12 Lecture 12, November 10, 2014

12.1 Concentration of the number of prime factors: Turan's proof of a theorem of Hardy and Ramanujan

Now for an application of near-pairwise independence in number theory.

Let $m(k)$ be the number of primes dividing k. Hardy and Ramanujan showed that for large k this number is almost always close to $\log\log k$. Specifically, let $k \in_U [n]$, and let M be the rv

$$M = m(k).$$

Theorem 53

$$\Pr(|M - \log\log k| > \lambda\sqrt{\log\log k}) < \frac{1+o(1)}{\lambda^2}$$

We show an elegant later proof of this due to Turan.

Proof: Before we begin the proof in earnest let's simplify things. Observe that the function $\log\log$, besides being monotone, is so slowly growing that it hardly distinguishes between n and $\sqrt{n}$. Specifically, $\log\log n = \log 2 + \log\log\sqrt{n}$, so we can write:

$$\begin{aligned}
& \Pr(|M - \log\log k| > \lambda\sqrt{\log\log k}) \\
\leq\ & \Pr(k \leq \sqrt{n}) \\
& + \Pr(|M - \log\log n| + \log 2 > \lambda(\sqrt{\log\log n - \log 2})) \\
\leq\ & \frac{1}{\sqrt{n}} + \Pr(|M - \log\log n| > (1-o(1))\lambda\sqrt{\log\log n})
\end{aligned}$$

The theorem will follow, therefore, from showing:

Proposition 54

$$\Pr(|M - \log\log n| > \lambda\sqrt{\log\log n}) < \frac{1+o(1)}{\lambda^2}.$$

Now to show the proposition. For prime p let $[\![p|k]\!]$ be the indicator for p dividing k. Note $M = \sum_p [\![p|k]\!]$.

$$E([\![p|k]\!]) \ = \ \lfloor n/p \rfloor / n$$

So

$$1/p - 1/n \leq E([\![p|k]\!]) \leq 1/p$$

$$-1 + \sum_{\text{prime } p \leq n} \frac{1}{p} \leq E(M) \leq \sum_{\text{prime } p \leq n} \frac{1}{p} \tag{32}$$

For $k \geq 1$ let $\pi(k) = |\{p : p \leq k, p \text{ prime}\}|$. We remind ourselves of the

Theorem 55 (Prime number theorem) $\pi(k) \in (1+o(1))k/\log k$.

We use the following corollary (proof omitted in class):

Lemma 56 $\sum_{prime\ p \leq n} 1/p \in (1+o(1))\log\log n$.

So, from Eqn. 32 and Lemma 56 we know that

$$E(M) \in (1 + o(1)) \log\log n.$$

Now for the variance of M. The proposition will follow from showing

$$\mathrm{Var}(M) = (1 + o(1)) \log\log n \tag{33}$$

and an application of the Chebyshev inequality. So the remainder of the argument is to show Eqn (33).

As always we can write

$$\mathrm{Var}(M) = \sum_{\text{prime } p \le n} \mathrm{Var}([\![p|k]\!]) + \sum_{\text{primes } p \ne q \le n} \mathrm{Cov}([\![p|k]\!], [\![q|k]\!])$$

The key is that we are in a very nearly pairwise-independent situation, because the sum of covariances is very small.

We already noted in a previous lecture that for $\{0,1\}$-valued rvs Y, $\mathrm{Var}(Y) = E(Y)(1 - E(Y)) \le E(Y)$. Applying this we have

$$\sum_{\substack{\text{prime} \\ p \le n}} \mathrm{Var}([\![p|k]\!]) \le \sum_{\substack{\text{prime} \\ p \le n}} E([\![p|k]\!]) \in (1 + o(1)) \log\log n.$$

Now to handle the covariances. Observe that for primes $p \ne q$, $[\![p|k]\!][\![q|k]\!]$ is the indicator rv $[\![pq|k]\!]$. Just as for primes, $E([\![pq|k]\!]) = \lfloor \frac{n}{pq} \rfloor / n \le \frac{1}{pq}$. So

$$\begin{aligned}
\mathrm{Cov}([\![p|k]\!], [\![q|k]\!]) &= E([\![pq|k]\!]) - E([\![p|k]\!])E([\![q|k]\!]) \\
&\le \frac{1}{pq} - \left(\frac{1}{p} - \frac{1}{n}\right)\left(\frac{1}{q} - \frac{1}{n}\right) \\
&\le \frac{1}{n}\left(\frac{1}{p} + \frac{1}{q}\right)
\end{aligned}$$

This is a very low covariance, which is crucial to the theorem.

$$\begin{aligned}
\sum_{\substack{\text{primes} \\ p \ne q \le n}} \mathrm{Cov}([\![p|k]\!], [\![q|k]\!]) &\le \sum_{\substack{\text{primes} \\ p \ne q \le n}} \frac{1}{n}\left(\frac{1}{p} + \frac{1}{q}\right) \\
&= (1 + o(1)) \frac{2}{n} \pi(n) \sum_{\substack{\text{primes} \\ p \le n}} \frac{1}{p} \quad \text{prime number thm} \\
&= (1 + o(1)) \frac{2}{n} \pi(n) \log\log n \quad \text{Lemma (56)} \\
&= (1 + o(1)) \frac{2 \log\log n}{\log n}
\end{aligned}$$

So the first term of $\mathrm{Var}(M)$ is by far the dominant one and we have established Eqn (33). □

12.2 Lower tail bound on random walk using 4th moment. Application to Gale-Berlekamp

In an earlier lecture we used a strong hammer, the CLT, to conclude that the value of the Gale-Berlekamp game is $\Omega(n^{3/2})$. Specifically we applied the CLT to show that for a symmetric random walk of length n, $X = \sum_1^n X_i$ with $X_i \in_U \{1, -1\}$, $E(|X|) \in \Omega(n^{1/2})$. Now we will show this from first principles—and more importantly, using only information about the 2nd and 4th moments.

This is not only of methodological interest. It makes the conclusion more robust, specifically the conclusion holds for any 4-wise independent space, and therefore implies a poly-time deterministic algorithm to find a G-B solution of value $\Omega(n^{3/2})$, as we will return to discuss.

Theorem 57 *Let $X = \sum_1^n X_i$ where the X_i are 4-wise independent and $X_i \in_U \{1, -1\}$. Then $E(|X|) \in \Omega(n^{1/2})$.*

Proof: We start with two calculations. These calculations are made easy by the fact that for any product of the form $X_{i_1}^{b_1} \cdots X_{i_4}^{b_4}$, with $i_1, \ldots, i_4$ distinct and $b_i \geq 0$ integer,

$$E(X_{i_1}^{b_1} \cdots X_{i_4}^{b_4}) = \begin{cases} 0 \text{ if any } b \text{ is odd} \\ 1 \text{ otherwise} \end{cases}$$

So now

$$E(X^2) = \sum_{i,j} E(X_i X_j) = \sum_i E(X_i^2) = n$$

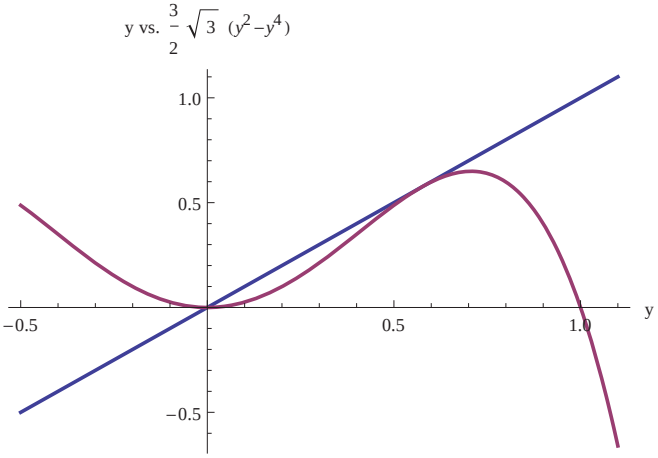
$$E(X^4) = 3\sum_{i,j} E(X_i^2 X_j^2) - 2\sum E(X_i^4) = 3n^2 - 2n.$$

One is tempted to apply Chebyschev's inequality to the rv X^2, because we know both its expectation and its variance. Unfortunately, the numbers are not favorable! $\mathrm{Var}(X^2) = 3n^2 - 2n - n^2 = 2n^2 - 2n > n^2 = E(X^2)^2$.

But there is something more clever we can do. We start with an elementary inequality.

Lemma 58 *For all $a, x > 0$, $\frac{3^{3/2}}{2a}(x^2 - x^4/a^2) \leq x$.*

Proof: Write $y = x/a$ and this becomes the claim $y < \frac{3^{3/2}}{2}(y^2 - y^4)$ for $y > 0$. The quartic has simple roots at $-2/\sqrt{3}$, 0, and a double root at $1/\sqrt{3}$. See Fig. 12.2. □



Consequently for any $a > 0$,

$$\begin{aligned} E(|X|) &\geq \frac{3^{3/2}}{2a} E(X^2 - X^4/a^2) \\ &= \frac{3^{3/2}}{2a}\left(n - \frac{3n^2 - 2n}{a^2}\right) \end{aligned}$$

Using $a = 3\sqrt{n}$ this is

$$\begin{aligned} &= \frac{\sqrt{3}}{2\sqrt{n}}\left(n - \frac{3n-2}{9}\right) \\ &\geq \sqrt{n/3} \end{aligned}$$

12.3 Hashing

Hashing is an old and important primitive for data structures and also for communication problems. It is simple to accomplish using pairwise independence. This is our second application of the fact that a sample space is small, the first being the reproof of the Shannon coding theorem.

The general situation we consider is that there are some n keys in the set $[u]$, but with $u \gg n$. So these are long names for the keys, and although we want to store some records associated with the keys, we can't afford to allocate an array of size u in memory. Instead we need a quick way of mapping a key in u down to an array index in $[m]$, for m "not much larger" than n, while avoiding collisions between the records.

It turns out that this is not too much easier than simply achieving low probability of collision between any two keys, as formalized by the following definition.

Let H be a family of functions $H : [u] \to [m]$. (More formally, we consider a sequence of such families as $u, m \to \infty$.) Say that H is a *weak univeral family* if for all $x \neq y \in [u], \Pr_{h \in_U H}(h(x) = h(y)) \leq O(1/m)$.

Obviously the family of *all* functions from u to m, also called $[m]^{[u]}$, is such a family (with precisely $1/m$ collision probability, which makes it a "strong" universal family). But it is much too large. We need to have concise descriptions of the functions h. This is the same problem we encountered in the first proof of Shannon's coding theorem.

So better to use the following construction. Let us suppose that m is prime. (Usually we don't have a precise value m in mind anyway; the real parameters to the problem are u and n, and we simply want an array that is not much larger than n. So pick a prime m that is a bit bigger than n. Say, less than $10n$.)

Now, convert any key κ, $0 \leq \kappa < u$, into a vector over the $\mathbb{Z}/m$: $\kappa = (\kappa_0, \ldots, \kappa_r)$ where $r \leq \log_m u$.

The family H consists of all functions $h_{h_0,\ldots,h_r}$ where $h_i \in \mathbb{Z}/m$ and

$$h_{h_0,\ldots,h_r}(\kappa_0, \ldots, \kappa_r) = \sum h_i \kappa_i \bmod m$$

Suppose $\kappa \neq \kappa'$. Then

$$\Pr_{h \in_U H}(h(\kappa) = h(\kappa')) = \Pr(\sum h_i(\kappa_i - \kappa_i') = 0 \bmod m) = \frac{1}{m}$$

This is actually strongly universal, and the hash family is of size only $m^{r+1} \leq m^{1+\log_m u} = um \in O(un)$, which is far less than m^u. The functions in the family are of course also very easy to apply. All seems nice but we still haven't looked at what happens to all keys at once. There are n of them, and the number of pairs that might collide is $\binom{n}{2}$. The expected number of collisions in this scheme is what you get from the birthday paradox:

$$E(\#\text{ collisions}) = \binom{n}{2}\frac{1}{m} \leq \frac{n-1}{2}.$$

That seems like a lot of collisions. You might be tempted to think we should have taken m quadratic in n, but there is a better solution. We do not need a quadratic size storage array, linear will suffice.

The idea, due to Fredman, Komlós and Szemerédi [24] (and see improvements [17]), is to use a secondary hashing scheme to resolve collisions.

Begin by picking h uniformly from the above family. For each $z \in [m]$, let R_z =number of keys mapping to z. The total number of collisions is $C = \sum_z \binom{R_z}{2}$; as above, $E(C) \leq \frac{n-1}{2}$. If $C > n$ pick h again; otherwise continue. The expected number of attempts is at most two.

For each z, link the register at position z to a register of size $m_z = 2\binom{R_z}{2}$. Now construct, in the same manner as above, a hash function $f_z : [u] \to [m_z]$. In this case, the expected number of collisions is $1/2$, so by repeated sampling we can get an f_z with no collisions (and again the expected number of attempts is at most two). We store its label in the main register at location z.

The total time to construct this mapping is $O(n)$. The total number of registers required for storage is $\leq 3n$, if we use $m = n$; this can't quite be done, necessarily, with m prime, so one may want to use slight

variations of this idea. However as noted earlier, in practice there are yet other aspects of the design to be optimized, and these have been studied with great care. We do not pursue the topic further here. One reason, besides lack of time, is that there are newer hashing methods that have been adopted due to some practical advantages and their ability to handle dynamic updates fairly naturally. Specifically *cuckoo hashing* [55] and see [49, 43], all starting from an insight on "the power of two choices" [6].