Strassen's Algorithm for Matrix Multiplication

ESO207

Indian Institute of Technology, Kanpur

Introduction

- We will consider the problem of multiplying two $n \times n$ matrices. For example

$$\begin{pmatrix} 1 & 4 & 6 \\ 2 & 7 & 5 \\ -1 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 3 & -1 & 2 \\ 2 & 5 & 3 \\ -2 & -1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 13 & 14 \\ 10 & 28 & 25 \\ -5 & 13 & 7 \end{pmatrix}$$

- Consider multiplication of two $n \times n$ matrices $A \times B$. If $A = (a_{ij})$, and $B = (b_{ij})$, where, $1 \leq i, j \leq n$, then, the product $C = A \times B$ is the $n \times n$ matrix $C = (c_{ij})$, with

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Classical Matrix Multiplication

```
SQUARE-MATRIX-MULTIPLY (A, B, n) {
// A, B are n × n matrices
    Let C be a new n × n matrix
    for i = 1 to n {
        for j = 1 to n {
            c_ij = 0
            for k = 1 to n {
                c_ij = c_ij + a_ik · b_kj
            }
        }
    }
    return C
}
```

Square Matrix Multiplication: Classical

- The c_{ij} entry is the dot product of the ith row of A and the jth column of B.
- Each c_{ij} entry is computed as a dot product involving n multiplications and n additions. Time taken is $O(n)$.
- There are n^2 entries (c_{ij}) for $1 \leq i, j \leq n$ to compute.
- Total time required is $n^2 \times O(n) = n^3$.
- Alternatively: this is a triple loop with indices running from 1 to n each, giving $O(n^3)$ total time.

Summary

- It might appear that square matrix multiplication should take time $\Omega(n^3)$.
- We will see Strassen's algorithm that solves square matrix multiplication in time $O(n^{2.81\cdots}) = O(n^{\log_2 7})$.
- We will first look at a simple divide-and-conquer algorithm and then understand Strassen's algorithm.

Simple Divide and Conquer Algorithm

- Goal: To compute $C = A \times B$.
- For simplicity, assume that n is an exact power of 2.
- View A, B, C as 2×2 block matrices where, each block is of size $n/2 \times n/2$.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

- Rewrite equation $C = A \cdot B$ as

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Block matrix multiplication

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

gives four equations.

$$\begin{aligned} C_{11} &= A_{11} \times B_{11} + A_{12} \times B_{21} \\ C_{12} &= A_{11} \times B_{12} + A_{12} \times B_{22} \\ C_{21} &= A_{21} \times B_{11} + A_{22} \times B_{21} \\ C_{22} &= A_{21} \times B_{12} + A_{22} \times B_{22} \end{aligned}$$

- In the above $\times$ denotes multiplication of $n/2 \times n/2$ matrices and $+$ denotes sum of $n/2 \times n/2$ matrices.

Translating into Recursive matrix multiplication

```
SQ-MAT-MULT-RECURSIVE(A, B, n)
    Let C be a new n × n matrix
    if n==1
        c11 = a11 · b11
    else  partition A, B, C into block sub-matrices as done earlier
        C11 = SQ-MAT-MULT-RECURSIVE(A11, B11)
            + SQ-MAT-MULT-RECURSIVE(A12, B21)
        C12 = SQ-MAT-MULT-RECURSIVE(A11, B12)
            + SQ-MAT-MULT-RECURSIVE(A12, B22)
        C21 = SQ-MAT-MULT-RECURSIVE(A21, B11)
            + SQ-MAT-MULT-RECURSIVE(A22, B21)
        C22 = SQ-MAT-MULT-RECURSIVE(A21, B12)
            + SQ-MAT-MULT-RECURSIVE(A22, B22)
```

Index Calculations

- To partition A, B, C into the $n/2 \times n/2$ blocks, there is no need to copy the sub-matrices.
- This overhead is not needed, we represent a $2^k \times 2^k$ sub-matrix A starting at row position i and column j as $(A, i, j, 2^k)$.
- Represent submatrices by index calculations.
- That is, we represent sub-matrices slightly more generally than we represent the original matrix.

Analysis

- Let $T(n)$ be the time to multiply two $n \times n$ matrices using the block-multiplication algorithm.
- Base Case: $T(1) = \Theta(1)$.
- Consider one of the four block multiplications:

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$$

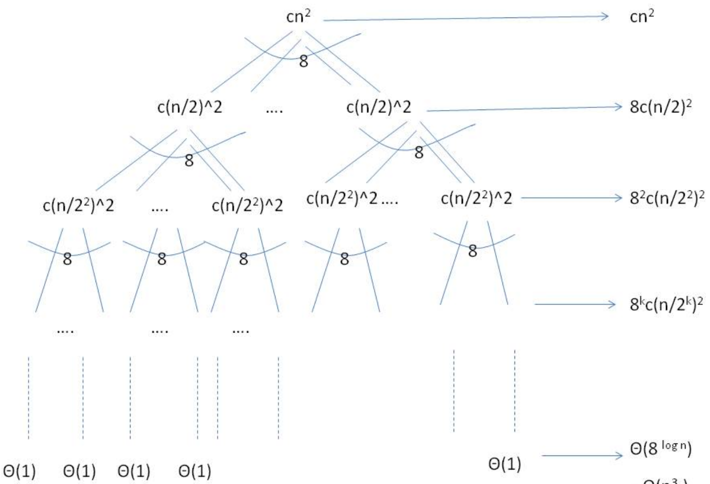
  where, $A_{11}, \ldots A_{22}$ and $B_{11}, \ldots, B_{22}$ are each $n/2 \times n/2$ matrices.
- The addition operation requires $\Theta(n^2)$ steps, since it adds two $n/2 \times n/2$ matrices.
- The multiplication operation is done recursively, and this requires $T(n/2) + T(n/2) = 2T(n/2)$ time.
- There are three other block equalities for C_{12}, C_{21} and C_{22}. Each of them requires $2T(n/2) + \Theta(n^2)$ time.

Analysis: Basic recursive algorithm

- The recurrence equation is (assuming n is an exact power of 2).

$$
\begin{aligned}
T(n) &= 8T(n/2) + \Theta(n^2) \\
T(1) &= \Theta(1)
\end{aligned}
$$

Recursion tree



Cost = $cn^2 + 2cn^2 + 4cn^2 + \ldots + 2^{\log n}(cn^2) = \Theta(n^3)$

Recurrence solution

- Thus solution to the recurrence equation

$$T(n) = 8T(n/2) + \Theta(n^2)$$
$$T(1) = \Theta(1)$$

is $T(n) = \Theta(n^3)$.

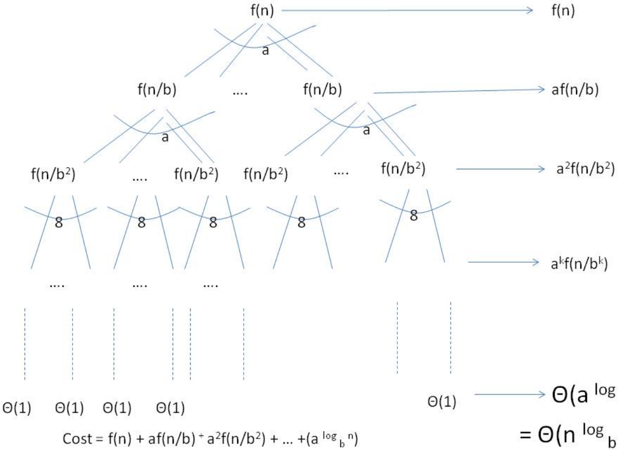
Master Theorem: a cookbook for recurrence equations

- Master Theorem (part-I). Consider the recurrence equation

$$T(n) = \begin{cases} aT(n/b) + f(n) & \text{if } n \geq b \\ \Theta(1) & \text{if } n = 1 \end{cases}.$$

  where, $a \geq 1$ and $b > 1$ and n/b means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.
- If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then, $T(n) = \Theta(n^{\log_b a})$.

Master Theorem -I: Recurrence tree



$$\Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$$

Master Theorem -I: Cost

- Assume n is a power of b.
- The total cost is

$$= f(n) + af(n/b) + a^2 f(n/b^2) + \ldots + a^{\log_b n}\Theta(1)$$

- Since,

$$a^{\log_b n} = b^{(\log_b a)(\log_b n)} = \left(b^{\log_b n}\right)^{\log_b a} = n^{\log_b a}$$

- total cost is $= \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$

$$= \Theta\left(n^{\log_b a} + \frac{n^{\log_b a}}{a} f(b) + \frac{n^{\log_b a}}{a^2} f(b^2) + \ldots + f(n)\right)$$

Master theorem-I: Cost

- The condition $f(n) = O(n^{\log_b a - \epsilon})$ implies that

$$f(b^j) = O(b^{j(\log_b a - \epsilon)}) = a^j \cdot b^{-j\epsilon}$$

Solution to recurrence equation

- Therefore, total cost is

$$= \Theta\left(n^{\log_b a} + \frac{n^{\log_b a}}{a} f(b) + \frac{n^{\log_b a}}{a^2} f(b^2) + \ldots + f(n)\right)$$

$$= \Theta\left(\sum_{j=0}^{\log_b n} \frac{n^{\log_b a}}{a^j} f(b^j)\right)$$

$$= O\left(\sum_{j=0}^{\log_b n} n^{\log_b a} b^{-j\epsilon}\right), \text{ since, } f(b^j) = O(a^j b^{-j\epsilon})$$

$$= O\left(n^{\log_b a}\right) \sum_{j=0}^{\log_b n} b^{-j\epsilon}$$

$$= O\left(n^{\log_b a} \cdot \frac{1}{1 - b^{-\epsilon}}\right) = O\left(n^{\log_b a}\right)$$

Total Cost

- Since the first term is $\Theta(n^{\log_b a})$, the total cost is $\Theta(n^{\log_b a})$.
- The recursion tree is cost-wise bottom-heavy.

Application of Master Theorem-I

- Consider the recurrence equation

$$T(n) = \begin{cases} 8T(n/2) + \Theta(n^2) & \text{if } n = 2^k, k \geq 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

- Applying Master Theorem-I, $a = 8, b = 2$ and $f(n) = \Theta(n^2)$, we have,

$$n^2 = O(n^{\log_2 8 - \epsilon}) = O(n^{3-\epsilon})$$

  for every $0 < \epsilon < 1$.

- Hence, by Master theorem, the solution is

$$T(n) = \Theta(n^{\log_2 8}) = \Theta(n^3) \quad .$$

Strassen's Algorithm: Overview

- Strassen's algorithm performs only seven recursive multiplications of $n/2 \times n/2$ matrices instead of eight.
- It performs several new additions of $n/2 \times n/2$ matrices, but only a constant number. Total cost of additions remains $\Theta(n^2)$.

Strassen's Algorithm: Overview

1. Divide the input matrices A and B and output matrix C into $n/2 \times n/2$ submatrices, as before. This takes $O(1)$ time by index calculation.
2. Create 10 matrices $S_1, S_2, \ldots, S_{10}$ each of which is $n/2 \times n/2$ and is the sum or difference of two matrices created in Step 1. This takes time $\Theta(n^2)$.
3. Using the submatrices of A, B and C, and $S_1, \ldots, S_{10}$, recursively compute seven matrix products $P_1, P_2, \ldots, P_7$. Each matrix P_i is $n/2 \times n/2$.
4. The output sub-matrices $C_{11}, C_{12}, C_{21}, C_{22}$ of the result matrix C is computed by adding and subtracting various combinations of the P_i matrices. This step can be done in time $\Theta(n^2)$ time.

Recurrence for Strassen's Algorithm

- We will see the exact calculations later, but we can write the recurrence relation for Strassen's method.
- Steps 1,2, and 4 require time $\Theta(n^2)$.
- Step 3 uses 7 recursive matrix multiplications of $n/2 \times n/2$ matrices. This gives

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1 \end{cases}$$

- By Master-theorem-I, the solution is

$$T(n) = \Theta(n^{\log_2 7}) = \Theta(n^{2.81\cdots}) \ .$$

Strassen's method

- Recall that we have four submatrices for each of A, B and C.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

- Create the following ten matrices $S_1, \ldots, S_{10}$.

$$\begin{aligned} S_1 &= B_{12} - B_{22} & S_2 &= A_{11} + A_{12} \\ S_3 &= A_{21} + A_{22} & S_4 &= B_{21} - B_{11} \\ S_5 &= A_{11} + A_{22} & S_6 &= B_{11} + B_{22} \\ S_7 &= A_{12} - A_{22} & S_8 &= B_{21} + B_{22} \\ S_9 &= A_{11} - A_{21} & S_{10} &= B_{11} + B_{12} \end{aligned}$$

- These addition and subtraction of $n/2 \times n/2$ matrices can be done in time $\Theta(n^2)$.

Strassen's method

- Recursively multiply $n/2 \times n/2$ matrices seven times.

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 = A_{11} \cdot B_{12} - A_{11} \cdot B_{22} \\
P_2 &= S_2 \cdot B_{22} = A_{11} \cdot B_{22} + A_{12} \cdot B_{22} \\
P_3 &= S_3 \cdot B_{11} = A_{21} \cdot B_{11} + A_{22} \cdot B_{11} \\
P_4 &= A_{22} \cdot S_4 = A_{22} \cdot B_{21} - A_{22} \cdot B_{11} \\
P_5 &= S_5 \cdot S_6 = A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \\
P_6 &= S_7 \cdot S_8 = A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22} \\
P_7 &= S_9 \cdot S_{10} = A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}
\end{aligned}
$$

- Note that the only multiplications are those in the middle column. The *RHS* shows the equivalent products.

Strassen's method

- Step 4 adds and subtracts various P_i matrices to construct the four $n/2 \times n/2$ sub-matrices of the product C.

$$C_{11} = P_5 + P_4 - P_2 + P_6$$

$$\begin{array}{l}
A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \\
\qquad\qquad\qquad\qquad - A_{22} \cdot B_{11} \qquad\qquad + A_{22} \cdot B_{21} \\
\qquad\quad - A_{11} \cdot B_{22} \qquad\qquad\qquad\qquad\qquad\qquad - A_{12} \cdot B_{22} \\
\qquad\qquad\qquad\qquad\qquad - A_{22} \cdot B_{22} - A_{22} \cdot B_{21} + A_{12} \cdot B_{22} + A_{12} \cdot B_{21} \\
\hline
A_{11} \cdot B_{11} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + A_{12} \cdot B_{21}
\end{array}$$

which is the expression for C_{11}.

Strassen's method

- Set

$$C_{12} = P_1 + P_2$$

C_{12} equals

$$\begin{array}{l}
A_{11} \cdot B_{12} - A_{11} \cdot B_{22} \\
\qquad\qquad + A_{11} \cdot B_{22} + A_{12} \cdot B_{22} \\
\hline
A_{11} \cdot B_{12} \qquad\qquad + A_{12} \cdot B_{22}
\end{array}$$

which corresponds to the expression for C_{12}.

- Setting

$$C_{21} = P_3 + P_4$$

makes C_{21} equal to

$$\begin{array}{l} A_{21} \cdot B_{11} + A_{22} \cdot B_{11} \\ \qquad\qquad - A_{22} \cdot B_{11} + A_{22} \cdot B_{21} \\ \hline A_{21} \cdot B_{11} \qquad\qquad\qquad + A_{22} \cdot B_{21} \end{array}$$

which is the expression for C_{21}.

- Finally, set

$$C_{22} = P_5 + P_1 - P_3 - P_7$$

so that C_{22} equals

$$\begin{array}{l}
A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \\
\qquad - A_{11} \cdot B_{22} \qquad\qquad\qquad\qquad + A_{11} \cdot B_{12} \\
\qquad\qquad\qquad - A_{22} \cdot B_{11} \qquad\qquad\qquad\qquad\qquad - A_{21} \cdot B_{11} \\
- A_{11} \cdot B_{11} \qquad\qquad\qquad\qquad\qquad - A_{11} \cdot B_{12} + A_{21} \cdot B_{11} + A_{21} \cdot B_{12} \\
\hline
\qquad\qquad\qquad\qquad\qquad + A_{22} \cdot B_{22} \qquad\qquad\qquad\qquad + A_{21} \cdot B_{12}
\end{array}$$

which is the expression for C_{22}.