Random Graphs — Assignment 2

Yuval Filmus

January 8, 2020

Question 1. Let X_k be the number of k-cliques in $G(n,1/2)$. Recall that

$$N_k := \mathbb{E}[X_k] = \binom{n}{k} 2^{-\binom{k}{2}}$$

and

$$\frac{\mathbb{E}[X_k^2]}{\mathbb{E}[X_k]^2} = \sum_{\ell=0}^{k} \frac{\binom{k}{\ell}\binom{n-k}{k-\ell}}{\binom{n}{k}} 2^{\binom{\ell}{2}} =: \sum_{\ell=0}^{k} J_\ell.$$

Let k_0 be the maximal k such that $N_k \geq 1$, and recall that $k_0 = 2\log_2 n - 2\log_2\log_2 n + O(1)$. The goal of this exercise is to show that with high probability, $G(n,1/2)$ contains a k-clique, for $k = k_0 - 1$.

(a) Calculate $J_{\ell+1}/J_\ell$.

Answer. We have

$$\frac{J_{\ell+1}}{J_\ell} = \frac{\binom{k}{\ell+1}}{\binom{k}{\ell}} \frac{\binom{n-k}{k-\ell-1}}{\binom{n-k}{k-\ell}} 2^{\binom{\ell+1}{2}-\binom{\ell}{2}} = \frac{(k-\ell)^2}{(\ell+1)(n-2k+\ell+1)} 2^\ell. \qquad \square$$

(b) Calculate $(J_{\ell+2}/J_{\ell+1})/(J_{\ell+1}/J_\ell)$.

Answer. We have

$$\frac{J_{\ell+2}}{J_{\ell+1}} \Big/ \frac{J_{\ell+1}}{J_\ell} = \frac{(k-\ell-1)^2}{(k-\ell)^2} \cdot \frac{\ell+1}{\ell+2} \cdot \frac{n-2k+\ell+1}{n-2k+\ell+2} 2^{\ell+1-\ell} =$$
$$\left(1 - \frac{1}{k-\ell}\right)^2 \left(1 - \frac{1}{\ell+2}\right)\left(1 - \frac{1}{n-2k+\ell+2}\right) \cdot 2. \qquad \square$$

(c) Show that there exists N_1 such that if $n \geq N_1$ and $0 \leq \ell \leq \frac{1}{2}\log_2 n$ then $J_{\ell+1} < J_\ell$.

Answer. If $\ell \leq \frac{1}{2}\log_2 n$ then

$$\frac{J_{\ell+1}}{J_\ell} \leq \frac{k^2}{n-2k}\sqrt{n} = O\left(\frac{\log^2 n}{\sqrt{n}}\right).$$

Hence for large enough n, $J_{\ell+1}/J_\ell < 1$. $\square$

(d) Show that there exists N_2 such that if $n \geq N_2$ and $k > \ell \geq \frac{3}{2}\log_2 n$ then $J_{\ell+1} > J_\ell$.

Answer. If $\ell \geq \frac{3}{2}\log_2 n$ then

$$\frac{J_{\ell+1}}{J_\ell} \geq \frac{1}{k(n-k)} n^{3/2} = \Omega\left(\frac{\sqrt{n}}{\log n}\right).$$

Hence for large enough n, $J_{\ell+1}/J_\ell > 1$. $\square$

(e) Show that there exists N_3 such that if $n \geq N_3$ then the sequence $J_{\ell+1}/J_\ell$ is increasing for $\frac{1}{2}\log_2 n \leq \ell \leq \frac{3}{2}\log_2 n$.

Answer. If $\frac{1}{2}\log_2 n \leq \ell \leq \frac{3}{2}\log_2 n$ then

$$\frac{J_{\ell+2}}{J_{\ell+1}} \Big/ \frac{J_{\ell+1}}{J_\ell} \geq$$
$$\left(1 - \frac{1}{(1-o(1))2\log_2 n - \frac{3}{2}\log_2 n}\right)^2 \left(1 - \frac{1}{\frac{1}{2}\log_2 n + 2}\right)\left(1 - \frac{1}{n - O(\log n)}\right) \cdot 2 =$$
$$2 - o(1).$$

Hence for large enough n, the double ratio is strictly larger than 1. $\square$

(f) Deduce that for $N \geq \max(N_1, N_2, N_3)$ the sequence $J_0, \ldots, J_k$ is unimodal (decreasing and then increasing, with perhaps two identical values at the middle).

Answer. Suppose that $N \geq \max(N_1, N_2, N_3)$, and let $\rho_\ell = J_{\ell+1}/J_\ell$. The foregoing shows that $\rho_0, \ldots, \rho_{\frac{1}{2}\log_2 n} < 1$, $\rho_{\frac{3}{2}\log_2 n}, \ldots, \rho_{k-1} > 1$, and the sequence $\rho_{\frac{1}{2}\log_2 n}, \ldots, \rho_{\frac{3}{2}\log_2 n}$ is increasing. This means that for some $\frac{1}{2}\log_2 n \leq \ell_0 < \frac{3}{2}\log_2 n$, it holds that $\rho_\ell \leq 1$ if $\ell \leq \ell_0$ while $\rho_\ell \geq 1$ if $\ell > \ell_0$, which is what we wanted to show. $\square$

(g) Deduce that for $N \geq \max(N_1, N_2, N_3)$ the maximum of $J_1, \ldots, J_k$ is attained at one of the endpoints.

Answer. Suppose that $N \geq \max(N_1, N_2, N_3)$. Since $J_0, \ldots, J_k$ is unimodal, the maximum of *any* subsequence is attained at an endpoint. $\square$

(h) Conclude that $\mathbb{E}[X_k^2]/\mathbb{E}[X_k]^2 = 1 + o(1)$, and so with high probability $G(n, 1/2)$ contains a k-clique.

Answer. We can assume that $N \geq \max(N_1, N_2, N_3)$. In the lecture notes, we have shown that $J_1, J_k = o(1/\log n)$. Therefore $J_1 + \cdots + J_k = o(1)$. Since $J_0 = 1 - o(1)$, it follows that $J_0 + \cdots + J_k = 1 + o(1)$. $\square$

Question 2. In this exercise, we will empirically explore algorithms for finding cliques in $G(n, 1/2)$.

(a) Consider the heuristic which constructs a clique by iteratively picking an arbitrary vertex which is connected to all vertices chosen so far. How large a clique does this heuristic find, when $n = 1000$?

Answer. I get roughly 9.7 on average, with a standard deviation of roughly 0.76. The largest clique found in 10^5 experiments was 14, and the smallest 7. $\square$

(b) Modify this heuristic by always picking a vertex of maximal degree. Does this improve on the original heuristic when $n = 1000$?

Answer. I get roughly 11.3 on average, with a standard deviation of 3.49. The largest clique found in 10^5 experiments was 26, and the smallest 7. $\square$

(c) Optional: Come up with a better heuristic.

In both cases, I suggest performing at least 10^4 experiments. In each experiment, generate a $G(n, 1/2)$ random graph, run the heuristic, and record the result. After running all experiments, report the average, using two significant digits.

Due to speed considerations, I recommend using a compiled language like C/C++/Java rather than an interepreted language such as Python/Matlab.

Note. Both experiments were run on the same graphs. C code is attached in the appendix. $\square$

Question 3. Let $p_{n,k}$ be the probability that the heuristic in Question 2(a) constructs a clique of size k when run on $G(n, 1/2)$.

(a) Write a recurrence relation for $p_{n,k}$.

Answer. The base case is $p_{0,0} = 1$ and $p_{0,k} = 0$ for $k \neq 0$. There are two ways to get a clique of size k at time n: either we had a clique of size k at time $n-1$ and the new vertex was not connected to the k clique vertices (which happens with probability $1 - 2^{-k}$), or we had a clique of size $k-1$ at time $n-1$ and the new vertex was connected to the $k-1$ clique vertices (which happens with probability $2^{-(k-1)}$). This leads to the recurrence

$$p_{n,k} = (1 - 2^{-k})p_{n-1,k} + 2^{-(k-1)}p_{n-1,k-1}.$$ $\square$

(b) Compute the expected size of the clique when $n = 1000$ exactly (but display the result as a decimal). You can use a computer algebra system such as Mathematica, Maple or Sage, or a library such as `libgmp`, which supports multi-precision integer or floating point arithmetic.

Answer. The expected number of vertices in the clique is 9.69399833091716, and the standard deviation is 0.760935117399371. $\square$

Here is sample code in Sage:

```
def distribution(n):
    p = [1]
    for m in range(1,n+1):
        p = [0] + [(1 - 2^(-k)) * p[k] + 2^(-(k-1)) * p[k-1] for k in range(1,m)] + [p[m-1] * 2^(-(m-1))]
    return p
RR(sum(p * x for (p,x) in enumerate(distribution(1000))))
```

Question 4 (Bonus)**.** Let $H := \rhd\!\!-$. In the last assignment we calculated the probability that $G(n, c/n)$ contains no copy of H. Now we calculate the entire distribution:

$$p_k := \lim_{n\to\infty} \Pr[G(n, c/n) \text{ contains exactly } k \text{ copies of } H].$$

Say that a triangle has *type* $t = (a, b, c)$ if $0 \le a \le b \le c$ and the degrees of vertices in the triangle are $2 + a, 2 + b, 2 + c$ (that is, the triangle has a edges dangling from one vertex, b edges from another, and c from the remaining vertex). Let $|t| = a + b + c$.

Fix an arbitrary ordering on types. For a vector $\tau = (t_1, \ldots, t_m)$ of non-decreasing types, define

$$q_\tau = \lim_{n\to\infty} \Pr[G(n, c/n) \text{ contains exactly } m \text{ triangles, of types } t_1, \ldots, t_m].$$

Let $|\tau| = \sum_{i=1}^m |t_i|$.

(a) Show that with high probability, any two copies of H in $G(n, c/n)$ are either disjoint or share a triangle.

Answer. Let H_1, H_2 be two non-vertex-disjoint copies of H, which intersect at a subgraph K. The density of $H_1 \cup H_2$ is

$$\frac{e(H_1) + e(H_2) - e(K)}{v(H_1) + v(H_2) - v(K)} = \frac{8 - d(K)v(K)}{8 - v(K)},$$

where $d(K) = e(K)/v(K)$. If $d(K) < 1$ then the density of $H_1 \cup H_2$ is larger than 1, and so with high probability it doesn't appear in $G(n, c/n)$. The only proper subgraph K of H with $d(K) \ge 1$ is the triangle. $\square$

(b) Show that

$$p_k = \sum_{|\tau|=k} q_\tau.$$

(Hint: use part (a) and modify the argument of Question 3 in Assignment 1.)

Answer. For every τ such that $|\tau| = k$, if $G(n, c/n)$ contains exactly the triangles described by τ, then it has exactly k copies of H. Furthermore, these events are disjoint for different τ. This shows that for every *finite* collection T of such τ,

$$p_k \ge \sum_{\tau\in T} q_\tau.$$

This shows, in particular, that $q_k := \sum_{|\tau|=k} q_\tau$ converges. For every $\epsilon > 0$ we can find a finite set T such that $\sum_{\tau\in T} q_\tau \ge q_k - \epsilon$, and so for every $\epsilon > 0$, $p_k \ge q_k - \epsilon$. It follows that $p_k \ge q_k$.

In the other direction, let T_m be the set of all vectors τ such that $|\tau| = k$ and τ contains at most m types; note that T_m is finite. If $G(n, c/n)$ contains exactly k copies of H then it either contains two non-vertex-disjoint copies of H which don't intersect in a triangle (which happens with probability $o(1)$), or more than m triangles (which happens with some limiting probability ϵ_m), or conforms to one of the vectors in T_m. Hence

$$p_k \leq \sum_{\tau \in T_m} q_\tau + \epsilon_m.$$

In the previous assignment we showed that $\epsilon_m \to 0$. Since $\sum_{\tau \in T_m} q_\tau \to q_k$, it follows that $p_k \leq q_k$. □

(c) Calculate q_τ.

Answer. Let $s_1 \prec \cdots \prec s_\ell$ be the different types appearing in τ, say s_i appears r_i times. The number of choices for the triangles (with ordered vertices) is

$$\sim \frac{n^{3\sum_i r_i}}{\prod_i r_i!}.$$

For a triangle of type $s_i = (a_i, b_i, c_i)$, let $\sigma_i = 1$ if all of a_i, b_i, c_i are different, $\sigma_i = 2$ if two are identical, and $\sigma_i = 3$ if all are identical. The number of choices for the edges emanating from the triangles is

$$\sim \prod_i (n^{a_i+b_i+c_i}/\sigma_i)^{r_i}.$$

The probability for this particular choice is

$$\sim (c/n)^{\sum_i (3+a_i+b_i+c_i)r_i} \cdot \left(1 - \frac{c}{n}\right)^{3n\sum_i r_i} \cdot e^{-c^3/6} \sim (c/n)^{\sum_i (3+a_i+b_i+c_i)r_i} e^{-3c\sum_i r_i - c^3/6}.$$

In total,

$$q_\tau = \frac{c^{3\sum_i r_i}}{\prod_i r_i!} \cdot \prod_i (e^{-3c}/\sigma_i)^{r_i} \cdot e^{-c^3/6} = \prod_i \frac{(c^3 e^{-3c}/\sigma_i)^{r_i}}{r_i!} \cdot e^{-c^3/6}.$$ □

(d) Let τ be a vector not including any isolated triangles. Calculate z_τ, which is the sum of $q_{\tau'}$ over all vectors τ' obtained from τ by adding an arbitrary number of isolated triangles.

Answer. Using the notation of the previous item, if we add k isolated triangles then there is an additional factor of

$$\frac{(c^3 e^{-3c}/6)^k}{k!}.$$

Summing this over all k, we obtain

$$e^{c^3 e^{-3c}/6}.$$

Therefore

$$z_\tau = \frac{c^{3\sum_i r_i}}{\prod_i r_i!} \cdot \prod_i (e^{-3c}/\sigma_i)^{r_i} \cdot e^{-c^3/6} = \prod_i \frac{(c^3 e^{-3c}/\sigma_i)^{r_i}}{r_i!} \cdot e^{-(1-e^{-3c})c^3/6}.$$ □

(e) Calculate p_0, p_1, p_2.

Answer. Let $z = e^{-(1-e^{-3c})c^3/6}$. Then

$$\begin{aligned} z_{(0,0,1)} &= \frac{c^3 e^{-3c}}{2} \cdot z, \\ z_{(0,0,1),(0,0,1)} &= \frac{c^6 e^{-6c}}{8} \cdot z, \\ z_{(0,0,2)} &= \frac{c^6 e^{-6c}}{2} \cdot z. \end{aligned}$$

Therefore

$$\begin{aligned} p_0 &= e^{-(1-e^{-3c})c^3/6}, \\ p_1 &= e^{-(1-e^{-3c})c^3/6} \cdot \frac{1}{2} c^3 e^{-3c}, \\ p_2 &= e^{-(1-e^{-3c})c^3/6} \cdot \frac{5}{8} (c^3 e^{-3c})^2. \end{aligned}$$

□

Sample code for Question 2.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXN (1000)

int graph[MAXN][MAXN] = {0};

void randomize(int n) {
  for (int i = 0; i < n; i++) {
    graph[i][i] = 0;
    for (int j = 0; j < i; j++)
      graph[i][j] = graph[j][i] = random() & 1;
  }
}

int clique[MAXN];

int random_cliquer(int n) {
  int nclique = 0, v = 0, i;

  clique[nclique++] = v++;

  while (v < n) {
    for (i = 0; i < nclique; i++)
      if (graph[i][v] == 0)
        break;
    if (i == nclique)
      clique[nclique++] = v++;
    else
      v++;
  }

  return nclique;
}

int degree[MAXN];

int greedy_cliquer(int n) {
  int nclique = 0;

  for (int i = 0; i < n; i++) {
    degree[i] = 0;
    for (int j = 0; j < n; j++)
      degree[i] += graph[i][j];
  }

  while (nclique < n) {
    int best = -1, bestv = -1;
    for (int v = 0; v < n; v++) {
      int i;
      for (i = 0; i < nclique; i++)
        if (graph[i][v] == 0)
          break;
      if (i < nclique)
        continue;
      if (degree[v] > best)
        bestv = v;
    }
    if (bestv == -1)
      return nclique;
    clique[nclique++] = bestv;
  }
  return nclique;
}

int histogram[MAXN+1] = {0};
```

```c
int main(int argc, char *argv[]) {
  if (argc < 3) {
    fprintf(stderr, "usage: %s n trials [random/greedy]\n", *argv);
    exit(1);
  }

  int n = atoi(*++argv);
  int trials = atoi(*++argv);
  int (*cliquer)(int) = random_cliquer;
  if (*++argv != 0) {
    if (strcmp(*argv, "random") == 0)
      cliquer = random_cliquer;
    else if (strcmp(*argv, "greedy") == 0)
      cliquer = greedy_cliquer;
    else {
      fprintf(stderr, "unknown heuristic %s\n", *argv);
      exit(2);
    }
  }

  if (n > MAXN) {
    fprintf(stderr, "n > %d\n", MAXN);
    exit(3);
  }

  srandom(1);

  int total = 0, total2 = 0;
  for (int i = 0; i < trials; i++) {
    randomize(n);
    int clique = cliquer(n);
    total += clique;
    total2 += clique*clique;
    histogram[clique]++;
  }
  double average = (double)total / trials;
  double average2 = (double)total2 / trials;
  printf("mean %.1f std %.2f\n", average, average2 - average*average);

  int min = 0;
  while (histogram[min] == 0)
    min++;
  int max = n;
  while (histogram[max] == 0)
    max--;

  for (int i = min; i <= max; i++)
    printf("size %2d frequency %.3f\n", i, (double)histogram[i] / trials);
}
```