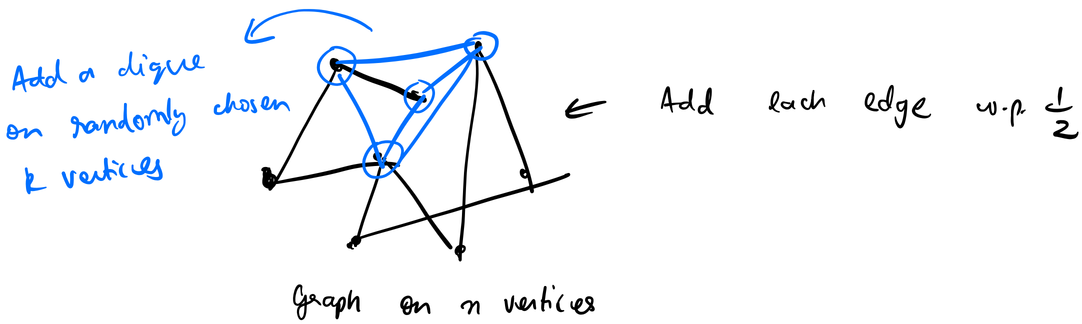
Lecture 20

* HW3 posted.

Computational - Statistical tradeoffs



Def (Planted clique problem).

Given a graph generated from one of the following 2 distributions, decide which distribution generated the graph

① $G(n, \frac{1}{2})$

② Generate an instance of $G(n, \frac{1}{2})$ & plant a clique on k randomly chosen vertices of the graph.

What is the smallest k at which these two distributions are information-theoretically distinguishable?

Lemma An Erdös-Renyi random graph $G(n, \frac{1}{2})$ does not have a clique of $k \geq 3\log n$, w.h.p.

For $k \gtrsim 3\log n$:

Algorithm (Brute-force search)

→ Search over every subset of k vertices

→ If $\exists$ a clique on any subset

return (graph comes from a planted clique mode)

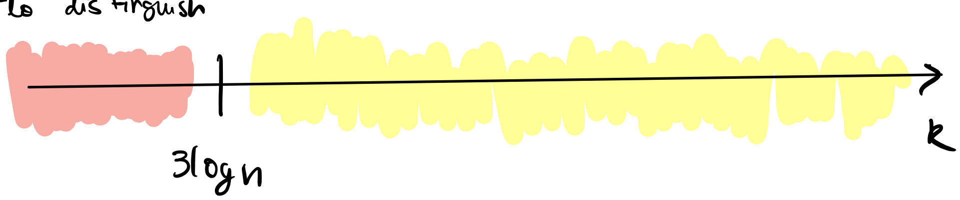
→ Else

return (graph comes from $G(n, \frac{1}{2})$)

Running time : $k^2 \binom{n}{k} = \Omega\left(n^{\log n}\right)$.

for $k << \log n$ information-theoretically impossible to distinguish

information theoretically possible to detect if $k \geq 3\log n$



When can we do this efficiently ?

Simple alg. when $k >> \sqrt{n \log n}$

Lemma The # edges in an Erdős-Renyi graph $G(n, \frac{1}{2})$ lies in $\left[\frac{n \cdot n-1}{4} - 100 n\sqrt{\log n}, \frac{n \cdot n-1}{4} + 100 n \sqrt{\log n}\right]$, whp.

By adding a clique on k vertices,

we will add $\simeq \binom{k}{2} \cdot \frac{1}{2}$ edges (repeat previous lemma for rigorous detail)

$$= \frac{k \cdot k-1}{4} \text{ edges}$$

$k = \sqrt{C n \log n}$ for some large C.

we will add $\simeq \frac{C n \log n}{4}$ edges

whp, for $k = \sqrt{c n \log n}$, $G(n, \frac{1}{2})$ graph with planted clique has at least

$$\frac{n \cdot n-1}{4} + \frac{1000 n \log n}{4} \text{ edges}$$

for large enough c.

for $k \gg \sqrt{n \log n}$,

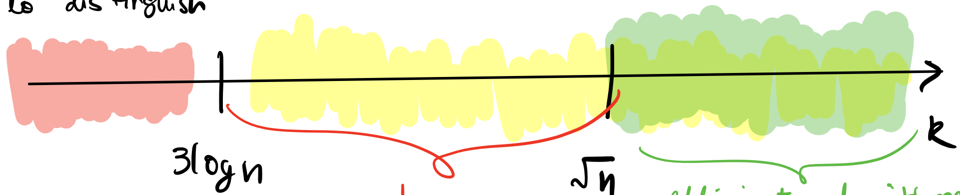
we can efficiently detect the clique whp by counting # edges in the graph.

for $k \gg \sqrt{n}$,

there is an efficient algorithm based on eigenvalue decomposition of the adjacency matrix of the graph.

for $k << \log n$ information-theoretically impossible to distinguish

information theoretically possible to detect if $k \geq 3\log n$



believed to be impossible to efficiently detect the presence of a planted clique in this regime.

efficient algorithms exist!

Planted clique conjecture: There is no efficient algorithm to detect a planted clique of size $k = o(\sqrt{n})$ in a $G(n, \frac{1}{2})$ graph.

Known to be be true for restricted class of algorithms:

1) a version of SQ algo.
2) generalizations of Semi-Definite Programs (SDPs)
3) Markov-Chain-Monte-Carlo (MCMC) methods

⋮

Memory-sample tradeoffs

What is the tradeoff b/w available memory & # samples needed for learning.

Memory-sample tradeoffs for parity learning:

Data comes in streaming fashion: Get datapoints one at a time, only get a single pass over your data stream.

Parity function:

$\mathcal{X}^d = \{0,1\}^d$

$\mathcal{Y} = \{0,1\}$

$\mathcal{C} = \{ w(x) = \langle w, x \rangle \bmod 2 : w \in \{0,1\}^d \}$

There is some unknown $w^* \in \{0,1\}^d$ which we want to find.

<u>at $t=1$</u>

get $x_1 \sim \mathrm{Unif}(\{0,1\}^d)$

get $b_1 = \langle x_1, w^* \rangle \bmod 2$

<u>at $t=2$</u>

get $x_2 \sim \mathrm{Unif}(\{0,1\}^d)$

get $b_2 = \langle x_2, w^* \rangle \bmod 2$

⋮

What is the tradeoff b/w the available memory & # samples needed for learning?

<u>Algorithm 1</u>

Store $n = O(d)$ examples in memory, solve linear system.

$$\begin{pmatrix} — x_1 — \\ — x_2 — \\ \vdots \\ — x_n — \end{pmatrix} \begin{pmatrix} w^* \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \pmod 2$$

Since w^* is d-dimensional, with $n \gg 100d$ examples, the system is full-rank whp.

Samples $= n = O(d)$

Memory $\simeq nd$ bits $= \Omega(d^2)$

Algorithm 2

Brute force search

- For every $w \in \{0,1\}^d$

  check if w is consistent over the next $O(d)$ examples we receive

  If consistent

    return w

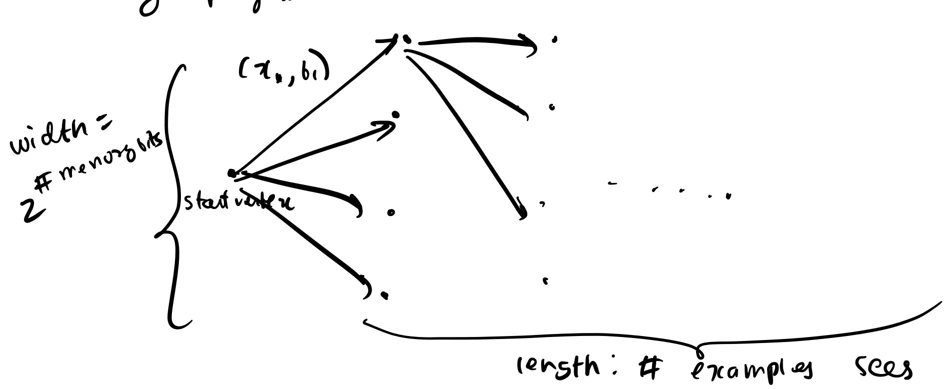
Memory $\simeq O(d)$

Samples $\simeq d\, 2^d$

Question: What else is possible?

$$\boxed{\text{Thm (Raz'17): Any algorithm for solving the above parity problem either requires } \Omega(d^2) \text{ memory, or at least } 2^{\Omega(d)} \text{ samples!}}$$

Branching program:



Thm (Garg-Raz-Tal '18): Consider a hypothesis class $\mathcal{H}$ with SQ-dim$(\mathcal{H}) = s$. Then any algorithm for learning $\mathcal{H}$ either requires $\Omega(\log^2 s)$ memory, or at least $s^{\Omega(1)}$ (poly(s)) samples.