Computational Complexity of Matrix Multiplication : An Analysis of Strassen's Algorithm

Joseph Kaplan

University of Utah

Contents

History

- Linear Algebra is a fundamental and extremely important aspect of Computer Science
  - However, matrix multiplication , a common operation of Linear Algebra, can be computationally expensive
- Volker Strassen was a German mathematician, born in 1936.
- In 1969, Strassen introduced Strassen's algorithm, which could outperform the traditional cubic time complexity associated with a naïve matrix multiplication algorithm.
- This algorithm was an extremely important breakthrough in computing and mathematics, allowing matrix multiplication to be performed quicker than previously thought possible.

Traditional Matrix Multiplication

Given two 2 by 2 matrices, A and B, find the product of these matrices using Traditional Matrix Multiplication:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},\ B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

The product of these two matrices, C, is defined below:

$C = A \times B = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$, where each entry of C requires two multiplications defined as:

$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$ $\quad C_{12} = A_{11} \times B_{12} + A_{12} \times B_{22}$

$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21}$ $\quad C_{22} = A_{21} \times B_{12} + A_{22} \times B_{22}$

Thus, traditional matrix multiplication requires 8 multiplications and 4 additions.

Traditional Matrix Multiplication Example

Given two 2 by 2 matrices, A and B, compute their product using Traditional Matrix Multiplication

$A=\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} -8 & 4 \\ 3 & 12 \end{bmatrix}$, $B=\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ -2 & 0 \end{bmatrix}$

The product of these two matrices, C, is defined below:

$C = A \times B = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$, where each entry of C requires two multiplications defined as:

$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$ $C_{12} = A_{11} \times B_{12} + A_{12} \times B_{22}$

$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21}$ $C_{22} = A_{21} \times B_{12} + A_{22} \times B_{22}$

Thus:

$C_{11} = -8 \times 3 + 4 \times -2 = 24 - 8 = 16$

$C_{12} = -8 \times 9 + 4 \times 0 = -72 + 0 = -72$

$C_{21} = 3 \times 3 + 12 \times -2 = 9 - 24 = -15$

$C_{21} = 3 \times 9 + 12 \times 0 = 27 + 0 = 27$

Upon constructing matrix C, we get our final matrix which represents the product of matrix A and matrix B:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 16 & -72 \\ -15 & 27 \end{bmatrix}$$

Strassen's Algorithm

Instead of following the computations outlined by traditional matrix multiplication, Strassen's algorithm defines new values, which will be used to find the product of two matrices, A and B. These values are outlined below:

$M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) \times B_{11}$

$M_3 = A_{11} \times (B_{12} - B_{22})$

$M_4 = A_{22} \times (B_{21} - B_{11})$

$M_5 = (A_{11} + A_{12}) \times B_{22}$

$M_6 = (A_{21} - A_{11}) \times (B_{21} + B_{12})$

$M_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$

Thus, we can define matrix C, the product of matrix A and matrix B, as follows:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}$$

Example of Strassen's Algorithm

Given two 2 by 2 matrices, A and B, find the product of A and B: $A = \begin{bmatrix} 1 & 6 \\ 2 & 4 \end{bmatrix}$, $B = \begin{bmatrix} -2 & 0 \\ 3 & -5 \end{bmatrix}$

Using the values defined by Strassen, we can compute each entry of matrix C, the product of matrix A and B

$M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22}) = (1 + 4) \times (-2 + (-5)) = 5 \times -7 = -35$
$M_2 = (A_{21} + A_{22}) \times B_{11} = (2 + 4) \times -2 = 6 \times -2 = -12$
$M_3 = A_{11} \times (B_{12} - B_{22}) = 1 \times (0 - (-5)) = 5$
$M_4 = A_{22} \times (B_{21} - B_{11}) = 4 \times (3 - (-2)) = 4 \times 5 = 20$
$M_5 = (A_{11} + A_{12}) \times B_{22} = (1 + 6) \times -5 = 7 \times -5 = -35$
$M_6 = (A_{21} - A_{11}) \times (B_{11} + B_{12}) = (2 - 1) \times (-2 + 0) = 1 \times -2 = -2$
$M_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22}) = (6 - 4) \times (3 + (-5)) = 2 \times -2 = -4$

Thus, we can create matrix C:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}$$

$$= \begin{bmatrix} -35 + 20 - (-35) + (-4) & 5 + (-35) \\ -12 + 20 & -35 - (-12) + 5 + (-2) \end{bmatrix} = \begin{bmatrix} 16 & -30 \\ 8 & -20 \end{bmatrix}$$

Via traditional matrix multiplication:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} -2 + 18 & 0 - 30 \\ -4 + 12 & 0 - 20 \end{bmatrix}$$

$$= \begin{bmatrix} 16 & -30 \\ 8 & -20 \end{bmatrix}$$

Example of Strassen's Algorithm "Padding"

Given two 2 by 2 matrices, A and B, find the product of A and B: $A = \begin{bmatrix}1\\2\end{bmatrix}$, B = [1 −4]

Using the values defined by Strassen, we can compute each entry of matrix C, the product of matrix A and B (Note: We will pad empty rows and columns with 0's)

$M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22}) = (1 + 0) \times (1 + 0) = 1 \times 1 = 1$
$M_2 = (A_{21} + A_{22}) \times B_{11} = (2 + 0) \times 1 = 2 \times 1 = 2$
$M_3 = A_{11} \times (B_{12} - B_{22}) = 1 \times (-4 - 0) = -4$
$M_4 = A_{22} \times (B_{21} - B_{11}) = 0 \times (0 - 1) = 0 \times -1 = 0$

$M_5 = (A_{11} + A_{12}) \times B_{22} = (1 + 0) \times 0 = 1 \times 0 = 0$
$M_6 = (A_{21} - A_{11}) \times (B_{11} + B_{12}) = (2 - 1) \times (1 + (-4)) = 1 \times -3 = -3$
$M_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22}) = (0 - 0) \times (0 + 0) = 0 \times 0 = 0$

Thus, we can create matrix C:

$$C = \begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix} = \begin{bmatrix}M_1 + M_4 - M_5 + M_7 & M_3 + M_5\\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6\end{bmatrix}$$

$$= \begin{bmatrix}1 + 0 - 0 + 0 & -4 + 0\\ 2 + 0 & 1 - 2 + (-4) + (-3)\end{bmatrix} = \begin{bmatrix}1 & -4\\ 2 & -8\end{bmatrix}$$

Via traditional matrix multiplication:

$$C = \begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix} = \begin{bmatrix}1 \times 1 & 1 \times -4\\ 2 \times 1 & 2 \times -4\end{bmatrix}$$

$$= \begin{bmatrix}1 & -4\\ 2 & -8\end{bmatrix}$$

Example of Winograd's Algorithm

Given two 2 by 2 matrices, A and B, find the product of A and B: $A = \begin{bmatrix} 1 & 6 \\ 2 & 4 \end{bmatrix}$, B = $\begin{bmatrix} -2 & 0 \\ 3 & -5 \end{bmatrix}$

Using the values defined by Winograd, we can compute each entry of matrix C, the product of matrix A and B

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} A & C \\ B & D \end{bmatrix} = \begin{bmatrix} t + b \times B & w + v + (a + b - c - d) \times D \\ w + u + d \times (B + C - A - D) & w + u + v \end{bmatrix}$$

$$where\ t = a \times A, u = (c - a) \times (C - D), v = (c + d) \times (C - A), w = t + (c + d - a) \times (A + D - C)$$

Thus, we can create matrix C (Process simplified to fit on page) :

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} -2 + 6 \times 3 & -37 + 12 + (1) \times -5 \\ -37 + 5 + 4 \times (10) & -37 + 5 + 12 \end{bmatrix} = \begin{bmatrix} 16 & -30 \\ 8 & -20 \end{bmatrix}$$

Via traditional matrix multiplication:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 1 \times -2 + 6 \times 3 & 1 \times 0 + 6 \times -5 \\ 2 \times -2 + 4 \times 3 & 2 \times 0 + 4 \times -5 \end{bmatrix}$$

$$= \begin{bmatrix} 16 & -30 \\ 8 & -20 \end{bmatrix}$$

Asymptotic complexity of Strassen's Algorithm

- Strassen's' algorithm only requires 7 multiplications instead of the 8 multiplications normally required by traditional matrix multiplication
- Strassen's' algorithm has a complexity of $O(N^{2.81})$, which is significantly better than the complexity of $O(N^3)$ that is required for traditional matrix multiplication.
  - We will examine a Python implementation of both matrix multiplication techniques in a few slides, where the difference in the coding techniques for both techniques will become apparent.
- While the difference between a time complexity of $O(N^{2.81})$ and $O(N^3)$ may seem almost negligible, this difference provides a monumental improvement in the time complexity of matrix multiplication as the size of the matrices being multiplied grows.

Implementation considerations

- Why does Strassen's algorithm yield a better time complexity than a traditional, naïve matrix multiplication algorithm?
  - Generally, a naïve approach to matrix multiplication requires 3 nested for-loops, giving a time complexity of $O(N^3)$. Strassen's algorithm avoids for-loops, opting for a divide-and-conquer technique instead.
- How does Strassen's algorithm scale as matrices grow?
  - Strassen's algorithm scales well as matrices grow and outperforms traditional matrix multiplication algorithms for large matrices.
- At what size does Strassen's algorithm outperform a traditional, naïve matrix multiplication algorithm?
  - For matrices of size 128x128 up to 1024x1024, Strassen's algorithm seems to perform the most efficiently. However, even though Strassen's algorithm is quite good for evaluating matrices within this size range, once matrices exceed this size range, Strassen's algorithm can encounter recursion errors or improper evaluations.

Strassen's Algorithm Implementation Via Python

Current State-of-the-Art Algorithms

- Coppersmith-Winograd Algorithm (1981/1990)
  - Reduced the asymptotic complexity of Strassen's algorithm from $O(N^{2.81})$ to $O(N^{2.496})$ in 1981. Further reduced the asymptotic complexity of Strassen's algorithm to $O(N^{2.3755})$ in 1990.
- Vassilevska Williams' Algorithm (2012)
  - Reduced the asymptotic complexity of Strassen's algorithm from $O(N^{2.81})$ to $O(N^{2.3729})$
- Le Galls Improvement (2014)
  - Reduced the asymptotic complexity of Strassen's algorithm from $O(N^{2.81})$ to $O(N^{2.3728})$
- Williams, Xu, Xu , and Zhou (2024)
  - Reduced the asymptotic complexity of Strassen's algorithm from $O(N^{2.81})$ to $O(N^{2.3715})$

Open Problems

- Can a matrix multiplication algorithm be found that has an asymptotic complexity of $O(N^2)$?
  - Is it possible to reach the theoretical lower bound of matrix multiplication complexity?

- Can Strassen's algorithm be repurposed to compute the product of smaller matrices more efficiently?
  - Could we remove inefficiencies and overhead when applying Strassen's algorithm to smaller matrices?

- Does Strassen's algorithm provide merits for certain matrix types, such as sparse or symmetric matrices?
  - Real-world matrices often have special properties. Could Strassen's algorithm offer performance gains for certain matrix types?

References

- Visit my GitHub (https://github.com/joeykap123) for all references used for this project. All references are in PDF format.