Taller No 2: Introducción al cómputo científico con Numpy

Profesores: Carlos A. Fajardo y Harold H. Rodriguez
Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones
Actualizado: August 13, 2025

Parte 1:

Instrucciones generales: Resuelva los siguientes ejercicios utilizando arreglos de `NumPy` y un estilo de programación *pythonic*. Se espera el uso de máscaras booleanas, y funciones propias de `NumPy` (`np.where()`, entre otras), evitando estructuras repetitivas innecesarias como bucles explícitos. Priorice la claridad, concisión y eficiencia del código.

1

Reemplazo binario simple: Dado un array de números enteros, reemplaza todos los valores mayores a 5 por 1 y el resto por 0.

- **Ejemplo de entrada**: `np.array([5, 2, 9, 1, 6])`
- **Resultado esperado**: `[0, 0, 1, 0, 1]`

2

Clasificación por etiquetas: Crea un array de cadenas que indique si cada valor del array es "alto" (mayor a 10) o "bajo".

- **Ejemplo de entrada**: `np.array([15, 8, 22, 4, 30])`
- **Resultado esperado**: `["alto", "bajo", "alto", "bajo", "alto"]`

3

Reemplazo en matriz: Reemplaza todos los valores menores que 5 en una matriz 2D por -1, y deja los demás iguales.

- **Ejemplo de entrada**: `np.array([[1, 5, 9], [2, 8, 3]])`
- **Resultado esperado**: `[[-1, 5, 9], [-1, 8, -1]]`

4

Multiplicación condicional: Multiplica por 2 los valores pares y por 3 los impares en un array 1D.

- **Ejemplo de entrada**: `np.array([2, 7, 5, 10])`
- **Resultado esperado**: `[4, 21, 15, 20]`

5

Conversión de temperaturas: Dado un arreglo de temperaturas registradas en una ciudad durante varios días, cree una función que exprese las temperaturas en grados Fahrenheit solo para aquellas que estén por debajo de 10°C. Para las temperaturas iguales o superiores a 10°C, conserve el valor original en grados Celsius.

- **Ejemplo de entrada**: `np.array([-2, 5, 12, 8, 15])`
- **Transformación esperada**: Solo los valores menores a 10°C deben convertirse a Fahrenheit utilizando la fórmula `F = C * 1.8 + 32`. Los demás valores deben conservarse en grados Celsius.
- **Resultado esperado**: `[28.4, 41.0, 12.0, 46.4, 15.0]`

6

Clasificación con múltiples rangos: Clasifica los elementos como "bajo" si son menores a 60, "medio" si están entre 60 y 80, y "alto" si son mayores a 80.

- **Ejemplo de entrada**: `np.array([45, 70, 82, 60, 95])`
- **Resultado esperado**: `["bajo", "medio", "alto", "medio", "alto"]`

7

Máscara binaria de imagen: Crea una máscara binaria (1 si el valor es mayor a 100, si no 0) para una matriz de intensidades de imagen.

- **Ejemplo de entrada**: `np.array([[10, 150, 200], [80, 255, 0]])`
- **Resultado esperado**: `[[0, 1, 1], [0, 1, 0]]`

8

Clasificación de ganancias y pérdidas: Dado un arreglo con los valores de ganancia o pérdida diaria de una empresa durante un período, cree una función que clasifique cada día como `"ganancia"`, `"pérdida"` o `"neutral"` (si el valor es exactamente cero).

- **Ejemplo de entrada**: `np.array([200, -150, 0, 80, -20])`
- **Clasificación esperada**: `["ganancia", "pérdida", "neutral", "ganancia", "pérdida"]`

9

Conversión de etiquetas a índices: En tareas de clasificación en *Machine Learning*, a menudo se requiere convertir etiquetas categóricas a enteros para alimentar modelos o realizar análisis previos.

Dada una lista de etiquetas categóricas, reemplaza cada categoría por su índice entero según el orden en que aparece en el conjunto de clases únicas.

- **Ejemplo de entrada**: `['perro', 'gato', 'pez', 'gato', 'perro', 'pez', 'gato']`
- **Clases detectadas (ordenadas alfabéticamente)**: `['gato', 'perro', 'pez']`
- **Asignación de índices**:
  - `'gato'` $\rightarrow$ `0`
  - `'perro'` $\rightarrow$ `1`
  - `'pez'` $\rightarrow$ `2`
- **Resultado esperado**: `[1, 0, 2, 0, 1, 2, 0]`

Pregunta: Escriba una función en Python que reciba una lista de etiquetas categóricas y retorne una lista con los índices numéricos correspondientes a cada categoría.

10

Codificación categórica (One-hot encoding): La codificación *one-hot* es una técnica ampliamente utilizada en ciencia de datos y aprendizaje automático (*Machine Learning*) para transformar variables categóricas en vectores numéricos binarios. Esta transformación es fundamental para que los modelos computacionales puedan interpretar datos no numéricos.

Cada categoría distinta se representa como un vector binario de longitud igual al número total de clases. En cada vector, solo una posición contiene el valor `1` (indicando la clase presente) y el resto contiene `0`.

Dada una lista de etiquetas categóricas, conviértelas a su representación *one-hot*.

- **Ejemplo de entrada:** `['perro', 'gato', 'pez', 'gato', 'perro', 'pez', 'gato']`
- **Clases detectadas:** `['perro', 'gato', 'pez']`
- **Representación one-hot:**
  - `'perro'` → `[1, 0, 0]`
  - `'gato'` → `[0, 1, 0]`
  - `'pez'` → `[0, 0, 1]`
- **Resultado esperado:** `[[1, 0, 0], [0, 1, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 0, 1], [0, 1, 0]]`

Pregunta: Escriba una función en Python que reciba una lista de datos categóricos y devuelva su codificación *one-hot*.

Parte 2: Simulación manual (ejercicios hechos a mano)

Instrucciones generales: Los siguientes ejercicios deben ser desarrollados a mano (lápiz y papel).

11

Asuma que usted es un compilador de Python e imprima la salida correcta del siguiente código.

- **Código de entrada:**

```
import numpy as np

a = np.array([[0, 0, 0],
              [10, 10, 10],
              [20, 20, 20],
              [30, 30, 30]])

print(a[1:, 1])
```

12

Indexación en tres dimensiones (NumPy): Asuma que usted es un compilador de Python e imprima la salida correcta del siguiente código.

- **Código de entrada:**

```
import numpy as np

a = np.arange(24).reshape((2, 3, 4))
print(a[1:3:2,1:,::2])
```

13

Slicing multidimensional (NumPy): Asuma que usted es un compilador de Python e imprima la salida correcta del siguiente código.

- **Código de entrada:**

```
import numpy as np

a = np.arange(24).reshape((2, 3, 4))
print(a[:1, ::2, 1:])
```

14

A continuación se muestra un arreglo tridimensional. Se proporciona también la salida esperada. Escriba la instrucción de `print(...)` que produce exactamente ese resultado utilizando slicing.

- **Array base**:

```
a = np.arange(24).reshape((2, 3, 4))
print(¿_________?)
```

- **Salida esperada**:

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

- **Pregunta**: ¿Cuál es el código de slicing que genera exactamente esta salida?

15

A continuación se muestra un arreglo tridimensional. Se proporciona también la salida esperada. Escriba la instrucción de `print(...)` que produce exactamente ese resultado utilizando slicing.

- **Array base**:

```
a = np.arange(27).reshape((3, 3, 3))
print(¿_________?)
```

- **Salida esperada**:

```
array([[[ 9, 10, 11],
        [12, 13, 14]]])
```

16

Analice el siguiente código. ¿La operación es válida?, ¿Cuál es el resultado o el motivo del error? Explique en detalle cualquiera que sea su respuesta.

- **Código**:

```
import numpy as np

a = np.arange(12).reshape((4, 3, 1))
b = np.arange(12).reshape((1, 3, 4))

print(a + b)
```

17

Analice el siguiente código. ¿La operación es válida?, ¿Cuál es el resultado o el motivo del error? Explique en detalle cualquiera que sea su respuesta.

- **Código**:

```
import numpy as np
```

```
a = np.arange(8).reshape((2, 4))
b = np.arange(12).reshape((3, 4))

print(a + b)
```

Parte 3: Operaciones entre arreglos

Instrucciones generales: Los siguientes ejercicios muestran diferentes operaciones efectuadas entre arreglos empleados durante el entrenamiento de modelos de deep learning. Resuelva dichos ejercicios utilizando arreglos de NumPy.

18

Sean los vectores y y $\hat{y}$ las etiquetas reales y a las predicciones obtenidas por un modelo de deep learning respectivamente:
Calcule el *error cuadrático* dado por la siguiente ecuación:

$$\rightarrow \text{SE} = (y - \hat{y})^2 \tag{1}$$

Donde las operaciones de `resta` y `al cuadro` son operaciones elemento a elemento.

Pregunta: Escriba una función en Python que reciba:

- y de dimensión $(1 \times n)$
- $\hat{y}$ de dimensión $(1 \times n)$
- Y devuelva SE de dimensión $(1 \times n)$

19

Modifique el ejercicio anterior, para que calcule *el error cuadrático medio*, dado por la siguiente ecuación:

$$\rightarrow \text{MSE} = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

Donde la sumatoria es la suma de todos los elementos del vector.

Pregunta: Escriba una función en Python que reciba:

- y de dimensión $(1 \times n)$
- $\hat{y}$ de dimensión $(1 \times n)$
- Y devuelva el MSE que es un escalar.

20

Cálculo de gradiente (versión matriz): Escriba una función en Python que calcule el gradiente $\frac{\partial J}{\partial w}$ a partir de y, $\hat{y}$ y X, donde:

$$\frac{\partial J}{\partial w} = (y - \hat{y}) * \mathbf{X}$$

Explicación:
Sean los vectores fila y y $\hat{y}$ y la matriz de características X:

$$y = \begin{bmatrix} 9, & 8, & 11 \end{bmatrix}, \hat{y} = \begin{bmatrix} 8.9, & 8.1, & 10.6 \end{bmatrix} \tag{3}$$

$$\mathbf{X} = \begin{bmatrix} 0.5 & 0.7 & 0.2 \\ 0.4 & 0.9 & 0.6 \end{bmatrix}$$

Calcule el gradiente (versión matriz) con respecto a w, definido como:

$$\frac{\partial J}{\partial w} = (y - \hat{y}) * \mathbf{X}$$

- $y - \hat{y}$ es el error :
$$e = y - \hat{y} = [0.1,\ -0.1,\ 0.4]$$
- La multiplicación $(y - \hat{y}) * \mathbf{X}$ es:
$$e * \mathbf{X} = \begin{bmatrix} 0.1 \cdot 0.5 & -0.1 \cdot 0.7 & 0.4 \cdot 0.2 \\ 0.1 \cdot 0.4 & -0.1 \cdot 0.9 & 0.4 \cdot 0.6 \end{bmatrix}$$
- Resultado final:
$$\frac{\partial J}{\partial w} = (y - \hat{y}) * X = e * \mathbf{X} = \begin{bmatrix} 0.05 & -0.07 & 0.08 \\ 0.04 & -0.09 & 0.24 \end{bmatrix}$$

Pregunta: Escriba una función en Python que reciba:

- y de dimensión $(1 \times n)$
- $\hat{y}$ de dimensión $(1 \times n)$
- X de dimensión $(m \times n)$

Y devuelva $\frac{\partial J}{\partial w}$ de dimensión $(m \times n)$

21

Cálculo del gradiente (versión vector): Modifique el ejercicio anterior para que calcule el gradiente, versión vector, dado por la fórmula:

$$\frac{\partial J}{\partial w} = \sum_{i=1}^{n} [(y_i - \hat{y}_i) * \mathbf{X}_{i,:}]$$

Donde la sumatoria se realiza **por columnas**, es decir, es la suma de las columnas de la matriz obtenidad en el ejercicio anterior.
Respuesta esperada:

$$\frac{\partial J}{\partial w} = \sum_{i=1}^{m} [(y_i - \hat{y}_i) * \mathbf{X}_{i,:}] = \begin{bmatrix} 0.06 \\ 0.19 \end{bmatrix}$$

Pregunta: Escriba una función en Python que reciba:

- y de dimensión $(1 \times n)$
- $\hat{y}$ de dimensión $(1 \times n)$
- X de dimensión $(m \times n)$

Y devuelva $\frac{\partial J}{\partial w}$ de dimensión $(1 \times n)$