HW1

ESSA.ALQALLAF 222121385

February 2024

1 Solution

I will use Strassen's algorithm to solve matrix multiplication that uses the divide-and-conquer method to have better time complexity which reduces the time from $O(n^3)$ to $O(n^2.81)$. Simply, in this algorithm each matrix will be divided to four sections and six formulas as shown below will be applied on these sections. Then the results can be computed in the result formula as shown below.

p1 = a(f - h)
p2 = (a + b)h
p3 = (c + d)e
p4 = d(g - e)
p5 = (a + d)(e + h)
p6 = (b - d)(g + h)
p7 = (a - c)(e + f)

The A x B can be calculated using above seven multiplications.
Following are values of four sub-matrices of result C

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} p5 + p4 - p2 + p6 & p1 + p2 \\ p3 + p4 & p1 + p5 - p3 - p7 \end{bmatrix}$$

A B C

A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2
p1, p2, p3, p4, p5, p6 and p7 are submatrices of size N/2 x N/2

2 The code using python

```
import numpy as np

def split(matrix):
    """
    Splits a given matrix into quarters.
```

```
    Input: nxn matrix
    Output: tuple containing 4 n/2 x n/2 matrices corresponding to a, b, c, d
    """
    row, col = matrix.shape
    row2, col2 = row//2, col//2
    return matrix[:row2, :col2], matrix[:row2, col2:], matrix[row2:, :col2], mat

def strassen(x, y):
    """
    Computes matrix product by divide and conquer approach, recursively.
    Input: nxn matrices x and y
    Output: nxn matrix, product of x and y
    """

    # Base case when size of matrices is 1x1
    if len(x) == 1:
        return x * y

    # Splitting the matrices into quadrants. This will be done recursively
    # until the base case is reached.
    a, b, c, d = split(x)
    e, f, g, h = split(y)

    # Computing the 7 products, recursively (p1, p2...p7)
    p1 = strassen(a, f - h)
    p2 = strassen(a + b, h)
    p3 = strassen(c + d, e)
    p4 = strassen(d, g - e)
    p5 = strassen(a + d, e + h)
    p6 = strassen(b - d, g + h)
    p7 = strassen(a - c, e + f)

    # Computing the values of the 4 quadrants of the final matrix c
    c11 = p5 + p4 - p2 + p6
    c12 = p1 + p2
    c21 = p3 + p4
    c22 = p1 + p5 - p3 - p7

    # Combining the 4 quadrants into a single matrix by stacking horizontally an
    c = np.vstack((np.hstack((c11, c12)), np.hstack((c21, c22))))

    return c
```

3 Strassen's algorithm complexity O($2^8$1)

As shown in the code there are addition and subtraction methods in the code, so their complexity will be O[n^2) and we divided the mercies to four subs (N/2), so we can conclude that$T(N) = 7T(N/2) + O(N^2) = O(N^Log7) = O(N^2.81)$ [1]

References

[1] GeeksforGeeks. *Divide and Conquer Set 5 Strassen s Matrix Multiplication.* URL: `https://www.geeksforgeeks.org/strassens-matrix-multiplication/`.