Lecture (20): 3/17/22

Lecturer: Xiaorui Sun *Scribe: Robert Finedore*

1 Last time and today

- Matrix multiplication
  - Naive Algorithm
  - Strassen Algorithm
  - Tensor

2 Matrix Multiplication Overview

2.1 Problem Definition and Naive Solution

Input: Matrix A and B where the dimensions of A is n*m and the dimensions of B is m*p. The output is A * B where the resulting matrix is of dimensions n*p and where the ith row and the jth column is equal to

$$\sum_{k=1}^{m} a_{ik} * b_{jk}$$

Visually matrix multiplication looks like the following:

ith row from Matrix A:

$$\begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

jth column from Matrix B:

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Output Matrix C where the ith and jth element is equal to

$$\begin{bmatrix} a_1 * b_1 + a_2 * b_2 \end{bmatrix}$$

Let n=m=p then the trivial runtime is

$$O(n^3)$$

Each entry needs to be evaluated when taking the product of two matrices thus the lower bound is

$$\Omega(n^2)$$

Let ω denote the best exponent for the running time for matrix multiplication

$$2 \leq \omega \leq 3$$

Current best runtime is $\omega \approx 2.37$. This runntime requires the use of tensors.

3 Strassen Algorithm

3.0.1 Overview

- Runtime is $O(n^{\log_2(7)})$ which is $\approx O(n^{2.81})$
- Observe matrix A+B where the ith row and the jth column is equal to

$$a_{ij} + b_{ij} = O(n^2)$$

- Since addition and subtraction is less expensive we would like to use more of those operations instead of multiplication.
- General idea of Strassen Algorithm is to divide the matrix into smaller matrices. Then do the addition/multiplication of the smaller matrices.

3.0.2 Example

Below is an example doing matrix multiplication with matrices A and B the naive way.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$C = A * B = \begin{bmatrix} a_{11} * b_{11} + a_{12} * b_{21} & a_{11} * b_{12} + a_{12} * b_{22} \\ a_{21} * b_{11} + a_{22} * b_{21} & a_{21} * b_{12} + a_{22} * b_{22} \end{bmatrix}$$

- Notice that there are 8 multiplications and 4 additions.
- With Strassen Algorithm we can now create partitions in the following manner:
  - $$p_1 = (a_{11} + a_{22}) * (b_{11} + b_{22})$$
  - $$p_2 = (a_{21} + a_{22}) * b_{11}$$
  - $$p_3 = a_{11} * (b_{12} - b_{22})$$
  - $$p_4 = a_{22} * (b_{21} - b_{11})$$
  - $$p_5 = b_{22} * (a_{11} + a_{12})$$
  - $$p_6 = (a_{21} - a_{11}) * (b_{11} + b_{12})$$
  - $$p_7 = (a_{12} - a_{22}) * (b_{21} + b_{22})$$
- With these partitions we can now find our output matrix , C , in the following manner:
  - $$C_{11} = p_1 + p_4 - p_5 + p_7$$

- $$C_{12} = p_3 + p_5$$
- $$C_{21} = p_2 + p_4$$
- $$C_{22} = p_1 - p_2 + p_3 + p_6$$
- Using Strassen Algorithm there are only 7 multiplications and 18 additions.
- Overall reduction in multiplication operation.

• More generally, let matrix A and B be size n where

$$n = 2^k$$

. We can recursively partition the larger matrices in the following way:

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

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

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

• With each partition calling Strassen(A,B) to generate smaller partitions.

• We can now see how to use Strassen Algorithm in a more general sense.

3.0.3 Matrix Multiplication Recursion

Let T(n) be the running time for matrix multiplication for a n*n matrices.
Recursion for Strassen's Matrix Multiplication Algorithm would be as followed:

$$T(n) = \begin{cases} Constant, & \text{if } n = 1 \\ 7 * T(n/2) + 18 * O(n^2), & \text{n} > 1 \end{cases} \tag{1}$$

The constant factor of 18 comes from the 18 additions. Thus we have

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

Recursion for matrix multiplication the naive way would be as followed:

$$T(n) = \begin{cases} Constant, & \text{if } n = 1 \\ 8 * T(n/2) + O(n^2), & \text{n} > 1 \end{cases} \tag{2}$$

Thus we have

$$T(n) = O(n^3)$$

4 Definitions

4.0.1 Quadratic Problem

Let variables $x_1, x_2 ... x_n$ exist in $\mathbb{R}$. Let $F = \{f_1, f_2 ... f_k\}$ be a set of quadratic functions. Where

$$f_k = \sum_{i,j=1}^{n} t_{i,j,k} * x_i * x_j$$

and where $t_{i,j,k}$ is a fixed coefficient.

The primary goal of this problem is to compute F. We can think of F as the output of A*B in matrix multiplication.

4.0.2 Bilinear Problem

Let define variables $x_1, x_2 ... x_n$ and $y_1, y_2 ... y_n$ In this case x variables represent matrix A and y variables represent matrix B. Then we can define

$$F = \{f_1, f_2 ... f_k\}$$

where once again F represents A*B and each

$$f_k = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{i,j,k} * x_i * y_j$$

If we look at

$$\{t_{i,j,k}\} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{k}$$

We can notice it is 3 dimensional making it a 3-tensor. Knowing this tensor values allows us to compute matrix multiplication faster.

5 Matrix Multiplication as Bilinear Problem

Let $A = \begin{bmatrix} x_{11} & x_{12} & ... & x_{1n} \\ x_{21} & x_{22} & ... & x_{2n} \\ ... & ... & ... & x_{nn} \end{bmatrix}$ And let $B = \begin{bmatrix} y_{11} & y_{12} & ... & y_{1n} \\ y_{21} & y_{22} & ... & y_{2n} \\ ... & ... & ... & y_{nn} \end{bmatrix}$

Ultimately with these two matrices we have $2n^2$ variables.

Our output: A*B is defined as $\begin{bmatrix} z_{11} & z_{12} & ... & z_{1n} \\ z_{21} & z_{22} & ... & z_{2n} \\ ... & ... & ... & z_{nn} \end{bmatrix}$

Let $f(i,j) = z_{i,j}$

$$f(i,k) = \sum_{(i_0,j_0),(j_1,k_1)} t_{(i_0,j_0),(j_1,k_1),(i,k)} * x_{(i_0,j_0)} * y_{(j_1,k_1)}$$

We can separate out the tensor constant and have the following:

$$f(i,k) = \sum_{j=1}^{n} x_{i,j} * y_{j,k}$$

$$T_{(i_0,j_0),(j_1,k_1),(i,k)} = \begin{cases} 1, & \text{if } i_0 = i, j_0 = j_1, k = k_1 \\ 0, & \text{Otherwise} \end{cases} \tag{3}$$