

Álgebra lineal en Python

Escalar:

Es un número, a diferencia de la mayoría de los otros objetos estudiados en Álgebra Lineal, que son generalmente una colección de múltiples números.

Vector:

Es una serie de números. Los números tienen un orden preestablecido, y podemos identificar cada número individual por su índice en ese orden. Podemos pensar en los vectores como la identificación de puntos en el espacio, con cada elemento que da la coordenada a lo largo de un eje diferente. Existen dos tipos de vectores, los vectores de fila y los vectores de columna.

$$f = [0 \quad 1 \quad -1] ; c = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

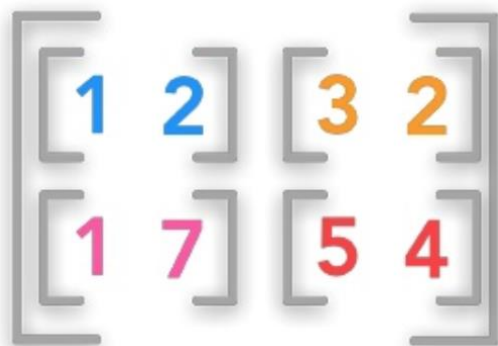
Matriz:

Es un arreglo bidimensional de números ordenados en renglones y columnas, donde una fila es cada una de las líneas horizontales de la matriz y una columna es cada una de las líneas verticales. En una matriz cada elemento puede ser identificado utilizando dos índices, uno para la fila y otro para la columna en que se encuentra. Las podemos representar de la siguiente manera, A es una matriz de 3x2.

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 2 \\ -2 & 3 \end{bmatrix}$$

Tensor:

En algunos casos necesitaremos una matriz con más de dos ejes. En general, una serie de números dispuestos en una cuadrícula regular con un número variable de ejes es conocido como un tensor.


$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$$

Vectores

```
>>>v1 = [2, 4, 6]
>>>v1
[2, 4, 6]

>>>import numpy as np
>>>v2 = np.ones(3)
>>>v2
array([1., 1., 1.])

>>>v3 = np.array([1, 3, 5])
>>>v3
array([1, 3, 5])

>>>v4 = np.arange(1, 8)
>>>v4
array([1, 2, 3, 4, 5, 6, 7])
```

Operaciones con vectores

Las operaciones más comunes que utilizamos cuando trabajamos con vectores son la suma, la resta y la multiplicación por escalares.

Cuando sumamos dos vectores, vamos sumando elemento por elemento de cada vector.

$$x + y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} := \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

De forma similar funciona la operación de resta.

$$x - y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} := \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \\ \vdots \\ x_n - y_n \end{bmatrix}$$

La multiplicación por escalares es una operación que toma a un número γ , y a un vector x y produce un nuevo vector donde cada elemento del vector x es multiplicado por el número γ .

$$\gamma x := \begin{bmatrix} \gamma x_1 \\ \gamma x_2 \\ \vdots \\ \gamma x_n \end{bmatrix}$$

```

>>>x = np.arange(1,5)
>>>y = np.array([2,4,6,8])
>>>x, y
(array([1,2,3,4]), array([2,4,6,8]))

>>>x+y
array([3,6,9,12])

>>>x-y
array([-1,-2,-3,-4])

>>>x*2
array([2,4,6,8])

>>>y*3
array([6,12,18,24])

```

Producto escalar o interior.

El producto escalar de dos vectores se define como la suma de cada elemento de x multiplicado por cada elemento de y. Cuando multiplicamos vectores, nos resulta otro vector, a este vector hacemos sumatoria de todos sus elementos.

Suma de x*y

Dos vectores son ortogonales o perpendiculares cuando forman ángulo recto entre sí. Si el producto escalar de dos vectores es cero, ambos vectores son ortogonales.

```

tt
>>>sum(x*y)
60

>>>np.dot(x,y)
60

```

Vectores ortogonal

```

>>>v1 = np.array([3,4])
>>>v2 = np.array([4, -3])
>>>sum(v1*v2)
0

```

Norma

La norma es la longitud del vector y se calcula mediante la siguiente formula:

Si a es un vector la representación de la norma es |a|, donde:

$$|a| = \sqrt{a_1 + a_2 + \dots + a_n}$$

La norma de “a” equivale a la raíz cuadrada de la sumatoria de los elementos de “a” al cuadrado.

```
>>> np.linalg.norm(x)
5.477225575051661
```

Operaciones con matrices

Al igual que con los vectores, las matrices se pueden sumar, restar y multiplicar por escalares.

Multiplicación por escalares:

$$\gamma A = \begin{bmatrix} \gamma a_{11} & \cdots & \gamma a_{1k} \\ \vdots & \vdots & \vdots \\ \gamma a_{n1} & \cdots & \gamma a_{nk} \end{bmatrix}$$

```
>>> A * 2
array([[2, 6, 4],
       [2, 0, 0],
       [2, 4, 4]])
```

Suma de matrices:

$$A + B = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \vdots & \vdots \\ b_{n1} & \cdots & b_{nk} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1k} + b_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} + b_{n1} & \cdots & a_{nk} + b_{nk} \end{bmatrix}$$

```
>>> A = np.array([[1, 3, 2],
                  [1, 0, 0],
                  [1, 2, 2]])
```

```
>>> B = np.array([[1, 0, 5],
                  [7, 5, 0],
                  [2, 1, 1]])
```

```
>>> A + B #Suma
array([[2, 3, 7],
       [8, 5, 0],
       [3, 3, 3]])
```

Resta de matrices:

$$A - B = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix} - \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \vdots & \vdots \\ b_{n1} & \cdots & b_{nk} \end{bmatrix} = \begin{bmatrix} a_{11} - b_{11} & \cdots & a_{1k} - b_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} - b_{n1} & \cdots & a_{nk} - b_{nk} \end{bmatrix}$$

```
>>> A - B #Resta
```

```
array([[ 0,  3, -3],
       [-6, -5,  0],
       [-1,  1,  1]])
```

Para los casos de suma y resta, hay que tener en cuenta que solo se pueden sumar o restar matrices que tengan las mismas dimensiones, es decir que si tengo una matriz A de 3x2 (3 filas y 2 columnas) solo puedo sumarla o restarla a una matriz que también sea de 3x2.

Para ver la dimensión de la matriz se utiliza:

```
>>>A.shape
(3, 3)
```

Para ver la cantidad de elementos de la matriz se utiliza:

```
>>> A.size
9
```

Multiplicación o Producto de matrices

La regla para la multiplicación de matrices generaliza la idea del producto interior que vimos con los vectores; y esta diseñada para facilitar las operaciones lineales básicas.

Cuando multiplicamos matrices, el número de columnas de la primera matriz debe ser igual al número de filas de la segunda matriz; y el resultado de esta multiplicación va a tener el mismo número de filas que la primer matriz y el número de la columnas de la segunda matriz.

Es decir, que si yo tengo una matriz A de dimensión 3x4 y la multiplico por una matriz B de dimensión 4x2, el resultado va a ser una matriz C de dimensión 3x2.

$$3 \times 4 \quad 4 \times 2 = 3 \times 2$$

Algo a tener en cuenta a la hora de multiplicar matrices es que la propiedad conmutativa no se cumple. $A \times B$ no es lo mismo que $B \times A$.

*

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
array([[ 0, 1],
       [ 2, 3],
       [ 4, 5],
       [ 6, 7]])
```

```

# Ejemplo multiplicación de matrices
>>>A = np.arange(1, 13).reshape(3, 4) #matriz de dimension 3x4
>>>A
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
>>>B = np.arange(8).reshape(4,2) #matriz de dimension 4x2
>>>B
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7]])
# Multiplicando A x B
>>>A @ B #resulta en una matriz de dimension 3x2
>>>array([[ 40,  50],
       [ 88, 114],
       [136, 178]])
# Multiplicando B x A
>>>B @ A

```

```

-----
ValueError                                Traceback (most recent call last)
<ipython-input-28-b55e34ad9c31> in <module>()
      1 # Multiplicando B x A
----> 2 B @ A

```

```
ValueError: shapes (4,2) and (3,4) not aligned: 2 (dim 1) != 3 (dim 0)
```

La matriz identidad, la matriz inversa, la matriz transpuesta y el determinante

La matriz identidad es el elemento neutro en la multiplicación de matrices, es el equivalente al número 1. Cualquier matriz multiplicada por la matriz identidad nos da como resultado la misma matriz. La matriz identidad es una matriz cuadrada (tiene siempre el mismo número de filas que de columnas); y su diagonal principal se compone de todos elementos 1 y el resto de los elementos se completan con 0. Suele representarse con la letra I

Por ejemplo la matriz identidad de 3x3 sería la siguiente:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```

# Creando una matriz identidad de 2x2
>>I = np.eye(2)
>>I
array([[1.,  0.],
       [0.,  1.]])
# Multiplicar una matriz por la identidad nos da la misma matriz
>>A = np.array([[4, 7],

```

```

[2, 6]])
>>A
array([[4, 7],
       [2, 6]])
>>A @ I # A x I = A
array([[4., 7.],
       [2., 6.]])

```

El determinante es un número especial que puede calcularse sobre las matrices cuadradas. Se calcula como la suma de los productos de las diagonales de la matriz en una dirección menos la suma de los productos de las diagonales en la otra dirección. Se representa con el símbolo $|A|$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$|A| = (a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}) - (a_{31}a_{22}a_{13} + a_{32}a_{23}a_{11} + a_{33}a_{21}a_{12})$$

Calculando el determinante de la matriz A

```

>>>np.linalg.det(A)
10.000000000000002

```

La matriz transpuesta es aquella en que las filas se transforman en columnas y las columnas en filas. Se representa con el símbolo A^T

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}^T := \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

Trasponiendo una matriz

```

>>A = np.arange(6).reshape(3, 2)
>>A
array([[0, 1],
       [2, 3],
       [4, 5]])
>>>np.transpose(A)
array([[0, 2, 4],
       [1, 3, 5]])

```

La matriz inversa funciona de la siguiente manera, si tenemos una matriz A, la matriz inversa de A, que se representa como A^{-1} es aquella matriz cuadrada que hace que la multiplicación $A \times A^{-1}$ sea igual a la matriz identidad I. Es decir que es la matriz recíproca de A.

$$A \times A^{-1} = A^{-1} \times A = I$$

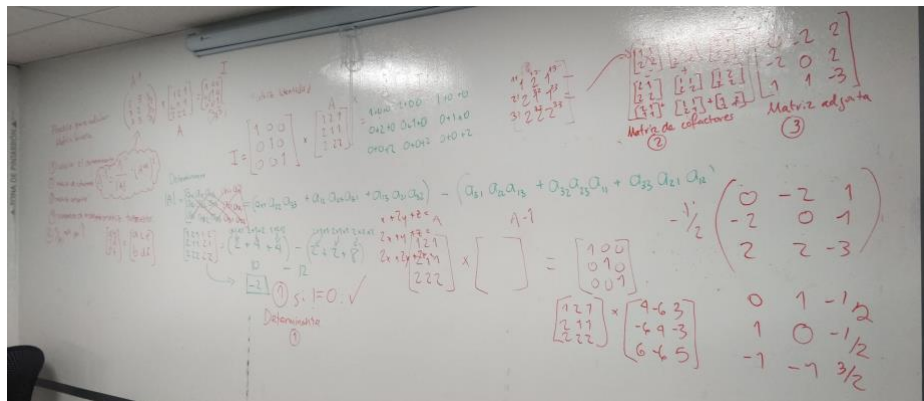
Tener en cuenta que esta matriz inversa en muchos casos puede no existir. En este caso se dice que la matriz es singular o degenerada. Una matriz es singular si y solo si su determinante es nulo.

Fórmula de la Matriz Inversa:

$$A^{-1} = \frac{1}{|A|} \cdot (Adj(A)^t)$$

Proceso para calcular la Matriz Inversa:

1. Calcular el determinante, si el determinante es diferente de 0 continúa, en caso contrario la matriz no tiene inversa.
2. Calcular la matriz de cofactores, que se calcula tachando cada renglón y columna del elemento en cuestión y los elementos que quedan toman la posición del elemento en cuestión en la nueva matriz.
3. Calcular la matriz adjunta que es obtener el determinante de las matrices que resultaron en la matriz de cofactores.
4. Calcular la traspuesta de la matriz adjunta que consiste en cambiar los renglones a columnas.
5. Por último, se multiplica $1/|A|$ o uno sobre el determinante, por la traspuesta de la adjunta.



Calculando la inversa de A.

```
>>A = np.array([[4, 7],
                [2, 6]])

>>A
array([[4, 7],
       [2, 6]])

>>A_inv = np.linalg.inv(A)
>>A_inv
```



```
array([[ 0.6, -0.7],
       [-0.2,  0.4]])
# A x A_inv nos da como resultado I.
>>A @ A_inv
array([[1.,  0.],
       [0.,  1.]])
```