Python para Análisis de Datos

Módulo 03

Numpy

Álgebra lineal

Hemos visto hay numerosas operaciones que se pueden hacer con arrays. Algunas son elemento a elemento, otras son sobre el conjunto de los datos, o en alguno de los ejes.

Además de todo eso, Numpy implementa muchas operaciones de álgebra lineal, como producto escalar entre vectores, producto matricial, cálculo de determinante, matriz inversa, etc.

Producto escalar

Recordemos qué es el producto escalar de dos vectores. Si tenemos los vectores `v` y `w` el producto escalar es:

$$v = (v_1, v_2, v_3)$$
$$w = (w_1, w_2, w_3)$$
$$v \cdot w = v_1 w_1 + v_2 w_2 + v_3 w_3$$

O en general:

$$v \cdot w = \sum_{i=1}^{n} v_i w_i$$

Para realizar el producto escalar entre dos vectores (arrays unidimensionales) podemos usar el método **dot**. Para esta operación también existe el operador **@**.

```
v, w
```

```
(array([5, 3, 2, 6]), array([1, 2, 0, 8]))
```

```
v.dot(w)
```

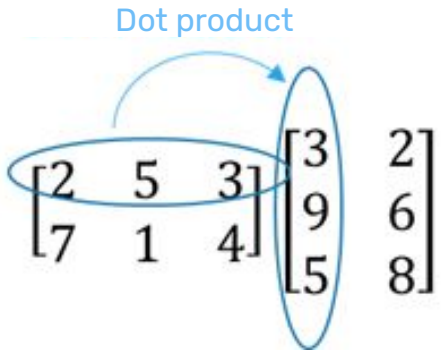
```
59
```

```
v @ w
```

```
59
```

Producto matricial

El método **dot** y su correspondiente operador **@** se pueden usar también para efectuar productos matriciales, en donde cada elemento de la nueva matriz es el producto escalar de una fila de la primer matriz y una columna de la segunda. La cantidad de columnas de la primer matriz debe ser igual a la cantidad de filas de la segunda.




```
A
```

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

```
B
```

```
array([[3, 2],
       [9, 6],
       [5, 8]])
```

```
A.dot(B)
```

```
array([[66, 58],
       [50, 52]])
```

Transpuesta

Cuando se trabaja con matrices es muy usual tener que obtener la matriz transpuesta. Para obtenerla podemos usar el método `transpose` o el atributo `T`.

```
A
```

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

```
A.T
```

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

Determinante

Numpy tiene un módulo dedicado a álgebra lineal: el módulo *linalg*, donde se implementan muchas funciones de álgebra lineal. Una de ellas es `det`, que calcula el determinante de una matriz cuadrada.

```
M
```

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

```
np.linalg.det(M)
```

```
167.99999999999997
```

Inversa

También podemos encontrar la inversa de una matriz cuadrada de determinante no nulo con la función `inv`.

```
M
```

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

```
np.linalg.inv(M)
```

```
array([[-0.08333333, -0.08333333,  0.25      ],
       [ 0.14285714,  0.        ,  0.        ],
       [-0.04761905,  0.16666667,  0.        ]])
```

Revisión

1. Repasar las funcionalidades de Numpy para álgebra lineal.
2. Construir vectores y matrices.
3. Aplicar las funciones del módulo `linalg`.

¡Muchas gracias!

¡Sigamos trabajando!