Scribed by Keyhan Vakil

Lecture 3

In which we complete the study of Independent Set and Max Cut in $G_{n,p}$ random graphs.

1 Maximum Independent Set

Last time we proved an upper bound of $O\left(\frac{1}{p}\log np\right)$ to the probable value of the maximum independent set in a $G_{n,p}$ random graph. This bound also holds if p is a function of n. There is a simple greedy algorithm which can be shown to achieve an independent set of size $\Omega(n/d)$ where d is the average degree of the graph. For a $G_{n,p}$ random graph, this gives us an independent of size $\Omega(1/p)$. However we will see how to specialize this analysis to *sparse* $G_{n,p}$ random graphs, and close the remaining gap between the probable value and the greedy algorithm.

Consider the greedy algorithm below.

- $S := \emptyset$
- for each $v \in V$
  - if v has no neighbors in S then $S := S \cup \{v\}$
- return S

1.1 First attempt

We might try to model our analysis of this algorithm based on our discussion from Lecture 2.

To wit, let R be the set of vertices not in S which have no neighbors in S. Let R_i be the size of R when S contains i vertices. If $R_k = 0$, then our algorithm outputs an independent set of size k. Therefore we can determine the expected size of the algorithm's output (up to a constant factor) by determining k such that $\mathbb{E}[R_k] = O(1)$.

Now we determine $\mathbb{E}[R_{i+1} \mid R_i]$. A proportion of p vertices are connected to the $(i+1)$th vertex in expectation. Of the R_i vertices, we expect that $1-p$ of them will remain unconnected to all the vertices in S. This gives us that $\mathbb{E}[R_{i+1} \mid R_i] = (1-p)R_i$, and by induction $\mathbb{E}[R_k] = (1-p)^k n$.

Let k be such that $\mathbb{E}[R_k] = 1$. Then:

$$\mathbb{E}[R_k] = (1-p)^k n = 1 \implies k = \log_{\frac{1}{1-p}} n \approx \frac{1}{p} \ln n$$

We conclude that our independent set has expected size $\Theta(\frac{1}{p} \log n)$. However if we take $p = \Theta(1/n)$, that would lead us to believe that we could get an independent set of size $\Theta(n \log n)$ in a graph with only n vertices, which is impossible.

The error is that $\mathbb{E}[R_{i+1} \mid R_i]$ should be $(1-p)(R_i - 1)$, not $(1-p)R_i$. Note that once we add the $(i+1)$th vertex to S, it can no longer be in R by definition. When p is a constant, the difference is negligible, but when p is small then the difference becomes more significant.

It is possible to salvage this analysis, but the result is less elegant. Instead we will now present a different analysis, which will also let us conclude more about higher moments as well.

1.2 Analysis of the greedy algorithm

To analyze the algorithm, consider the following random variables: let t_i be the number of for-loop iterations between the time the i-th element is added to S and the time the $(i+1)$-th element is added to S. We leave t_i undefined if the algorithm terminates with a set S of size less than $i+1$. Thus the size of the independent set found by the algorithm is the largest i such that t_{i-1} is defined.

Consider the following slightly different probabilistic process: in addition to our graph over n vertices $\{1, \ldots, n\}$, we also consider a countably infinite number of other vertices $n+1, n+2, \ldots$. We sample an infinite super-graph of our graph over this larger vertex set, so that each possible edge has probability p of being generated.

We continue to run the greedy algorithm for every vertex of this infinite graph, and we call t_i the (now, always defined) number of for-loop iterations between the i-th and the $(i+1)$-th time that we add a node to S. In this revised definition, the size of the independent set found by algorithm in our actual graph is the largest k such that $t_0 + t_1 + \cdots + t_{k-1} \leq n$.

Now we will reason about the distribution of t_i. Say that we have i vertices in S and we are trying to determine if we should add some vertex v to S. Note that the probability of v being disconnected from all of S is $(1-p)^i$. So we add a vertex at each iteration with probability $(1-p)^i$, which shows that t_i is geometrically distributed with success probability $(1-p)^i$.

Based on this, we can find the expected value and variance of our sum from before

$$\mathbb{E}\left[t_0 + t_1 + \cdots t_{k-1}\right] = \frac{\frac{1}{(1-p)^k} - 1}{\frac{1}{1-p} - 1} \leq \frac{\frac{1}{(1-p)^k}}{\frac{1}{1-p} - 1} = \frac{1}{p \cdot (1-p)^{k-1}}$$

and likewise

$$\begin{aligned}
\mathbf{Var}[t_0 + t_1 + \cdots t_{k-1}] &\leq \sum_{i=0}^{k-1} \frac{1}{(1-p)^{2i}} \\
&= \frac{\frac{1}{(1-p)^{2k}} - 1}{\frac{1}{(1-p)^2} - 1} \\
&\leq \frac{1}{(1-(1-p)^2) \cdot (1-p)^{2k-2}} \\
&\leq \frac{1}{p \cdot (1-p)^{2k-2}} \\
&= p\left(\mathbb{E}[t_0 + \cdots + t_{k-1}]\right)^2 .
\end{aligned}$$

We want to choose k so that the sum is at most n with high probability. Let

$$k = \log_{\frac{1}{1-p}} \frac{pn}{2} \approx \frac{1}{p} \ln pn.$$

This makes the expected value of the sum $\leq n/2$ and the standard deviation $\leq \sqrt{p}n/2$. Thus, if $p(n) \to 0$ sufficiently fast, the greedy algorithm has a $1 - o(1)$ probability of finding an independent set of size $\Omega(p^{-1} \log pn) = \Omega\left(\frac{n}{d} \log d\right)$, where $d := np$ is a measure of the average degree.

1.3 Certifiable upper bound

We now derive a polynomial time computable upper bound certificate for maximum independent set in $G_{n,p}$. We use the following lemma without proof. Note its similarity to Lemma 2 from Lecture 1.

Lemma 1 *If $p = p(n) \geq \frac{\log n}{n}$, G is sampled from $G_{n,p}$, A is the adjacency matrix of G, and J is the matrix of all ones, then there is a $1 - o(1)$ probability that*

$$\|A - pJ\| \leq O(\sqrt{pn})$$

Since $A - pJ$ is a real symmetric matrix its spectral norm can be computed as:

$$\|A - pJ\| = \max_{\mathbf{x} \neq \mathbf{0}} \frac{|\mathbf{x}^T (A - pJ)\mathbf{x}|}{\mathbf{x}^T \mathbf{x}} .$$

If S is an independent set of size k, then $\mathbf{1}_S^T A \mathbf{1}_S = 0$, $\mathbf{1}_S^T J \mathbf{1}_S = k^2$, and $\mathbf{1}_S^T \mathbf{1}_S = k$, so that

$$\begin{aligned}
\|A - pJ\| &\geq \frac{|\mathbf{1}_S^T (A - pJ)\mathbf{1}_S|}{\mathbf{1}_S^T \mathbf{1}_S} \\
&= pk.
\end{aligned}$$

This bound holds for any independent set, so it also holds for the largest one. If we denote by $\alpha(G)$ the size of the largest independent set in G, we have that

$$\alpha(G) \leq \frac{1}{p}\|A - pJ\|.$$

For a $G_{n,p}$ random graph, the above upper bound is $O(\sqrt{n/p}) = O(n/\sqrt{d})$ with high probability.

2 Max Cut

We will now reconsider Max Cut for the general case $G_{n,p}$. In Lecture 2, we dealt with the special case of $p = \frac{1}{2}$. Unlike maximum independent set, our arguments for the case $p = \frac{1}{2}$ apply to Max Cut without much modification.

2.1 High probability upper bound

Let G be a random graph from $G_{n,p}$, and define $d := pn$ as a measure of its average degree. We will prove that the size of a maximum cut of G is at most $dn/4 + O(\sqrt{d}n)$ with high probability. The proof of this statement is nearly identical to the version in Lecture 2, where it was presented for the case $p = \frac{1}{2}$. We know that the expected value of a cut S is $|S| \cdot |V - S| \leq dn/4$. By a Chernoff bound, the probability that any particular cut exceeds expectation by an additive factor of $O(\epsilon n)$ is exponentially decreasing by a factor of $\epsilon^2 dn$. By taking $\epsilon = 1/\sqrt{d}$ and taking a union bound over all 2^n possible cuts S, we have that our expected cut has value at most $dn/4 + O(\sqrt{d}n)$ with probability $1 - 2^{-\Omega(n)}$.

2.2 Greedy algorithm

Consider the greedy algorithm

- $A := \emptyset$
- $B := \emptyset$
- for each $v \in V$
  - if v has more neighbors in B than in A then $A := A \cup \{v\}$
  - else $B := B \cup \{v\}$
- return (A, B).

Label $V = \{1, \ldots, n\}$. Let A_i and B_i be the sets A and B when vertex i is considered in the for-loop. For the purpose of analysis, we delay the random decisions in G until a vertex is considered. In particular, we delay the choice of which of $1, 2, \ldots, i-1$ is a neighbor until

i is vertex i is considered. Note that no edge needs to be considered twice, and so we can treat each one as an independent biased coin flip.

Let a_i and b_i be the neighbors of i in A_i and B_i respectively. We can show that $|a_i - b_i| = \max(a_i, b_i) - \frac{1}{2}(a_i + b_i)$, and so $\sum_i |a_i - b_i|$ is the gain our algorithm achieves over cutting half the edges.

Now $|a_i - b_i|$ has expectation $\Omega(\sqrt{pi})$ and variance $O(pi)$. Adding over all i, the sum of the differences has mean $\Omega(n\sqrt{pn})$ and variance $O(pn^2)$. This gives us an expected gain of $\Omega(n\sqrt{pn}) = \Omega(n\sqrt{d})$ with $1 - o(1)$ probability. The value of cutting half the edges is approximately $dn/4$. This gives a final value of $dn/4 + \Omega(n\sqrt{d})$ w.h.p. as stated.

2.3 Certifiable upper bound

Again, we will derive a certifiable upper bound by looking at the spectral norm. If $(S, V-S)$ is a cut with value $\frac{dn}{4} + C$, then we have

$$\mathbf{1}_S^T A \mathbf{1}_{V-S} = \frac{dn}{4} + C$$

$$\mathbf{1}_S^T pJ \mathbf{1}_{V-S} = p \cdot |S| \cdot |V - S| \leq p \cdot \frac{n^2}{4} = \frac{dn}{4}$$

$$\|\mathbf{1}_S\| \cdot \|\mathbf{1}_{V-S}\| = \sqrt{|S| \cdot |V - S|} \leq \sqrt{\frac{n^2}{4}}$$

so

$$C \leq 2n \cdot \|\mathbf{1}_S\| \cdot \|\mathbf{1}_{V-S}\|.$$

This means that, in every graph, the maximum cut is upper bounded by

$$\frac{dn}{4} + \frac{n}{2} \left\| A - \frac{d}{n} J \right\|$$

which if $d \geq \log n$ is with high probability at most $\frac{dn}{4} + O(n\sqrt{d})$ (by Lemma 1).

3 Conclusion

We conclude with the following table, which summarizes our results for a random graph sampled from $G_{n,d/n}$.

Problem	**Expected Value**	**Greedy Algorithm**	**Certifiable Upper Bound**
Independent Set	$O\left(\frac{n}{d} \log d\right)$	$\Omega\left(\frac{n}{d} \log d\right)$ w.h.p.	$O\left(\frac{n}{\sqrt{d}}\right)$ w.h.p.*
Max Cut	$\frac{dn}{4} + O(n\sqrt{d})$	$\frac{dn}{4} + \Omega(n\sqrt{d})$ w.h.p.	$\frac{dn}{4} + O(n\sqrt{d})$ w.h.p.*

* Note that both certifiable upper bounds require $d \geq \log n$.

Both greedy algorithms perform very well in comparison to the probable value. In Max Cut, our greedy algorithm is particularly strong, matching our certifiable upper bound up to a

lower order term. This supports one of our major theses: while greedy algorithms exhibit poor worst-case performance, they tend to do well over our given distribution.