Lecture 20

- Learning heavy Fourier coeffs (with queries) (cont.)
- weak learning of monotone fctns

Recall Fourier Transform:

$$\chi_S(x) = \prod_{i \in S} x_i$$

$$\langle f, g \rangle = \frac{1}{2^n} \sum_x f(x)\, g(x)$$

lemma

$$\hat{f}(S) = \langle f, \chi_S \rangle \quad = 1 - 2 \cdot \Pr[f(x) \neq \chi_S(x)]$$

$$\forall f, \quad f(x) = \sum \hat{f}(S) \chi_S(x)$$

Plancherel $\langle f, g \rangle = \sum_S \hat{f}(S)\, \hat{g}(S)$

Learning Heavy Fourier Coeffs

[Goldreich Levin]
[Kushilevitz Mansour]

$$x \longrightarrow \boxed{f} \longrightarrow f(x)$$

not just low degree S

$\downarrow$

all close linear fctns

Given f, θ

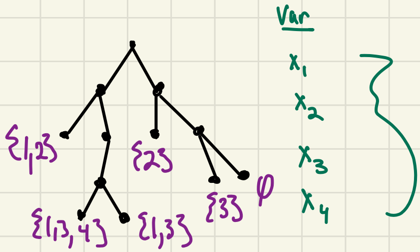
- Output all coeffs S s.t. $|\hat{f}(S)| \geq \theta$
- Only output S s.t. $|\hat{f}(S)| \geq \frac{\theta}{2}$ $\Leftarrow$ no junk

Probably can't do it with only random examples

What if can query f at any input?

Main Idea: "exhaustive search with good pruning"

leaves $\sim$ Fourier coeffs



left $\Rightarrow x_i \in S$
right $\Rightarrow x_i \notin S$

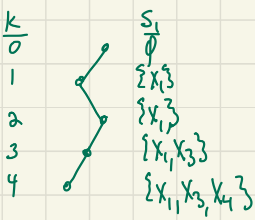
ONLY OUTPUT THOSE THAT REACH BOTTOM LEVEL

recursive algorithm:

- each node $\sim$ setting of $x_1 \cdots x_i$
- estimate "total energy" of subtrees $x_1 \cdots x_i \circ (x_{i+1} = +1)$ & $x_1 \cdots x_i \circ (x_{i+1} = -1)$
- only go down paths with high enough energy

How to prune?

Define quantity:



Fix $0 \le k \le n$ current "level" of search

$S_1 \subseteq [k]$ current "node" of search

2^k such fctns (for each S_1)

$$\boxed{\begin{aligned} &f_{k,S_1} : \{\pm 1\}^{n-k} \to \mathbb{R} \\ &\text{s.t. } f_{k,S_1}(x) = \underbrace{\sum_{T_2 \subseteq \{k+1..n\}}}_{\text{all extensions of } S_1 \text{ to indices in } \{k+1\ldots n\}} \hat{f}(S_1 \cup T_2)\, \chi_{T_2}(x) \end{aligned}}$$

all Fourier coeffs which agree on first k elements

could be $S_1 \cup T_2$ but no need since $\chi_{S_1 \cup T_2} = \underbrace{\chi_{S_1}}_{\text{same for all}} \cdot \chi_{T_2}$

notation:
index $1 \to$ prefix
$2 \to$ suffix

where are S_2 & T_1 in analysis?

Sanity Checks:

1) $k=0$

$$f_{0,\emptyset}(x) = \sum_{T_2 \subseteq [n]} \hat{f}(T_2)\, \chi_{T_2}(x) = f(x)$$

since $k=0$

since $S_1 = \emptyset$

2) $k=n$ $\quad f_{n,S_1}(x) = \hat{f}(S_1)$

since $T_2 = \emptyset$

sum over $T_2 = \emptyset$

Plan Only go down paths with $\mathbb{E}_{k,s}[f^2(x)] \geq \theta^2$

1. can we compute it?
2. does it bring us to right leaves?
   - do we get to all heavy leaves?
   - do we get junk? (light leaves)
3. how many paths do we take?

   lots of dead ends?
   is runtime good?

Not too many paths! (answer to 3)

Lemma "not too many" at any stage in algorithm

$f : \{\pm 1\}^n \to \{\pm 1\}$

(1) $\leq \frac{1}{\theta^2}$ S's satisfy $|\hat{f}(S)| \geq \theta$

(2) $\forall\ 0 \leq k \leq n$, $\leq \frac{1}{\theta^2}$ fctns f_{k,S_1} have $E_x[f_{k,S_1}^2] \geq \theta^2$

(Proved last lecture)

Useful claim (proved last time):

Claim: $\forall$ k, $S_1 \subseteq k$

$$E_x[f_{k,S_1}(x)^2] = \sum_{T_2 \subseteq \{k+1,\dots,n\}} \hat{f}(S_1 \cup T_2)^2$$

Does algorithm bring us to good leaves?

(answer to 2)

Fact: "not missing out" $\Longrightarrow$ find all big Fourier coeffs

For any S_1, if $\exists\, T_2$ s.t.

$$|\hat{f}(S_1 \cup T_2)| > \theta$$

then $E_x[f^2_{k,S_1}(x)] = \sum_{T_2} \hat{f}(S_1 \cup T_2)^2$ via claim

$$\geq \theta^2$$

$\Longrightarrow$ $E_x[f^2_{k,S_1}(x)]$ is a good measure to use when deciding whether to investigate subtree!

So we find all good leaves
we don't spend too much time

but do we output junk?

No junk (answer to 2)

Simple fix:

For each "candidate" to S,

estimate it's Fourier coeff & make sure it is big enough before outputting it recall:

$f: \{\pm 1\}^n \to \{\pm 1\}$

$0 \le k \le n$

$S_1 \subseteq [k]$

Can we estimate $f_{k,S_1}(x)$?

(answer to 1)

$$f_{k,S_1}(x) = \sum_{T_2 \subseteq \{k+1..n\}} \hat{f}(S_1 \cup T_2)\,\chi_{T_2}(x)$$

$$E_x[f_{k,S_1}(x)^2] = \sum_{T_2 \subseteq \{k+1,\dots,n\}} \hat{f}(S_1 \cup T_2)^2$$

Bad idea: estimate each $\hat{f}(S_1 \cup T_2)$ $\forall T_2$ $\longleftarrow$ too much time

Lemma "$f_{k,S_1}(x)$ Estimation lemma"

for $x \in \{\pm 1\}^{n-k}$

"agreement"

$$f_{k,S_1}(x) = E_{y \in \{\pm 1\}^k}[f(yx)\chi_{S_1}(y)]$$

concatenation

use this to estimate $f_{k,S_1}(x)$

Pf

Fourier representation $\Rightarrow$ $f(yx) = \sum_T \hat{f}(T)\chi_T(yx)$

think of as fctn of y since x is fixed throughout

$T = T_1 \cup T_2$ $\quad T_1 \subseteq [k]$ $\quad T_2 \subseteq \{k+1..n\}$

so $\chi_T(yx) = \chi_{T_1}(y) \cdot \chi_{T_2}(x)$

$$E_y[f(yx)\chi_{S_1}(y)]$$

$$= E_y\Big[\sum_{T_1}\sum_{T_2}\hat{f}(T_1 \cup T_2)\,\chi_{T_1}(y)\,\chi_{T_2}(x)\cdot\chi_{S_1}(y)\Big] \quad \text{above}$$

$$= \sum_{T_1}\sum_{T_2}\hat{f}(T_1 \cup T_2)\,\chi_{T_2}(x)\,\underbrace{E_y[\chi_{T_1}(y)\,\chi_{S_1}(y)]}_{=\begin{cases}0 & \text{if } T_1 \neq S_1\\ 1 & \text{if } T_1 = S_1\end{cases}}$$

$$= \sum_{T_2}\hat{f}(S_1 \cup T_2)\,\chi_{T_2}(x)$$

$$= f_{k_1,S_1}(x)$$

■

Overall Algorithm: Pick random x's
for each, pick random y's
estimate $E_y[f(yx)\,\chi_{S_1}(y)]$
Estimate $E_x[f_{k,S_1}(x)^2]$

Chernoff + samples

$\Rightarrow$ Can get γ-additive estimate with prob $\geq 1-\delta$ in $O\left(\frac{1}{\gamma^2}\log\frac{1}{\delta}\text{ queries}\right)$

$\Rightarrow$ Can get $\frac{\gamma^2}{2}$-additive est of $f^2_{k_1 S_1}(x)$

Algorithm:

Ideal [KM] algorithm (given k, S_1):

test S_1 is indeed heavy

if $k = n$

(*) if $\frac{\theta}{4}$-additive-estimate of $\hat{f}(S_1) \geq \frac{3}{4}\theta$ output S_1

else (1) if $E_x[f^2_{k, S_1 \cup \{k+1\}}(x)] \geq \frac{\theta^2}{2}$ — check out left subtree

recurse on $(k+1, S_1 \cup \{k+1\})$

(else kill this subtree)

(2) if $E_x[f^2_{k, S_1}(x)] \geq \frac{\theta^2}{2}$ — check out right subtree

recurse on $(k+1, S_1)$

(else kill this subtree)

Thm $\forall \theta > 0$, KM-alg outputs

$S' = \{S_1 \ldots S_\ell\}$ s.t. $\ell = O(1/\theta^2)$ & with prob $\geq 1-\delta$

$\forall S_i \in S'$ $|\hat{f}(S_i)| \geq \frac{\theta}{2}$ no junk

$\forall S \notin S'$ $|\hat{f}(S)| \leq \theta$ no misses

& query time is poly $(n, \frac{1}{\theta}, \log \frac{1}{\delta})$

Pf. if $\hat{f}(S) < \frac{\theta}{2}$, $\hat{f}(S)^2 \leq \frac{\theta^2}{4}$

& test $*$ prevents it from being output

if $\hat{f}(S) > \theta$, $\forall k$ if S_1 & S agree on $[1..k]$

"not missing out" fact $\Rightarrow$

$$E_x[f^2_{k,S_1}(x)] \geq \theta^2$$

total # nodes explored at level $k \leq \frac{1}{\theta^2}$

$\Rightarrow$ total nodes explored $\leq \frac{n}{\theta^2}$ ∎

Applications

- size $\leq t$ decision trees
- fctns of small L_1-norm

$$L_1(f) = \sum_S |\hat{f}(S)|$$

set $\theta \leftarrow \frac{\varepsilon}{L_1(f)}$

in time poly$(n, L_1(f), 1/\varepsilon)$

if don't know $L_1(f)$

assume $1, 2, 4, 8$

test hypothesis at each round

& continue if not good

Monotone Functions

def. partial order $\leq$: $x \leq y$ iff $\forall i \;\; x_i \leq y_i$

monotone fctn f : $x \leq y \Rightarrow f(x) \leq f(y)$

Are there fast learning algorithms for the class of monotone functions?

Occam's razor:

poly $(\log |C|)$ samples suffice

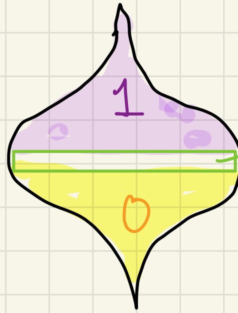
↑ Class of monotone fctns

$\geq 2^{2^n/\sqrt{n}}$ monotone fctns

So only gives exponential bound

Why so many monotone fctns?

Consider "slice" fctns:



set middle row in all possible ways w/o violating monotonicity

$2^{\binom{n}{n/2}}$ options

all are monotone!

H.W.: $2^{O(\sqrt{n})}$ random samples suffice for unif dist

<u>Today</u>: what if you have queries?

Can get very slight "win"

All monotone fctns have weak agreement with <u>some</u> dictator fctn.

<u>Thm</u> $\forall f$ monotone, $\exists g \in \{\pm 1, x_1, x_2, \dots x_n\} \equiv S$

s.t. $\Pr_x [f(x) = g(x)] \geq \frac{1}{2} + \Omega(\frac{1}{n})$

uniform distribution

slightly better than random guessing

(can get $\frac{1}{2} + \Omega(\frac{1}{\sqrt{n}})$ if add majority)

<u>note</u> slice fctns have weak agreement with all dictators on uniform dist

$\Rightarrow$ learning algorithm:

estimate agreement of f with all members of S

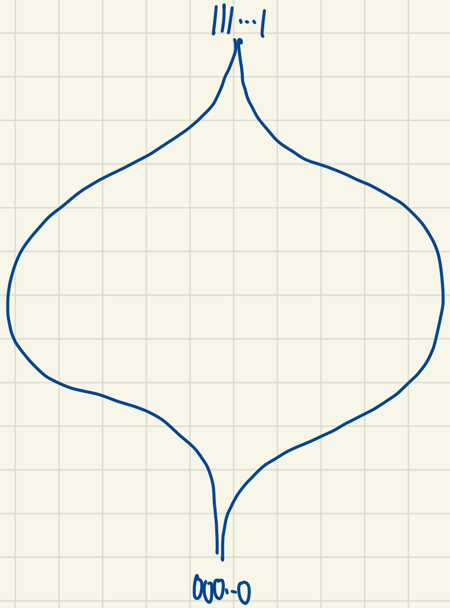
output best

<u>Pf.</u>

Case 1: $f(x)$ has weak agreement with $+1$ or -1 ✓

Case 2: otherwise $\Pr[f(x) = 1] \in [\frac{1}{4}, \frac{3}{4}]$

Boolean Cube



hypercube

level k:

nodes labeled by

k 1's + n-k 0's

nodes on level k:

$\binom{n}{k}$

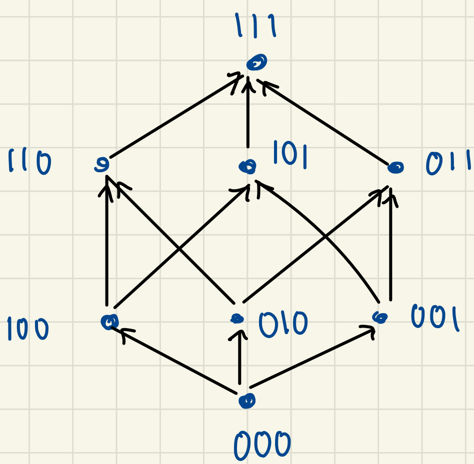
edges:

$x \to y$

if flip one 0 in x to a 1 to get y

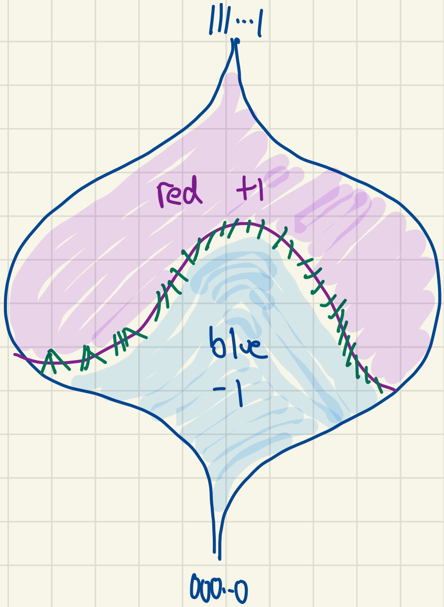
nodes: 2^n

edges: $\frac{n \cdot 2^n}{2}$



example for $n=3$

Monotone Functions on Boolean Cube



monotone ⇒ no blue above any red

$x \le y$ if $\forall i \;\; x_i \le y_i$

f monotone if

$\forall x \le y, \;\; f(x) \le f(y)$

Influence of f:

$$\mathrm{Inf}_i(f) = \frac{\#\text{ red-blue edges in } i^{th} \text{ dir}}{2^{n-1}}$$

$$= \Pr_x\left[f(x) \ne f(x^{\oplus i})\right]$$

x with i^{th} bit flipped

$$\mathrm{Inf}(f) = \frac{\#\text{ red-blue edges}}{2^n}$$

$$= \sum_{i=1}^{n} \mathrm{Inf}_i(f)$$

Thm f monotone $\Rightarrow$ $\mathrm{inf}_i(f) = \hat{f}(\{i\})$

Thm majority fctn $f(x) \equiv \mathrm{sign}\left(\sum_{i=1}^{n} x_i\right)$ (odd n)
maximizes influence among
monotone fctns

Pfs on h.w.