COMS 6998: Algebraic Techniques in TCS (Fall'21) Oct 26, 2021

Lecture 7: Kronecker Products, DFT, and Matrix Rigidity

Instructor: *Josh Alman* Scribe notes by: *Sandip Nair, Shengyue Guo*

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

1 Kronecker Product

Review of Homework 2. Recall in Problem 1 from Homework 2 we had a matrix R which was defined recursively as follows:

$$R_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix},$$
$$R_n = \begin{bmatrix} R_{n-1} & R_{n-1} \\ R_{n-1} & 0 \end{bmatrix} \in \{0,1\}^{2^n \times 2^n}.$$

We saw how we can compute $R_n \cdot v$ given a vector $v \in \{0,1\}^{2^n}$ in $O(2^n \cdot n)$ operations. Here, computing $R_n \cdot v$ is equivalent to evaluating an n-variable multilinear polynomial on all inputs in $\{0,1\}^n$. This helped us construct an algorithm for the orthogonal vectors problem.

Besides this matrix, there are many important matrices for which we would want to compute a matrix vector product as efficiently as possible. For example, the Fourier Transform shows up in fast evaluation of polynomials. We now define a matrix product which lets us express matrices like R above in a more elegant way.

Definition 1 (Kronecker product $\otimes$)**.** *Consider two matrices $A \in \mathbb{F}^{n_a \times m_a}$ and $B \in \mathbb{F}^{n_b \times m_b}$ over a field $\mathbb{F}$. The Kronecker product of these matrices is a matrix $A \otimes B \in \mathbb{F}^{n_a n_b \times m_a m_b}$, and is defined as $\forall i \in [n_a], j \in [n_b], k \in [m_a], l \in [m_b]$,*

$$(A \otimes B)[(i,j),(k,l)] = A[i,k] \cdot B[j,l],$$

where the entries of $A \otimes B$ are indexed by pairs of indices from the original matrices.

This can be thought of as putting copies of the first matrix A in every position of matrix B multiplied by the entry at that position (matrix-scalar multiplication).

Using the Kronecker product, we can define the matrix R as

$$R_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad R_n = R_{n-1} \otimes R_1, \quad \implies R_n = R_1^{\otimes n},$$

where $R_1^{\otimes n} = \underbrace{R_1 \otimes R_1 \otimes \cdots \otimes R_1}_{n \text{ times}}$. This is a nice method to recursively define many important matrices.

For example, consider the following matrix:

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_n = H_1^{\otimes n}.$$

This matrix, called the "Walsh Hadamard transform" differs from R in that H_1 has a -1 where R_1 has a 0. Computing $H_n \cdot v$ for $v \in \{0,1\}^{2^n}$ is equivalent to evaluating an n-variable multilinear polynomial on all inputs in $\{-1,1\}^n$. Note that by replacing the -1 in H_1 by a, we can obtain a matrix for which the same result holds for all inputs in $\{1,a\}^n$.

Hence, we can efficiently compute any Kronecker product power times a vector using the same recursive idea we have seen for the matrix R.

Theorem 2. *If $A_1 \in \mathbb{F}^{2\times 2}$, and $A_n = A_1^{\otimes n}$, then given $V \in \mathbb{F}^{2^n}$, we can compute $A_n \cdot v$ in $O(2^n \cdot n)$ field operations.*

Proof. Same proof as Problem 1 of Homework 2. □

2 Discrete Fourier Transform

Next, we look at an important linear transformation called the Discrete Fourier Transform.

Let $N = 2^n$. Let $\omega_N = e^{-\frac{2\pi i}{N}} \in \mathbb{C}$. Note that ω_N is an N-th root of unity, as it satisfies $\omega_N^N = e^{-2\pi i} = 1$. The Discrete Fourier Transform is a matrix $F_N \in \mathbb{C}^{N\times N}$ where

$$F_N[a,b] = \omega_N^{ab}. \tag{1}$$

There is an $O(N \log N)$ algorithm for computing the vector $F_N \cdot v$, popularly known as the Fast Fourier Transform or FFT.

Theorem 3 (Discrete FFT)**.** *Let $F_N \in \mathbb{C}^{N\times N}$ be the discrete Fourier transform matrix. Given any $v \in \mathbb{C}^N$, we can compute the vector $F_N \cdot v$ in $O(N \log N)$ operations.*

This algorithm makes use of the following recursive structure of the matrix F_N.

Claim 4. *The discrete Fourier transform $F_N \in \mathbb{C}^{N\times N}$ has the following structure:*

$$\begin{bmatrix} F_{N/2} & D_1 \cdot F_{N/2} \\ F_{N/2} & D_2 \cdot F_{N/2} \end{bmatrix},$$

where the columns are rearranged as $0, 2, 4, \cdots, (N-2); 1, 3, 5, \cdots, (N-1)$, and the rows are still ordered as $0, 1, 2, \cdots, N-1$, and $D_1, D_2 \in \mathbb{C}^{N/2\times N/2}$ are two diagonal matrices.

Proof. The left two block matrices consist of even numbered columns, and the right two block matrices consist of odd numbered columns. We will use a to denote the index of the columns, and b to denote the index of the rows.

1. **Top-left matrix:** Since the top-left matrix have columns $a = 0, 2, 4, \cdots, N-2$, and rows $b = 0, 1, 2, \cdots, N/2-1$, from the definition of F_N in Eq. (1), the top-left matrix has the form:

$$\left[\omega_N^{ab}\right]_{a\in\{0,2,4,\cdots,N-2\}}^{b\in\{0,1,2,\cdots,N/2-1\}}.$$

Let $a' = \frac{a}{2}$ since a is even. The top-left matrix is equivalent to

$$\begin{aligned}\left[\omega_N^{2a'b}\right]_{a'\in\{0,1,2,\cdots,N/2-1\}}^{b\in\{0,1,2,\cdots,N/2-1\}} &= \left[\omega_{N/2}^{a'b}\right]_{a'\in\{0,1,2,\cdots,N/2-1\}}^{b\in\{0,1,2,\cdots,N/2-1\}} && (\because \omega_N^2 = \omega_{N/2})\\ &= F_{N/2}.\end{aligned}$$

2. **Bottom-left matrix:** The bottom-left matrix have columns $a = 0, 2, 4, \cdots, N-2$, and rows $b = N/2, N/2+1, \cdots, N-1$, so it has the following form:

$$\begin{aligned}\left[\omega_N^{ab}\right]_{a\in\{0,2,4,\cdots,N-2\}}^{b\in\{N/2,N/2+1,\cdots,N-1\}} &= \left[\omega_{N/2}^{a'b}\right]_{a'\in\{0,1,2,\cdots,N/2-1\}}^{b\in\{N/2,N/2+1,\cdots,N-1\}} && (\text{by defining } a' = a/2)\\ &= \left[\omega_{N/2}^{a'b'+\frac{N}{2}a'}\right]_{a'\in\{0,1,2,\cdots,N/2-1\}}^{b'\in\{0,1,2,\cdots,N/2-1\}} && (\text{by defining } b' = b - N/2)\\ &= \left[\omega_{N/2}^{a'b'}\right]_{a'\in\{0,1,2,\cdots,N/2-1\}}^{b'\in\{0,1,2,\cdots,N/2-1\}} && (\because (\omega_{N/2})^{N/2} = 1)\\ &= F_{N/2}.\end{aligned}$$

3. **Top-right matrix:** The top-right matrix have columns $a = 1, 3, 5, \cdots, N-1$, and rows $b = 0, 1, 2, \cdots, N/2-1$, so it has the following form:

$$\begin{aligned}\left[\omega_N^{ab}\right]_{a\in\{1,3,5,\cdots,N-1\}}^{b\in\{0,1,2,\cdots,N/2-1\}} &= \left[\omega_N^{b}\cdot\omega_N^{ab}\right]_{a\in\{0,2,4,\cdots,N-2\}}^{b\in\{0,1,2,\cdots,N/2-1\}}\\ &= \left[\omega_N^{b}\cdot\omega_{N/2}^{a'b}\right]_{a'\in\{0,1,2\cdots,N/2-1\}}^{b\in\{0,1,2,\cdots,N/2-1\}} && (\text{by defining } a' = a/2)\\ &= D_1\cdot F_{N/2},\end{aligned}$$

where $D_1 \in \mathbb{C}^{N/2\times N/2}$ is a diagonal matrix defined as

$$D_1[b,b] = \omega_N^b.$$

4. **Bottom-right matrix:** Carrying out a similar transformation, we can define $D_2 = -D_1$ (since $\omega_N^{N/2} = -1$), and the bottom-right matrix is exactly $D_2 \cdot F_{N/2}$. □

Once we have written F_N like this, we can cosntruct a recursive algorithm to compute $F_N \cdot v$ for a given vector v.

Proof of Theorem 3. **Fast Fourier Transform:** Given $v \in \mathbb{C}^N$, to compute $F_N \cdot v$, the algorithm works as follows:

1. Write $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ where $v_1, v_2 \in \mathbb{C}^{N/2}$.
2. Compute $u_1 = F_{N/2}\cdot v_1$ and $u_2 = F_{N/2}\cdot v_2$ recursively.

3. Compute $D_1 \cdot u_2$ and $D_2 \cdot u_2$.

4. Output $\begin{bmatrix} u_1 + D_1 u_2 \\ u_1 + D_2 u_2 \end{bmatrix}$.

The correctness of this algorithm directly follows from Claim 4.
Running time: Since each computation of input dimension N requires two recursive computations of dimensions $N/2$ along with matrix-vector multiplications and vector additions, which take $O(N)$ time each, we have the following recurrence for the running time:

$$\begin{aligned} T(N) &= 2 \cdot T(N/2) + O(N) \\ &= O(N \log N). \end{aligned}$$

□

Notice how this algorithm is almost the same as computing $R^{\otimes n} \cdot v$, with the exception of the diagonal matrices D_1, D_2. There are a few more matrices which can be multiplied by a vector efficiently, and they follow a similar procedure.
Question: Is this algorithm optimal? It is conjectured that $O(N \log N)$ time is necessary.

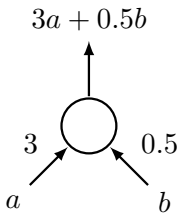
Next, we introduce the concepts of matrix rigidity in order to prove lower bounds on these kinds of matrix-vector computations. If we can show that a given matrix is "rigid", then we cannot multiply it by a vector in linear time. For this, we have to first introduce the model in which we shall be working in, which is the model of linear circuits.

3 Linear Circuit

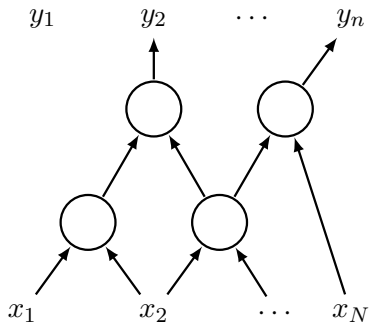
We first define linear circuits over field $\mathbb{F}$.

Definition 5 (Linear circuits over $\mathbb{F}$)**.** *There are N inputs and N outputs from $\mathbb{F}$. Each gate has fan-in 2, and computes a $\mathbb{F}$-linear combination of inputs.*

See Figure 1a for an illustration of a $\mathbb{F}$-linear combination, and see Figure 1b for an illustration of a linear circuit.



(a) Example of a $\mathbb{F}$-linear combination.



(b) Example of a linear circuit.

Parameters. Any circuit C corresponds to matrix $M \in \mathbb{F}^{N\times N}$, and C outputs $y = Mx$ on input $x \in \mathbb{F}^N$. We characterize a circuit using the following parameters:

- Size: # of wires
- Depth: length of the longest path from input to output

Question: For a given matrix M, what size of the circuit is needed to compute M?

We have the following three known facts about the size of the circuit.

Fact 6. *Every $M \in \mathbb{F}^{N\times N}$ can be computed by a circuit of size $O(N^2)$.*

We can construct an $O(N^2)$-size circuit by naively transforming $M \cdot v$ into a circuit.

Fact 7. *Every $M \in \mathbb{F}^{N\times N}$ has a circuit of size $O(N^2/\log N)$.*

(Not proved in class.)

Fact 8. *R_N, H_N, F_N has circuits size $O(N \log N)$, depth $O(\log N)$.*

This follows from the arguments in Section 1 and 2.

Open Question: Find an explicit family of matrices which do not have $O(N)$ size and $O(\log N)$ depth.

Here "explicit family" means a family of matrices $\{M_1, M_2, M_3, \cdots\}$ such that for any $M_N \in \mathbb{F}^{N\times N}$ there is a deterministic poly(N) time algorithm which outputs M_N.

- Why depth $O(\log N)$?

  Most matrices cannot be computed by $o(\log N)$-depth. The fan-in of a gate is 2, so when the depth is d, a circuit has at most 2^d inputs, in order to allow N inputs, we need at least $O(\log N)$ depth.

- Why deterministic?

  A random 0/1 matrix needs $\Omega(N)$-size and $\Omega(\log N)$-depth with high probability.

- Why poly(N) time?

  The problem is still open even if we relax it to either $2^{o(N)}$ time, or poly(N) non-deterministic time.

4 Rigidity

The notion of matrix rigidity was introduced to address the previous open question.

Definition 9 (Rigidity)**.** *Given a matrix $M \in \mathbb{F}^{N\times N}$ and a positive integer r, we define the rank r rigidity of M as $\mathscr{R}_M(r) := \min \#$of entries of M one must change to make its rank $\leq r$.*

Valiant's Approach. In the 70's, Valiant proposed a possible approach to find an explicit family of matrices that do not have $O(N)$-size and $O(\log N)$-depth circuits. His approach consists of two steps:

1. Show that if a matrix can be computed by a $O(N)$-size $O(\log N)$-depth circuit, then it is non-rigid.
2. Show that there exist some explicit family of matrices that are rigid.

Together these two steps will construct the explicit family that we aim for. Valiant proved Step 1, which we will show shortly. However, Step 2 (finding an explicit family of rigid matrices) is still a big **open** problem today!

4.1 Proof of Valiant's Step 1

Next we show the proof of Step 1. We first prove a helpful lemma.

Lemma 10. *Let G be a directed acyclic graph with s edges and depth $d = 2^k - 1$. Then there is a set of $\frac{s}{k}$ edges whose removal makes G have depth $\leq \frac{d-1}{2} = 2^{k-1} - 1$.*

Proof. Since G is a directed acyclic G, there exists a depth function $D : V(G) \to \{0, 1, 2, \cdots\}$ such that for any edge $(a, b) \in E(G)$, $D(a) < D(b)$. G has depth $\leq d$ is equivalent to say that the range of the depth function is $\{0, 1, \cdots, d\}$. So in our case the depth function is

$$D : V(G) \to \{0, 1, \cdots, 2^k - 1\}.$$

We partition the edges of G to $E(G) = E_1 \cup E_2 \cup \cdots E_k$ such that $(a, b) \in E_i$ if $D(a)$ and $D(b)$ first differ in the i-th bit. There must exist an i^* such that

$$|E_{i^*}| \leq s/k.$$

We remove the edges in E_{i^*}.

Consider the new depth function $D' : V(G) \to \{0, 1, \cdots, 2^{k-1} - 1\}$ such that $D'(a)$ is $D(a)$ without the i^*-th bit, e.g., if $D(a) = (c_1, c_2, \cdots, c_k) \in \{0, 1\}^k$, then $D'(a) = (c_1, c_2, \cdots c_{i^*-1}, c_{i^*+1}, \cdots c_k) \in \{0, 1\}^{k-1}$.

D' is a valid depth function since for any edge $(a, b) \in E_j$ for $j \neq i^*$, $D(a)$ and $D(b)$ first differed in the j-th bit, and $D'(a)$ and $D'(b)$ still first differ there. Thus after removing E_{i^*}, the graph G has depth $2^{k-1} - 1$. □

Now we are ready to prove the main theorem of Valiant's Step 1.

Theorem 11. *For any s, d, t, if $M \in \mathbb{F}^{N \times N}$ has circuit size s, depth d, then one can change $N \cdot 2^{d/t}$ entries of M to make its rank $\leq \frac{s \log t}{\log d - \log t}$.*

Proof. W.l.o.g. assume s, d, t, N are all powers of 2. Let C be the circuit for M. Apply Lemma 10 to the circuit C for $\log t$ times. In this way we can find a set T of $|T| \leq \frac{s \log t}{\log d - \log t}$ edges such that the removal of T makes C have depth $\leq d/t$.

Since C has low depth, each output depends on at most $2^{d/t}$ inputs, so the corresponding matrix of C has $\leq 2^{d/t}$ non-zero entries per row, and in total it has $\leq N \cdot 2^{d/t}$ non-zero entries. Each edge in T that we remove was computing some linear combinations of the inputs, thus removing one edge corresponds to a rank-1 update to M. □

Plug in $d = c \cdot \log N$, $s = O(N)$, and set $t = \frac{c}{\epsilon}$, we get that if the matrix M can be computed by a size-s depth-d circuit, then $\mathscr{R}_M(O(\frac{N}{\log \log N})) \leq N^{1+\epsilon}$.

We say a matrix A is "Valiant-rigid" if $\mathscr{R}_A(O(\frac{N}{\log \log N})) > N^{1+\epsilon}$. Thus we have shown that if a matrix can be computed by a $O(N)$-size $O(\log N)$-depth circuit, then it is not Valiant-rigid. This finishes the proof of Step 1 of Valiant's approach.