Lecture 11:

Monotonicity Testing

Property Testers For Monotonicity:

Given list $y_1 \cdots y_n$

Output sorted?

i.e. if $y_1 \le y_2 \le \dots \le y_n$ output PASS (with prob $\ge 3/4$)

if $y_1 \cdots y_n$ ε-far from sorted (need to delete εn? need to change εn?)

output FAIL (with prob $\ge 3/4$)

e.g.

Sorted	1	2	4	5	7	11	14	19	20	21	23
close	1	4	2	5	7	11	14	19	20	39	23
far	45	39	23	1	38	4	5	21	20	19	2

An easy case: $y_i \in \{0,1\}$ $\forall i$

Can do it in poly$(1/\varepsilon)$ time. (hw.)

A first attempt:

Proposed algorithm: "neighbor test"

Pick random i, test $y_i \le y_{i+1}$

Bad input:

$1, 2, 3, 4, 5, \ldots, n/4, 1, 2, 3, 4, \ldots, n/4, 1, 2, 3, 4, \ldots n/4, 1, 2, 3, 4, \ldots n/4$

- $\frac{3}{4}n$- far from monotone
- only 3 choices of i fail

A second attempt:

Proposed algorithm: "random pair test"

Pick random $i < j$, test $y_i \le y_j$

Bad input: $n/4$ groups of 4 decreasing elements

$\underbrace{4,3,2,1}, \underbrace{8,7,6,5}, \underbrace{12,11,10,9}, \underbrace{16,15,14,13}, \ldots$

- largest monotone sequence size $n/4$
- must pick i, j in same group to fail, prob $\le \frac{1}{n}$

if see $o(\sqrt{n})$ samples, prob $o(1)$

A minor simplification:

Lets assume list is distinct

Claim This is wlog

why? (old trick used in parallel computation)

$x_1 \cdots x_n \rightarrow (x_1, 1), (x_2, 2), \ldots, (x_n, n)$

"virtually" (at runtime)
append i to each x_i

breaks ties w/o changing order

ie. if $x_i \leq x_{i+1}$ then $(x_i, i) < (x_{i+1}, i+1)$

A test: Given $x_1 \cdots x_n$

Repeat $O(\frac{1}{\varepsilon})$ times:

runtime $O(\frac{\log n}{\varepsilon})$

Pick $i \in_R [n]$

$z \leftarrow x_i$

do binary search on $x_1 \cdots x_n$ for z

if see any inconsistency FAIL + halt

ie. left is bigger
right is smaller

if end up at locn $j \neq i$ FAIL + halt

Pass

Why does it work?

- If $x_1 < x_2 < \ldots < x_n$ then always passes

- To show: if need to change $\geq \varepsilon n$ x_i's then test fails whp

  equivalently: if test likely to pass, x_i's ε-close to monotone

  <u>defn</u>. i "good" if bin search for $z \in x_i$ successful

  <u>restatement of test</u>:

  Pick $O(\frac{1}{\varepsilon})$ i's randomly + pass if all are good

  if test likely to pass, $\geq 1-\varepsilon$ fraction of i's are good

  (otherwise, in $O(\frac{1}{\varepsilon})$ samples, likely to hit a bad i)

  <u>main observation</u>:

  "good" elements form increasing subsequence

  <u>Proof</u> if $i<j$ both good, let k be least common ancestor in bin search tree.

  When hit x_k, search for x_i went left + search for x_j went right.

  so $x_i < x_k < x_j$ ■

Mon 5

Monotonicity over Posets:

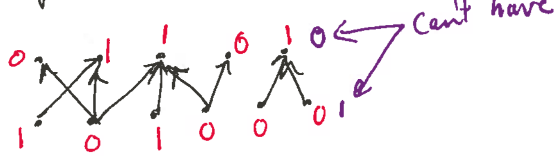
<u>def</u>. f is monotone over poset P if

$\forall\ x \prec y$

then $f(x) \leq f(y)$

<u>examples</u>: Can represent via dags

- bipartite posets



- hypercube

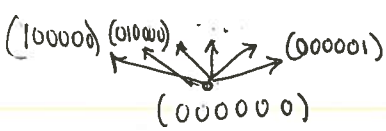
top level: all 1's

level 5: five 1's

⋮

level i: i 1's

⋮



In h.w.: Show testing monotonicity of arbitrary poset can be transformed into "equivalent" monotonicity testing problem on bipartite poset

If can test monotonicity, can also test:

1) Given 2CNF ϕ along with assignment $A = \{a_1 \dots a_n\}$ $a_i \in \{T, F\}$

- Pass if $\phi(A) = T$
- Fail if $\forall A'$ s.t. A ε-close to A' $\phi(A) = F$

whp

2) Given G with $U \subseteq V$

- Pass if U is VC
- Fail if $\forall U'$ s.t. U ε-close to U', U' not VC

\# nodes in $U' \Delta U$

3) Given G with $U \subseteq V$

- Pass if U is clique
- Fail if $\forall U'$ s.t. U' ε-close to U, U' not clique

Thm For bipartite graphs (n nodes on each side) ε-mon test can be done in $O(\sqrt{n/\varepsilon})$ queries

Pf. h.w.

Thm ε-mon test requires $n^{\Omega(1)}$ queries if nonadaptive

$\xRightarrow{\text{h.w.}}$ $\Omega(\log n)$ queries adaptive

open problem: Can we improve this to $\Omega(\sqrt{n})$? for adaptive queries ??

What about grids?

$f: [n] \times [n] \to [m]$

Can test monotonicity in $O(\log^2 n)$ time

actually $O(\frac{1}{\varepsilon} \log n \log m)$

$f: [n]^d \to [m]$

Can test monotonicity in $O(\frac{d}{\varepsilon} \log n \log m)$

$f: 2^d \to \{0,1\}$

Can test monotonicity in $O(\frac{d^{1/2}}{\text{poly}(\varepsilon)} \text{poly}(\log d))$

need $\Omega(d^{1/4})$ queries (even for adaptive algorithms!)

Monotonicity in L_1 distance

Problem: given $f : [n] \to [0,1]$

Pass f if monotone

Fail if f is ε-far in L_1-distance from any monotone fctn.
(ie. L_1-dist is $\geq \varepsilon \cdot n$)

How does it compare to Hamming distance?

when $f : [n] \to \{0,1\}$, Hamming distance equals L_1 distance

for $f : [n] \to [0,1]$, $HD \geq L_1\text{-dist}$

for $f : [n] \to [0, \dots, d]$ $HD \cdot d \geq L_1\text{-dist}$

Thm Can test if f monotone with respect to L_1-distance in $O(\frac{1}{\varepsilon})$ queries

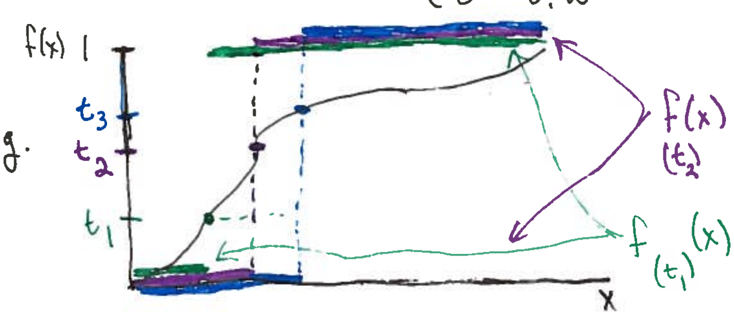
compare to $O(\log n)$ for Hamming distance !!!

Pf. idea: reduce to Boolean fctn monotonicity testing!

def. for $t \in [0,1]$, threshold fctn $f_{(t)} : [n] \to \{0,1\}$,

$$f_{(t)}(x) = \begin{cases} 1 & \text{if } f(x) \geq t \\ 0 & \text{o.w} \end{cases}$$

e.g.



Express f as sum of fctns mapping to $\{0,1\}$:

For *any* fctn f (including non-monotone) s.t. $f(0)=0$:

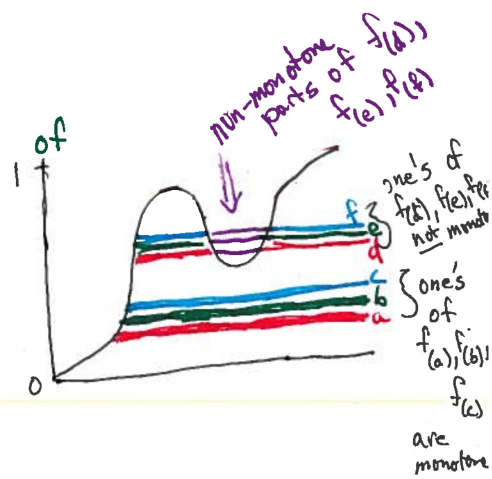
$$f(x) = \int_0^{f(x)} dt = \int_0^1 f_{(t)}(x)\, dt$$

$$= \int_0^{f(x)} \underbrace{f_{(t)}(x)}_{=1 \text{ in this range}} dt + \int_{f(x)}^1 \underbrace{f_t(x)}_{=0} dt$$

Let $L_1(f, M) \equiv L_1$ distance of f to closest monotone fctn.

<u>Lemma</u> $L_1(f,M) = \int_0^1 L_1(f_{(t)}, M)\, dt$

ie. can express dist to monotonicity in terms of distances of threshold fctns.



<u>Pf idea</u>

To change f into monotone, must make each "row" monotone. (row = 0,1 fctn from choice of t)
need to change at least $L_1(f_{(t)}, M)$ in each row

$$\Rightarrow L_1(f, M) \geq \int_0^1 L_1(f_{(t)}, M)\, dt$$

When make each row monotone, let g_t be resulting fctn s.t. g_t is closest monotone fctn to $f_{(t)}$ & let $g \equiv \int_0^1 g_t\, dt$
g monotone since sum of monotone fctns. $\Rightarrow L_1(f,M) \leq \|f-g\|_1$
$\leq \int_0^1 L_1(f_{(t)}, M)\, dt$

Pf

I. $L_1(f, M) \le \int_0^1 L_1(f_{(t)}, M)\, dt$:

$\forall t$, let $g_t \leftarrow$ closest monotone fctn to $f_{(t)}$

(remember that $f_{(t)}$ + g_t are 0/1 fctns!)

let $g \leftarrow \int_0^1 g_t\, dt$

g is monotone since sum of monotone fctns.

so $L_1(f, M) \le \|f - g\|_1$

$= \| \int_0^1 f_{(t)}\, dt - \int_0^1 g_{(t)}\, dt \|_1$

$= \| \int_0^1 f_{(t)} - g_{(t)}\, d_t \|_1$

$\le \int_0^1 \|f_{(t)} - g_{(t)}\|_1\, d_t = \int_0^1 L_1(f_{(t)}, M)\, dt$

II. $L_1(f, M) \ge \int_0^1 L_1(f_{(t)}, M)\, dt$:

let $g \leftarrow$ closest mon fctn ~~to~~ f w.r.t. L_1

so $g_{(t)}$ mon $\forall t \in [0,1]$

$$L_1(f,M) = \|f-g\|_1$$

$$= \left\| \int_0^1 f_{(t)} - g_{(t)} \, dt \right\|_1$$

$$= \sum_{X \text{ s.t. } f(x) \geq g(x)} \int_0^1 \left(f_{(t)} - g_{(t)}\right)(x) \, dt + \sum_{X \text{ s.t. } f(x) < g(x)} \int_0^1 \left(g_{(t)} - f_{(t)}\right)(x) \, dt$$

$$= \int_0^1 \sum_{\substack{X \text{ s.t.} \\ f(x) \geq g(x)}} \left(f_{(t)} - g_{(t)}\right)(x) + \sum_{\substack{X \text{ s.t.} \\ f(x) < g(x)}} \left(g_{(t)} - f_{(t)}\right)(x) \, dt$$

Claim: $f(x) \geq g(x)$ iff $\forall t \in [0,1]$ $f_t(x) \geq g_t(x)$

claim $\Rightarrow$ all terms in both sums are positive

$$= \int_0^1 \|f_{(t)} - g_{(t)}\|_1 \, dt = \int_0^1 L_1(f_{(t)}, M) \, dt$$

Why is characterization helpful?

Lemma

always passes monotone f

if T is a nonadaptive, 1-sided error tester for $f: D \to \{0,1\}$ w.r.t. Hamming measure

$\Rightarrow$ " " " " " " " " " $f: D \to [0,1]$ " L_1 measure

same test, same query complexity,
"same" error ε (but w.r.t. different measure)
more general domain!!

Pf.

if f monotone:
T queries pts in Q & passes iff f monotone on Q
$\Rightarrow T$ accepts (uses 1-sided error of T)

if f st. $L_1(f, M) \geq \varepsilon \cdot n$:

$$\text{Lemma} \Rightarrow L_1(f, M) = \int_0^1 L_1(f_{(t)}, M)\, dt$$

$$\Rightarrow \exists\, t^* \text{ st. } L_1(f_{(t^*)}, M) \geq \varepsilon n$$

Since $f_{(t^*)}$ Boolean, Hamming dist & L_1 dist of $f_{(t^*)}$ from monotone are same!

$\Rightarrow$ T fails $f_{(t^*)}$ for $\geq 3/4$ choices of Q :

i.e. with prob $\geq 3/4$, Q contains $x < y$

s.t. $f_{(t^*)}(x) > f_{(t^*)}(y)$ } maybe not a problem since algorithm only compares $f(x), f(y)$? NO!!!

$\Downarrow$

$f(x) \geq t^* > f(y)$

$\Downarrow$

so Q will output fail

$\blacksquare$