Fast Matrix Multiplication

Jinxin Xia

WAKE FOREST
UNIVERSITY

Sep 24, 2019

Strassen's Algorithm

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

$$\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}$$

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

Strassen's Algorithm

Strassen's algorithm

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

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

Strassen's factor matrices:

$$
\mathbf{U} = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & -1
\end{pmatrix}
$$

$$
\mathbf{V} = \begin{pmatrix}
1 & 1 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & -1 & 0 & 1 & 0 & 1
\end{pmatrix}
$$

$$
\mathbf{W} = \begin{pmatrix}
1 & 0 & 0 & 1 & -1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & -1 & 1 & 0 & 0 & 1 & 0
\end{pmatrix}
$$

Strassen's Algorithm

Strassen's algorithm

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

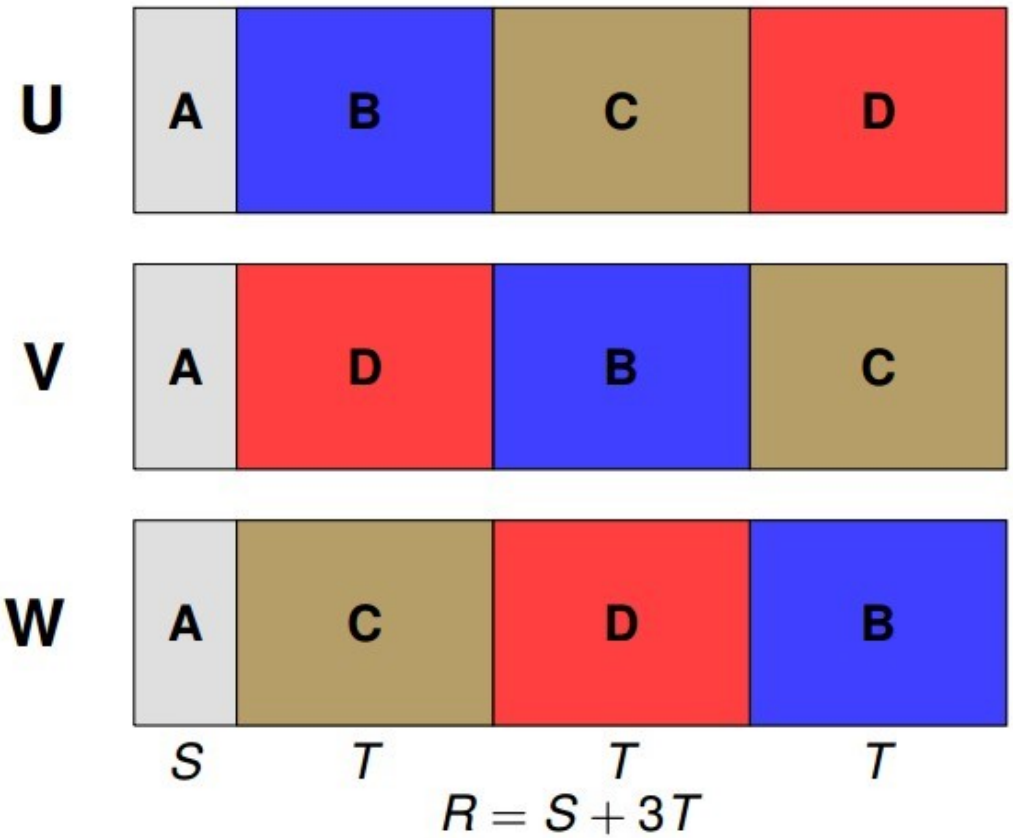
$$
\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}
$$

Strassen's factor matrices:

		M_1	M_2	M_3	M_4	M_5	M_6	M_7
U	A_{11}	1		1		1	−1	
	A_{12}					1		1
	A_{21}		1				1	
	A_{22}	1	1		1			−1
V	B_{11}	1	1		−1		1	
	B_{12}			1			1	
	B_{21}				1			1
	B_{22}	1		−1		1		1
W	C_{11}	1			1	−1		1
	C_{21}		1		1			
	C_{12}			1		1		
	C_{22}	1	−1	1			1	

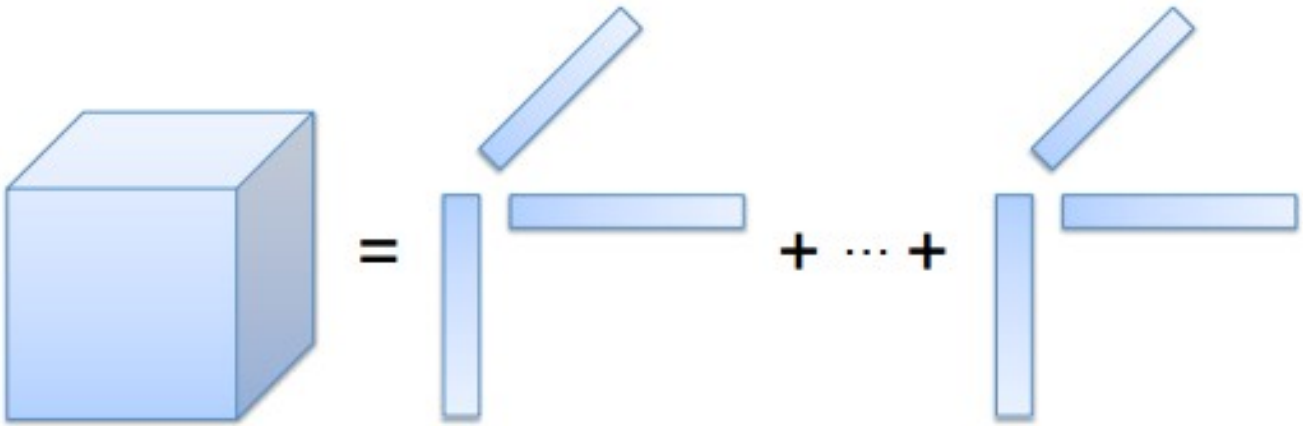
Strassen's Algorithm

- U, V, W also have the following properties with the same row dimension I.



Strassen's Algorithm

Problem: Find $[\![\mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$ such that $\mathcal{T} = \sum \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$



$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$$

The problem

$$\underset{U,V,W}{\text{minimize}} \quad \|\tau - \langle Uo \quad Vo \quad W\rangle\|$$

Let $\tau - \langle UoVoW\rangle = \gamma$. Then after vectorize the original optimization problem, the minimization problem becomes a nonlinear least squares problem.

$$\underset{X}{\text{minimize}} \quad \|\gamma\|_2^2$$

where X is the combination of vectorized U, v, and W.

The problem

- We use *ijk* to represent the row number of the vectorized γ elements, and the following is the equation for getting γ.

$$\gamma_{ijk} = \tau_{ijk} - \sum_{s=1}^{S} a_{is} a_{js} a_{ks} - \sum_{t=1}^{T} (b_{it} c_{jt} d_{kt} + d_{it} b_{jt} c_{kt} + c_{it} d_{jt} b_{kt})$$

Gaussian Newton Method

- Gaussian Newton Method fits the new nonlinear least squares problem very well, and the formula for finding next X is

$$X^{(k+1)} = (J^{(k)T} J^{(k)})^{-1} J^{(k)T} (J^{(k)} X^k - \gamma(X^{(k)}))$$

Methodology

Next Steps

1. Find jacobian matrix J
2. Apply Gaussian Newton
3. Evaluate the results

The jacobian matrix of the γ involves the vectorized matrices and tensor, thus to get there we need to get the jacobian matrix for vectorized matrices A, B, C, and D.

Jacobian Matrices

$$J_A = (A \odot A) \otimes I_I + \Pi^{(2)}((A \odot A) \otimes I_I) + \Pi^{(3)}((A \odot A) \otimes I_I)$$

$$J_B = (C \odot D) \otimes I_I + \Pi^{(2)}((D \odot C) \otimes I_I) + \Pi^{(3)}((C \odot D) \otimes I_I)$$

$$J_C = (B \odot D) \otimes I_I + \Pi^{(2)}((D \odot B) \otimes I_I) + \Pi^{(3)}((B \odot D) \otimes I_I)$$

$$J_D = (B \odot C) \otimes I_I + \Pi^{(2)}((C \odot B) \otimes I_I) + \Pi^{(3)}((B \odot C) \otimes I_I)$$

where $\Pi^{(2)}$ and $\Pi^{(3)}$ are permutation matrices.

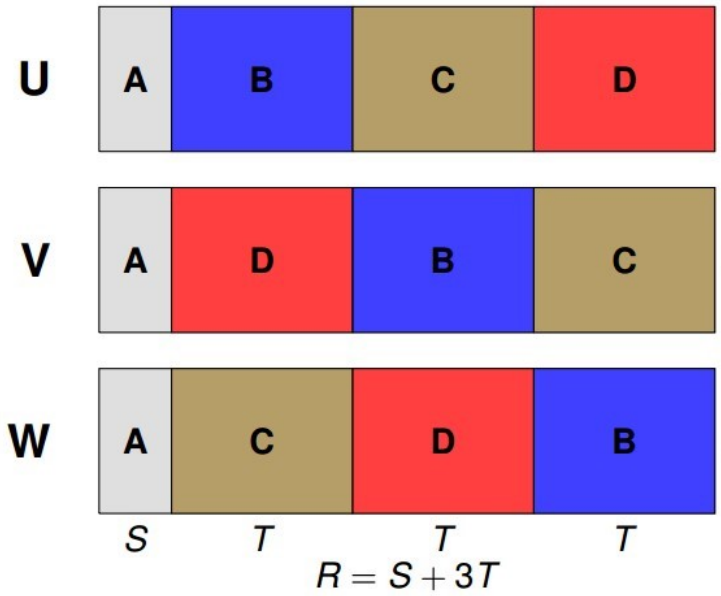
$$\Pi^{(2)} : ijk \to jik$$

$$\Pi^{(3)} : ijk \to kij$$

Then the jacobian matrix of γ is $J = [J_A J_B J_C J_D]$.

Updated algorithm

- Because of the special structures that matrices U, V, and W have. It's easy to get matrix A correct but hard to get B, C, and D correct even if the residual pass the tolerance.



$$\gamma_{ijk} = \tau_{ijk} - \sum_{s=1}^{S} a_{is} a_{js} a_{ks} - \sum_{t=1}^{T} (b_{it} c_{jt} d_{kt} + d_{it} b_{jt} c_{kt} + c_{it} d_{jt} b_{kt})$$

Updated algorithm

- Therefore, we put penalty part in the nonlinear least squares problem by adding

$$\lambda \|p - p^*\|_2^2$$

where $p = X^{(k+1)} - X^{(k)}$ and $p^* = X^{(k)} - X^{(k-1)}$.

- We also put regularization on each p^* by changing the elements value to 1 if it's greater than 1 or changing it to -1 if it's smaller than -1.
- Then the new problem becomes

$$\underset{X}{\text{minimize}} \quad \|\gamma\|_2^2 + \lambda \|p - p^*\|_2^2$$

Results

```
>> [num_converg,norm_r]=fastmtm2(2,9,1,2,1e-3,2000,1e-10,1e-7);

  1    5.63547e-12   2.86931e-12      4    >> [A B C D]

ans =

     1     0     0     0     1     1    -1
     0     1     0     0     0     0     1
     0     0     0     1     1     0     0
     1     1     1    -1     0     0     0
```

The first number in the first row is the norm of γ and the second number is the norm of p, and the last number is how many steps does the algorithm need to find the solution. If the initial guess is close to the true answer then the algorithm can find it within 10 steps.

Results

```
>> [A B C D]

ans =

     0     1     0     0     0     0     0     0     0     0     0     0     1    -1     0     0    -1     0     0     0     0     1
     0     0     0     1     0     0     0     0     0     0     0     0     1    -1     1     0     0     0     0     0     0    -1
     0     0     0     0    -1     0     0     0    -1     0     0     0    -1     1    -1     0     0     0     0    -1     1     0     1
     0     0     0     0     0     0     1     0     0     0    -1     0     0     0     0     1     0     0     1     1     0     0     0
     1     0     0     1     0     0     0     0    -1     1     0    -1     0     0     0     0    -1     0     0     0     0     1     0
     0     0     0     0     0     0    -1     1     0     0     0     0     0     0     0    -1     1     0     0    -1     0     0     0
     0     0     0     0     0    -1     1     0     0     0     0     0     0     0    -1     0     0    -1     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     1     0    -1     0     0     1     0     0     0     0     0     0     0     0
     1     0     1     0     0     0    -1     1     0     0     0     0     0     0    -1    -1     0     0     0     0     0     0     0
```

Here is the 3×3 case. Also if the initial guess is close to the true answer we can get the true answer around 500 steps.

Conclusion

- The algorithm works for finding the Strassen's algorithm matrix, and it can also work for 3×3 case. With more modification on the algorithm, it could have better performance in finding the 3 by 3 case.

- The outcomes of the algorithm are not the exact answer that we want if we begin with a random guess. It might because the optimization questions are not well defined.

Fast Matrix Multiplication

Jinxin Xia

WAKE FOREST
UNIVERSITY

Sep 24, 2019