CSC006P1M: Design and Analysis of Algorithms
Lecture 14 *(Matrix Multiplication)*

Sumit Kumar Pandey

September 26, 2022

Matrix Multiplication

Let

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}; B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix}$$

and $C = AB$

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

where $c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$.

Time Complexity of Matrix Multiplication

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

- Total Multiplications $= n$.
- Total Additions $= n - 1$.

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

- Total Multiplications $= n^2 \cdot n = n^3$.
- Total Additions $= n^2 \cdot (n - 1)$.

Winograd's Algorithm

For simplicity, assume that n is even. Let

$$A_i = \sum_{k=1}^{n/2} a_{i,2k-1} a_{i,2k}$$

and

$$B_j = \sum_{k=1}^{n/2} b_{2k-1,j} b_{2k,j}$$

Then,

$$c_{i,j} = \sum_{k=1}^{n/2} (a_{i,2k-1} + b_{2k,j}) \cdot (a_{i,2k} + b_{2k-1,j}) - A_i - B_j.$$

Winograd's Algorithm

$$A_i = \sum_{k=1}^{n/2} a_{i,2k-1} a_{i,2k}$$

Total number of multiplications $= n/2$, additions $= n/2 - 1$.

$$B_j = \sum_{k=1}^{n/2} b_{2k-1,j} b_{2k,j}$$

Total number of multiplications $= n/2$, additions $= n/2 - 1$.

$$c_{i,j} = \sum_{k=1}^{n/2} (a_{i,2k-1} + b_{2k,j}) \cdot (a_{i,2k} + b_{2k-1,j}) - A_i - B_j.$$

Total number of multiplications $= n/2 + n/2 + n/2 = 3n/2$,
Additions $= n + (n/2 - 1) + (n/2 - 1) + (n/2 - 1) + 2 = 5n/2 - 1$.

Winograd's Algorithm

First Approach

$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$

- Total Multiplications $= n$, Additions $= n - 1$.

Winograd's Algorithm

$c_{i,j} = \sum_{k=1}^{n/2} (a_{i,2k-1} + b_{2k,j}) \cdot (a_{i,2k} + b_{2k-1,j}) - A_i - B_j.$

- Total Multiplications $= 3n/2$, Additions $= 5n/2 - 1$.

No Improvement???

No. Wrong calculation regarding Winograd's algorithm.

Winograd's Algorithm

Note that,

- A_i for $1 \leq i \leq n$ need to be calculated only once.
- Similarly, B_j for $1 \leq j \leq n$ also need to be calculated only once.

Total number of multiplications and additions for $\{A_i\}_{i=1}^{n} = n^2/2$ and $n(n/2-1)$ respectively. And so for B_j also. So,

Winograd's Algorithm

$c_{i,j} = \sum_{k=1}^{n/2}(a_{i,2k-1} + b_{2k,j}) \cdot (a_{i,2k} + b_{2k-1,j}) - A_i - B_j.$

for $1 \leq i,j \leq n$ require $n^2(n/2) + n^2/2 + n^2/2 = n^3/2 + n^2$ multiplications and
$n^2(2 \cdot n/2 + (n/2-1) + 2) + 2n(n/2-1) = 3n^3/2 + 2n^2 - 2n$ additions.

Winograd's Algorithm

	Usual	Winograd's
Multiplications	n^3	$n^3/2 + n^2$
Additions	$n^3 - n^2$	$3n^3/2 + 2n^2 - 2n$

Can we do better?

Strassen's Algorithm

Let

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}; B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

and $C = AB$

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

$T(n) = 8T(n/2) + \Theta(n^2)$
$T(n) = \Theta(n^3)$.

Strassen's Algorithm

- $C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$
- $C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$
- $C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
- $C_{2,2} = A_{2,1}B_{1,2} + A_{1,2}B_{2,2}$

Clearly, there are 8 matrix multiplications of order $n/2 \times n/2$.
Moreover, there are $4(n/2)^2 = n^2$ additions.
Therefore,
$T(n) = 8T(n/2) + \Theta(n^2)$ which implies
$T(n) = \Theta(n^{\lg 8}) = \Theta(n^3)$.

Can we do better?

Strassen's Algorithm

Once again, let A, B and C be $n \times n$ matrices.

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}; B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

and $C = AB$

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

where $A_{i,j}, B_{i,j}$, and $C_{i,j}$ for $1 \leq i, j \leq 2$ are $n/2 \times n/2$ matrices.

Strassen's Algorithm

Compute,

1. $S_1 = B_{1,2} - B_{2,2}$,
2. $S_2 = A_{1,1} + A_{1,2}$,
3. $S_3 = A_{2,1} + A_{2,2}$,
4. $S_4 = B_{2,1} - B_{1,1}$,
5. $S_5 = A_{1,1} + A_{2,2}$,
6. $S_6 = B_{1,1} + B_{2,2}$,
7. $S_7 = A_{1,2} - A_{2,2}$,
8. $S_8 = B_{2,1} + B_{2,2}$,
9. $S_9 = A_{1,1} - A_{2,1}$,
10. $S_{10} = B_{1,1} + B_{1,2}$,

There are 10 additions of $n/2 \times n/2$ matrices, a total of $10 \cdot n^2/4$ additions which is $\Theta(n^2)$.

Then compute,

1. $P_1 = A_{1,1}S_1 = A_{1,1}B_{1,2} - A_{1,1}B_{2,2}$,
2. $P_2 = S_2B_{2,2} = A_{1,1}B_{2,2} + A_{1,2}B_{2,2}$,
3. $P_3 = S_3B_{1,1} = A_{2,1}B_{1,1} + A_{2,2}B_{1,1}$,
4. $P_4 = A_{2,2}S_4 = A_{2,2}B_{2,1} - A_{2,2}B_{1,1}$,
5. $P_5 = S_5S_6 = A_{1,1}B_{1,1} + A_{1,1}B_{2,2} + A_{2,2}B_{1,1} + A_{2,2}B_{2,2}$,
6. $P_6 = S_7S_8 = A_{1,2}B_{2,1} + A_{1,2}B_{2,2} - A_{2,2}B_{2,1} - A_{2,2}B_{2,2}$,
7. $P_7 = S_9S_{10} = A_{1,1}B_{1,1} + A_{1,1}B_{1,2} - A_{2,1}B_{1,1} - A_{2,1}B_{1,2}$

There are 7 multiplications of $n/2 \times n/2$ matrices.

Strassen's Algorithm

Finally compute,

1. $C_{1,1} = P_5 + P_4 - P_2 + P_6$,
2. $C_{1,2} = P_1 + P_2$,
3. $C_{2,1} = P_3 + P_4$,
4. $C_{2,2} = P_5 + P_1 - P_3 - P_7$.

There are 8 additions of $n/2 \times n/2$ matrices, a total of $8 \cdot n^2/4$ additions which is $\Theta(n^2)$.

Time Complexity of Strassen's Algorithm:

$$T(n) = 7T(n/2) + \Theta(n^2).$$

Thus,

$$T(n) = \Theta(n^{\lg 7}) = \Theta(n^{2.81})$$

Strassen found that 7 multiplications are sufficient to compute the product of two 2×2 matrices. We will see how to achieve this? Observe that,

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & g \\ f & h \end{bmatrix} = \begin{bmatrix} p & s \\ r & t \end{bmatrix}$$

is equivalent to

$$\begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} p \\ r \\ s \\ t \end{bmatrix}$$

We write the above matrix multiplication as $A \cdot X = Y$.

Type	Product	No. of Mults.
$\alpha)$	$\begin{bmatrix} a & a \\ a & a \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} a(e+f) \\ a(e+f) \end{bmatrix}$	1
$\beta)$	$\begin{bmatrix} a & a \\ -a & -a \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} a(e+f) \\ -a(e+f) \end{bmatrix}$	1
$\gamma)$	$\begin{bmatrix} a & 0 \\ a-b & b \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ae \\ ae+b(f-e) \end{bmatrix}$	2
$\delta)$	$\begin{bmatrix} a & b-a \\ 0 & b \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} a(e-f)+bf \\ bf \end{bmatrix}$	2

Come to the multiplication again.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & g \\ f & h \end{bmatrix} = \begin{bmatrix} p & s \\ r & t \end{bmatrix}$$

is equivalent to

$$\begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} p \\ r \\ s \\ t \end{bmatrix}$$

We write the above matrix multiplication as $A \cdot X = Y$.

$A = B + C + D + E + F$, where

$$A = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}, B = \begin{bmatrix} b & b & 0 & 0 \\ b & b & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & c & c \\ 0 & 0 & c & c \end{bmatrix},$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ c-b & 0 & 0 & c-b \\ b-c & 0 & 0 & b-c \\ 0 & 0 & 0 & 0 \end{bmatrix}, E = \begin{bmatrix} a-b & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ c-b & 0 & a-c & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & d-b & 0 & b-c \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d-c \end{bmatrix}.$$

Strassen's Algorithm

$AX = BX + CX + DX + EX + FX$ requires 7 multiplications because

- BX needs 1 multiplication (type α),
- CX needs 1 multiplication (type α),
- DX needs 1 multiplication (type β),
- EX needs 2 multiplications (type γ),
- FX needs 2 multiplications (type δ).

The same analysis is valid when we replace a, b, c, d, e, f, g, h with $n/2 \times n/2$ matrices $A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1$.

Strassen's Algorithm

- Strassen's algorithm uses Divide-and-Conquer paradigm.
- Strassen's algorithm works better than the straightforward $\Theta(n^3)$ algorithm when $n \geq 100$ as per empirical study.
- Strassen's algorithm is less stable than the straightforward algorithm, i.e. for similar errors in the input, Strassen's algorithm create larger errors in the output.
- Strassen's algorithm is much more complicated than the straightforward algorithm.
- Strassen's algorithm cannot be easily parallelized, whereas the regular algorithm can.
- Nevertheless, it is a useful algorithm when $n \geq 100$.

Thank You