Guía de Laboratorio: Introducción a NumPy

Docente: Carlos R. P. Tovar
Curso: Inteligencia Artificial

1. Introduccion

NumPy es una biblioteca fundamental en Python para el manejo de arreglos numericos y operaciones matematicas eficientes. Es ampliamente utilizada en ciencia de datos, machine learning y computacion cientifica debido a su capacidad para realizar calculos vectorizados de manera rapida y eficiente. Esta guía de laboratorio esta disenada para introducir los conceptos basicos de NumPy, incluyendo la creacion de arreglos, operaciones matematicas y manipulacion de datos.

2. Objetivos

- Comprender el concepto de arreglos en NumPy y su diferencia con listas de Python.
- Aprender a crear y manipular arreglos de diferentes dimensiones.
- Realizar operaciones matematicas y estadisticas basicas con NumPy.
- Explorar funciones de indexacion y segmentacion de arreglos.
- Aplicar NumPy en problemas basicos de inteligencia artificial.

3. Requisitos previos

- Tener instalado Python (version 3.6 o superior).
- Instalar la biblioteca NumPy (`pip install numpy`).
- Un entorno de desarrollo como Jupyter Notebook, Visual Studio Code o cualquier editor de texto.

4. Ejercicios Practicos

4.1. Ejercicio 1: Creacion de arreglos

En este ejercicio, crearas diferentes tipos de arreglos utilizando NumPy.

1. Crea un arreglo 1D con los numeros del 1 al 5.
2. Crea un arreglo 2D de 2x3 con valores enteros consecutivos.

3. Crea un arreglo de ceros de tamano 3x4.

4. Crea un arreglo con 10 valores espaciados uniformemente entre 0 y 1.

Codigo de ejemplo:

```
import numpy as np

# Arreglo 1D
arr1 = np.array([1, 2, 3, 4, 5])
print("Arreglo 1D:", arr1)

# Arreglo 2D
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print("Arreglo 2D:\n", arr2)

# Arreglo de ceros
arr3 = np.zeros((3, 4))
print("Arreglo de ceros:\n", arr3)

# Arreglo con valores espaciados
arr4 = np.linspace(0, 1, 10)
print("Arreglo espaciado:\n", arr4)
```

Tarea: Modifica el codigo para crear un arreglo 2D de 4x2 con valores aleatorios entre 0 y 10.

4.2. Ejercicio 2: Operaciones con arreglos

Realiza operaciones matematicas basicas con arreglos de NumPy.

1. Crea dos arreglos 1D de tamano 4 con valores enteros.

2. Realiza la suma, resta, multiplicacion y division elemento por elemento.

3. Calcula el producto punto de los dos arreglos.

Codigo de ejemplo:

```
import numpy as np

# Crear dos arreglos
a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])

# Operaciones elemento por elemento
suma = a + b
resta = a - b
multiplicacion = a * b
division = a / b

print("Suma:", suma)
print("Resta:", resta)
print("Multiplicacion:", multiplicacion)
print("Division:", division)

# Producto punto
producto_punto = np.dot(a, b)
print("Producto punto:", producto_punto)
```

Tarea: Calcula el cuadrado de cada elemento del arreglo `a` y la raiz cuadrada del arreglo `b`.

4.3. Ejercicio 3: Indexacion y segmentacion

Explora como acceder a elementos y subarreglos en NumPy.

1. Crea un arreglo 2D de 3x3 con valores del 1 al 9.
2. Accede al elemento en la posicion (1, 2).
3. Extrae la primera fila y la ultima columna.
4. Selecciona un subarreglo de 2x2 desde la esquina superior izquierda.

Codigo de ejemplo:

```
import numpy as np

# Crear arreglo 2D
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Arreglo 2D:\n", arr)

# Acceder a un elemento
elemento = arr[1, 2]
print("Elemento en (1, 2):", elemento)

# Extraer primera fila
primera_fila = arr[0, :]
print("Primera fila:", primera_fila)

# Extraer ultima columna
ultima_columna = arr[:, -1]
print("Ultima columna:", ultima_columna)

# Subarreglo 2x2
subarreglo = arr[0:2, 0:2]
print("Subarreglo 2x2:\n", subarreglo)
```

Tarea: Extrae un subarreglo que contenga las filas 1 y 2, y las columnas 0 y 1.

4.4. Ejercicio 4: Funciones estadisticas

Usa funciones de NumPy para calcular estadisticas basicas.

1. Crea un arreglo 1D con 10 valores aleatorios entre 1 y 100.
2. Calcula la media, mediana, desviacion estandar y suma de los valores.
3. Encuentra el valor maximo y su indice.

Codigo de ejemplo:

```python
import numpy as np

# Crear arreglo con valores aleatorios
arr = np.random.randint(1, 101, 10)
print("Arreglo aleatorio:", arr)

# Calcular estadisticas
media = np.mean(arr)
mediana = np.median(arr)
desviacion = np.std(arr)
suma = np.sum(arr)
maximo = np.max(arr)
indice_max = np.argmax(arr)

print("Media:", media)
print("Mediana:", mediana)
print("Desviacion estandar:", desviacion)
print("Suma:", suma)
print("Valor maximo:", maximo, "en indice:", indice_max)
```

Tarea: Crea un arreglo 2D de 3x3 con valores aleatorios y calcula la suma de cada fila y columna.

4.5. Ejercicio 5: Aplicacion en IA - Normalizacion de datos

La normalizacion de datos es un paso crucial en el preprocesamiento para algoritmos de IA.

1. Crea un arreglo 2D que simule un dataset de caracteristicas (100 muestras, 5 caracteristicas).
2. Normaliza los datos para que cada caracteristica tenga media 0 y desviacion estandar 1.
3. Verifica que la normalizacion fue correcta.

Codigo de ejemplo:

```python
import numpy as np

# Crear dataset simulado (100 muestras, 5 caracteristicas)
np.random.seed(42)  # Para resultados reproducibles
X = np.random.randn(100, 5) * 10 + 5  # Media ~5, desviacion ~10
print("Dataset original - Media:", np.mean(X, axis=0), "Desviacion:",
    np.std(X, axis=0))

# Normalizacion (Standard Scaling)
X_normalized = (X - np.mean(X, axis=0)) / np.std(X, axis=0)

print("Dataset normalizado - Media:", np.mean(X_normalized, axis=0), "
    Desviacion:", np.std(X_normalized, axis=0))
```

Tarea: Modifica el codigo para normalizar los datos en el rango [0, 1] (Min-Max scaling).

5. Conclusion

En esta guia, has aprendido a crear y manipular arreglos en NumPy, realizar operaciones matematicas, indexar subarreglos y calcular estadisticas basicas. NumPy es una herramienta poderosa que optimiza el manejo de datos numericos en Python. Te recomendamos explorar mas funciones avanzadas como `np.where`, `np.concatenate` y operaciones con matrices para profundizar tus conocimientos.

6. Recursos adicionales

- Documentacion oficial de NumPy: https://numpy.org/doc/stable/
- Tutoriales interactivos en Jupyter Notebook: https://jupyter.org/
- Curso gratuito de NumPy en DataCamp: https://www.datacamp.com/courses/intro-to-python-for-data-science