COMS 6998: Algebraic Techniques in TCS (Fall'21) Nov 16, 2021

Lecture 9: Algorithms for Matrix Multiplication Part I

Instructor: *Josh Alman* Scribe notes by: *Shyamal Patel*

Disclaimer: This draft may be incomplete or have errors. Consult the course webpage for the most up-to-date version.

1 Tensors and Matrix Multiplication

1.1 Definitions

We begin by defining a tensor. To do so, we start by recalling that we can view a matrix $M \in \mathbb{F}^{a \times b}$ as a

- 2D array with entries from $\mathbb{F}$,
- linear map $M : \mathbb{F}^a \to \mathbb{F}^b$,
- bilinear map $M : \mathbb{F}^a \times \mathbb{F}^b \to \mathbb{F}$, i.e., $u^T M v$ for $u \in \mathbb{F}^a$ and $v \in \mathbb{F}^b$,
- a bilinear polynomial $\sum_{i=1}^{a} \sum_{j=1}^{b} M[i,j] \cdot x_i y_j$ – plugging in a set of values for x and y corresponds to evaluating the bilinear map above.

Similarly we can define a tensor. We will primarily be concerned with 3D tensors. Namely, we can think of a tensor $T \in \mathbb{F}^{a \times b \times c}$ as a

- 3D array with entries from $\mathbb{F}$,
- linear map $M : \mathbb{F}^a \to \mathbb{F}^{b \times c}$,
- bilinear map $M : \mathbb{F}^a \times \mathbb{F}^b \to \mathbb{F}^c$,
- trilinear map $M : \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^c \to \mathbb{F}$,
- a trilinear polynomial $\sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{c} T[i,j,k] \cdot x_i y_j z_k$.

We'll primarily think of tensors as trilinear polynomials.

It is now natural to wonder why we are thinking about tensors for matrix multiplication. Indeed, we can think of matrix multiplication as bilinear map $\mathbb{F}^{n^2} \times \mathbb{F}^{n^2} \to \mathbb{F}^{n^2}$. As such we can define a corresponding tensor for matrix multiplication (MM):

Definition 1 (MM Tensor)**.** *The $a \times b \times c$ MM tensor denoted $\langle a, b, c \rangle$ is given by*

$$\sum_{i=1}^{a} \sum_{k=1}^{b} \sum_{j=1}^{c} x_{i,k} \cdot y_{k,j} \cdot z_{i,j}.$$

It corresponds to a bilinear map $\mathbb{F}^{ac} \times \mathbb{F}^{bc} \to \mathbb{F}^{ab}$.

Note that if we plug in x and y from the matrix entries then the coefficient of $z_{i,j}$ corresponds to the (i,j)th entry in the product.

1.2 Tensor Rank and Matrix Multiplication

We now turn to describing how the complexity of the MM tensor relates to the time it takes to multiply two matrices. To do so, we will first need a definition.

Definition 2 (Tensor Rank)**.** *Let T be a tensor over finite sets of variables X, Y, Z. A tensor has rank 1 if for coefficients $\alpha_x, \beta_y, \gamma_z \in \mathbb{F}$,*

$$T = \left(\sum_{x \in X} \alpha_x x\right) \cdot \left(\sum_{y \in Y} \beta_y y\right) \cdot \left(\sum_{z \in Z} \gamma_z z\right).$$

We then say the rank of a tensor T, denoted as $R(T)$, is the minimum number of rank one tensors that sum to T.

Note that this is a natural extension of the rank of a matrix. It turns out that the rank of the MM tensor is closely related to the time it takes to compute matrix multiplication.

Theorem 3. *If $R(\langle q, q, q\rangle) \leq r$, then we can multiply $n \times n$ matrices in $O(n^{\log_q(r)})$ operations.*

Proof. The proof follows similar to Strassen's algorithm. We will think of r, q as constants. By definition of tensor rank, we have that

$$\sum_{i=1}^{q}\sum_{j=1}^{q}\sum_{k=1}^{q} x_{i,k}y_{k,j}z_{i,j} = \sum_{\ell=1}^{r}\left(\sum_{i,k=1}^{q} \alpha_{i,k,\ell}x_{i,k}\right)\left(\sum_{k,j=1}^{q} \beta_{k,j,\ell}y_{k,j}\right)\left(\sum_{i,j=1}^{q} \gamma_{i,j,\ell}z_{i,j}\right). \tag{1}$$

Now given matrices $X, Y \in \mathbb{R}^{n\times n}$ we block them into $q \times q$ blocks of matrices, $X_{i,k}$ and $Y_{k,j}$, of size $\frac{n}{q} \times \frac{n}{q}$. We can then plug in these matrices into the right hand side of equation (1) and compute the coefficient of each $Z_{i,j}$ to compute the product $X \times Y$. To evaluate the products of the $\frac{n}{q} \times \frac{n}{q}$ matrices that arise as intermediates we recursively use this procedure.

We now use make a recurrence relation to compute the runtime of this procedure. Note that to compute the value of a given rank one matrix it takes at most $3q^2$ total additions of $n/q \times n/q$ matrices and one multiplication. This gives us that the total number of operations to compute the sum of all r rank one tensors in the decompositions is

$$T(n) = r \cdot T(n/q) + O(n^2).$$

The master theorem then tells us that $T(n) = O(n^{\log_q(r)})$. □

With this, we now define the exponent of matrix multiplication.

Definition 4 (Exponent of MM)**.** *The exponent of matrix multiplication is defined by*

$$\omega := \liminf_{q \in \mathbb{N}} \frac{\log(R(\langle q, q, q\rangle))}{\log(q)}.$$

Using our theorem, it follows that we can multiply matrices in $n^{\omega+\epsilon}$ time for any $\epsilon > 0$. That said, this isn't a morally justified definition as there could be other algorithms that do not use tensor rank. However, we will now turn to show that many algorithms can be formulated as bounding the tensor rank.

Proposition 1. *Suppose there is an arithmetic circuit with T gates $(+, -, \times, \div)$ for multiplying $q \times q$ matrices, then $R(\langle q, q, q\rangle) \leq 2T$.*

Of course, there could still be other ways of computing matrix multiplication using say bit wise operations, but all known approaches for matrix multiplication can be phrased in this way.

This proposition means that if there exists an algorithm that computes $q \times q$ by $q \times q$ matrix multiplication in $T = O(q^c)$ operations, then $\omega \leq \frac{\log(R(\langle q,q,q\rangle))}{\log(q)} \leq \frac{\log(2T)}{\log(q)} = \frac{\log(O(q^c))}{\log(q)} \to c$. Thus it is reasonable to define ω using tensor rank.

Sketch of Proof. We start by assuming that there are no division gates in our circuit. It then follows that each gate computes a polynomial over the inputs. Next, we transform the circuit so that each gate computes a homogeneous polynomial of degree at most 2. Essentially, we split each gate into three parts – one for each of the degree 0, 1 and 2 parts. If a gate computes a polynomial with higher degree terms, we simply throw out these terms. (This won't be a problem since at the end the circuit must output a degree 2 polynomial.) This procedure will simply blow up the size of the circuit by a constant factor.

Formally, we can do this inductively layer by layer of the circuit. For addition, suppose we have a gate $p + r$ then we know by the inductive hypothesis that its inputs have been split into p_0, p_1, p_2 and r_0, r_1, q_2 where p_i, r_i are the degree i parts of p and r. We can then transform the gate into three homogeneous ones as shown below.

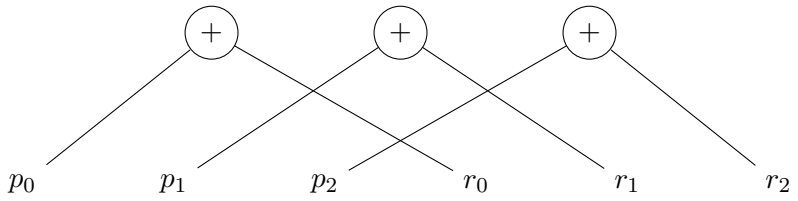

Figure 1: Replacing an addition gate for $p + r$.

We can do something similar for subtraction. For multiplication, we can replace the gate $p \times r$ by

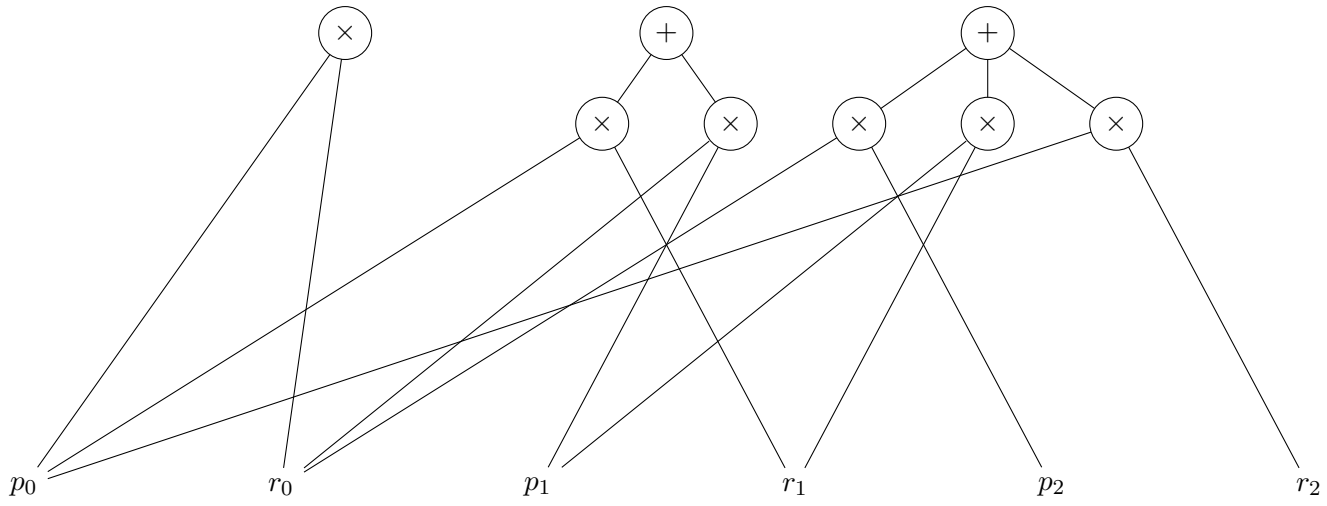

Figure 2: Replacing a multiplication gate for $p \times r$.

We can then prove inductively that at each layer we correctly compute the degree ≤ 2 parts from the original circuit. This implies that after making these changes the outputs will still be correct.

Given this, we'll now assume that each gate computes a homogenous polynomial. We then note that every degree 2 part is a linear combination of degree 1 parts that were computed earlier. In particular, every output is a linear combination of the at most T different products of degree one parts that we computed at multiplication gates. So there are T different products of the form

$$\left(\sum_{x\in X}\alpha_x x + \sum_{y\in Y}\beta_y y\right)\left(\sum_{x\in X}\alpha'_x x + \sum_{y\in Y}\beta'_y y\right).$$

Note that this is almost what we want; we just need to separate the x and y terms. To do this, we split the product into two:

$$\left(\sum_{x\in X}\alpha_x x\right)\left(\sum_{y\in Y}\beta'_y y\right) + \left(\sum_{x\in X}\alpha'_x x\right)\left(\sum_{y\in Y}\beta_y y\right). \tag{2}$$

Note that we can safely get rid of the xx' terms for $x, x' \in X$ and yy' for $y, y' \in Y$ since the outputs don't have terms of this form. So, we then get that all of the outputs are linear combination of the at most $2T$ products arising from equation (2). This is the rough idea of the proof, but note that we are skipping some details.

To handle divisions, we can replace any division operation by multiplication by a formal power series, i.e., $1/a = 1 + (1-a) + (1-a)^2 + \cdots$. We can then make this a finite multiplication since we only care about terms of degree at most 2. All that said, this is very handwavy since it's not clear what the constant term is in $1/a$ for instance. However, trick can be made to work and we refer the interested reader to Section 4.1 of [Blä13]. □

2 Tensor Rank for Matrix Multiplication

We now further motivate using tensor rank for matrix multiplication. So far, everything we've done can still be thought of in terms of identities like in Strassen's algorithm, so it's not clear why we should use tensors instead. As we'll see, tensors have various nice properties. For instance, Strassen's algorithm looks very "asymmetric" in the x's, y's, and z's. On the other hand, the matrix multiplication tensor is very symmetric. For instance, note that we have that

$$R(\langle a, b, c\rangle) = R(\langle b, c, a\rangle).$$

This follows very quickly for tensor rank from just swapping the variables, but it is not clear how to see this from something like Strassen's identity.

2.1 Rectangular MM Rank Bounds Imply Square MM Rank Bounds

To illustrate another nice aspect of tensors for matrix multiplication, we will consider the natural question of whether bounds on the rank of rectangular matrix multiplication $\langle a, b, c\rangle$ can give us bounds on the rank of square MM tensors $\langle q, q, q\rangle$. To do this, we will need a crucial definition.

Definition 5 (Kronecker Product)**.** *Let T be a tensor over X, Y, Z and T' a tensor over X', Y', Z', then $T \otimes T'$ is a tensor over $X \times X'$, $Y \times Y'$, and $Z \times Z'$. For $x \in X, x' \in X', y \in Y, y \in Y', z \in Z, z \in Z'$, we have that*

$$T \otimes T'[(x, x'), (y, y'), (z, z')] = T[x, y, z] \cdot T'[x', y', z']$$

If you are familiar with the tensor product, the Kronecker product is similar. The main difference is that the tensor product of two 3 dimensional tensors is a 6 dimensional tensor. The Kronecker product "smashes" this object back to being a 3D tensor instead.

With this definition in hand, we now turn to prove a few basic facts.

Fact 6. *If $R(T) = R(T') = 1$, then $R(T \otimes T') = 1$.*

Proof. Since T is rank one there exists linear polynomials $\sum_{x \in X} \alpha_x \cdot x$, $\sum_{y \in Y} \beta_y \cdot y$, $\sum_{z \in Z} \gamma_z \cdot z$ over X, Y, and Z respectively such that

$$T[x, y, z] = \alpha_x \cdot \beta_y \cdot \gamma_z.$$

Similarly, we have that

$$T'[x', y', z'] = \alpha'_{x'} \cdot \beta'_{y'} \cdot \gamma'_{z'}.$$

We then observe that

$$T \otimes T'[(x, x'), (y, y'), (z, z')] = \alpha_x \cdot \beta_y \cdot \gamma_z \cdot \alpha'_{x'} \cdot \beta'_{y'} \cdot \gamma'_{z'}.$$

This implies that the tensor is rank one since $\alpha_x \cdot \alpha'_{x'}$ only depends on (x, x'), $\beta_y \cdot \beta_{y'}$ only depends on y, and $\gamma_z \cdot \gamma_{z'}$ only depends on (z, z'). □

Fact 7. $R(T \otimes T') \leq R(T) \cdot R(T')$.

Proof. Note that by definition, there exist rank one tensors T_i and T'_j such that

$$T = \sum_{i=1}^{R(T)} T_i, \qquad T' = \sum_{j=1}^{R(T')} T'_j.$$

If we think of these as polynomials and distribute out the terms we can see that

$$T \otimes T' = \sum_{i=1}^{R(T)} \sum_{j=1}^{R(T')} T_i \otimes T'_j.$$

By the previous fact, the Kronecker product of rank one tensors is rank one. So we have that $R(T \otimes T') \leq R(T) \cdot R(T')$ as desired. □

Note that for matrices the rank of the Kronecker product of two matrices is exactly the product of their ranks. For tensors, however, we only get an inequality.

Fact 8. $\langle a, b, c \rangle \otimes \langle a', b', c' \rangle = \langle aa', bb', cc' \rangle$.

We omit the proof, but it corresponds to the fact that when multiplying two matrices you can partition them into blocks and multiply the blocks as if they were entries. It can also be seen by simply working through the definition.

We can now combine these facts to lift rank bounds for $\langle a, b, c\rangle$ to square matrices. Namely we observe that if we have that $R(\langle a, b, c\rangle) \leq r$ then

$$R(\langle b, c, a\rangle) \leq r, \qquad R(\langle c, a, b\rangle) \leq r.$$

We now note that

$$\begin{aligned} R(\langle abc, abc, abc\rangle) &= R(\langle a, b, c\rangle \otimes \langle b, c, a\rangle \otimes \langle c, b, a\rangle) \\ &\leq R(\langle a, b, c\rangle) \cdot R(\langle b, c, a\rangle) \cdot R(\langle c, b, a\rangle) \\ &\leq r^3, \\ \implies \omega &\leq \frac{\log(r^3)}{\log(abc)}. \end{aligned}$$

2.2 Bounds on Tensor Rank of MM

Now that we've established a rough equivalence between tensor rank and matrix multiplication, we turn to review some of the bounds that have been shown. We remind the reader to refer to the corresponding handout throughout this section.

We start with the classical divide and conquer algorithm of Strassen for matrix multiplication.

Theorem 9 ([Str69])**.** $R(\langle 2, 2, 2\rangle) \leq 7$

We now know that this is tight for $\langle 2, 2, 2\rangle$, so to prove a stronger bound we will need to consider larger MM tensors. That said, we are already unsure of the rank of $R(\langle 2, 3, 3\rangle)$. We know that

$$14 \leq R(\langle 2, 3, 3\rangle) \leq 15.$$

Note that the upper bound gives us that the MM exponent $\omega \leq 2.811$, which is notably worse than Strassen's bound. On the other hand, if we could show that $R(\langle 2, 3, 3\rangle) = 14$ this would gives us that $\omega \leq 2.74$.

Along these lines, after 9 years, Pan was able to prove the first improvement to Strassen's algorithm.

Theorem 10 ([Pan78])**.** $R(\langle 70, 70, 70\rangle) \leq 143640$.

This implies that $\omega \leq 2.80$. In general, it is quite hard to bound the rank of tensors. As such, people showed that there were weaker notions of rank that were sufficient to bound ω. One such notion is border rank by Bini.

Theorem 11 ([BCRL79])**.** $\underline{R}(\langle 3, 2, 2\rangle) \leq 10$.

To prove this, Bini first showed a statement about the rank of a tensor corresponding to approximately multiplying matrices of the form

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & 0 \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

He then used this to bound the tensor rank of approximately multiplying a 3×2 matrix and 2×2 matrix as

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \left(\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix} \right) \times \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \\ c_{3,1} & c_{3,2} \end{bmatrix}.$$

We will now define what we mean by approximately multiplying matrices. Namely, Bini proved that there is a tensor T' such that for any $\epsilon > 0$, $R(\langle 3,2,2\rangle + \epsilon T') \leq 10$. Ideally, we could take $\epsilon = 0$ and get a bound on the rank of the matrix multiplication tensor. Unfortunately, ϵ^{-1} terms appear which prevent us from doing this. That said, we can take ϵ arbitrarily small to get a tensor that is close to the MM tensor.

While Bini initially presented his identity as a method for approximate matrix multiplication, he later found he could use it for exact multiplication. To see how to do this, we start with a definition.

Definition 12 (Border Rank)**.** *We say a tensor T has border rank at most r, denoted by $\underline{R}(T)$, if for some d there are tensors T_h such that*

$$T + \sum_{h=1}^{3d} \epsilon^h T_h = \sum_{\ell=1}^{r} \left(\sum_{x \in X} \alpha_{x,\ell}(\epsilon) \cdot x \right) \left(\sum_{y \in Y} \beta_{y,\ell}(\epsilon) \cdot y \right) \left(\sum_{z \in Z} \gamma_{z,\ell}(\epsilon) \cdot z \right),$$

where $\alpha_{x,\ell}(\epsilon), \beta_{y,\ell}(\epsilon), \gamma_{z,\ell}(\epsilon)$ are polynomials in $\epsilon, \frac{1}{\epsilon}$ of degree at most d.

While this definition above is somewhat strange, it is equivalent to the more topological notion that there exists a sequence of low rank tensors that converges to the tensor we care about.

Now suppose that T is a tensor of border rank at most r. We take the Nth Kronecker product of both sides of the definition to get

$$T^{\otimes N} + \sum_{h=1}^{3Nd} \epsilon^h T'_h = \sum_{\ell=1}^{r^N} \left(\sum_{x \in X^N} \alpha_{x,\ell}(\epsilon) x \right) \left(\sum_{y \in Y^N} \beta_{y,\ell}(\epsilon) y \right) \left(\sum_{z \in Z^N} \gamma_{z,\ell}(\epsilon) z \right),$$

where each coefficient $\alpha_{x,\ell}(\epsilon), \beta_{y,\ell}(\epsilon), \gamma_{z,\ell}(\epsilon)$ has degree at most dN in $\epsilon, \frac{1}{\epsilon}$. We now note that we only care about terms of degree 0 in ϵ. Equating the constant terms on both sides, we then get that

$$T^{\otimes N} = \sum_{\substack{a,b,c=-dN \\ a+b+c=0}}^{dN} \sum_{\ell=1}^{r^N} \left(\sum_{x \in X^N} \alpha_{x,\ell}[\epsilon^a] x \right) \left(\sum_{y \in Y^N} \beta_{y,\ell}[\epsilon^b] y \right) \left(\sum_{z \in Z^N} \gamma_{z,\ell}[\epsilon^c] z \right).$$

where we use $p[\epsilon^a]$ to denote the coefficient of ϵ^a in p. So it follows that

$$R(T^{\otimes N}) \leq (2dN)^3 r^N.$$

In particular, if $T = \langle a, b, c\rangle$, then we have

$$\omega \leq \frac{3\log((2dN)^3 r^N)}{\log((abc)^N)} \xrightarrow{N \to \infty} \frac{3\log(r)}{\log(abc)}.$$

Applying this to Bini's result, we get that

$$R(\langle 3^N, 2^N, 2^N \rangle) \leq (4N)^3 10^N.$$

Using our reduction from rectangular to square matrices, we conclude that

$$\omega \leq \lim_{N \to \infty} \frac{3 \log((4N)^3 \cdot 10^N)}{\log((3 \cdot 2 \cdot 2)^N)} = \frac{3 \log(10)}{\log(12)} \approx 2.78.$$

References

[BCRL79] Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Inf. Process. Lett.*, 1979.

[Blä13] Markus Bläser. Fast matrix multiplication. *Theory of Computing*, pages 1–60, 2013.

[Pan78] V Ya Pan. Strassen's algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *19th Annual Symposium on Foundations of Computer Science (FOCS 1978)*, pages 166–176. IEEE, 1978.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.