Universidad Autónoma de San Luis Potosí
Facultad de ingeniería
Inteligencia Artificial Aplicada
Practica 2
Conceptos básicos de numpy
Ana Sofía Medina Martínez

Fecha 28/08/2024

Objetivo

Que el estudiante adquiera habilidades en el manejo de la biblioteca NumPy para procesar, analizar y manipular datos de diferentes tipos y dimensiones.

Procedimiento

2.1.- Inicie Jupyter Notebooks y abra los notebooks "fundamentos", “seleccionar_datos” y “matemáticas_arreglos” proporcionados.

2.2.- Siga las instrucciones en los notebooks para explorar los conceptos básicos de Numpy.

2.3.- Resuelva los ejercicios proporcionados en el notebook “ejercicios”

Resultados

Conceptos básicos de NumPy

Crear arreglos

En esta sección se trabajará con un Sudoku.

Como primer paso importe NumPy con el alias *np*

```
[1]  import numpy as np
```

Por el momento el Sudoku esta almacenado en una lista de Python llamada *sudoku_list*.

```
sudoku_list = [
    [0, 0, 4, 3, 0, 0, 2, 0, 9],
    [0, 0, 5, 0, 0, 9, 0, 0, 1],
    [0, 7, 0, 0, 6, 0, 0, 4, 3],
    [0, 0, 6, 0, 0, 2, 0, 8, 7],
    [1, 9, 0, 0, 0, 7, 4, 0, 0],
    [0, 5, 0, 0, 8, 3, 0, 0, 0],
    [6, 0, 0, 0, 0, 0, 1, 0, 5],
    [0, 0, 3, 5, 0, 8, 6, 9, 0],
    [0, 4, 2, 9, 1, 0, 3, 0, 0]
]
```

Convierta la lista de Python, en un arreglo de NumPy y almacenelo en una variable llamada *sudoku_array*.

```
sodoku_array = np.array(sudoku_list)
print(sodoku_array)
```

```
[[0 0 4 3 0 0 2 0 9]
 [0 0 5 0 0 9 0 0 1]
 [0 7 0 0 6 0 0 4 3]
 [0 0 6 0 0 2 0 8 7]
 [1 9 0 0 0 7 4 0 0]
 [0 5 0 0 8 3 0 0 0]
 [6 0 0 0 0 0 1 0 5]
 [0 0 3 5 0 8 6 9 0]
 [0 4 2 9 1 0 3 0 0]]
```

En ocasiones es necesario crear arreglos desde cero.

Para probar alguna de estas funciones cree un arreglo de ceros con 3 renglones y 4 columnas.

```
[4] zeros_array = np.zeros((3,4))
    print(zeros_array)
```

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

Dimensionalidad

Podemos modificar las dimensiones utilizando el método *reshape*.

Convierta el arreglo *sudoku_array* a un arreglo unidimensional, almacenelo en una variable llamada *sudoku_flatten* e imprimalo.

```
[9] sudoku_flatten = sodoku_array.flatten()
    print(sudoku_flatten)
```

```
[0 0 4 3 0 0 2 0 9 0 0 5 0 0 9 0 0 1 0 7 0 0 6 0 0 4 3 0 0 6 0 0 2 0 8 7 1
 9 0 0 0 7 4 0 0 0 5 0 0 8 3 0 0 0 6 0 0 0 0 0 1 0 5 0 0 3 5 0 8 6 9 0 0 4
 2 9 1 0 3 0 0]
```

Regrese el arreglo *sudoku_flatten* a las dimensiones originales y almacene el resultado en la variable *sudoku_reshaped*.

Imprima la variable sudoku_reshaped para observar que el orden de los elementos no se haya afectado.

```
[10] sudoku_reshaped = sudoku_flatten.reshape(9, 9)
     print(sudoku_reshaped)
```

```
[[0 0 4 3 0 0 2 0 9]
 [0 0 5 0 0 9 0 0 1]
 [0 7 0 0 6 0 0 4 3]
 [0 0 6 0 0 2 0 8 7]
 [1 9 0 0 0 7 4 0 0]
 [0 5 0 0 8 3 0 0 0]
 [6 0 0 0 0 0 1 0 5]
 [0 0 3 5 0 8 6 9 0]
 [0 4 2 9 1 0 3 0 0]]
```

Tipos de dato

Imprima el tipo de dato del arreglo sudoku_array.

```
sodoku_array.dtype
```

```
dtype('int64')
```

Para optimizar memoria cambiemos el tipo de dato del arreglo *sudoku_array* a *uint8*

```
[12] sodoku_array = sodoku_array.astype('uint8')
     sodoku_array.dtype
```

```
dtype('uint8')
```

Accediendo a elementos del arreglo

En esta sección se utilizará un arreglo con los datos de un censo de los árboles de Nueva York.

Esta es la informacion que contiene:

- La primer columna es el identificador del arbol.
- La segunda columna es el identificador del bloque al que pertenece el arbol.
- La tercer columna es el diametro del tronco.
- La cuarta columna es el diametro del tocón.

```
[13] import numpy as np
```

Utilice la función `np.load()` para cargar los datos del archivo `tree_census.npy`.

Almacene el arreglo en una variable llamada `tree_census`

```
from google.colab import drive
drive.mount('/content/drive')

# Ruta al archivo en Google Drive
file_path = '/content/drive/My Drive/IAA/Practica2_CONCEPTOS BÁSICOS DE NUMPY/datasets/tree_census.npy'

# Cargar el archivo .npy
tree_census = np.load(file_path)

# Verifica el contenido
print(tree_census)
```

```
Mounted at /content/drive
[[     3 501451     24      0]
 [     4 501451     20      0]
 [     7 501911      3      0]
 ...
 [  1198 227387     11      0]
 [  1199 227387     11      0]
 [  1210 227386      6      0]]
```

Seleccione todos los renglones de la segunda columna (con indice 1) y almacenelos en una variable llamada *block_ids*

```
[15] block_ids = tree_census[:, 1]
```

De la variable *block_ids* imprima:

- El octavo elemento.
- Todos los elementos hasta el sexto.
- 5 elementos empezando en el indice 10.

```
[16] block_ids[7]
     block_ids[:6]
     block_ids[10:15]
```

```
array([501911, 501911, 501909, 501909, 501909])
```

Del arreglo *tree_census* selecciones todos los árboles (renglones) con un diametro del tronco menor a 10.

```
[17] tree_census[tree_census[:, 2] < 10]
```

```
array([[     7, 501911,      3,      0],
       [     8, 501911,      3,      0],
       [     9, 501911,      4,      0],
       ...,
       [  1186, 226832,      0,     20],
       [  1187, 226832,      0,     21],
       [  1210, 227386,      6,      0]])
```

Investigadores descubrieron 2 nuevos árboles, ambos estan almacenados en la variable *new_trees*, agrege sus datos al arreglo *tree_census*

```
[18] new_trees = np.array([[1211, 227386, 20, 0], [1212, 227386, 8, 0]])
```

```
[19] tree_census = np.concatenate((tree_census, new_trees), axis=0)
     print(tree_census)
```

```
[[     3 501451     24      0]
 [     4 501451     20      0]
 [     7 501911      3      0]
 ...
 [  1210 227386      6      0]
 [  1211 227386     20      0]
 [  1212 227386      8      0]]
```

Comprensión

1. **¿Qué es Numpy?**

   Es una librería de Python que nos permite manejar números como matrices y se puede realizar muchas operaciones matemáticas.

2. **¿Cuál es la diferencia entre una lista de Python y un arreglo Numpy?**

Los arreglos de numpy son mejores que las listas de pyhton porque guardan los datos de forma homogénea y permite operaciones matemáticas.

3. **¿Qué es el broadcasting en Numpy y por qué es importante?**
   El broadcasting permite realizar operaciones entre arrays de diferentes tamaños sin necesidad de duplicar datos

4. **¿Cuál es la importancia de Numpy en el ámbito del aprendizaje automático y la ciencia de datos?**
   Numpy resulta ser muy importante en el ámbito porque tiene la capacidad de procesar grandes cantidades de datos numéricos en operaciones complejas y esto optimiza el rendimiento computacional.

Conclusiones

En conclusión, numpy es una de las bibliotecas más importantes de Python para el ámbito de aprendizaje automático y la ciencia de datos debido a que es muy eficiente para manipular grandes cantidades de datos, así como de realizar operaciones con estos datos.