

Manual de Ingeniería II: Computación Numérica y Lógica

De Python Puro a la Vectorización

Semana 02 - Curso de IA Agroindustrial

10 de enero de 2026

Índice

1. Introducción: La Necesidad de Velocidad

En la Semana 1 dominamos el sistema operativo. En la Semana 2, entramos al núcleo del procesamiento de datos.

En IA, no procesamos 10 datos; procesamos millones (imágenes satelitales, series temporales de sensores). Python, por sí solo, es un lenguaje interpretado y lento para bucles masivos. Aquí es donde entra la **Computación Vectorial**.

2. Parte 1: Lógica y Estructuras de Datos (Python Puro)

Antes de correr, debemos caminar. El script `main.py` aborda la limpieza de datos secuencial.

2.1. El Costo de las Listas en Python

Una lista en Python (`list`) no es un array contiguo de memoria como en C. Es un array de **punteros** a objetos.

Lista = [ptr → Obj(24.5), ptr → Obj(None), ...]

Cada vez que iteramos sobre una lista, el procesador debe:

1. Leer el puntero.
2. Ir a la dirección de memoria del objeto.
3. Verificar el tipo de dato (Dynamic Typing check).
4. Obtener el valor.

Esto genera lo que llamamos *Overhead* computacional.

2.2. Algoritmo de Limpieza (Data Cleaning)

En el taller, implementamos un filtro de "paso bajo" lógico:

```
1 # Complejidad Temporal: O(N) - Lineal
2 for lectura in sensores:
3     if lectura is None: continue          # Manejo de Nulls
4     if 0 <= lectura <= 50:                # Validacion de Rango Fisico
5         datos_limpios.append(lectura)    # Mutacion de lista
```

Concepto Clave: La complejidad es $O(N)$, donde N es el número de sensores. Si $N = 1,000,000$, el bucle se ejecutará un millón de veces secuencialmente.

3. Parte 2: Vectorización con NumPy

NumPy (Numerical Python) resuelve el problema de la velocidad cambiando la estrategia:

1. **Memoria Contigua:** Los datos están uno al lado del otro en RAM (Localidad de Caché).
2. **Tipado Estático:** Todos los elementos son del mismo tipo (ej. `float64`).
3. **SIMD (Single Instruction, Multiple Data):** El procesador aplica la misma operación a múltiples datos simultáneamente.

3.1. Broadcasting y Operaciones Element-wise

En el script `simulacion.py`, no usamos bucles `for`. Usamos operaciones vectoriales.

Ejemplo Matemático: Si queremos normalizar una matriz A restando el mínimo μ :

$$A_{norm} = A - \mu$$

En Python puro, esto requiere iterar. En NumPy, gracias al *Broadcasting*, se resta μ a cada elemento de la matriz en paralelo (a nivel de CPU).

3.2. Filtrado Booleano (Masking)

Una de las herramientas más potentes en IA es la máscara booleana.

```
1 # Genera una matriz de True/False
2 mapa_alertas = mapa_termico > 40.0
```

Esto evalúa 1, 000, 000 de píxeles casi instantáneamente. Matemáticamente, es una función indicador:

$$I(x) = \begin{cases} 1 & \text{si } x > 40 \\ 0 & \text{si } x \leq 40 \end{cases}$$

4. Análisis de los Retos Propuestos

4.1. Reto 1: Normalización Min-Max (Feature Scaling)

Las Redes Neuronales aprenden mejor cuando los datos están entre 0 y 1. La fórmula formal que deben aplicar es:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

En NumPy, esto se escribe en una sola línea gracias al broadcasting, sin bucles.

4.2. Reto 2: Búsqueda Espacial (Argmax)

Encontrar el máximo es fácil ($\max(A)$). Encontrar **dónde** está es más complejo.

- `np.argmax`: Devuelve el índice como si la matriz fuera una lista plana de 1 dimensión.
- `np.unravel_index`: Convierte ese índice plano de vuelta a coordenadas (x, y) .

4.3. Reto 3: Imputación Condicional (np.where)

En ingeniería de datos, “Imputar” significa reemplazar un dato faltante o erróneo.

```
1 # np.where(condicion, valor_si_true, valor_si_false)
2 mapa_corregido = np.where(mapa < 21, promedio, mapa)
```

Esto es equivalente a escribir un millón de if-else, pero ejecutado en C optimizado.

5. Conclusión: El Perfil del Ingeniero de IA

- Un programador normal escribe bucles.
- Un ingeniero de datos piensa en **Vectores y Matrices**.

Al eliminar los bucles explícitos de su código, no solo lo hace más rápido, sino más legible y cercano a la notación matemática de los algoritmos de aprendizaje automático.