Lecture 16:

Hypothesis Testing
A useful tool: Hypothesis Testing

Given collection of distributions \( \mathcal{H} \), at least one has high accuracy for describing \( p \) given via samples.output one of collection that is close to \( p \).

How many samples in terms of \( |\mathcal{H}| + \text{domain size} \)?

Why is this different than testing closeness, uniformly?

Do we have the same lower bounds?

\[ \textbf{NO} \]

Since \( p \) is guaranteed to be close to some \( q \in \mathcal{H} \), all bets are off!!

A subtool: allows comparing two hypotheses

\[ \textbf{Thm} \]

Given sample access to \( p \)

Given \( h_1, h_2 \) hypothesis distributions (fully known to algorithm)

Given accuracy parameter \( \epsilon' \), confidence \( \delta' \)

Algorithm "Choose" takes \( O(\log(1/\delta')/\epsilon'^2) \) samples & outputs

\[ h \in \mathcal{E}_{h_1, h_2, 3} \]. If one of \( h_1, h_2 \) has \( \|h_1 - p\|_1 < \epsilon' \)

Then with prob \( \geq 1 - \delta' \), output \( h \) has \( \|h - p\|_1 \leq 2\epsilon' \)

\[ \text{if both } h_1, h_2 \text{ far, no guarantees} \]

\[ \text{if one is close, you output something pretty close.} \]
Actually, will prove something stronger!

**Thm**  \[ p \] given via samples

\[ h_1, h_2 \] fully known \( \Rightarrow \) is \( \varepsilon' \)-close to at least one of \( h_1, h_2 \)

\( \varepsilon', \delta' \) given

Algorithm "Choose" takes \( O(\log(\varepsilon'))(1/\varepsilon')^2 \) samples

\( \Rightarrow \) outputs \( h \in \{h_1, h_2\} \) satisfying:

1. if \( h \) more than \( 12 \varepsilon' \)-far from \( p \), unlikely to output it as winner

\( \Rightarrow \) very bad

2. if \( h \) more than \( 10 \varepsilon' \)-far, unlikely to output as winner

\( \Rightarrow \) not that bad

\( \Rightarrow \) might tie but won't win

\[ \text{Why such crazy constants??} \]
Proof of "Subgoal":

idea: wlog \( h_1 \) is \( \varepsilon' \)-close,
if \( h_2 \) is \( 10\varepsilon' \)-close, then either output ok as "winner" or "tie"
else, if \( h_2 \) is not \( 10\varepsilon' \)-close but is \( 12\varepsilon' \)-close, ok to "tie" or output \( h_0 \)
else, \( h_2 \) is \( 12\varepsilon' \)-far, from \( h_1 \) + \( 11 \varepsilon' \)-far from \( p \)
so samples from \( p \) will fall in "difference" between \( h_1 + h_2 \) will output \( h_1 \)

Algorithm
Choose: \( P, h_1, h_2 \)

First some definitions:

\[ A = \{ x \mid h_1(x) > h_2(x) \} \]

\[ a_1 = h_1(A) \]
\[ a_2 = h_2(A) \]

note \( ||h_1-h_2|| = 2(a_1-a_2) \)

1. if \( a_1-a_2 \leq 5\varepsilon' \) declare "tie" + return \( h_1 \)
   (no samples needed)

2. draw \( m = 2 \cdot \frac{\log \frac{1}{\delta}}{(\varepsilon')^2} \) samples \( s_1 \ldots s_m \) from \( P \)

3. \( \alpha \leftarrow \frac{1}{m} \sum_{i=1}^{m} s_i \in A \)

4. if \( \alpha > a_1 - \frac{3}{2} \varepsilon' \) return \( h_1 \)
   else if \( \alpha < a_2 + \frac{3}{2} \varepsilon' \) return \( h_2 \)
   else declare "tie" + return \( h_1 \)
Why does it work?

\[ E[\alpha] = p(A) \]

- if reach step 2, whp (via Chernoff) \( |\alpha - E[\alpha]| \leq \frac{\varepsilon'}{2} \)

if \( \|p - h_1\|_1 > 12\varepsilon' \) then since other is \( \leq \varepsilon' \) distance,
or \( \|p - h_2\|_1 > 12\varepsilon' \)

\( \|h_1 - h_2\|_1 > 3\varepsilon' \)

so will reach step 2

if \( p \varepsilon' \)-close to \( h_1 \), whp \( \alpha > \alpha_1 - \varepsilon' - \varepsilon'' \)

so output \( h_1 \)

else, \( p \) is \( 12\varepsilon' \) far from \( h_1 \)

but \( \varepsilon' \)-close to \( h_2 \)

whp \( \alpha > \alpha_2 + \varepsilon' + \varepsilon'' \)

if one of \( h_1, h_2 \) \( \varepsilon' \)-close, and other is \( >10\varepsilon' \) far but not \( 12\varepsilon' \) far,

if \( \alpha_1 - \alpha_2 \leq 5\varepsilon' \) then declares draw, so neither are declared "winner"

else \( \|h_1 - h_2\|_1 > 9\varepsilon' \) far

+ Similar reasoning shows that medium far will not win (in fact, will lose)

- if both are \( 10\varepsilon' \)-close, might output \( h_1, h_2 \) or "tie"

Recall: \( \|h_1 - h_2\|_1 = 2(\alpha_1 - \alpha_2) \)
The Cover Method

A method for learning distributions

\[ \text{def } C \text{ is a } \varepsilon \text{-cover of } \mathcal{P} \text{ if } \forall p \in \mathcal{P} \]

\[ \exists \ v \in C \text{ s.t. } \|p - v\|_1 \leq \varepsilon \]

Why useful?

Hopefully \( C \) is much smaller than \( \mathcal{P} \) allows us to approximate.

Note: \( C \) not unique

Big improvement: union bound over size of \( C \) not \( |\mathcal{P}| \)!

**Theorem:** Algorithm given \( p \in \mathcal{P} \), which takes \( O(\frac{1}{\varepsilon^2 \log |C|}) \) samples of \( p \) outputs \( h \in C \)

\[ \text{st. } \|h - p\|_1 \leq 6\varepsilon \] with prob \( \geq 9/10 \)

**Proof:**

Since \( p \in \mathcal{P} \), \( \exists v \in C \) s.t. \( \|p - v\|_1 \leq \varepsilon \)

(but there could be more than 1) \(-\) we just need to find one, not even required to return

will run Choose on \( p \) with every pair \( q_1, q_2 \in C \)

if \( q \) doesn't win all of its "matches" then it loses to someone that is not so bad

Furthermore can't show that with \( p \) there is a \( q' \) s.t.

\( q' \) always satisfies all matches (best \( q \) never loses, anyone that is \( \leq 10\varepsilon \) far)

\( q' \) needs all matches to give correct output - union bound on \( \binom{|C|}{2} \) matches
The cover method

Example 1: learning distribution of a coin

domain = \{0, 1\}

need to learn bias

Here C^o = \mathbb{Z} = \{0, 1, \frac{1}{k}, \frac{2}{k}, ... \frac{k-1}{k}, \frac{k}{k} = 1\}

if \text{ use } C^o = \mathbb{Z} \text{ then } \forall \text{ bias } p, \quad \text{let } \frac{i}{k} \leq p \leq \frac{i+1}{k}

then picking \hat{p} = \frac{i}{k} \text{ gives } \|p - \hat{p}\|_1 = \left| \frac{i}{k} - p \right| + \left( \frac{i-\left\lfloor \frac{i}{k} \right\rfloor}{k} \right) \left| p \right| \leq \frac{2}{k}

so using \quad k = \Theta\left(\frac{1}{\varepsilon} \right) \text{ gives } \|p - \hat{p}\|_1 \leq \varepsilon

|C^o| = k + 1 = 6(\varepsilon) \quad \# \text{ samples needed by cover method is } O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right)

Example 2: 2-bucket distributions

now need to specify \alpha \quad \text{and } \beta

so \quad C^o = \mathbb{Z} \left( \frac{\alpha}{k}, \frac{\beta}{k} \right), \quad \alpha, \beta \in \mathbb{Z}\quad \#\text{samples}

|C^o| = \Theta\left(\frac{1}{\varepsilon^2}\right) \quad \# \text{ samples is } O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right)

Example 3: monotone distributions

Birge \implies C^o = \mathbb{Z} \left( \frac{\lambda_1}{k}, \frac{\lambda_2}{k}, ... \frac{\lambda_n}{k} \right), \lambda_1, \lambda_2, ... \in \mathbb{Z} \quad \#\text{samples}

|C^o| = \Theta\left(\frac{1}{\varepsilon \log \frac{n}{\varepsilon}}\right) \quad \# \text{ samples is } O\left(\frac{1}{\varepsilon^3} \cdot \log n \cdot \log \frac{1}{\varepsilon}\right)