Блочне Множення Матриць

Виконав Пристайчук Дмитро, ММШІ-1

Цей звіт описує алгоритм блочного множення матриць, реалізований у Python з використанням NumPy. Алгоритм ділить матриці на блоки фіксованого розміру `L x L` і виконує множення на рівні цих блоків.

Крок 1: Ініціалізація результуючої матриці

Створюється нульова матриця `C` розміром `n x p`, де `n` - кількість рядків матриці `A`, а `p` - кількість стовпців матриці `B`.

```
# Initialize the result matrix
C = np.zeros((n, p))
```

Крок 2: Ітерація по блоках матриці C

Алгоритм ітерує по блоках результуючої матриці `C`. Зовнішні цикли проходять по рядках (`i`) та стовпцях (`j`) з кроком `L`.

```
# Iterate through blocks of C
for i in range(0, n, L):
    i_end = min(i + L, n)
    for j in range(0, p, L):
        j_end = min(j + L, p)
        # ... process block C[i:i_end, j:j_end] ...
```

Крок 3: Обчислення кожного блоку C

Для кожного блоку `C[i:i_end, j:j_end]`, ініціалізується нульовий блок `C_block`. Потім виконується ітерація по відповідних блоках матриць `A` та `B` (внутрішній цикл по `k` з кроком `L`). Важливо зазначити, що функція множення не має доступу до повних матриць `A` та `B` одночасно. Замість цього, вона отримує необхідні блоки матриць `A` та `B` за запитом через функцію `get_block`. Ці матриці заздалегідь розділені на блоки розміром не більше `L x L`.

```
        # Initialize block of C
        C_block = np.zeros((i_end - i, j_end - j))

        # Multiply the corresponding blocks from A and B
        for k in range(0, m, L):
            k_end = min(k + L, m)
            # ... fetch and multiply A_block and B_block ...
```

Крок 4: Отримання та множення блоків A та B

На кожній ітерації внутрішнього циклу (`k`) отримуються відповідні блоки `A_block` (розміром `(i_end-i) x (k_end-k)`) та `B_block` (розміром `(k_end-k) x (j_end-j)`) за допомогою функції `get_block`. Ці блоки перемножуються стандартним множенням матриць (`@`), і результат додається до `C_block`.

```
            # Get blocks from A and B
            A_block = get_block('A', i, i_end, k, k_end)
            B_block = get_block('B', k, k_end, j, j_end)

            # Multiply and accumulate result
            C_block += A_block @ B_block
```

Крок 5: Запис результату в матрицю C

Після завершення внутрішнього циклу (`k`), обчислений блок `C_block` записується у відповідне місце в результуючій матриці `C`.

```
        # Set the result in C
        C[i:i_end, j:j_end] = C_block
```

Крок 6: Повернення результату

Після обробки всіх блоків функція повертає повну результуючу матрицю `C`.

```
return C
```

Повна реалізація функції в Python

```
import numpy as np

def block_matrix_multiply(n, m, p, L, get_block):
    """
    Multiply matrices A and B using block approach.
    Matrices are divided into blocks of size LxL.

    Parameters:
    -----------
    n, m, p : int
        Dimensions of matrices (A is n×m, B is m×p)
    L : int
        Size of blocks for matrix division
```

```
    get_block : callable
        Function that returns a block of either A or B matrix
        Signature: get_block(matrix_name, row_start, row_end, col_start,
col_end)
        matrix_name should be either 'A' or 'B'

    Returns:
    --------
    C : numpy.ndarray
        Result matrix C = A×B
    """
    # Initialize the result matrix
    C = np.zeros((n, p))

    # Iterate through blocks of C
    for i in range(0, n, L):
        i_end = min(i + L, n)
        for j in range(0, p, L):
            j_end = min(j + L, p)

            # Initialize block of C
            C_block = np.zeros((i_end - i, j_end - j))

            # Multiply the corresponding blocks from A and B
            for k in range(0, m, L):
                k_end = min(k + L, m)

                # Get blocks from A and B
                A_block = get_block('A', i, i_end, k, k_end)
                B_block = get_block('B', k, k_end, j, j_end)

                # Multiply and accumulate result
                C_block += A_block @ B_block

            # Set the result in C
            C[i:i_end, j:j_end] = C_block

    return C
```

Результати виконання (Приклад)

Результати для множення матриць 1024x1024 з розміром блоку L=64:

```
Dividing matrices into blocks of size 64×64...
Multiplying matrices 1024×1024 and 1024×1024 using blocks 64×64
Block multiplication time: 0.0306 seconds
NumPy time: 0.0076 seconds
Maximum difference: 1.19e-12
```