Algebra lineal

Téllez Gerardo Rubén

2/5/2021

Declarar matrices, filas y colúmnas

Vanilla

```
# Declarar filas
row = [1, 2, 3]
print(row)

# Declarar columnas
```

```
## [1, 2, 3]
```

```
col = [[1], [2], [3]]
print(col)

# Delcarar matríces, que en realidad son una tupla de listas
```

```
## [[1], [2], [3]]
```

```
M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(M)
```

```
## [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Declarar submatrices

```
#Primera fila
M[0]

#Primer elemento
```

```
## [1, 2, 3]
```

```
M[0][0]
```

```
## 1
```

Usando NumPy

```
import numpy as np

#Declarar matrices
M1 = np.array(([1,2,3], [4, 5, 6], [7, 8, 9]))
print(M1)
```

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

```
M2 = np.array(([1,2,3], [4, 5, 6], [7, 8, 9]), dtype='complex')
print(M2)

## De ceros
```

```
## [[1.+0.j 2.+0.j 3.+0.j]
##  [4.+0.j 5.+0.j 6.+0.j]
##  [7.+0.j 8.+0.j 9.+0.j]]
```

```
M3 = np.zeros((2,3))
print(M3)

# De unos
```

```
## [[0. 0. 0.]
##  [0. 0. 0.]]
```

```
M4 = np.ones((2,3))
print(M4)


#Para conocer el orden de la matríz
```

```
## [[1. 1. 1.]
##  [1. 1. 1.]]
```

```
print(np.shape(M1))
```

```
## (3, 3)
```

Sumar matrices

```
A = np.array(([1, 2], [3, 4]))
B = np.array(([3,0], [1, -1]))

# Suma de matrices
print(A + B, "\n")

# Producto tensorial
```

```
## [[4 2]
##  [4 3]]
```

```
print(A * B, "\n")

# Producto de matrices
```

```
## [[ 3  0]
##  [ 3 -4]]
```

```
print(A.dot(B), "\n") #M.dor(M1) es la operación de producto matricial

# Matrices transpuestas
```

```
## [[ 5 -2]
##  [13 -4]]
```

```
print(A.transpose(), "\n")

# Suma de los elementos de la diagonal
```

```
## [[1 3]
##  [2 4]]
```

```
print(A.trace(), "\n")

# Potencia de la matriz
```

```
## 5
```

```
print(np.linalg.matrix_power(A, 2))
```

```
## [[ 7 10]
##  [15 22]]
```

Paquete *np.linalg.*, con múltiples operaciones de álgebra lineal.

```
# Rango de la matriz
print(np.linalg.matrix_rank(A), "\n")

# Potencia de la matriz (A^5)
```

```
## 2
```

```
print(np.linalg.matrix_power(A, 5), "\n")

# Determinante
```

```
## [[1069 1558]
##  [2337 3406]]
```

```
print(np.linalg.det(A), "\n")

# Inversa de matriz
```

```
## -2.0000000000000004
```

```
print(np.linalg.inv(A), "\n")

# Matriz identidad
```

```
## [[-2.   1. ]
##  [ 1.5 -0.5]]
```

```
print(np.linalg.inv(A).dot(A), "\n")

# Matriz no invertible
```

```
## [[1.0000000e+00 4.4408921e-16]
##  [0.0000000e+00 1.0000000e+00]]
```

```
Mni = np.array(([1, 2, 3], [0, 0, 5], [0, 0, 6]))
print(Mni, "\n")
```

```
## [[1 2 3]
##  [0 0 5]
##  [0 0 6]]
```

```
print(np.linalg.matrix_rank(Mni), "\n")
```

```
## 2
```

```
try:
  print(np.linalg.inv(Mni))
except Exception as ex:
  print("La matríz es no invertible", ex, type(ex))
```

```
## La matríz es no invertible Singular matrix <class 'numpy.linalg.LinAlgError'>
```

Las matrices con un rango inferior a su número de filas/columnas o que no son cuadradas no se pueden invertir.

Para imprimir la fila de una matriz:

```
for row in M1:
  print(row[0])
```

```
## 1
## 4
## 7
```