soluções do sistema homogêneo $A\mathbf{x} = 0$.

Esses espaços são úteis para entender propriedades como a dimensão, o posto e a nulidade de uma matriz, além de serem amplamente aplicados em problemas de otimização, análise de dados e aprendizado de máquina.

Espaço colunar

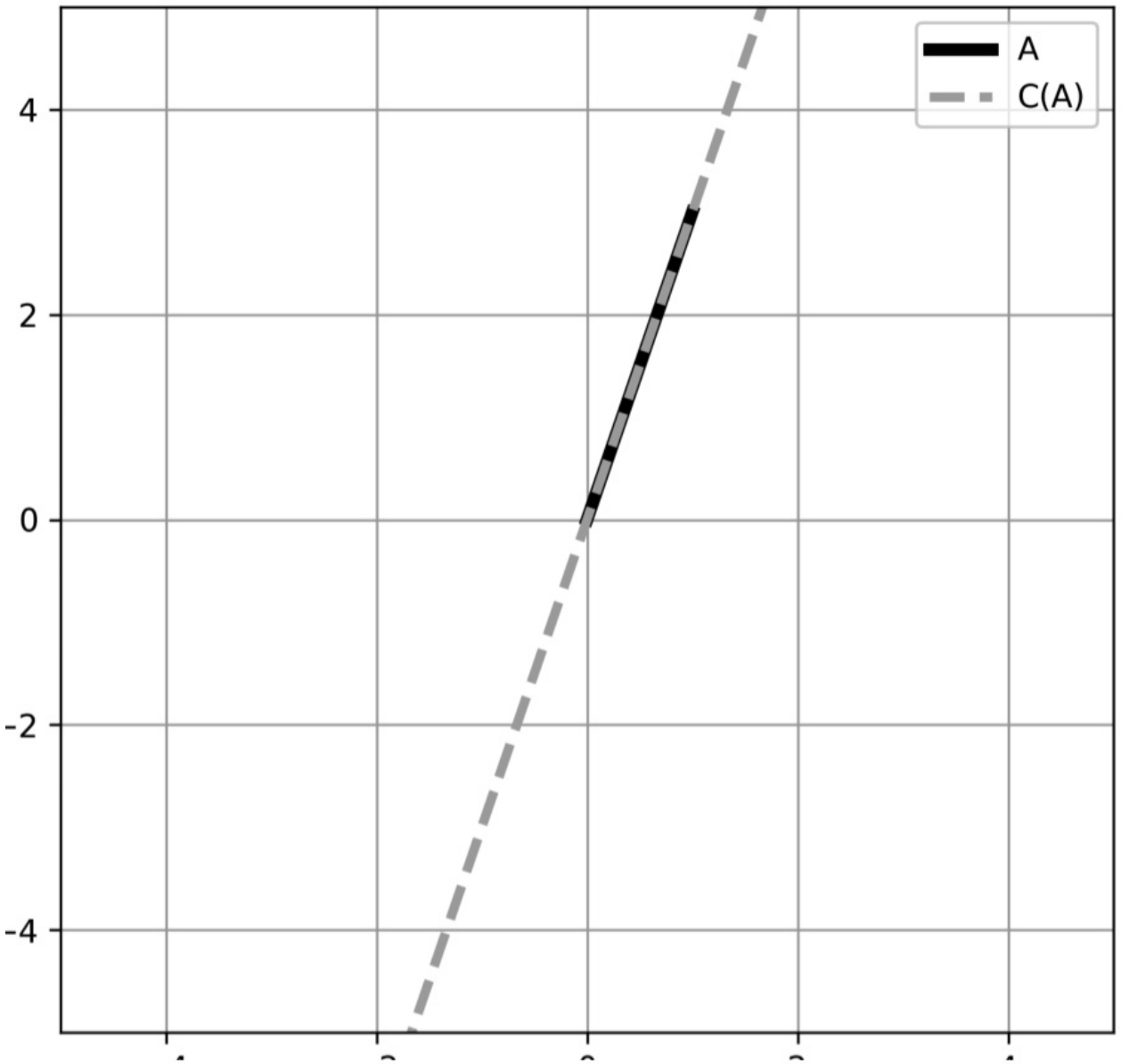
Primeiro, conceitualizamos uma matriz como um conjunto de vetores coluna. Em seguida, consideramos um conjunto infinito de valores reais escalares, em vez de trabalhar com um conjunto específico de escalares. Isso resulta em infinitas formas de combinar os vetores de um conjunto. O conjunto resultante é chamado de espaço colunar de uma matriz.

$$C\left(\begin{bmatrix}1\\3\end{bmatrix}\right) = \lambda\begin{bmatrix}1\\3\end{bmatrix}, \quad \lambda \in \mathbb{R}$$

O $C(\boldA)$ indica o espaço colunar da matriz $\boldA$.

Por exemplo, o vetor $\begin{bmatrix}-2\\-6\end{bmatrix}$ está no espaço colunar, pois pode ser expresso com $\lambda = -2$. No entanto, o vetor $\begin{bmatrix}1\\4\end{bmatrix}$ não pertence ao espaço colunar da matriz, pois não existe um escalar que, ao multiplicar a matriz, produza esse vetor.

Para uma matriz com uma única coluna, o espaço colunar é uma linha que passa pela origem na direção do vetor no espaço vetorial e se estende ao infinito em ambas as direções.



In [6]:
```python
# Definindo uma matriz
matriz = np.array([[1, 2], [3, 6]])

# Verificando se um vetor pertence ao espaço colunar
vetor = np.array([4, 12])
combinacao = np.linalg.lstsq(matriz, vetor, rcond=None)[0]
print(f"Combinação linear para gerar o vetor: {combinacao}")
```

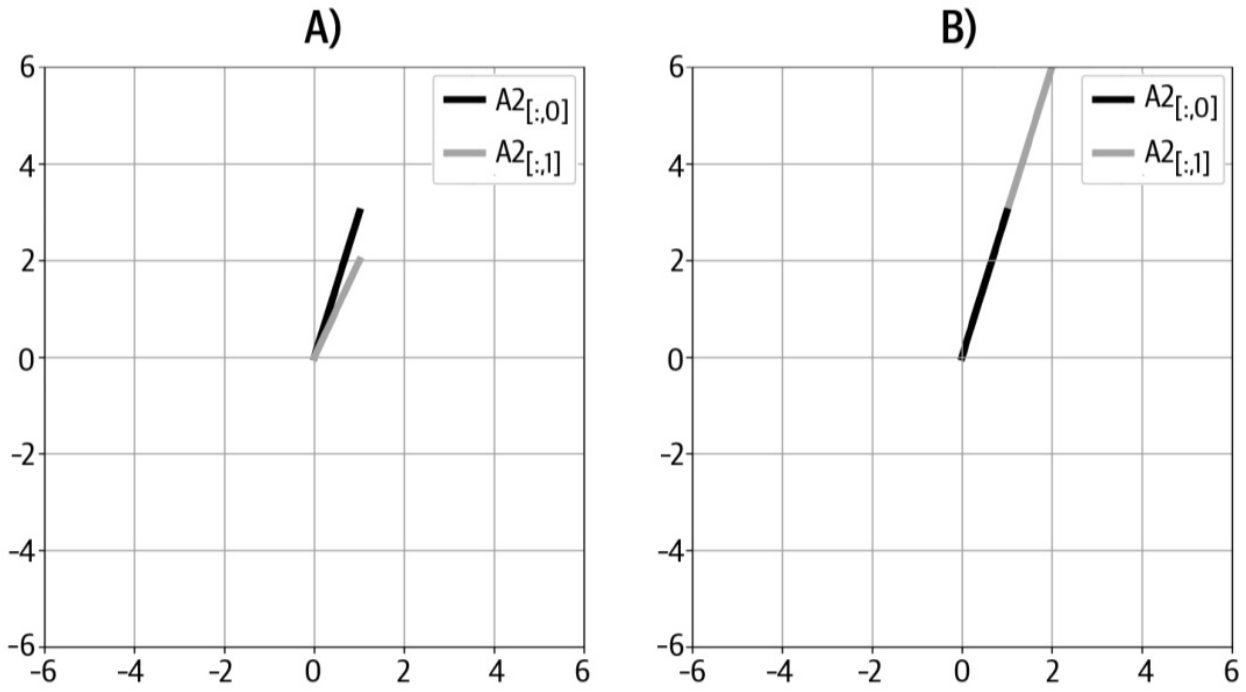
```
Combinação linear para gerar o vetor: [0.8 1.6]
```

Considerando uma matriz com mais colunas:

$$C\left(\begin{bmatrix}1 & 1\\ 3 & 2\end{bmatrix}\right) = \lambda_1 \begin{bmatrix}1\\3\end{bmatrix} \lambda_2 \begin{bmatrix}1\\2\end{bmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{R}$$

Com duas colunas, temos dois λ diferentes. O conjunto de todos os vetores que podem ser alcançados por alguma combinação linear desses dois vetores coluna é o espaço $\mathbb{R}^2$.

Por exemplo, o vetor $\begin{bmatrix}-4\\3\end{bmatrix}$ pode ser obtido escalonando as duas colunas por 11 e -15, respectivamente (esses escalares foram obtidos pelo método de projeção dos mínimos quadrados).



Outro exemplo é:

$$C\left(\begin{bmatrix}1 & 2\\ 3 & 6\end{bmatrix}\right) = \lambda_1 \begin{bmatrix}1\\3\end{bmatrix} + \lambda_2 \begin{bmatrix}2\\6\end{bmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{R}$$

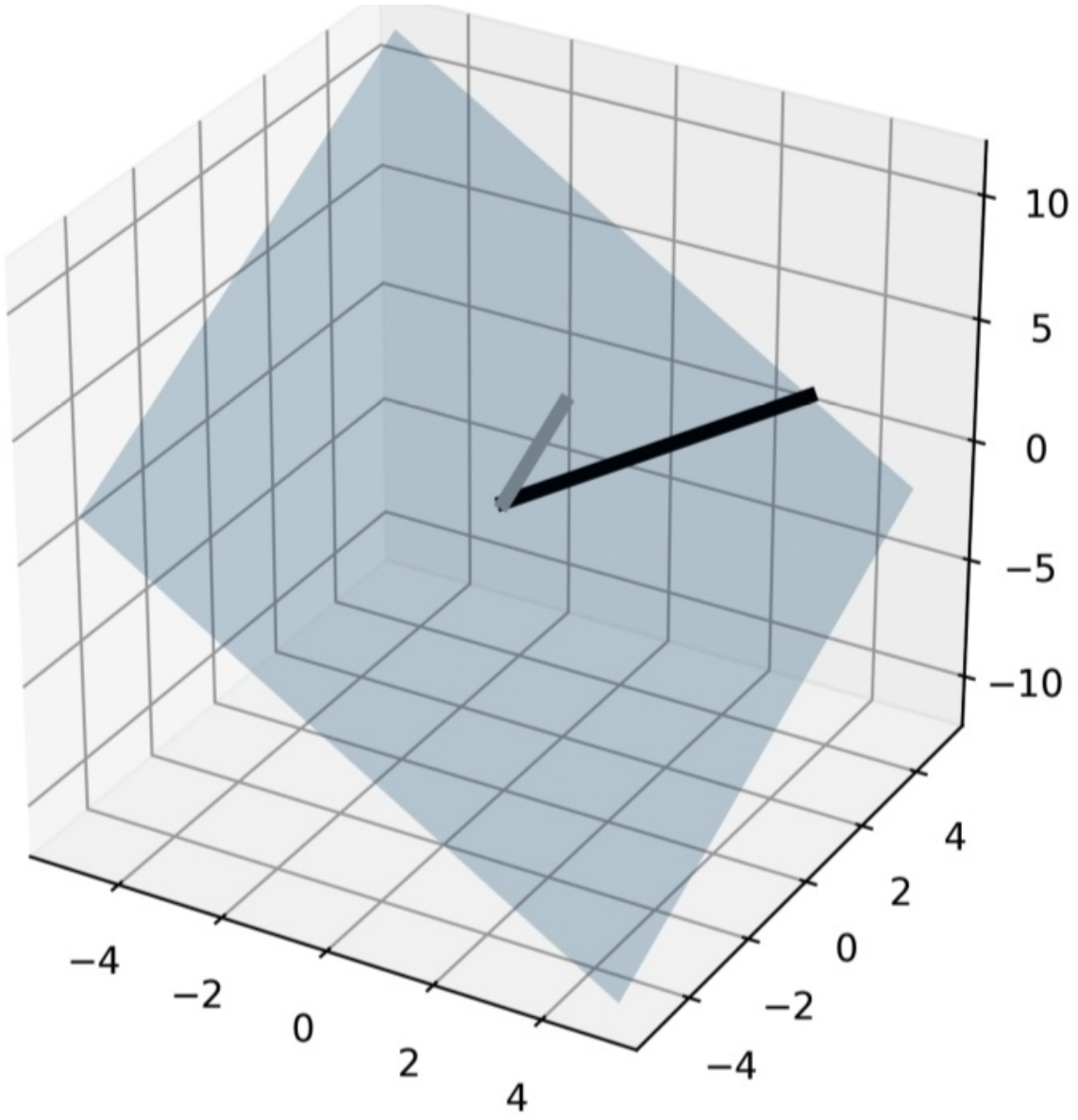
Essas duas colunas são colineares, pois uma é uma versão escalada da outra. Isso significa que o espaço colunar dessa matriz 2×2 é apenas uma linha — um subespaço 1D.

A dimensionalidade do espaço colunar é igual ao número de colunas **apenas se as colunas forem linearmente independentes**.

Um ultimo exemplo:

$$C\left(\begin{bmatrix}3 & 0\\ 5 & 2\\ 1 & 2\end{bmatrix}\right) = \lambda_1 \begin{bmatrix}3\\5\\1\end{bmatrix} + \lambda_2 \begin{bmatrix}0\\2\\2\end{bmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{R}$$

Agora há 2 colunas em $\mathbb{R}^3$. Essas colunas são **linearmente independentes**, o que significa que o espaço colunar dessa matriz está em 2D. No entanto, esse plano 2D está embutido em $\mathbb{R}^3$.



Há muitos vetores no plano, mas há muito mais vetores fora dele.

Espaço linear

O espaço linear de uma matriz segue o mesmo conceito do espaço colunar, mas, em vez de considerar as combinações lineares das colunas, analisamos as combinações lineares das linhas da matriz.

$$R(A) = C(A)^T$$

O espaço linear é invariante sob operações de redução de linha. Isso significa que, ao realizar operações elementares nas linhas da matriz, o espaço linear permanece o mesmo.

Para matrizes simétricas, os espaços linear e colunar são idênticos, ou seja:

$$R(A) = C(A)^T$$

Espaços nulos

O espaço nulo de uma matriz, também chamado de núcleo, é o conjunto de todos os vetores que, ao serem multiplicados pela matriz, resultam no vetor nulo. Ele pode ser representado pela equação:

$$A\mathbf{y} = 0$$

Isso significa que o espaço nulo contém todos os vetores $\mathbf{y}$ que satisfazem essa equação. Em contraste, o espaço colunar está relacionado à equação:

$$A\mathbf{x} = \mathbf{b}$$

Uma pergunta importante ao analisar o espaço nulo é: "Podemos encontrar um conjunto de pesos que não sejam todos zero e que produzam o vetor nulo?"

Se tal conjunto de pesos existir, ele estará no espaço nulo da matriz $\boldA$, que é denotado por $N(A)$.

Há uma relação direta entre a dimensionalidade do espaço nulo e a independência linear das colunas de uma matriz:

- O espaço nulo é **vazio** (contém apenas o vetor nulo) quando as colunas da matriz são **linearmente independentes**.
- Caso contrário, o espaço nulo terá dimensão maior que zero, indicando que existem combinações lineares não triviais das colunas que resultam no vetor nulo.

In [7]:
```
A = np.array([[1, -1], [-2, 2]])
B = np.array([[1, -1], [-2, 3]])

print("Espaço nulo da matriz A:")
print(scipy.linalg.null_space(A))

print("\nEspaço nulo da matriz B:")
print(scipy.linalg.null_space(B))
```

```
Espaço nulo da matriz A:
[[0.70710678]
 [0.70710678]]

Espaço nulo da matriz B:
[]
```

Dado o número infinito de vetores possíveis no espaço nulo, o Python retorna um vetor unitário como representante. Vetores unitários são convenientes para trabalhar, pois possuem propriedades úteis, como:

- **Estabilidade numérica**: Operações com vetores unitários tendem a ser mais estáveis em cálculos computacionais.
- **Normalização**: O vetor unitário tem norma igual a 1, o que facilita comparações e análises.

Rank (Posto)

O **posto** de uma matriz é um número associado a ela que está relacionado à dimensionalidade dos subespaços gerados por suas linhas ou colunas. Ele possui importantes implicações para operações matriciais, como a inversão de matrizes e a determinação do número de soluções de um sistema de equações lineares.

Propriedades do Posto:

- É um número inteiro não negativo
- Toda matriz possui um único posto (o posto é uma característica da matriz, não de suas linhas ou colunas individualmente).
- É indicado por $r(\boldA)$ ou $rank(\boldA)$.
- O posto máximo possível de uma matriz é igual ao menor valor entre o número de linhas e o número de colunas.
- Uma matriz com o posto máximo possível é chamada de **matriz de posto completo** (full-rank). Se $r < \min(M, N)$, ela é chamada de **matriz de posto reduzido**.
- Multiplicações escalares não afetam o posto da matriz (exceto quando o escalar é zero).

Interpretações e Definições do Posto:

- O maior número de colunas (ou linhas) que formam um conjunto linearmente independente.
- A dimensionalidade do espaço colunar (que é igual à dimensionalidade do espaço das linhas).
- O número de dimensões que contêm informações na matriz.
- A quantidade de valores singulares diferentes de zero na matriz.

In [43]:
```
# Definindo uma matriz
matriz = np.random.randint(1, 10, (3, 3))

# Calculando o posto da matriz
posto = np.linalg.matrix_rank(matriz)

print(f"Matriz:\n{matriz}")
print(f"\nPosto da matriz: {posto}")
```

```
Matriz:
[[7 1 3]
 [4 3 1]
 [4 7 8]]

Posto da matriz: 3
```

Postos de Matrizes Especiais

Vetores:

Todos os vetores têm posto 1. Isso ocorre porque vetores possuem apenas uma coluna (ou linha) de informação. A única exceção é o vetor zero, que tem posto 0.

In [45]:
```
# Vetor (posto 1)
vetor = np.array([[1], [2], [3]])
posto_vetor = np.linalg.matrix_rank(vetor)
print(f"Vetor:\n{vetor}")
```

```
print(f"\nPosto: {posto_vetor}")
```

```
Vetor:
[[1]
 [2]
 [3]]

Posto: 1
```

Matriz zero:

Uma matriz zero de qualquer tamanho (incluindo o vetor zero) tem posto 0.

In [46]:
```
# Matriz zero (posto 0)
matriz_zero = np.zeros((3, 3))
posto_matriz_zero = np.linalg.matrix_rank(matriz_zero)
print(f"Matriz zero:\n{matriz_zero}")
print(f"\nPosto: {posto_matriz_zero}")
```

```
Matriz zero:
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

Posto: 0
```

Matrizes identidade:

O posto de uma matriz identidade é igual ao número de linhas (ou colunas). Ou seja, $r(\boldI_N) = N$.

In [47]:
```
# Matriz identidade (posto igual ao número de linhas/colunas)
matriz_identidade = np.eye(4)
posto_matriz_identidade = np.linalg.matrix_rank(matriz_identidade)
print(f"Matriz identidade:\n{matriz_identidade}")
print(f"\nPosto: {posto_matriz_identidade}")
```

```
Matriz identidade:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

Posto: 4
```

Matrizes diagonais:

O posto de uma matriz diagonal é igual à quantidade de elementos diferentes de zero na diagonal. Isso ocorre porque cada linha contém, no máximo, um elemento diferente de zero, e é impossível criar um número diferente de zero por meio de combinações ponderadas de zeros.

In [48]:
```
# Matriz diagonal (posto igual ao número de elementos diferentes de zero na diagona
matriz_diagonal = np.diag([1, 0, 3, 0])
posto_matriz_diagonal = np.linalg.matrix_rank(matriz_diagonal)
print(f"Matriz diagonal:\n{matriz_diagonal}")
print(f"\nPosto: {posto_matriz_diagonal}")
```

```
Matriz diagonal:
[[1 0 0 0]
 [0 0 0 0]
 [0 0 3 0]
 [0 0 0 0]]

Posto: 2
```

Matrizes triângulares:

Uma matriz triangular tem posto completo apenas se todos os elementos da diagonal forem diferentes de zero. Caso haja pelo menos um zero na diagonal, a matriz será de posto reduzido. O posto exato dependerá dos valores numéricos da matriz.

In [49]:
```
# Matriz triangular (posto depende dos elementos da diagonal)
matriz_triangular = np.array([[1, 2, 3], [0, 4, 5], [0, 0, 0]])
posto_matriz_triangular = np.linalg.matrix_rank(matriz_triangular)
print(f"Matriz triangular:\n{matriz_triangular}")
print(f"\nPosto: {posto_matriz_triangular}")
```

```
Matriz triangular:
[[1 2 3]
 [0 4 5]
 [0 0 0]]

Posto: 2
```

Matrizes aleátorias:

O posto de uma matriz aleatória é impossível de determinar a priori, pois depende da distribuição dos números dos quais os elementos da matriz foram extraídos e da probabilidade de extrair cada número. Matrizes criadas com np.random.randn() geralmente têm o maior posto possível.

In [57]:
```
# Matriz aleatória (posto geralmente é o máximo possível)
matriz_aleatoria = np.random.randn(3, 3)
posto_matriz_aleatoria = np.linalg.matrix_rank(matriz_aleatoria)
print(f"Matriz aleatória:\n{matriz_aleatoria}")
print(f"\nPosto: {posto_matriz_aleatoria}")
```

```
Matriz aleatória:
[[-0.59986625  0.92728525  0.49024393]
 [-1.53753592  1.03575983  0.10257277]
 [ 0.88263703 -0.16548291  1.44483606]]

Posto: 3
```

Posto de Matrizes Adicionadas e Multiplicadas

O posto de duas matrizes individuais fornece limites para o máximo posto possível das matrizes resultantes de suas operações:

$$\text{rank}(\boldsymbol{A} + \boldsymbol{B}) \leq \text{rank}(\boldsymbol{A}) + \text{rank}(\boldsymbol{B})$$

$$\text{rank}(\boldsymbol{AB}) \leq \min \text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B})$$

Pontos Importantes:

1. **Limites superiores:**

   - O posto de uma matriz somada ou multiplicada não pode ser maior que os

limites definidos pelas propriedades acima.

2. **Posto de matrizes somadas:**

   - O posto de uma matriz somada pode ser maior que o posto das matrizes individuais, mas nunca excederá a soma dos postos.

3. **Posto de matrizes multiplicadas:**

   - O posto de uma matriz multiplicada não pode ser maior que o menor posto entre as matrizes multiplicadas.

In [180…

```python
# Definindo duas matrizes
A = np.random.randint(1, 10, (2, 2))
B = np.random.randint(1, 10, (2, 2))

# Calculando os postos das matrizes individuais
posto_A = np.linalg.matrix_rank(A)
posto_B = np.linalg.matrix_rank(B)

# Calculando o posto da soma das matrizes
posto_soma = np.linalg.matrix_rank(A + B)

# Calculando o posto do produto das matrizes
posto_produto = np.linalg.matrix_rank(A @ B)

# Exibindo os resultados
print(f"Matriz A:\n{A}")
print(f"Posto de A: {posto_A}\n")

print(f"Matriz B:\n{B}")
print(f"Posto de B: {posto_B}\n")

print(f"Soma (A + B):\n{A + B}")
print(f"Posto da soma (A + B): {posto_soma}\n")

print(f"Produto (A @ B):\n{A @ B}")
print(f"Posto do produto (A @ B): {posto_produto}")
```

```
Matriz A:
[[4 3]
 [8 7]]
Posto de A: 2

Matriz B:
[[3 2]
 [2 7]]
Posto de B: 2

Soma (A + B):
[[ 7  5]
 [10 14]]
Posto da soma (A + B): 2

Produto (A @ B):
[[18 29]
 [38 65]]
Posto do produto (A @ B): 2
```

Posto de Matrizes Deslocadas

Simplificando, matrizes deslocadas possuem posto completo. Um dos principais objetivos de deslocar uma matriz quadrada é aumentar seu posto de $r < M$ para $r = M$, onde M

é o número de linhas (ou colunas) da matriz.

In [181…
```python
# Matriz original com posto reduzido
matriz_reduzida = np.array([[1, 2], [2, 4]])
posto_original = np.linalg.matrix_rank(matriz_reduzida)

# Adicionando um deslocamento (matriz identidade escalada)
deslocamento = np.eye(2)  # Matriz identidade 2x2
matriz_deslocada = matriz_reduzida + deslocamento
posto_deslocado = np.linalg.matrix_rank(matriz_deslocada)

# Exibindo os resultados
print("Matriz original (posto reduzido):")
print(matriz_reduzida)
print(f"Posto original: {posto_original}\n")

print("Matriz deslocada (posto completo):")
print(matriz_deslocada)
print(f"Posto deslocado: {posto_deslocado}")
```

```
Matriz original (posto reduzido):
[[1 2]
 [2 4]]
Posto original: 1

Matriz deslocada (posto completo):
[[2. 2.]
 [2. 5.]]
Posto deslocado: 2
```

Aplicações de Posto

No espaço coluna

Antes de explicar o algoritmo para determinar se $v \in C(\bold{A})$, é necessário entender o procedimento chamado aumento de uma matriz. Aumentar uma matriz significa adicionar colunas extras ao lado direito da matriz original. Por exemplo:

$$\begin{bmatrix} 4 & 5 & 6 \\ 0 & 1 & 2 \\ 9 & 9 & 4 \end{bmatrix} \cup \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 & 1 \\ 0 & 1 & 2 & 2 \\ 9 & 9 & 4 & 3 \end{bmatrix}$$

O algoritmo para determinar se um vetor pertence ao espaço coluna da matriz é o seguinte:

1. Aumente a matriz com o vetor: Adicione o vetor como uma nova coluna à matriz original. Vamos chamar a matriz original de $\bold{A}$ e a matriz aumentada de $\bold{\tilde{A}}$.
2. Calcule o posto das duas matrizes:
3. Compare os dois postos:
    - Se $\text{rank}(\bold{A}) = \text{rank}(\bold{\tilde{A}})$, o vetor v pertence ao espaço coluna da matriz $\bold{A}$.
    - Se $\text{rank}(\bold{A}) < \text{rank}(\bold{\tilde{A}})$, o vetor v não pertence ao espaço coluna da matriz $\bold{A}$.

In [182…
```python
# Definindo a matriz A
```

```python
A = np.array([[4, 5, 6],
              [0, 1, 2],
              [9, 9, 4]])

# Definindo o vetor v
v = np.array([1, 2, 3])

# Aumentando a matriz A com o vetor v
A_aumentada = np.column_stack((A, v))

# Calculando os postos
posto_A = np.linalg.matrix_rank(A)
posto_A_aumentada = np.linalg.matrix_rank(A_aumentada)

# Verificando se o vetor pertence ao espaço coluna
if posto_A == posto_A_aumentada:
    print("O vetor v pertence ao espaço coluna da matriz A.")
else:
    print("O vetor v NÃO pertence ao espaço coluna da matriz A.")

# Exibindo os resultados
print("\nMatriz A:")
print(A)
print("\nVetor v:")
print(v)
print("\nMatriz A aumentada:")
print(A_aumentada)
print(f"\nPosto de A: {posto_A}")
print(f"Posto de A aumentada: {posto_A_aumentada}")
```

```
O vetor v pertence ao espaço coluna da matriz A.

Matriz A:
[[4 5 6]
 [0 1 2]
 [9 9 4]]

Vetor v:
[1 2 3]

Matriz A aumentada:
[[4 5 6 1]
 [0 1 2 2]
 [9 9 4 3]]

Posto de A: 3
Posto de A aumentada: 3
```

Independência Linear de um Conjunto Vetor

O algoritmo para verificar a independência linear de um conjunto de vetores é simples: organize os vetores em uma matriz, calcule o posto dessa matriz e compare o posto ao número máximo de colunas (ou linhas) da matriz.

- Se $r = M$, o conjunto de vetores é **linearmente independente**.
- Se $r < M$, o conjunto de vetores é **linearmente dependente**.

In [183…

```python
# Definindo os vetores
v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
v3 = np.array([7, 8, 9])
```

```python
# Organizando os vetores em uma matriz
matriz = np.column_stack((v1, v2, v3))

# Calculando o posto da matriz
posto = np.linalg.matrix_rank(matriz)

# Verificando a independência linear
if posto == matriz.shape[1]:  # Número de colunas
    print("Os vetores são linearmente independentes.")
else:
    print("Os vetores são linearmente dependentes.")

# Exibindo os resultados
print("\nMatriz formada pelos vetores:")
print(matriz)
print(f"\nPosto da matriz: {posto}")
```

```
Os vetores são linearmente dependentes.

Matriz formada pelos vetores:
[[1 4 7]
 [2 5 8]
 [3 6 9]]

Posto da matriz: 2
```

Determinante

A determinante é um número associado a uma matriz quadrada. Na álgebra linear, ela é uma peça fundamental para muitas operações, como a inversão de matrizes e a solução de sistemas lineares. No entanto, calcular a determinante na prática pode ser numericamente instável para matrizes grandes, devido a problemas de subfluxo ou excesso de valores numéricos.

As duas propriedades mais importantes das determinantes são:

1. Ela só é válida para matrizes quadradas.
2. Seu valor é zero para matrizes de posto reduzido.

A determinante é representada como $\det(\boldsymbol{A})$ ou $|\boldsymbol{A}|$. A letra grega maiúscula delta (Δ) também pode ser usada quando não é necessário fazer referência a uma matriz específica.

A determinante possui uma interpretação geométrica: ela mede o fator pelo qual uma matriz escala o volume de um espaço vetorial durante a multiplicação matriz-vetor. Um valor negativo indica que a transformação inclui uma inversão (ou rotação) de um eixo.

Calcular a Determinante

A determinante de uma matriz 2x2 é calculada utilizando a seguinte fórmula:

$$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

Aqui, o valor da determinante é obtido subtraindo o produto dos elementos da diagonal secundária (bc) do produto dos elementos da diagonal principal (ad).

In [184…
```
# Definindo a matriz 2x2
matriz = np.array([[2, 3],
                   [4, 5]])

# Calculando a determinante
determinante = np.linalg.det(matriz)

# Exibindo os resultados
print("Matriz:")
print(matriz)
print(f"\nDeterminante da matriz: {determinante}")
```

```
Matriz:
[[2 3]
 [4 5]]

Determinante da matriz: -2.0
```

Determinante com Dependências Lineares

Como foi dito anteriormente, a determinante para qualquer matriz reduziada é zero

$$\begin{vmatrix} a & \lambda a \\ c & \lambda c \end{vmatrix} = ac\lambda - a\lambda c = 0$$

Esse conceito generaliza para matrizes maiores

In [186…
```
# Definindo uma matriz com dependências lineares
# A segunda coluna é um múltiplo da primeira
matriz = np.array([[1, 2],
                   [3, 6]])

# Calculando a determinante
determinante = np.linalg.det(matriz)

# Exibindo os resultados
print("Matriz:")
print(matriz)
print(f"\nDeterminante da matriz: {determinante}")
```

```
Matriz:
[[1 2]
 [3 6]]

Determinante da matriz: 0.0
```

A Característica Polinomial

A combinação da matriz deslocada com a determinante é chamada de **polinômio característico** da matriz. Ele é definido como:

$$\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = \Delta$$

Por exemplo, considere a matriz:

$$\det(\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix} \lambda\boldsymbol{I}) = 0$$

Expandindo o determinante:

$$\begin{vmatrix} 1-\lambda & 3 \\ 3 & 1-\lambda \end{vmatrix} = 0 \rightarrow (1-\lambda)^2 - 9 = 0 \rightarrow 1 - \lambda = \pm 3$$

Os valores próprios (autovalores) são:

$$\lambda_1 = -2 \quad \text{e} \quad \lambda_2 = 4$$

Substituindo os autovalores, obtemos as matrizes associadas:

Para ($\lambda_1 = -2$):

$$\begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$$

Para ($\lambda_2 = 4$):

$$\begin{bmatrix} -3 & 3 \\ 3 & -3 \end{bmatrix}$$

Ambas as matrizes têm posto 1. Além disso, possuem espaços nulos não triviais, o que significa que existem vetores ($\mathbf{y} \neq 0$) que satisfazem:

$$(\bold{A} - \lambda \bold{I})\mathbf{y} = 0$$

Referência

https://www.oreilly.com/library/view/practical-linear-algebra/9781098120603/