LECTURE 24

Last time
• $L_p$-testing of monotonicity
• Work investment strategy
• Testing via learning

Today
• Finish testing via learning
• Local Computation Algorithms (LCAs)
• Distributed LOCAL model
• Maximal Independent Set (MIS)

Project Reports are due Thursday, presentations next week
### Monotonicity Testers: Running Time

<table>
<thead>
<tr>
<th>$f$</th>
<th>$L_0$</th>
<th>$L_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[n]$ → ${0,1}$</td>
<td>$\Theta\left(\frac{1}{\epsilon}\right)$</td>
<td>$\Theta\left(\frac{1}{\epsilon^p}\right)$</td>
</tr>
<tr>
<td>$[n]^d$ → ${0,1}$</td>
<td>$O\left(\frac{d}{\epsilon} \cdot \log \frac{d}{\epsilon}\right)$</td>
<td>$O\left(\frac{d}{\epsilon^p} \log \frac{d}{\epsilon^p}\right)$ for $d = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega\left(\frac{1}{\epsilon^p} \log \frac{1}{\epsilon^p}\right)$ for $d = 2$ nonadaptive 1-sided error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Theta\left(\frac{1}{\epsilon^p}\right)$ for constant $d$ adaptive 1-sided error</td>
</tr>
</tbody>
</table>
Testing Monotonicity of $f: [n]^2 \rightarrow \{0, 1\}$

- For nonadaptive, 1-sided error testers, $\Omega\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)$ queries are needed.
- There is an adaptive, 1-sided error tester with $O\left(\frac{1}{\varepsilon}\right)$ queries. Method: testing via learning.
Partial Learning

- An \( \varepsilon \)-partial function \( g \) with domain \( D \) and range \( R \) is a function \( g : D \rightarrow R \cup \{?\} \) that satisfies \( \Pr_{x \in D} [g(x) =?] \leq \varepsilon \).

- An \( \varepsilon \)-partial function \( g \) agrees with a function \( f \) if \( g(x) = f(x) \) for all \( x \) on which \( g(x) \neq ? \).

- Given a function class \( \mathcal{C} \), let \( \mathcal{C}_\varepsilon \) denote the class of \( \varepsilon \)-partial functions, each of which agrees with some function in \( \mathcal{C} \).

- An \( \varepsilon \)-partial learner for a function class \( \mathcal{C} \) is an algorithm that, given a parameter \( \varepsilon \) and oracle access to a function \( f \), outputs a hypothesis \( g \in \mathcal{C}_\varepsilon \) or fails. Moreover, if \( f \in \mathcal{C} \) then it outputs \( g \) that agrees with \( f \).

**Lemma (Conversion from Learner to Tester)**

If there is an \( \varepsilon \)-partial learner for a function class \( \mathcal{C} \) that makes \( q(\varepsilon) \) queries then \( \mathcal{C} \) can be \( \varepsilon \)-tested with 1-sided error with \( q(\varepsilon/2) + O(1/\varepsilon) \) queries.
**Partial Learner of Monotone functions** \( f : [n]^2 \rightarrow \{0, 1\} \)

**Lemma**
There is an \( \varepsilon \)-partial learner for the class of monotone Boolean functions over \([n]^2\) that makes \(O(1/\varepsilon)\) queries.

**Idea:**
- Divide the grid into quarters.
- Query the bottom left and the top right corner for each quarter.
- If the value of the function is NOT determined by the corners, recurse.

**Details:** Keep a quad tree and stop at \( \log \frac{1}{\varepsilon} + 1 \) levels.
- If \( \geq 2^{j+1} \) nodes at level \( j \) are \(?\), fail.
Correctness of the Learner

Claim

If the input function is monotone, level $j$ will have fewer than $2^{j+1}$ nodes.

Proof:

$m = 2^j$
## Monotonicity Testers: Running Time

<table>
<thead>
<tr>
<th>$f$</th>
<th>$L_0$</th>
<th>$L_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[n] \rightarrow {0,1}$</td>
<td>$\Theta\left(\frac{1}{\epsilon}\right)$</td>
<td>$\Theta\left(\frac{1}{\epsilon^p}\right)$</td>
</tr>
<tr>
<td>$[n]^d \rightarrow {0,1}$</td>
<td>$O\left(\frac{d}{\epsilon} \cdot \log \frac{d}{\epsilon}\right)$</td>
<td>$O\left(\frac{d}{\epsilon^p} \log \frac{d}{\epsilon^p}\right)$ for $d = 2$ nonadaptive 1-sided error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega\left(\frac{1}{\epsilon^p} \log \frac{1}{\epsilon^p}\right)$ for $d = 2$ nonadaptive 1-sided error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Theta\left(\frac{1}{\epsilon^p}\right)$ for constant $d$ adaptive 1-sided error</td>
</tr>
</tbody>
</table>
### Monotonicity Testers: Running Time

<table>
<thead>
<tr>
<th>$f$</th>
<th>$L_0$</th>
<th>$L_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[n]$ → $[0,1]$</td>
<td>$\Theta \left( \frac{\log n}{\varepsilon} \right)$</td>
<td>$\Theta \left( \frac{1}{\varepsilon^p} \right)$</td>
</tr>
<tr>
<td></td>
<td>[Ergün Kannan Kannan Kumar Rubinfeld Viswanathan 00, Fischer 04]</td>
<td></td>
</tr>
<tr>
<td>$[n]^d$ → $[0,1]$</td>
<td>$\Theta \left( \frac{d \cdot \log n}{\varepsilon} \right)$</td>
<td>$O \left( \frac{d}{\varepsilon^p \log \frac{d}{\varepsilon^p}} \right)$, $\Omega \left( \frac{1}{\varepsilon^p \log \frac{1}{\varepsilon^p}} \right)$ for $d = 2$</td>
</tr>
<tr>
<td></td>
<td>[Chakrabarty Seshadhri 13]</td>
<td>Nonadaptive 1-sided error</td>
</tr>
</tbody>
</table>

* Hiding some log $1/\varepsilon$ dependence
Approximating $L_1$-distance to monotonicity $\pm \varepsilon$ w. p. $\geq 2/3$

<table>
<thead>
<tr>
<th>$f$</th>
<th>$L_0$</th>
<th>$L_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[n] \rightarrow [0, 1]$</td>
<td>$\text{polylog } n \cdot \left(\frac{1}{\varepsilon}\right)^{O(1/\varepsilon)}$</td>
<td>$\Theta\left(\frac{1}{\varepsilon^2}\right)$</td>
</tr>
</tbody>
</table>

[Saks Seshadhri 10]

- Time complexity of tolerant $L_1$-testing for monotonicity is

$$O\left(\frac{\varepsilon_2}{(\varepsilon_2 - \varepsilon_1)^2}\right).$$
Open Problems

• Our $L_1$-tester for monotonicity is nonadaptive, but adaptivity helps for Boolean range.

  Is there a better adaptive tester?

• All our algorithms for $L_p$-testing for $p \geq 1$ were obtained directly from $L_1$-testers.

  Can one design better algorithms by working directly with $L_p$-distances?

• We designed tolerant tester only for monotonicity (d=1,2).

  Tolerant testers for higher dimensions?

  Other properties?
**Local Computation Algorithms (LCAs)**

**Motivation:** to have sublinear-time algorithms for problems with long output
- User should be able to “probe” bits of the output.
- If there are multiple possible outputs, LCA should be giving answers consistent with one.
- The order of the probes should not affect the answers (instantiations of LCA should be able to consistently answer probes in parallel)
- They can have access to the same random string.
- [Rubinfeld, Tamir, Vardi, Xie 11]
Maximal Independent Set (MIS)

For a graph $G = (V, E)$, a set $M \subseteq V$ is a maximal independent set if

- $M$ is independent: $\forall u, v \in M$, the pair $(u, v) \notin E$
- $M$ is maximal: no larger independent set contains $M$ as a subset.

Example:

- MIS can be found in poly time by greedily adding vertices to $M$ and removing them and their neighbors from consideration.
- It is NP-hard to compute a maximum independent set.

Goal: An LCA for MIS

- Given probe access to a graph $G$ of maximum degree $\Delta$, provide query access to an MIS $M$:
  
  $\text{in-MIS}(v)$: Is $v$ in $M$?

Main idea: modify an existing distributed algorithm for MIS.

Based on Ronitt Rubinfeld’s and Sepehr Assadi’s lecture notes
Distributed LOCAL Model

- The input graph is a communication network; each node is a processor.
- In each round:
  - **Communication**: each vertex can send any message to each neighbor (possibly different messages to different neighbors).
  - **Computation**: each vertex can decide on its actions for the next round, based on received messages.
- At the end of the last round, each vertex decides on its final status (e.g., whether it is in the MIS $M$)
- **Goal**: to minimize the number of rounds.

(A variant of) Luby’s MIS Algorithm for the LOCAL Model

1. Initialize $Active(v) = True; M(v) = False$ for all $v \in V$.
2. For each (out of $R$) rounds, all vertices $v$ run the following in parallel:
   a. Vertex $v$ selects itself with probability $\frac{1}{2\Delta}$
   b. If $Active(v) = True$, $v$ is selected, and no neighbor of $v$ is selected then set $M(v) = True$ and $Active(u) = False \ \forall u \in \{v\} \cup N(v)$
Correctness of Luby’s Algorithm

(A variant of) Luby’s MIS Algorithm for the LOCAL Model

1. Initialize $Active(v) = True; M(v) = False$ for all $v \in V$.
2. For each (out of $R$) rounds, all vertices $v$ run the following in parallel:
   a. Vertex $v$ selects itself with probability $\frac{1}{2\Delta}$
   b. If $Active(v) = True$, $v$ is selected, and no neighbor of $v$ is selected then set $M(v) = True$ and $Active(u) = False$ $\forall u \in \{v\} \cup N(v)$

Correctness Theorem

Let $M$ be the set of vertices for which $M(v) = True$.
1. After every round, $M$ is an independent set
2. When $Active(v) = False$ for all $v \in V$ then $M$ is an MIS.

Proof:
Analyzing the Number of Rounds

Termination Theorem
Fix $v \in V$ and round $R \geq 1$. Then

$\Pr[\text{Active}(v) = \text{True} \text{ after } R \text{ rounds of Luby's algorithm}] \leq \exp\left(-\frac{R}{4\Delta}\right)$

Proof: For each $v \in V$ and round $r \geq 1$, define the following events.

- $A_r(v)$: the event that $\text{Active}(v) = \text{True}$ after round $r$
- $S_r(v)$: the event that $v$ is selected in round $r$
- $M_r(v)$: the event that $v$ is added to $M$ in round $r$

\[
\Pr\left[\overline{A_r(v)} \mid A_{r-1}(v)\right] \geq \Pr\left[\overline{M_r(v)} \mid A_{r-1}(v)\right] \\
= \Pr\left[S_r(v) \land \forall u \in N(v): \overline{S_r(v)}\right] \\
= \Pr[S_r(v)] \cdot \Pr\left[\forall u \in N(v): \overline{S_r(v)}\right] \\
\geq \Pr[S_r(v)] \cdot \left(1 - \sum_{u \in N(v)} \Pr[S_r(u)]\right) \\
\geq \frac{1}{2\Delta} \cdot \left(1 - \Delta \cdot \frac{1}{2\Delta}\right) = \frac{1}{4\Delta}
\]

If $v$ is added to $M$, it is no longer active.

By union bound.

If $v$ is active, it will be deactivated in this round w.p. $\geq \frac{1}{4\Delta}$.
Analyzing the Number of Rounds

Termination Theorem

Fix $v \in V$ and round $R \geq 1$. Then

$$\Pr[A_R(v) = \text{True} \text{ after } R \text{ rounds of Luby's algorithm}] \leq \exp\left(-\frac{R}{4\Delta}\right)$$

Proof: For each $v \in V$ and round $r \geq 1$, define the following events.

$A_r(v)$: the event that $\text{Active}(v) = \text{True}$ after round $r$

$$\Pr\left[ A_r(v) \mid A_{r-1}(v) \right] \geq \frac{1}{4\Delta}$$

$$\Pr[A_R(v)] = \prod_{r=1}^{R} \Pr[A_r(v) \mid A_{r-1}(v)]$$

$$\leq \left(1 - \frac{1}{4\Delta}\right)^R \leq \exp\left(-\frac{R}{4\Delta}\right)$$

Conclusion: Set $R = 8\Delta \cdot \ln n$.

- Then a specific vertex remains active after $R$ rounds w.p. at most $1/n^2$
- By a union bound, no vertex remains active w.p. at least $1 - 1/n$