Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 September 2022 doi:10.20944/preprints202209.0150.v1

Article

A short review on fast ways to multiply two matrices

Ishan Banerjee [1,†,‡]

[1] Affiliation 1; ishanbanerjee.314@gmail.com
* Correspondence: ishanb@cmi.ac.in;
† Current address: AC Nath Street, Kantadhar, Ichapore, Kolkata-743144, West Bengal, India.

Abstract: Matrix multiplication is probably one of the most basic and important operations having a wide range of applications in studying Linear algebra. We will be primarily focused on reviewing different algorithms for the multiplication of two $n \times n$ matrices. The most Naive method has a complexity of $O(n^3)$ but interestingly, algorithms with better complexity have been achieved. We will get to see how the complexity can be improved further, occasionally giving a survey.

Keywords: Matrix multiplication; Complexity; Matrix multiplication constant ; Algorithm ; Tensors; Tensor product ; Tensor power

1. Introduction

Matrix multiplication forms one of the most basic operations which we need to frequently implement. However, the naive method is not optimal and hence will take a lot of time to deal with large data. Strassen[4] had made a remarkable discovery on the complexity of matrix multiplication, achieving an upper bound of $\log_2 7 \approx 2.807$ for the exponent. Following that marvelous discovery, progress began on improving the upper-bound. The language of tensors played an important role in this. Interestingly it was conjectured that for some $\epsilon > 0$ we can run an algorithm for matrix multiplication in $O(n^{2+\epsilon})$ [1], therefore the at-most we can do is to get closer to 2. As of December of 2020, the upper-bound for the exponent is 2.3728596. In this paper we will be focusing on various algorithms to multiply two matrices and how the upper-bound of the complexity can be improved.
Coming to the naive method, it is probably the most basic where for a matrix $A = [a_{ij}]$ and $B = [b_{ij}]$, the product gives us $AB = [c_{ij}] = \left[\sum_k a_{ik} b_{ki}\right]$, using the idea of divide and conquer. For square matrices of size $n \times n$, n steps are taken to give a single element of the product matrix. For a total of n^2 number of elements, we have the complexity of n^3. The following represents matrix multiplication in the naive method for a 2×2 matrix.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

We note that the total number of multiplications in this case is 2 for each element, which makes a total of 8 multiplications. Now for a general matrix, we can divide the matrix into sub-matrices of size $\frac{n}{2} \times \frac{n}{2}$ in the following way.

$$\left(\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right) \cdot \left(\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right) = \left(\begin{array}{c|c} ae+bg & af+bh \\ \hline ce+dg & cf+dh \end{array}\right)$$

Complexity: $T(n) = 8T(n/2) + O(n^2)$(Addition requires n^2 complexity)
Which gives a complexity of $O(n^3)$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 September 2022 doi:10.20944/preprints202209.0150.v1

2. Strassen's method

Strassen's method is pretty similar to the divide and conquer method but it reduces the number of multiplication using the following method.
We first need to divide the matrix into sub-matrices just like the naive method.

$$\left(\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right) \cdot \left(\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right)$$

We will then define the following

$$p_1 = a(f-h) \quad p_2 = (a+b)h$$
$$p_3 = (c+d)e \quad p_4 = d(g-e)$$
$$p_5 = (a+d)(e+h) \quad p_6 = (b-d)(g+h)$$
$$p_7 = (a-c)(e+f)$$

We can then get the final product from the following

$$\left(\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right) \cdot \left(\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right) = \left(\begin{array}{c|c} p_5+p_4-p_2+p_6 & p_1+p_2 \\ \hline p_3+p_4 & p_1+p_5-p_3-p_7 \end{array}\right)$$

We note that, unlike the naive method, we need to do only 7 multiplications with the sub-matrices.

Complexity: $T(n) = 7T(n/2) + O(n^2)$

Therefore the complexity becomes $O(n^{log7}) \approx O(n^{2.807})$.

3. Discussion

The following table shows the execution time for the naive and Strassen's method of multiplication for different dimensions of matrix.

Size($n \times n$)	Strassen's method Time(ms)	Naive method Time(ms)
32×32	135	11
64×64	847	66
128×128	7304	479
256×256	31626	3146
512×512	258892	17109

Table 1. Execution time of Naive and Strassen's algorithm. (The codes were written in Python 3)

We notice that for the following data, Naive algorithm beats Strassen's algorithm. However,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 September 2022 doi:10.20944/preprints202209.0150.v1

for really large matrices, we get to see a different picture. When the size of matrix gets around $2^{36} \times 2^{36}$ and above, Strassen's method starts leading significantly.

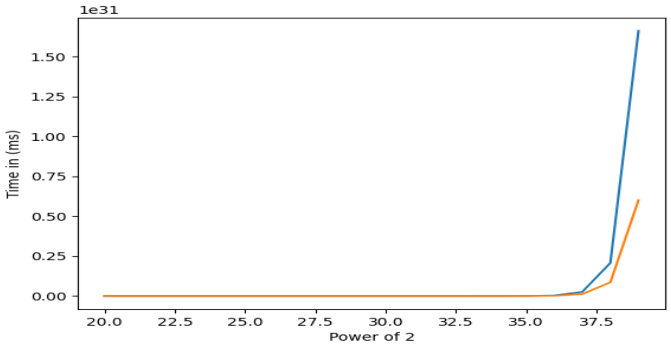

Figure 1. Blue line represents the Naive method and the orange denotes Strassen's method.

However, Strassen's method has a few disadvantages which makes it unproductive in real life applications.

1. Recursive stacks consume more memory.
2. The recursive calls add latency.
3. We face precision error with Strassen's method.

Thus for smaller matrices, its more practical to use the Naive method.

4. Algorithm for approximate matrix multiplication

For a given $m \times n$ matrix A and $n \times p$ matrix B, we will can use algorithms to give an approximation P for $A \cdot B$ where bounds on the norm of the error matrix ($\epsilon = P - A \cdot B$) is provable. In case of the naive algorithm, we will have the complexity as $O(mnp)$. Using the algorithms of AMM, we can reduce it to $O(mnt)$ where $t < p$. The entire idea here is to choose t columns and take only t columns from A, forming the $m \times t$ S and taking t rows from B, forming the $t \times n$ matrix R. The approximate matrix we get is $P = S \cdot R$. the strategy is to choose the t columns and rows effectively.
Theorem: For a fixed $t > 0$, there is a randomized algorithm which approximates the matrix $A \cdot B$ by P such that it runs in $O(tmp)$ and

$$E(||P - A \cdot B||_F^2) \leq \frac{1}{t}\left(\sum_{k=1}^{n} ||A_{(k)}B^{(k)}||\right)^2 \leq \frac{1}{t}||A||_F^2||B||_F^2$$

Where $||\cdot||$ is the Frobenius norm. The problem with this method is that we may get a huge variance which can render this method ineffective in real-life applications. The author in [9] states two algorithms which can run in $O(mn + np + mp)$ with a suitable choice of probability distribution.

5. Matrix multipication tensors

Before moving further we need to know the definition of the matrix multiplication constant ω. This constant is the least number such that for any $\epsilon > 0$, the complexity of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 September 2022 doi:10.20944/preprints202209.0150.v1

multiplying two matrices is $O(n^{\omega+\epsilon})$. It has been conjectured that $\omega = 2$.
Strassen had used the language of tensors to represent his algorithm.

$$\sum_{i,j,k=1}^{2} x_{ij}y_{jk}z_{ki} = (x_{11}+x_{22})(y_{11}+y_{22})(z_{11}+z_{22}) + (x_{21}+x_{22})y_{11}(z_{21}-z_{22}) +$$
$$x_{11}(y_{12}-y_{22})(z_{12}+z_{22}) + x_{22}(y_{21}-y_{11})(z_{11}+z_{21}) +$$
$$(x_{11}+x_{12})y_{22}(-z_{11}+z_{12}) + (x_{21}-x_{11})(y_{11}+y_{12})z_{22} +$$
$$(x_{12}-x_{22})(y_{12}+y_{22})z_{11}$$

We call this (can also be represented as $\langle 2,2,2\rangle$) the matrix multiplication tensor. In this case, the above expression represents the product of two 2×2 matrices. It is indeed possible to show that the rank of this tensor is at-most 7 (Strassen showed that it is exactly 7). But why do we care about the rank of the tensor? It is worth noting that the expression $R(\langle n,n,n\rangle) = n^{\alpha}$ says that a basis algorithm exists such that multiplication of $n \times n$ matrices can be done in n^{α} multiplications which can be further iterated to $n^k \times n^k$. Therefore the matrix multiplication constant must satisfy $\omega \leq \alpha$ which means that we can write $n^{\omega} \leq R(\langle n,n,n\rangle)$. Let's now have a look on some of the important results.

Theorem 1 (Lickteig): $\underline{R}(\langle n,n,n\rangle) \geq \frac{3n^2}{2} + \frac{n}{2} - 1$

Theorem 2 (Bläser): $\underline{R}(\langle n,n,n\rangle) \geq \frac{5}{2}n^2 - 3n$

Theorem 3 (Brockett-Dobkin): $R(\langle n,n,n\rangle) \geq 2n^2 - 1$

Corollary (Winograd): $R(\langle 2,2,2\rangle) = 7$

Theorem 4 : $\underline{R}(\langle 2,2,2\rangle) = 7$

Here $\underline{R}$ represents the boundary rank of the tensor.

5.1. Schönhage's asymptotic sum inequality

The asymptotic sum inequality is a vast generalization of this idea which says

$$\sum_{i=1}^{L}(n_i m_i p_i)^{\frac{\omega}{3}} \leq \underline{R}\left(\bigoplus_{i=1}^{L}\langle n_i, m_i, p_i\rangle\right)$$

Schönhage showed that $\underline{R}(\langle 4,1,4\rangle \oplus \langle 1,9,1\rangle) \leq 17$. On applying the inequality, we get $16^{\omega/3} + 9^{\omega/3} \leq 17$, which finally gives us the bound $\omega < 2.55$.

5.2. Laser Method

For an integer parameter q, Coppersmith and Winograd considered the following tensor

$$T = \sum_{i=1}^{q}(x_0 y_i z_i + x_i y_0 z_i + x_i y_i z_0) + x_0 y_0 z_{q+1} + x_0 y_{q+1} z_0 + x_{q+1} y_0 z_0$$

The above expression can also be written in the form of partitioned tensors as the following.

$$\langle 1,1,q\rangle^{[0,1,1]} + \langle q,1,1\rangle^{[1,0,1]} + \langle 1,q,1\rangle^{[1,1,0]} + \langle 1,1,1\rangle^{[0,0,2]}\langle 1,1,1\rangle^{[0,2,0]} + \langle 1,1,1\rangle^{[2,0,0]}$$

We will partition the x-variable into three parts, $X_0 = \{x_0\}$,$X_1 = \{x_1, x_2, \ldots x_q\}$ and $X_2 = \{x_{q+1}\}$. We will do the same thing with the y and z variables. It is worth noting that every term of the second expression is dependent on a single group of x, y and z variables (Which is represented by the superscripts). It was shown that $\underline{R}(T) \leq q + 2$.
The main idea behind the laser method is to take a high tensor power of T and vanish some groups of variables such that we get a sum of disjoint tensors. We can then apply the asymptotic sum inequality.
When we take high tensor product $T^{\oplus N}$, we get partitioned tensor with 6^N constituent

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 September 2022 doi:10.20944/preprints202209.0150.v1

tensors over $X^N \times Y^N \times Z^N$. We note that the x variables represented by X^N are partitioned into 3^N parts indexed by $\{0,1,2\}$ (Let's call them the x indices). Similarly we will have y and z indices. We also note that each constituent tensor has an index of x, y and z variables (Let's call the triple I, J, K). We will represent those tensors as $T^{\oplus N}_{I,J,K}$.
In the next step let's vanish all the x variables except those whose index are in a set $A \subseteq \{0,1,2\}$. We will do the same for the y and z variables. This will give us the final expression as

$$\sum_{(I,J,K)\in supp(T^{\oplus N})\cap(A\times B\times C)} T^{\oplus N}_{I,J,K}$$

Here $supp(T^{\oplus N})$ represents the support of $T^{\oplus N}$. Suppose all the summands are over disjoint variables. In this case $\underline{R}(T^{\oplus N}) \leq (q+2)^N$. We can now apply the asymptotic sum inequality which will give us

$$\sum_{(I,J,K)\in supp(T^{\oplus N})\cap(A\times B\times C)} Vol(T^{\oplus N}_{I,J,K})^{\omega/3} \leq (q+2)^N$$

Volume of a tensor $\langle m,n,p\rangle$ represented as $Vol(\langle m,n,p\rangle) = mnp$. Coppersmith and Winograd considered the quantity $V^{pr}_{\rho,N}(T)$ which is the maximum of

$$\sum_{(I,J,K)\in supp(T^{\oplus N})\cap(A\times B\times C)} Vol(T^{\oplus N}_{I,J,K})^{\rho/3}$$

Choosing $q = 6$, Coppersmith and Winograd computed the value of ρ such that $V^{pr}_{\rho}(T) = q+2$ and deduced that $\omega < \rho$ which finally gives the upper-bound $\omega < 2.3872$. Different generalizations of the Laser method gave even better bounds, like the recursive laser method gives us an upper-bound of 2.3755.

6. Conclusion

Improvements of the upper-bounds on ω is still continuing, getting even closer to the mythical goal of 2. Starting with the Naive method we saw that it computed matrix multiplication in $O(n^3)$. Strassen's method on the other hand, has a better complexity but its practical applications are limited as it's helpful only for really large matrices. Once we get into the language of tensors, we got to see how the upper-bounds can be made better using the asymptotic sum inequality and considering the higher powers of a tensor and vanishing a few variables and interestingly, different generalizations of the Laser method are capable of giving us even better bounds.

References

1. Alman, J. and Williams, V.V., 2021, A refined laser method and faster matrix multiplication. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA) (pp. 522-539). Society for Industrial and Applied Mathematics.
2. Bläser, M., 2013, Fast matrix multiplication. Theory of Computing, pp.1-60.
3. Huss-Lederman, S., Jacobson, E.M., Tsao, A., Turnbull, T. and Johnson, J.R., 1996, Implementation of Strassen's algorithm for matrix multiplication. In Proceedings of the 1996 ACM/IEEE Conference on Supercomputing (pp. 32-es).
4. Cohn, H., Kleinberg, R., Szegedy, B. and Umans, C., 2005, October. Group-theoretic algorithms for matrix multiplication. In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05) (pp. 379-388). IEEE.
5. Landsberg, J.M., 2012. Tensors: geometry and applications. Representation theory, 381(402), p.3.
6. Author 1, A.B.; Author 2, C.D.; Author 3, E.F. Title of presentation. In Proceedings of the Name of the Conference, Location of Conference, Country, Date of Conference (Day Month Year); Abstract Number (optional), Pagination (optional).
7. Francis, D.P. and Raimond, K., 2018. A practical streaming approximate matrix multiplication algorithm. Journal of King Saud University-Computer and Information Sciences.
8. Le Gall, F., 2014, July. Powers of tensors and fast matrix multiplication. In Proceedings of the 39th international symposium on symbolic and algebraic computation (pp. 296-303).
9. Drineas, P., Kannan, R. and Mahoney, M.W., 2006. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. SIAM Journal on Computing, 36(1), pp.132-157.