Guía de Ejercicios: Python Avanzado

NumPy Avanzado, Broadcasting, Dask y Numba

Prof. Miguel Cárcamo

9 de octubre de 2025

1. NumPy Avanzado

1.1. Ejercicios Básicos

1. Procesamiento de Visibilidades

- Simula 1,000,000 visibilidades complejas (correlaciones entre antenas) con amplitud y fase aleatorias
- Implementa una función que calcule la potencia promedio usando un loop de Python
- Implementa la misma función usando operaciones vectorizadas de NumPy
- Compara el rendimiento de ambas implementaciones
- ¿Cuál es la aceleración obtenida? ¿Por qué es importante para observaciones en tiempo real?

2. Optimización de Memoria en Datos Astronómicos

- Simula factores de ganancia de antenas (valores complejos) con diferentes precisiones: `int32`, `int64`, `float32`, `float64`
- Calcula el tamaño en memoria de cada array para 1000 antenas
- Realiza operaciones de calibración (multiplicación de visibilidades por factores de ganancia) y mide el tiempo
- ¿Qué tipo de datos es más eficiente? ¿Cuándo usarías cada uno en observaciones reales?

3. Vistas vs Copias

- Crea un array original de 10 elementos
- Crea una vista del array (sin usar `.copy()`)
- Crea una copia del array (usando `.copy()`)
- Modifica elementos en la vista y la copia
- Verifica qué arrays se modificaron y cuáles no
- Explica la diferencia entre vistas y copias

1.2. Ejercicios Intermedios

4. Análisis de Rendimiento

- Implementa tres versiones de una función que calcule la norma euclidiana:
  a) Versión con loop de Python
  b) Versión con NumPy vectorizado
  c) Versión con NumPy usando funciones específicas (`np.linalg.norm`)
- Compara el rendimiento para arrays de diferentes tamaños
- ¿Cuál es la diferencia de rendimiento entre las tres versiones?

5. Manipulación de Arrays

- Crea una matriz de 1000x1000 con valores aleatorios
- Implementa una función que encuentre el elemento máximo en cada fila
- Implementa una función que encuentre el elemento máximo en cada columna
- Implementa una función que normalice cada fila (media=0, std=1)
- Compara el rendimiento de tus implementaciones

2. Broadcasting

2.1. Ejercicios Básicos

6. Broadcasting Básico

- Crea un array 1D de 5 elementos
- Crea un escalar
- Realiza operaciones aritméticas entre el array y el escalar
- Explica cómo funciona el broadcasting en cada caso

7. Broadcasting con Arrays 2D

- Crea una matriz de 3x4
- Crea un vector de 4 elementos
- Realiza operaciones entre la matriz y el vector
- ¿Qué forma tiene el resultado? ¿Por qué?

8. None y np.newaxis

- Crea un array 1D de 6 elementos
- Usa `None` para crear una matriz de 6x1
- Usa `None` para crear una matriz de 1x6
- Realiza operaciones entre estos arrays y explica los resultados

2.2. Ejercicios Intermedios

9. Cálculo de Baselines del Interferómetro

- Simula las posiciones de 100 antenas en un interferómetro (coordenadas X, Y en metros)
- Calcula todas las distancias entre pares de antenas (baselines) usando broadcasting
- Usa broadcasting para evitar loops explícitos
- Compara con una implementación usando loops
- ¿Cuántas líneas de base únicas tienes? ¿Por qué es importante este cálculo?

10. Análisis de Correlación entre Antenas

- Simula datos de ruido de 1000 antenas durante 10 observaciones
- Calcula la matriz de covarianza del ruido usando broadcasting
- Compara con `np.cov()`
- ¿Son los resultados equivalentes? ¿Qué información te da esta matriz sobre el interferómetro?

11. Correlación Espacial de Fuentes

- Implementa análisis de correlación espacial usando broadcasting
- Crea un mapa de cielo con fuentes puntuales distribuidas aleatoriamente
- Calcula correlaciones espaciales entre todas las fuentes usando broadcasting
- Identifica cúmulos y estructuras en la distribución de fuentes
- Compara el rendimiento con una implementación usando loops
- ¿Qué información sobre la estructura del universo puedes extraer?

3. Dask

3.1. Ejercicios Básicos

12. Configuración de Dask

- Configura un cliente Dask con 4 workers usando `Client()`
- Crea un array Dask de 10000x10000 con chunks de 1000x1000
- Realiza operaciones básicas (suma, multiplicación, etc.)
- Usa `compute()` para obtener resultados
- ¿Cuánto tiempo toma cada operación?

13. Lazy Evaluation

- Crea una cadena de operaciones Dask (sin usar `compute()`)
- Inspecciona el grafo de tareas generado usando `dask.visualize()` o `result.visualize()`
- Usa `compute()` al final para ejecutar todo

- Compara con ejecutar cada operación por separado
- ¿Qué optimizaciones automáticas puedes observar en el grafo?

14. compute() vs persist()

- Crea un array Dask grande
- Aplica varias transformaciones
- Usa `persist()` para mantener datos intermedios
- Realiza múltiples operaciones sobre los datos persistidos
- Compara el rendimiento con usar `compute()` en cada operación

3.2. Ejercicios Intermedios

15. Procesamiento de Datos de ALMA

- Simula un dataset de visibilidades de ALMA con 10 millones de puntos por noche
- Implementa filtrado de ruido atmosférico usando Dask
- Calcula estadísticas descriptivas del ruido (media, desviación estándar, etc.)
- Detecta anomalías en las observaciones usando umbrales estadísticos
- ¿Puedes procesar estos datos sin Dask? ¿Por qué es crítico el procesamiento en tiempo real?

16. Análisis de Espectros de Radio

- Genera datos espectrales para 1 millón de canales de frecuencia (amplitud y fase)
- Implementa ajuste de líneas espectrales usando regresión distribuida con Dask
- Usa regresión para ajustar perfiles gaussianos a las líneas de emisión detectadas
- Calcula parámetros de las líneas (centro, ancho, amplitud) usando mínimos cuadrados distribuidos
- Identifica líneas de emisión y absorción en los espectros
- Compara con una implementación usando NumPy
- ¿Qué información astronómica puedes extraer de los parámetros de las líneas?

4. Numba

4.1. Ejercicios Básicos

17. Decorador @jit

- Implementa una función que calcule la serie de Fibonacci
- Aplica el decorador `@numba.jit`
- Compara el rendimiento con y sin el decorador
- ¿Cuál es la aceleración obtenida?

18. Decorador @njit

- Implementa una función que calcule el producto punto de dos vectores
- Usa `@numba.njit` para máximo rendimiento
- Compara con `np.dot()`
- ¿Cuál es más rápido? ¿Por qué?

19. Decorador @vectorize

- Implementa una función sigmoide
- Usa `@numba.vectorize` para crear una función universal
- Aplica la función a arrays de diferentes tamaños
- Compara con una implementación usando NumPy

4.2. Ejercicios Intermedios

20. prange para Paralelización

- Implementa una función que calcule la transformada de Fourier discreta (DFT)
- Usa `numba.prange` para paralelizar los loops
- Compara con una implementación secuencial
- ¿Cuál es la aceleración obtenida?

21. Simulación de Ruido Atmosférico

- Implementa una simulación Monte Carlo para modelar ruido atmosférico en observaciones
- Usa Numba para acelerar la simulación de 10 millones de muestras
- Compara con una implementación en Python puro
- ¿Cuántas iteraciones necesitas para una precisión dada? ¿Cómo afecta el ruido a la calidad de las observaciones?

5. Integración de Herramientas

5.1. Ejercicios Avanzados

22. Análisis Completo de Datos

- Simula un dataset de 100 millones de puntos con 20 características
- Usa Dask para manejar la escalabilidad
- Implementa funciones de procesamiento con Numba
- Usa broadcasting para operaciones eficientes
- Calcula estadísticas descriptivas y detecta patrones
- ¿Cuál es el tiempo total de procesamiento?

23. Machine Learning Distribuido

- Implementa regresión logística distribuida
- Usa Dask para datos que no caben en memoria
- Optimiza funciones críticas con Numba
- Implementa el algoritmo de descenso de gradiente
- Compara con implementaciones estándar
- ¿Cuál es la precisión y rendimiento obtenidos?

24. Simulación Científica

- Implementa una simulación de dinámica molecular simple
- Usa Dask para paralelizar múltiples simulaciones
- Optimiza cálculos de fuerzas con Numba
- Usa broadcasting para operaciones vectoriales
- Analiza los resultados y calcula propiedades termodinámicas
- ¿Cuántas partículas puedes simular eficientemente?

6. Problemas de Desafío

25. Procesamiento de Mapas de Cielo

- Implementa filtros especializados para mapas de radio (suavizado, detección de bordes, etc.)
- Usa Dask para procesar mapas de cielo de alta resolución (10000x10000 píxeles)
- Optimiza operaciones pixel por pixel con Numba para detectar fuentes astronómicas
- Usa broadcasting para operaciones eficientes de convolución
- Procesa un mapa completo del cielo y detecta fuentes puntuales
- ¿Cuál es el tiempo de procesamiento? ¿Cuántas fuentes puedes detectar?

26. Análisis de Series Temporales

- Simula datos de series temporales de alta frecuencia
- Implementa detección de anomalías en tiempo real
- Usa Dask para procesamiento en streaming
- Optimiza algoritmos de detección con Numba
- Implementa alertas automáticas
- ¿Puedes procesar datos en tiempo real?

27. Optimización de Portafolio

- Implementa optimización de portafolio usando Markowitz
- Usa Dask para cálculos de covarianza distribuida
- Optimiza algoritmos de optimización con Numba
- Implementa backtesting de estrategias
- Analiza riesgo y retorno de diferentes portafolios
- ¿Cuántos activos puedes optimizar eficientemente?

7. Instrucciones para los Ejercicios

7.1. Configuración del Entorno

Instalación de dependencias:

- `pip install numpy dask numba matplotlib pandas`

Importaciones básicas:

- `import numpy as np`
- `import dask.array as da`
- `import numba`
- `import time`
- `import matplotlib.pyplot as plt`

7.2. Criterios de Evaluación

- **Correctitud**: El código debe producir resultados correctos
- **Eficiencia**: Debe demostrar mejoras de rendimiento
- **Claridad**: El código debe ser legible y bien comentado
- **Innovación**: Soluciones creativas y optimizadas

7.3. Consejos para Resolver los Ejercicios

1. **Empezar simple**: Implementa primero una versión básica
2. **Medir rendimiento**: Siempre compara tiempos de ejecución
3. **Verificar resultados**: Asegúrate de que las optimizaciones no cambien los resultados
4. **Documentar**: Explica tus decisiones de optimización
5. **Experimentar**: Prueba diferentes configuraciones y parámetros

8. Recursos Adicionales

- **Documentación oficial**: NumPy, Dask, Numba
- **Tutoriales interactivos**: Jupyter notebooks
- **Ejemplos de código**: GitHub repositories
- **Benchmarks**: Comparaciones de rendimiento
- **Comunidades**: Stack Overflow, Reddit, Discord

¡Buena suerte con los ejercicios!