Lecture 17: Balls & bins (contd.)

Plan/outline

Last class, we started looking at hashing, using a simple abstraction of throwing balls into bins. We tried to ask various questions about the distribution of the bin sizes. This time we'll continue this line of reasoning. Some important tools like the linearity of expectation (which we also saw last time) and the union bound will feature in the analysis.

Throwing n balls into m bins at random

Recall the setting from last class: we have n balls that are thrown randomly and independently into m bins. We would like to understand different properties about how "evenly" the balls are distributed. We asked:

- What is the expected size (number of balls) of each bin?
- Suppose $n = m$. Then what is the expected number of bins with say precisely 4 balls?
- Again assuming $n = m$, what is the probability that *some* bin gets $\log n$ balls?

Using the linearity of expectation, we saw that the expected size of each bin is precisely n/m, as one can expect. We also saw that the answer to the second question (for n large enough) is roughly $n/65$. The main trick in the analyses of both these statements was to define the appropriate random variables, express them as a sum of "simpler", binary random variables, and work with their expectated values (which is easier to compute).

Let us now answer the final question above. If we fix a particular bin and ask what's the probability that it gets $\log n$ balls, it's the computation we did last time. For any i, the probability that bin i receives k balls is exactly: $\binom{n}{k}\frac{1}{n^k}\left(1-\frac{1}{n}\right)^{n-k}$.

Like last time, let us use $\left(1-\frac{1}{n}\right)^n \approx 1/e$, and a well known consequence of what is known as Stirling's approximation.

Approximation for $\binom{n}{k}$. Stirling's approximation is an asymptotic formula for $n!$. Using this, one can obtain a commonly used approximation for the binomial coefficient. Specifically, we get: $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$.

Plugging both of these into the expression above, we have: Pr[bin i receives k balls] $\leq \left(\frac{ne}{k}\right)^k \frac{1}{e} \frac{1}{(n-1)^k}$.

Using the fact that when $k \ll n$, $(n/(n-1))^k \approx 1$ (and definitely less than e), we obtain an upper bound of $\left(\frac{e}{k}\right)^k$.

We can then plug in $k = \log n$ to get an upper bound on the probability that bin i receives $\log n$ balls (and this is true for any i). Now, how do we argue about the probability that *some* ball receives $\log n$ balls? A general (and very simple) fact that helps us show such bounds is the so-called **union bound**.

Theorem. (Union bound) Let $E_1, E_2, \ldots, E_n$ be events in a probability space. Then we have $\Pr[E_1 \vee E_2 \vee \cdots \vee E_n] \leq \Pr[E_1] + \Pr[E_2] + \cdots + \Pr[E_n]$.

In the above, for two events A and B, $A \vee B$ is the "OR" of the two events, or the "union" of the two events, i.e., it occurs if either A or B occurs (or both).

The union bound is shown easily for the case of two events, and the version above follows by induction.

Also, equality in the union bound holds if and only if the events E_i are all disjoint. In fact, the union bound is a special case of another well-known formula, known as the **inclusion/exclusion formula**. For more on these, here's a good and short note. (Pay close attention to Corollary 3 in the note). For a longer note with other connections, see this.

Going back to our example, we obtained a bound on a specific bin i receving k balls. Let us call this event E_i. Then by definition, the probability that *some* ball receives k balls is precisely $\Pr[E_1 \vee E_2 \vee \cdots \vee E_n]$ (as we have $m = n$ bins). Thus by the union bound, the probability is at most n times the probability computed earlier, which is $n \cdot \left(\frac{e}{k}\right)^k$.

Plugging in $k = \log n$ and writing $k = e^{\log\log n}$, the above simplifies to $\frac{n^2}{n^{\log\log n}}$, which is tiny when n is large enough.

In fact, as an algebra exercise, one can verify that $n(e/k)^k$ starts becoming $\ll 1$ as we choose $k \geq c \cdot \log n / \log\log n$, for a constant c.

A second proof

Another proof suggested in class is the following; suppose we show that the probability of bin i receiving k balls is $\leq (e/k)^k$, as above. Now, what is the *expected number* of bins that receive k balls? (similar to what we studied last time with $k = 4$)

Let Y denote the random variable which is the number of bins receiving k balls. The nice thing is that the probability that there exists a bin with k balls is equivalent to the probability that $Y \geq 1$. Now, expressing Y as a sum of n binary variables like last time, and by the linearity of expectation, $\mathbb{E}[Y] \leq n(e/k)^k$. Now, we can use Markov's inequality. Note that we can re-state it as $\Pr[Y \geq \alpha] \leq \mathbb{E}[Y]/\alpha$.

Plugging in $\alpha = 1$, we get $\Pr[Y \geq 1] \leq \mathbb{E}[Y] \leq n(e/k)^k$.

This is precisely the same bound as above, but without using the Union bound. (In simple settings such as this, it's often possible to obtain multiple proofs, and you should try coming up with new ones!)

Conclusions -- balls and bins

Let us summarize what we learned about the balls and bins process, specifically in the case of $n = m$ (so we are throwing n balls into n bins uniformly at random).

The expected size (#balls) in each bin is precisely 1.

That said, not all bins have size exactly 1, there is a distribution. Furthermore, the sizes of the bins are **correlated** (to be precise, *anti-correlated*, i.e., if one bin happens to get a lot of balls, the others will likely get fewer balls, because the total number is fixed).

The number of bins of size k is at most $n(e/k)^k$, as we saw. [This is a "fairly tight" upper bound; we had a more precise bound earlier.] For $k = 4$, the expected number of bins of size k is $\approx n/65$. What one should note is that a constant fraction of the bins receive k balls, for any constant k.

We also computed the probability that there *exists* a bin of size $\log n$ and found it to be very small. In fact, for balls and bins with $m = n$, it turns out that the **maximum bin size** is, with high probability, $O\left(\frac{\log n}{\log \log n}\right)$. This bound is also tight; with high probability, there exists a bin that gets roughly $\log n / \log \log n$ balls.

Consequences for hashing. We noted earlier that the balls and bins process is an idealized way to study hashing. Assuming that a hash function is perfectly random, its behavior would correspond to that of balls and bins.

The bound above for the "max size" implies that if we were to use hashing in order to do element lookup (more precisely, suppose we're using a hash map to store a set of values, and that we perform linear probing to do lookup), then the *max lookup time* can be as large as $\log n / \log \log n$. Another scenario where this problem surfaces is the following: suppose we have n servers that can process queries, and suppose we have (roughly) n queries arriving one by one. Each time a query arrives, suppose we assign it to a random server. Then the above analysis shows that some server receives $\log n / \log \log n$ queries, and it turns out that a constant fraction of servers don't receive any queries! So random assignment is much more "lopsided" than say round-robin. [Round-robin has other problems in many applications: suppose a new server joins the "server pool" half way into the process. Then round-robin requires effort to "adapt", while the random process can go on unchanged.]

In the query assignment situation above, consider the following simple modification: instead of assigning a query to a random server, say we find *two* random servers, h_1 and h_2, and we assign the query to the **server with the smaller load** (of the two). Then it turns out that the max load is only $\log \log n$ --- considerably smaller than the earlier bound.

This trick can also be used in hash tables. We have two hash functions h_1 and h_2, and when inserting an element x, we look at the bins $h_1(x)$ and $h_2(x)$ and insert x into into the *smaller bin*. During lookup, we simply go through both the bins to check for x. We now obtain a query time that is $2 \cdot \log \log n$ (because we potentially go through two bins) in the worst case.

This phenomenon is often known as the **power of two choices**, discovered in around 1992. Proving the $\log \log n$ bound turns out to be rather tricky. There are many writeups that give a proof. Two that I liked are here and here.

Next class, we will study **sampling**, one of the other most common applications of probabilistic algorithms.