1 Introduction

Last time

During the last lecture we completed ABFR lower bound method. We finished the proof about computing PAR using circuits with a \textbf{MAJ} gate at the top and \textbf{\lor}, \textbf{\land}, \textbf{\lnot} everywhere else. In addition, we proved Razborov $\textbf{AC}^0$ lower bounds using PAR gates.

Today

Today we discuss about the linear threshold functions (\textbf{LTF}s). We will prove lower bounds for circuits that use LTF gates. We will present the main facts on LTF s and the basic Randomized Communication Complexity (RCC) model. Finally, we analyze LTF under RCC model.

2 Linear Threshold Functions (\textbf{LTF}s)

\textbf{Definition 1.} Linear Threshold Function (\textbf{LTF}) A boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is called a weighted majority or (linear) threshold function if it is expressible as $f(x) = \text{sgn}(w_1 x_1 + \cdots + w_n x_n - \theta) = \text{sgn}(\vec{w} \cdot \vec{x} - \theta)$ for some $w_1, \cdots, w_n, \theta \in \mathbb{R}$

A simple example of LTF is $\phi(x) = 1\{3x_1 + 4x_2 + \cdots + 10x_n \geq 71\}$

LTF s have also some several different names: weighted majority, weighted threshold functions, perceptrons, halfspaces, linear separators etc.

\textbf{Definition 2.} Polynomial Threshold Function (\textbf{PTF}) A function $f : \{0,1\}^n \rightarrow \{0,1\}$ is called a polynomial threshold function (\textbf{PTF}) of degree at most $k$ if it is expressible as $f(x) = \text{sgn}(p(x))$ for some real polynomial $p : \{0,1\}^n \rightarrow \mathbb{R}$ of degree at most $k$
Corollary 3. Every LTF is just a degree-1 PTF.

Fact 4. Is every Boolean function an LTF? No the PAR(x) is not linearly separable:

Figure 2: $f(x) = \text{XOR}(x_1, x_2)$

Fact 5. Any LTF $f(x)$ has many different representations based on $\vec{w}, \theta$. In fact there are infinite different representations over the binary hypercube $\{-1, 1\}^n$. Any coefficient can be perturbed $w_i$ in the range $[w_i - \frac{1}{cn}, w_i + \frac{1}{cn}]$, for a significant large constant $c$, without changing the value of $f(x)$ over $\{-1, 1\}^n$.

Corollary 6. Combining the previous fact and the density of the rational numbers in $\mathbb{R}$, it holds that there is always a representation of LTF with only rational coefficients.
Example 7. Both them are equivalent over $\{-1, 1\}^2$:
\[
\begin{align*}
&x_1 + x_2 \geq 15 \\
&0.99x_1 + 1.002x_2 \geq 15.005
\end{align*}
\]

Fact 8.
\[\text{sgn}(\vec{w} \cdot \vec{x} - \theta) = \text{sgn}(c(\vec{w} \cdot \vec{x} - \theta)) \quad \forall c > 0\]

Fact 9. Any LTF has an integer representation, that is $w_1, \ldots, w_n, \theta \in \mathbb{Z}$.

Proof. As we mentioned before, we can choose a very small $\epsilon_n$ such that $f(x) = \text{sgn}(\sum_{i \in [n]} w_i x_i - \theta)$ equals to $g(x) = \text{sgn}(\sum_{i \in [n]} \bar{w}_i x_i - \theta)$ for all $x \in \{-1, 1\}^n$, where $\bar{w}_i \in (w_i - \epsilon_n, w_i + \epsilon_n)$. As it is well-known from real analysis, the rational numbers are dense in the set of real numbers. This implies that $\forall i \in [n]$ there is at least one $\bar{w}_i$ in the range $(w_i - \epsilon_n, w_i + \epsilon_n)$, which is rational. Thus we can just adopt the following simple strategy: Every non rational coefficient $w_i, \theta$ can be perturbed to any rational number of its aforementioned corresponding range. After that, we can just multiply each coefficient with least common multiple of the denominators. \[\blacksquare\]

Definition 10 (The weight of an integer representation of LTF). The weight of an integer representation of LTF’s parameter $(\vec{w}, \theta)$ is $\sum_{i=1}^{n} |w_i|$.

Definition 11 (The weight of an LTF). The weight of an LTF function $f(x)$, denoted by $W(f(x))$ is the smallest weight over all the weights of the different integer representations of the function.

Example 12.
\[\text{OR}(x_1, \ldots, x_n) = \text{sgn}(\sum_{i=1}^{n} x_i - 1), \vec{x} \in \{0, 1\}^n\]

Notice that using the above representation we can easily argue that $W(\text{OR}) \leq n$. In addition we can easily argue that this is optimal. Let $g(x) = \text{sgn}(\sum_{i=1}^{n} w_i x_i - t)$ be the weight-optimal representation of OR function. By definition of OR function we have that $\forall i \in [n] : f(\vec{e}_i) = f\left((0, \ldots, 0, 1, \text{ith position}, \ldots, 0)\right) = 1$. This implies that $\forall i \in [n] : w_i \geq t$. Additionally, we have that $f(\vec{0}) = -1$. This implies that $t > 0$. Therefore it holds that $\forall i \in [n] : w_i > 0 \Rightarrow w_i \geq 1$, since they are integers. This conclude the proof that the weight of OR is at least $\geq n$ too.

Example 13. $\text{MAJ}(x_1, \ldots, x_n)$ has weight $\Theta(n)$.
\[\text{MAJ}(x_1, \ldots, x_n) = \begin{cases} \sum_{i=1}^{n} x_i \geq n/2 & \vec{x} \in \{0, 1\}^n \\ \sum_{i=1}^{n} x_i \geq 0 & \vec{x} \in \{-1, 1\}^n \end{cases}\]
To prove that the lower bound of $W(\text{MAJ}) = \Omega(n)$, i.e. for the $\{0, 1\}^n$ case, instead of $\{\vec{e}_i \in [n], \vec{0}\}$, one just can use all the $\binom{n}{n/2+1}$ positive examples of $\text{MAJ}$ function which contain $\lceil n/2 + 1 \rceil$ ones and all the $\binom{n}{n/2-1}$ negative examples which contain $\lceil n/2 - 1 \rceil$ examples. This family of vectors is sufficient to prove that $t \geq n/2$ and $w_i \geq 1$.

### 2.1 LTF’s weight upper bound

**Lemma 14** (Weight Upper bound). *Any $n$-variate LTF has weight $\leq 2^{\Theta(n \log n)}$*

**Proof sketch:** Each configuration $(x_1, \cdots, x_n), f(\vec{x})$ corresponds to a linear inequality.

- For example if $f(1, \cdots, 1) = 1$ then we get:

$$\pm w_1 + w_2 + \cdots + w_n - \theta \geq 1$$

- On the other hand, if $f(-1, 1, \cdots, 1) = -1$ then we get $-w_1 + \cdots + w_n - \theta \leq -1$

Since there is the guarantee of LTF property for $f(x)$, we know that the system is feasible. To be more precise, using LP arguments we can conclude that there exists a set of $(n + 1)$ inequalities, which if they will be converted to equalities, will give us a solution $\{w_1^*, \cdots, w_n^*\} = S$. Via LP arguments again, it is not difficult to prove that this solution is guaranteed that will satisfy all $2^n$ linear inequalities.

\[
\begin{bmatrix}
(\pm)w_1 & \cdots & -\theta &=& \pm 1 \\
\vdots & \ddots & -\theta &=& \pm 1 \\
\vdots & \cdots & -\theta &=& \pm 1
\end{bmatrix}
= : \text{System } M
\]

Notice that the matrix of the system has only $(\pm 1)$ entries. Solving the system with Cramer Rule gives us a rational solution. The determinant of the system at the worst scenario is upper bounded by $n! \leq n^n = 2^{n \log n}$. The determinant of the matrix is the denominator of the solution of the Cramer Rule. Numerators are also the determinants of minor matrices of the system. Therefore using the scaling trick that we discussed to transform all the coefficients to integers we get a weight of $2^{n \log n}$.

**Question:** The previous fact upper bounds the weight of an LTF. Is this huge size really necessary?
Example 15. Suppose the following decision list:

\[
L = \begin{bmatrix}
    x_1 & \rightarrow & x_2 & \rightarrow & \cdots & \rightarrow & x_n \\
    \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
    +1 & -1 & +1 & -1 & +1 & -1 & -1
\end{bmatrix} \Rightarrow LTF = \text{sgn}(2^n x_1 - 2^{n-1} x_2 + 2^{n-3} + \cdots)
\]

Notice that because of the priority of the decision list we have inequality constraints like:

\[
\begin{align*}
|w_1| & \geq \sum_{i=1}^{n} / 2 |w_{2i}| \\
|w_2| & \geq \sum_{i=1}^{n} / 2 |w_{2i+1}| \\
|w_3| & \geq \sum_{i=2}^{n} / 2 |w_{2i}|
\end{align*}
\]

Especially for this problem the smallest weight’s numbers are the Fibonacci sequence that are growing exponentially as \((1 + \sqrt{5})^n\)

Lemma 16 (Bad News[Has94]). \(\exists\) \(n\)-variate LTF with weights \(2^{\Omega(n \log n)}\)

Question: Can we approximate any LTF with a lower weight LTF?

Answer: Yes

Fact 17. Let \(f(x)\) be any LTF. For any \(\epsilon > 0\) : \(\exists\) LTF \(f'\) such that \(\text{dist}(f, f') = \Pr_{x \in \{0,1\}^n} [f(x) \neq f'(x)] \leq \epsilon\) where \(W(f'(x))\) is \((\frac{1}{\epsilon})^{\log^2(\frac{1}{\epsilon})} \text{poly}(n)\)

Example 18. For the previous representation of the decision list \(L\), we can just drop the lower significance’s numbers.

3 LTF as gates

First, note that LTF gates can be transformed to equivalent MAJ gates:

\[
f(x) = \text{sgn}(w_1 x_1 + w_2 x_2 + \cdots - \theta)
\]

\[
\uparrow
\]

\[
f(x) = \text{sgn}(\underbrace{x_1 + \cdots + x_1}_{w_1\text{ times}} + \underbrace{x_2 + \cdots + x_2}_{w_2\text{ times}} + \cdots + \underbrace{-1 - \cdots - 1}_{\theta})
\]
Suppose that we have an LTF function of the form: $f(x) = \text{sgn}(\vec{w} \cdot \vec{x} - \theta) = 1\{\vec{w} \cdot \vec{x} > \theta\}$. Then, we can convert it to a majority gate by duplicating each input wire $w_i$ times.

$$\text{LTF}(3x_1, \cdots, 2x_n) \equiv \text{MAJ}(x_1, x_1, x_1, \cdots, x_n, x_n)$$

So constructing circuits of LTF gates is equivalent with a MAJ gates but at a cost of the weight of the LTF.

Some additional interesting remarks:

1. We do not know of any counterexamples to the following claim:

   “Every $f$ in $\text{NP}$ (e.g. $f = \text{Clique}$) has poly$(n)$-size LTF circuits of depth 2”

2. Any poly$(n)$-size depth-$d$ LTF circuit can be computed by a poly$(n)$-size depth-$d + 1$ MAJ circuits

3. Furthermore it is not difficult to prove that any LTF circuit can be computed by depth--$O(1)$ AND/OR/NOT circuits with MAJ gates at the very bottom level.

### 3.1 LTF circuits lower bounds

**Fact 19** (Layered LTF circuits). Any depth-2 layered LTF circuit for parity must have $\Omega(\sqrt{n})$ LTF gates
Note: A circuit is called layered if and only if after a level-decomposition from the input to the output, every wire of the previous level goes only to the very next level.

Proof sketch:

• An edge in \([-1, 1]^n\) is just \((\vec{x}, \vec{x}^{e_i})\), where \(\vec{x}^{e_i} = \begin{cases} x_i^{e_i} = x_j & i \neq j \\ x_i^{e_i} = -x_i & \end{cases}\).

• It is not difficult to prove that if \(f(x)\) is an LTF that separates \([-1, 1]^n\) hypercube then \(f(x)\) slices the edge \((\vec{x}, \vec{x}^{e_i})\) if and only if \(f(\vec{x}) \neq f(\vec{x}^{e_i})\).

Definition 20 (Influence of a variable). The influence of a variable \(x_i\) is the number of the edges of the form \((y, y^{e_i})\) that are sliced by \(f\).

\[\text{Inf}_i(f) = \frac{1}{2^n} \left| \{(y, y^{e_i}) | \forall y \in \{-1, 1\}^n \text{ and } f(y) \neq f(y^{e_i})\} \right|\]

• For any boolean function \(|\hat{f}(i)| \leq \text{Inf}_i(f)\) and if \(f\) is and LTF then the equality is tight.

• For any boolean function, it holds:

\[
\begin{cases}
\sum_{S \subseteq [n]} \hat{f}^2(S) = 1 \\
\bigcup_{i \in [n]} \{i\} \subseteq 2^{[n]} \Rightarrow \sum_{i \in [n]} \hat{f}^2(\{i\}) \leq 1 \Rightarrow \\
\text{For any LTF } f : \sum_{i \in [n]} \left(\text{Inf}_i(f)\right)^2 \leq 1
\end{cases}
\]

Using Cauchy-Schwarz we get :

\[\sum_{i \in [n]} \text{Inf}_i(f) \leq \sqrt{n}\]

• Therefore, for any boolean function that LTF \(f\), \(f\) slices at most \(\left(\frac{1}{\sqrt{n}}\right)\) from all \(n2^{n-1}\) edges.

• Parity function has to slice all the edges. Therefore we need at least \(\sqrt{n}\) LTFs gates in the slice them.
Slicing the cube

Using LTFs arises many interesting questions. For example: How many hyperplanes do we need to slice all the \(n2^{n-1}\) edges of \([-1,1]^n\) hypercube?

1. \(n\) LTFs are enough. We use one for each variable.

2. For the \([-1,1]^6\) cube, it is possible to slice it using 5 LTFs.

3. Generalizing the last result, we can slice all the edges of \([-1,1]^n\) by \(5\frac{n}{6}\).

How many different LTFs exist? Using the Chow’s theorem we can enumerate \(2^{\Theta(n^2)}\). A tighter result gives us a number of \(2^{n^2/2-n/10}\) different LTFs.

What is a hard function for LTFs circuits?

Generally our preferred function is PAR. Here, we will use it via the backdoor.

Definition 21. Inner boolean product is a \((2,n)\)-variate function which corresponds to \(IP(x, y) = \langle x, y \rangle \mod 2\) or equivalently \(IP(x_1, \cdots, x_n, y_1, \cdots, y_n) = PAR(x_1 \land y_1, \cdots, x_n \land y_n)\)

Let’s recall the definition of the parity function:

\[
\chi_S(\vec{x}) = \bigoplus_{i \in S} x_i
\]

Therefore we have: \(\chi_S(\vec{x}) = IP(1_S(\vec{x}), \vec{x})\)

3.2 Lecture’s lower bounds:

Today we will prove the following lower bounds:

1. Any circuit of LTFs gates for IP must use at least \(\Omega(n/\log n)\) gates.

2. Any DT with LTFs as nodes for inner product must have depth \(\Omega(n/\log n)\)

3. Any circuit of the form \(\text{MAJ} \circ \text{(LTF)s for IP}\) uses at least \(2^{\Omega(n)}\) LTF gates

Question: Why will we use IP instead of PAR?

Answer: Under Randomized Communication Complexity with public coins, PAR has a very cheap randomized protocol. On the other hand, IP requires very high communication complexity even for the randomized protocol.
4 Randomized Communication Complexity

Firstly, let’s recall the model of the communication complexity. Let $A$ and $B$ be two cooperative parties, say Alice and Bob. There is a known $f : X \times Y \rightarrow Z$, so $f(x, y) = z$. However, $A$ only gets $x$ and $B$ only gets $y$. They must send information to each other to compute $f$ and at the end both must know the result of $f(x, y)$.

**Definition 22** (Deterministic Communication Complexity of function $f(x, y)$). For a protocol $P$, $\text{cost}(P) =$ the depth of the protocol tree (this is the number of bits of communication needed in the worst case). The deterministic communication complexity of $f$, $D(f)$ is the minimum $\text{cost}(P)$, $\forall P$ that compute $f$.

**Definition 23** (Random Communication Model using public coins). Let $A$ and $B$ be two cooperative parties, say Alice and Bob. There is a known $f : X \times Y \rightarrow Z$, so $f(x, y) = z$. However, $A$ only gets $x$ and $B$ only gets $y$. They must send information to each other to compute $f$ and at the end both must know the result of $f(x, y)$. Additionally, Alice and Bob have access to a common string of random bits.

Notice that Random Communication Model using public coins is just a probability distribution over all deterministic protocols.

So, let’s suppose that Alice and Bob draw some random bits and given that the drawn coins are public, their samples corresponds to a combined sampled protocol.

Thus, we have the following definition of the random communication complexity protocol:

**Definition 24** (Random Communication Complexity Model). A random protocol $P$ for computing $f(x, y)$ works thus: $A$, $B$ sample from a public uniform random string and based on this sample they agree on a deterministic protocol $\text{Protocol}$. Then $A$ and $B$ apply this protocol.

**Definition 25** (The cost of the protocol). For a random protocol $P$, we can define $\text{cost}(P)(x, y) =$ the depth of the protocol tree (this is the number of bits of communication needed in the worst case), given $(x, y)$. For a random protocol $P$, the cost of the protocol is $\text{cost}(P) = \max_{(x, y) \in X \times Y} \text{cost}(x, y)$

**Definition 26** (Random Communication Protocol error). A random protocol $P$ has error $\epsilon$ if and only if $\forall (x, y) \in X \times Y :$

$$\Pr_{r \in \text{public coins}}[P(x, y) \neq f(x, y)] \leq \epsilon$$

**Definition 27** (Randomized - $\epsilon$ communication complexity). The randomized - $\epsilon$ communication complexity of $f$, $\mathcal{R}_\epsilon(f)$ is the minimum $\text{cost}(P)$, $\forall P$ that compute $f$ with at most $\epsilon$ error.
Lemma 28.

\[ R_\epsilon(EQ(x, y)) = O(\log \frac{1}{\epsilon}) \]

Proof. It suffices to prove that there exists a protocol that achieves that bound.

Protocol \( \mathcal{P} \):

- Sample independently, uniformly and publicly \( \log(1/\epsilon) \) strings \( r_1, \ldots, r_{\log(1/\epsilon)} \) of length \( n \).

- Let’s imagine that Alice has the string \( x \) and Bob has the string \( y \).

- Alice computes \( \log(1/\epsilon) \) inner products

\[
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_{\log(1/\epsilon)}
\end{bmatrix} = \begin{bmatrix}
\langle x, r_1 \rangle \\
\langle x, r_2 \rangle \\
\vdots \\
\langle x, r_{\log(1/\epsilon)} \rangle
\end{bmatrix}
\]

- Bob computes two inner products

\[
\begin{bmatrix}
\beta_1 \\
\beta_2 \\
\vdots \\
\beta_{\log(1/\epsilon)}
\end{bmatrix} = \begin{bmatrix}
\langle y, r_1 \rangle \\
\langle y, r_2 \rangle \\
\vdots \\
\langle y, r_{\log(1/\epsilon)} \rangle
\end{bmatrix}
\]

- Alice sends to Bob these \( \log(1/\epsilon) \) bits

\[
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_{\log(1/\epsilon)}
\end{bmatrix}
\]

- Bob answers \( Q(a_1, b_1, a_2, b_2) = \land_{i=1}^{n} [a_i = b_i] \).

- Correctness

  - Firstly, if \( x = y = s \) then :

\[
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_{\log(1/\epsilon)}
\end{bmatrix} = \begin{bmatrix}
\langle x, r_1 \rangle \\
\langle x, r_2 \rangle \\
\vdots \\
\langle x, r_{\log(1/\epsilon)} \rangle
\end{bmatrix} = \begin{bmatrix}
\langle s, r_1 \rangle \\
\langle s, r_2 \rangle \\
\vdots \\
\langle s, r_{\log(1/\epsilon)} \rangle
\end{bmatrix} = \begin{bmatrix}
\langle y, r_1 \rangle \\
\langle y, r_2 \rangle \\
\vdots \\
\langle y, r_{\log(1/\epsilon)} \rangle
\end{bmatrix}
\]
Therefore, if \( x = y \) then Bob always answers correctly.

- Secondly, if \( x \neq y \) then there exists an index \( i \) such that \( x_i \neq y_i \).

Let’s imagine the equality and not-equality indices’ sets:
\[
E = \{ i | x_i = y_i \} \\
D = \{ i | x_i \neq y_i \}
\]

Given that \( x \neq y \Rightarrow |D| > 1 \).

\[
\Pr[\text{Bob fails}] = \Pr[ \bigcap_{i=1}^{\log(1/\epsilon)} (a_i = b_i) ] = \prod_{i=1}^{\log(1/\epsilon)} \Pr[(a_i = b_i)]
\]

Let’s analyze \( \Pr[(a_k = b_k)] \)

\[
\Pr[(a_k = b_k)] = \Pr[\langle x, r_k \rangle = \langle y, r_k \rangle] \\
= \Pr[\langle x, r_k \rangle = \langle y, r_k \rangle] \\
= \Pr[(x - y, r_k) = 0] \\
= \Pr[\sum_{i=1}^{n} (x_i - y_i) r_k \mod 2 = 0] \\
= \Pr[\sum_{i \in E} (x_i - y_i) r_k \mod 2 + \sum_{i \in D} (x_i - y_i) r_k \mod 2 = 0] \\
= \Pr[\sum_{i \in E} 0 \times r_k \mod 2 + \sum_{i \in D} 1 \times r_k \mod 2 = 0] \\
= \Pr[\sum_{i \in D} r_k \mod 2 = 0] = 1/2
\]

The above analysis is just the Random Subset identity. Finally:

\[
\Pr[\text{Bob fails}] = \prod_{i=1}^{\log(1/\epsilon)} \Pr[(a_i = b_i)] = \prod_{i=1}^{\log(1/\epsilon)} \frac{1}{2} = 2^{-\log(1/\epsilon)} = \epsilon
\]

- **Complexity**

  - As the coins are public the only bits that Alice and Bob exchange are these \( \log(1/\epsilon) \) inner-product bit.
So we need $O(\log(1/\epsilon))$ communication bits in order to solve with probability error at most $\epsilon$ to solve the equality problem.

Exercise 29. $\mathcal{R}_{1-\epsilon}(IP) \geq n - \log(1/\epsilon)$

5 Communication Complexity of Univariate Functions

Suppose that we want to analyze Communication Complexity of a univariate function $f(x)$ Then, in order to define the communication complexity of this class, we can just compute

\[
\text{Deterministic Communication Complexity}(f) = \max_{\text{all splits}} D(f)
\]

\[
\text{Random Communication Complexity}(f) = \max_{\text{all splits}} \mathcal{R}_\epsilon(f)
\]

In any case:

- If you want to compute an upper bound on the ratio, we should solve the problem for any partition.
- If you want to compute a lower bound on the ratio, we have to exhibit a specific partition.

Fact 30. For the class of the linear threshold functions, we know that the deterministic communication complexity is $D(LTF) \geq \Omega(n)$.

It seems obvious that this lower bound cannot be improved. Let’s analyze the random public protocols.

The best-known result is the following:

Theorem 31. $\mathcal{R}_\epsilon(LTF) \leq \log \log(n/\epsilon)$

Here, in class, We will prove a more simple result:

Theorem 32. $\mathcal{R}_\epsilon(LTF) \leq O(\log n)O(\log(\log(n/\epsilon)))$
Proof. As we mentioned, in order to upper bound the complexity, we have to design a protocol. Notice that the proposed protocol should be independent of any specific split.

So, let’s assume that $\vec{x}$ is split arbitrarily into

\[ \begin{align*}
    \text{Alice bits : } & X = (x_1, \ldots, x_k) \\
    \text{Bob bits : } & Y = (x_{k+1}, \ldots, x_n)
\end{align*} \]

In the beginning of the lecture for any LTF we showed that $|w_i| \leq 2^{n \log n}$ for all $i \in [n]$. Therefore each $w_i$ is described by $\log(\text{size}) = n \log n$ bits.

Protocol $\mathcal{P}$:

1. Alice computes $\alpha = \sum_{i=1}^{k} x_i \times w_i$
2. Bob computes $\beta = \theta - \sum_{i=k+1}^{n} x_i \times w_i$
   
   Note: They want to verify if $\alpha > \beta$. These numbers have $n \log n$ bits as length.
3. Alice and Bob execute binary search to find the most significant bit that $\alpha, \beta$ differs. Let’s call it $i^*$
4. If $i^* = -1$ then Alice and Bob agree to output 0.
5. If $\alpha_{i^*} = 1$ then Alice and Bob agree to output 1.
6. If $\alpha_{i^*} = 0$ then Alice and Bob agree to output 0

Their binary search scheme is the following: They are trying to identify what is the longest prefix of their numbers that differ.
Binary Search:

1. $\ell_u \leftarrow n \log n$
2. $\ell_d \leftarrow n \log n/2$
3. Alice looks at $\hat{a} \leftarrow (\alpha)_{\text{bit} = \ell_d}^{\ell_u}$
4. Bob looks at $\hat{b} \leftarrow (\beta)_{\text{bit} = \ell_d}^{\ell_u}$
5. Alice and Bob runs the $R_{\epsilon/(2 \log(n))}$ Equality protocol.
6. If $\hat{a} = \hat{b}$ then we extend our search to the suffix $(\ell_u, \ell_d \leftarrow \ell_d, \ell_d - (\ell_d - \ell_u)/2)$
7. If $\hat{a} \neq \hat{b}$ then we extend our search to the prefix $(\ell_u, \ell_d \leftarrow \ell_u, \ell_d - (\ell_d - \ell_u)/2)$
8. If $\ell_u - \ell_d = 0$ then $x = \ell_u = \ell_d$
   - If $x \neq 0$ then output $x$
   - If $x \neq 0$ then output $-1$ /* The numbers were equal */
Communication Complexity:
\[ O\left( \frac{\log(length|W(f)|)}{2} \right) \]
Number of the stages of the binary search
\[ \mathcal{O}\left( \frac{\log(\log(W(f)))}{\epsilon} \right) \]
The samples for a \( \mathcal{O}\left( \frac{\log(W(f))}{2\log(W(f))} \right) \) Equality Protocol

Since \( W(f) = 2^{O(n \log n)} \Rightarrow length|W(f)| \leq \log W(f) = n \log n \).
So, the total complexity is \( O(\log(n \log n) \log(\log(n/\epsilon))) \).

Correctness:
The probability error is by union bound \( \log(lengthW(f)) \times \epsilon/(2\log(n)) \leq \epsilon \).

Lemma 33. Let \( f \) be a boolean function that is computable by a circuit \( C \) of \( s \) gates that belong \( \mathcal{R} \) family. Then \( \mathcal{R}_{1/3}(f) \leq s\mathcal{R}_{1/3s}(\mathcal{R}) \)

Protocol \( P \):

- Sort topologically the circuit of \( f \) from the input-bottom layer to the output-top layer. Each layer is made ups of some gates.
- Parse each layer from the bottom to the top layer and apply the \( \mathcal{R}_{1/3s}(\mathcal{R}) \) protocol.
- Output the result of the final gate of the top layer

Proof.

- Communication Complexity: In each gate we are applying an \( \mathcal{R}_{1/3s}(\mathcal{R}) \) protocol. Since the number of the gates are at most \( s \). We are using \( s \times \mathcal{R}_{1/3s}(\mathcal{R}) \) for communication.
- Correctness: Indeed, by union bound the probability error is \( \frac{1}{3s} \times |\#gates| = \frac{1}{3s} \times s = \frac{1}{3} \).

5.1 Circuit lower bounds

Corollary 34. Any circuit of LTF gates for \( IP \) needs \( \Omega\left( \frac{n}{\log n} \right) \) gates:

Proof. As we mention : \( n - \log(1/\epsilon) \leq \mathcal{R}_{1/2}(\mathcal{R}) \).
Therefore \( n - 2 \leq \mathcal{R}_{1/3}(IP) \leq s\mathcal{R}_{1/3s}(LT\bar{F}) \).
Let’s assume now by contradiction that there was a circuit that uses $s = o\left(\frac{n}{\log n}\right)$. However given the previous result, we have that $sR_{1/3}(\text{LTF}) = o\left(\frac{n}{\log n}\right)R_{1/3}(\text{LTF}) = o(n)$.

But this means that $n - 2 \leq o(n)$, which is obviously a contradiction. ■

**Corollary 35.** Any DT with LTFs as nodes for inner product must have depth $\Omega(n/\log n)$

**Proof.** The proof is similar with the previous one.

The corresponding lemma for thin computation model is the following:

**Lemma 36.** Let $f$ be a boolean function that is computable by a decision tree $T$ with height $h$ using gates that belong to $\mathbb{R}$ family. Then $R_{1/3}(f) \leq hR_{1/3}(\mathbb{R})$

**Proof sketch:** The reason that this lemma is different with the previous one that refers to general gates is simple. During the simulation of a decision tree we will follow only a very specific branch. Therefore the actual circuit that will be used for the computation could be at most the longest branch. By definition the number of the longest branch of the tree is its depth. Thus in the worst case scenario, the communication bits that we will need, are $O(\text{depth}(T))R_{1/3}(\mathbb{R})$. ■

Let’s assume now by contradiction that there was a tree that has depth $d$. However given the previous result, we have that:

$$dR_{1/3}(\text{LTF}) = o\left(\frac{n}{\log n}\right)R_{1/3}(\text{LTF}) = o(n)$$

However as we have already mentioned:

$$n - 2 \leq R_{1/3}(\text{IP})$$

This implies that if there was any DT $T$ with depth $o\left(\frac{n}{\log n}\right)$ that uses as nodes LTF gates then $n - 2 \leq R_{1/3}(\text{IP}) \leq o(n)$, which is obviously a contradiction. ■

Let’s recall the last goal of this lecture:

**Theorem 37.** Any circuit of the form $\text{MAJ} \circ (\text{LTF})s$ for $\text{IP}$ uses at least $2^\Omega(n)$ LTF gates.
Figure 4: Any circuit \( c = \text{MAJ} \circ (\text{LTF})^s \) has two layers. The bottom layer is made up of \( s \) LTFs gates and the top layer is made up of the final MAJ gate.

**Proof.** Here we will need a new trick. We will prove the following lemma:

**Lemma 38.** For any function of the form \( f = \text{MAJ} \circ (\mathcal{R}) \), we have that: \( R_{1/2 - 1/4s}(f) \leq R_{1/4s}(\mathcal{R}) \), for any family of circuits \( \mathcal{R} \).

**Proof.**

- Again we have to implement a protocol for \( f \) using the best possible random protocol for \( \mathcal{R} \).
- The reader first of all should notice that there is an extremely simple \( O(1) \)-bit \( R_{1/2}(f) \) protocol. We can just toss a coin and decide the output of the Majority gate.
- However, given that the majority gates are defined for odd length input we can describe a protocol that boosts a little bit the success probability.

\[
\begin{align*}
\text{Question:} & \quad \text{What is the probability that the majority function agrees with a uniformly randomly chosen input variable } x_i? \\
& \quad \text{(where the probability is also over uniform random inputs.)} \\
\text{Answer:} & \quad \text{At least } \frac{n+1}{n} \quad \text{As the population of the majority is at least } \frac{n+1}{n}
\end{align*}
\]
Therefore:

\[
\Pr_{i \in \text{Size(Bottom Layer)}} \left[ f(x) = C_i(x) \mid C_i(x) \text{is computed correctly} \right] \geq \frac{(s + 1)/2}{s} = \frac{1}{2} + \frac{1}{2s}
\]

So the protocol is pretty simple:

**Protocol \( \hat{P} \):**

1. Choose uniformly at random a circuit \( C_s(x) \) and run the protocol \( R_{1/4s}(\mathcal{R}) \).
2. Output its result

Indeed, the protocol \( \hat{P} \) will output the correct result with probability:

\[
\Pr_{i \in \text{Size(Bottom Layer)}} \left[ f(x) = C_i(x) \mid C_i(x) \text{is computed correctly} \right] \Pr[C_i(x) \text{is computed correctly}] = (\frac{1}{2} + \frac{1}{2s})(1 - \frac{1}{4s}) \geq (\frac{1}{2} + \frac{1}{2s} - \frac{1}{4s}) = \frac{1}{2} + \frac{1}{4s}
\]

Now we are ready to prove our last lower bound: Suppose that \( s = 2^{o(n)} \).

We know that:

\[
n - \log(1/\epsilon) \leq \mathcal{R}_{1/2-\epsilon}(\mathcal{I}P) \quad \forall \epsilon > 0
\]

Let \( \epsilon = 1/4s \) then we have:

\[
n - o(n) \leq \mathcal{R}_{1/2-1/4s}(\mathcal{I}P)
\]

Additionally:

\[
\mathcal{R}_{1/2-1/4s}(\mathcal{I}P) \leq \mathcal{R}_{1/2-1/4s}(\text{MAJ} \circ \text{LTF}) \leq \mathcal{R}_{1/4s}(\text{LTF}) \leq \log(\log n \times 4s) \in O(\log n)
\]

Combining the last two inequalities we get \( \Theta(n) \leq O(\log n) \), which is a contradiction.

References