Matrix Computations (DAA, M.Tech + Ph.D.)

By:

Sunil Kumar Singh, PhD
Assistant Professor,
Department of Computer Science and Information Technology

School of Computational Sciences, Information and Communication Technology,
Mahatma Gandhi Central University, Motihari
Bihar, India-845401

Outline

- Matrix Computation
- Strassen's Algorithm
- Divide an conquer paradigm for multiplication
- Inverse of a triangular matrix
- Conclusion
- References

Matrix Computations

Input: $A = [a_{ij}], B = [b_{ij}]$

output: $C = [c_{ij}] = A.B \quad where\ i, j = \{1,2,3, \ldots \ldots n\}$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{22} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \ldots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{22} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} . \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{22} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \ldots & b_{nn} \end{bmatrix}$$

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

Standard Algorithm

```
for i ←1 to n
    do for j←1 to n
        do cij←0
        for k←1 to n
            do cij←cij+ aik·bkj
```

$$T(N) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} c\text{=c } N^3 = O(N^3)$$

Divide-and-Conquer

- Divide-and conquer is a general algorithm design paradigm:
  - ✓ Divide: divide the input data S in two or more disjoint subsets S_1, S_2 …
  - ✓ Solve the subproblems recursively
  - ✓ Conquer: combine the solutions for S_1, S_2 … into a solution for S
- The base case for the recursion are sub problems of constant size
- Analysis can be done using **recurrence equations**

Matrix multiplication through divide and conquer approach

1. **Divide:** Partition A and B into $\left(\frac{n}{2}\right) X \left(\frac{n}{2}\right)$ submatrices. Form terms to be multiplied using + and **–**

2. **Conquer:** Perform 7 multiplications of $\left(\frac{n}{2}\right) X \left(\frac{n}{2}\right)$ submatrices recursively.

3. **Combine:** Form C using + and – on $\left(\frac{n}{2}\right) X \left(\frac{n}{2}\right)$ submatrices

Matrix multiplication through divide and conquer approach

$n \times n\ matrix = 2 \times 2$ matrix of $(n/2) \times (n/2)$ submatrices:

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

$$C = A.B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}\right\} \text{ 8 multiplications and 4 adds of } \left(\frac{\text{n}}{2}\right) \text{X} \left(\frac{\text{n}}{2}\right)$$

Matrix multiplication through divide and conquer approach

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)$$

#submatrices

submatrix size

work adding submatrices

Complexity: $T(n) = \Theta(n^3)$

Strassen's Algorithm

- In linear algebra, the Strassen algorithm, named after Volker Strassen, is an algorithm for matrix multiplication.

- It is faster than standard matrix multiplication algorithm and is useful in practice for large matrices.

- But it would be slower than the fastest known algorithms for extremely large matrices.

Strassen's Idea

- Multiply 2×2 matrices with only 7 recursive multiplications.

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

$$\boldsymbol{P_1 = (A_{11} + A_{22})(B_{11} + B_{22})}$$

$$\boldsymbol{P_2 = (A_{21} + A_{22}) * (B_{11})}$$

$$\boldsymbol{P_3 = (A_{11}) * (B_{12} - B_{22})}$$

$$\boldsymbol{P_4 = (A_{22})(B_{11} + B_{22})}$$

$$\boldsymbol{P_5 = (A_{11} + A_{22}) * (B_{22})}$$

$$\boldsymbol{P_6 = (A_{21} - A_{11}) * (B_{11} + B_{12})}$$

$$\boldsymbol{P_7 = (A_{12} - A_{22}) * (B_{21} + B_{22})}$$

$$\boldsymbol{C_{11} = P_1 + P_4 - P_5 + P_7}$$

$$\boldsymbol{C_{12} = P_3 + P_5}$$

$$\boldsymbol{C_{21} = P_2 + P_4}$$

$$\boldsymbol{C_{11} = P_1 + P_3 - P_2 + P_6}$$

Analysis

$$T(n) = \begin{cases} b & n \leq 2 \\ 7T\left(\frac{n}{2}\right) + \Theta(n^2) & n > 2 \end{cases}$$

- When we apply the masters theorem:

$$T(n) = O(n^{\log_2 7})$$

$$T(n) = O(n^{2.81})$$

- The number 2.81 may not seem much smaller than 3, but because the difference is in the exponent, the impact on running time is significant. In fact, Strassen's algorithm beats the ordinary algorithm on today's machines for $n \geq 32$ or so.

Inverse of a Matrix

- Inverse of an $n \times n$ matrix A to be the $n \times n$ matrix, denoted by A^{-1}(if it exists), such that

$$AA^{-1} = I_n = A^{-1}A$$

- for example

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

- Many nonzero $n \times n$ matrices do not have inverse. A matrix without an inverse is called noninvertible, or singular.
- An example of a non-zero singular matrix is

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

- If a matrix has an inverse, it is called invertible, or nonsingular.

Inverse of a Matrix

- If A and B are nonsingular $n \times n$ matrices, then

$$(BA)^{-1} = A^{-1}B^{-1}$$

- Inverse operation commutes with the transpose operation:

$$(A^{-1})^T = (A^T)^{-1}$$

- Prove that an $n \times n$ matrix is singular if and only if det(A)=0.

References

1. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
2. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to algorithms second edition." *The Knuth-Morris-Pratt Algorithm, year* (2001).
3. Seaver, Nick. "Knowing algorithms." (2014): 1441587647177.

Thank You