Clase 019 — NumPy: ordenamiento y búsqueda

Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 2 § 2.8 Sorting Arrays · NumPy sorting reference. · Duración estimada: 60 min.

Clase 019 — NumPy: ordenamiento y búsqueda

> *Parte: 0 — Prerrequisitos · Fuente: VanderPlas, cap. 2 § 2.8 Sorting Arrays · NumPy sorting reference. · Duración estimada: 60 min.*

Objetivo

Que el alumno ordene arrays con criterio: sort vs argsort, ordenamiento por eje, partial sort con partition, y búsqueda binaria con searchsorted. Útil para top-K, rankings, alineación de series.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Ordenar con np.sort(arr) (devuelve copia) y arr.sort() (in-place).
2. Obtener índices del orden con argsort — base de top-K y rankings.
3. Ordenar por eje en matrices con axis=0 o axis=1.
4. Top-K eficiente con np.partition (no ordena completo, solo separa).
5. Búsqueda binaria con np.searchsorted en arrays ordenados (O(log n)).

Temas

#	Tema	Por qué importa
1	np.sort vs arr.sort()	Copia vs in-place.
2	argsort: el truco del top-K	Índices que ordenarían el array.
3	Ordenamiento por eje	Por fila o por columna.
4	np.partition para top-K	Más rápido que sort completo.
5	np.searchsorted — binaria O(log n)	Inserción en array ordenado.
6	np.unique	Únicos + opcionalmente cuentas.

Definiciones y características

np.sort vs arr.sort()

: np.sort(arr) devuelve copia ordenada (no muta). arr.sort() ordena in-place y retorna None. Mismo patrón que sorted(list) vs list.sort().

argsort

: Devuelve los índices que ordenarían el array. idx = arr.argsort(); ordenado = arr[idx]. Base de top-K, rankings, alineación entre arrays correlacionados.

np.partition

: Quick-select O(n): garantiza que los K menores quedan en las primeras K posiciones (no necesariamente ordenados entre sí), el resto después. Mucho más rápido que sort completo cuando solo necesitas top-K.

np.searchsorted

: Búsqueda binaria O(log n) en array ordenado. Devuelve el índice donde insertar un valor para mantener

orden. Útil para calcular percentiles, bucketing.

np.unique

: Devuelve únicos ordenados. Con return_counts=True devuelve también las frecuencias — alternativa rápida a Counter para datos numéricos.

Dataset / recursos

Sintético: puntajes de 1M estudiantes. Sin descarga.

Ejercicios

1. Top-10. Dado array de 1M puntajes, obtén los 10 más altos. Compara np.sort()[-10:] vs np.partition.

2. Ranking. Con argsort, asigna a cada estudiante su ranking (1 = mejor).

3. Ordena matriz por columna. Matriz 10×5; ordena cada columna por su valor.

4. Mediana por bisect. Implementa una función que dado un valor v y un array ordenado, devuelve su posición percentil usando searchsorted.

5. np.unique con cuentas. Dado array de categorías, obtén valores únicos y sus frecuencias.

Homework verificable

Notebook con array de 100k puntajes: (a) top-100 con partition y benchmark vs sort completo; (b) ranking con argsort.argsort(); (c) percentil de un valor dado con searchsorted; (d) unique con return_counts y barplot top-10 categorías.

Criterio de aceptación: partition >10× más rápido que sort completo para N=100k y K=100.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Ordeno con arr.sort() y la variable queda	sort() es in-place — modifica arr y retorn
Top-K con sort()[-K:] es lento para N gran	Sort completo es O(N log N). Fix: np.parti
argsort da resultado raro con matriz	Sin axis, ordena cada fila/columna indepen
np.searchsorted da índice fuera del array	Si el valor es mayor que todos, devuelve l
np.unique(arr_2d) aplana el array	Por default, unique trabaja sobre array ap

Preguntas frecuentes

¿Cuándo argsort y cuándo sort?

Si solo necesitas los valores ordenados, sort. Si necesitas el orden para aplicarlo a otros arrays correlacionados (ej: ordenar nombres por puntajes), argsort te da los índices.

¿partition o heapq.nlargest?

partition para arrays NumPy (vectorizado, C). heapq.nlargest(K, lst) para listas Python. Para K muy pequeño

(3-5) sobre N grande, similares; partition gana en arrays grandes.

¿Cómo ordeno por múltiples claves (lexicográfico)?

np.lexsort([clave_secundaria, clave_principal]) — devuelve índices. Atención: el orden es al revés (último argumento = key primaria).

¿np.unique preserva el orden de primera aparición?

No — siempre ordena. Para preservar orden de aparición: _, idx = np.unique(arr, return_index=True); arr[np.sort(idx)].

¿Existe equivalente a SQL ORDER BY x DESC?

arr[arr.argsort()][::-1] o arr[arr.argsort()[::-1]]. NumPy no tiene flag reverse= como sorted() Python — invierte tú.

Referencias

- VanderPlas, cap. 2 § 2.8.
- NumPy sorting reference

Siguiente clase

Clase 020 — NumPy: álgebra lineal con numpy.linalg

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
import numpy as np
import time
rng = np.random.default_rng(42)
```

Archivos complementarios

- notebook.ipynb