1. Probabilistic Methods

Two obvious facts that we will use over and over again are the following: If an event has positive probability of occurring, it’s possible for it to occur. If the expectation of a random variable, \( X \), is \( \mathbb{E}(X) \), the value of \( X \) will sometimes be at least \( \mathbb{E}(X) \) and sometimes at most \( \mathbb{E}(X) \). An immediate corollary to the first obvious fact is that if an event has probability less than 1 of occurring, it’s possible for it to not occur. The goal when looking at probabilistic methods is to use these two facts in every way possible. The first example we will consider is the question of the existence of \( k \)-balanced tournaments.

1.1. Balanced Tournaments.

**Definition 1.** A tournament is a directed graph on some vertex set, \( V \), so that for each pair of \( x, y \in V \), exactly one directed edge exists between them.

This can be thought of as a round-robin tournament, where every player plays every other player, and there are no draws. Below are two examples of tournaments on three vertices, or three players.

- ![Tournament example 1](image1.png)
- ![Tournament example 2](image2.png)

**Definition 2.** A tournament is \( k \)-balanced if for every subset \( S \) of \( V \) with \( |S| = k \), there is a vertex, \( x \), such that all the edges from \( S \) to \( x \) are directed away from \( x \).

This can be thought of as for any given set of \( k \) players, there is always at least one player that beats all of them. It can be quickly seen from the definition that a \( k \)-balanced tournament will also be \( k-1 \)-balanced. In the above examples, the first tournament is not 1-balanced because there is a player that no other player beats. The second tournament, however, is 1-balanced, but it is not 2-balanced.

Using standard techniques, the question of whether a \( k \)-balanced tournament can exist would come to a lot of casework and/or a clever usage of symmetry to construct and then check. If we take \( |V| = n = \) the number of players it is possible to show the existence of \( k \)-balanced tournaments for any given \( k \), provided \( n \) is large enough. The idea is to construct the tournament, \( T \), at random and show the probability it is \( k \)-balanced is positive.

First we need to decide how to randomly construct a tournament. Let \( (x,y) \) denote the edge directed from \( x \) to \( y \), i.e. \( x \) beats \( y \), for \( x, y \in V \). For every pair, \( \{x,y\} \), of vertices, let the edge be directed \( (x,y) \) with probability \( \frac{1}{2} \), and directed \( (y,x) \) with probability \( \frac{1}{2} \). Each pair, or each game, is thus independent of the others. In the three vertex tournaments above, there is probability \( \frac{1}{8} \) of either of them happening, as there are eight equally likely possibilities for directing the edges.

Instead of considering a lower bound on the probability of a \( k \)-balanced tournament directly, consider an upper bound on the probability of a tournament being not \( k \)-balanced. For a set, \( S \), with \( |S| = k \), we say it is “bad” if there is no player that beats everyone in \( S \). Abusing notation, we refer to this as there being no vertex with \( (x,S) \). For individual vertices, the probability of this happening is \( \frac{1}{2^k} \). What makes this work
For any two random variables, \( X_1 \) and \( X_2 \), \( \text{E}(X_1 + X_2) = \text{E}(X_1) + \text{E}(X_2) \).

Notice that there are no restrictions on \( X_1 \) and \( X_2 \). They can be dependent, independent, whatever. The “sophisticated-looking” proof of this from measure theory is essentially that the integral of a sum is the sum of the integrals, since an expectation, or the average value, of a variable is the integral of that variable with respect to the probability measure.

So here what we’re going to do is we’re going to say, for any edge, let \( f(e) = \begin{cases} 1 & \text{the edge is splits} \\ 0 & \text{else} \end{cases} \)
We know that $E(f(e)) = \frac{1}{2}$. So by linearity of expectation, the expected number of edges cut by a random cut is $\frac{1}{2}|e|$. And if the average split cuts half the edges, there must be a split that cuts half the edges.

1.3. Conditional Expectations. As it turns out, probability isn’t really necessary here. What we could have done instead was a sort of greedy algorithm: Place the vertices in an arbitrary order $\{v_1, \ldots, v_n\}$, and then place each $v_i$ one by one. At each step, always place $v_i$ in the set which cuts more of the edges between $v_i$ and $\{v_1, \ldots, v_{i-1}\}$.

This is a special case of what is termed a ”conditional expectation” argument. The idea is as follows: Imagine the random process as a sort of decision tree. First we decide whether to place vertex 1 in $A$ or $B$, then vertex 2 in $A$ or $B$, and so on. If all the steps are completely random, then on average we cut half the edges.

Now every time we have to make a decision, we always choose to move down the branch of the tree with higher conditional expectation given what we’ve done so far, assuming all future decisions are random.

Example: Suppose that vertex $v_1$ has already been placed in $A$, and vertex $v_2$ (the second to be placed) is adjacent to vertex 1. If we place $v_2$ in $A$, then the expected number of cut edges is $\frac{d-1}{2}$ (all edges but the $v_1v_2$ edge stand a $1/2$ chance of being cut, but the $v_1v_2$ edge is uncut). If we place $v_2$ in $B$, the expected number of cut edges if $\frac{d+1}{2}$ (same as before, but now $v_1v_2$ is cut), so we place $v_2$ in $B$.

In general, the conditional expectations argument behaves exactly like the greedy algorithm here. The point here is that the expectation was initially half the edges, and can only increase each time we make a decision (we always move to the better branch). At the end of the algorithm, the conditional expectation exactly equals the number of edges cut (there’s no randomness left), so we must have cut half the edges.

In theory, the same derandomization works for the random tournament example, but there’s a catch. Say I’m trying to find a 50-balanced tournament. Then to compute the conditional expectation I need to use to make my decision, I need to keep track of $\binom{n}{50}$ different sets, which is hard computationally.

2. Dominating Sets

So our philosophy so far has I guess been along the lines of, okay, if I want to make the set work, choose a random set, see if it works. If I want to see if a tournament works, choose a random tournament, see if it works. Let’s look at something that maybe a little bit more complicated. Again we’re going to start with a definition.

Definition 3. Given a graph, $G$, we call a subset, $S$, of vertices dominating if every vertex on the graph is either in $S$ or adjacent to something in $S$.

Intuitively, think about $S$ as sort of being here a set of sensors. A sensor can only see so far; it can only see things adjacent to it. A dominating set is a set of sensors that can see everything on the graph. Sensors are expensive; small dominating sets are good. I mean, if I took every vertex in the graph, that would be a dominating set, but that would be an expensive one. Now maybe, okay maybe in some cases I do need to take every vertex. If my original graph was the empty graph, then I can’t do anything better than just taking every vertex. But somehow the feeling is, if the graph has a lot of edges, if all the vertices have a lot of neighbors, we should be able to do better.

So a question. If we’re given a $d$, and we know that $G$ is $d$-regular on $n$ vertices, but I don’t know anything about $G$ other than that it’s $d$-regular. I want to know what can I say about the smallest dominating set.

So here’s a quick lower bound. $\frac{n}{d+1}$. To see this, let’s say I place my sensors one by one. I place my first sensor down, mark my first vertex in the graph, and it can see exactly $d+1$ vertices. Let’s
use better language: So far \(d+1\) vertices have been covered. After two sensors have been placed, there’s at most \(2(d+1)\) covered vertices. Not equal anymore, because maybe some of those \(d+1\) vertices were already seen by the first sensor. In the absolute best case for us, we still only got \(2(d+1)\) vertices covered. So if I place \(k\) sensors, there’s at most \(k(d+1)\) covered, and I only stop when \(n\) vertices covered. So \(k\) had better be at least \(\frac{n}{d+1}\). [After each placement, at most \(d+1\) new vertices are covered.]

But in general, I’m not going to be able to get \(\frac{n}{d+1}\), because that’s only if I can miraculously cover the graph perfectly. And I don’t know if I can cover the graph perfectly. I don’t know anything at all about \(G\). But it’s sort of the same way as before: I want my sensors sort of spread out over the graph so they don’t overlap too much. So maybe I should try placing the sensors at random.

There’s a couple of ways you can think about placing the sensors at random. You can think about it as saying, I’m going to pick \(k\) vertices on the graph, uniformly over all sets of size \(k\). That’s a little annoying, actually, to work with because there is not much independence when we do it that way (if I know that vertex \(x\) got a sensor, that makes it a tiny bit less likely that each other vertex gets a sensor). And I guess sort of an intuition is that when in doubt, try to maximize independence. It makes computations a lot nicer. So my random sensor placement, or my first try at random sensor placement, I’m going to place a sensor at each vertex with probability \(p\). Independently. I haven’t said what \(p\) is. \(p\) is something I’m going to pick later on in the argument, once I see what the bounds look like in terms of \(p\). Or if you want it in terms of the original language, probability that \(x \in S\) is \(p\) for every \(x\).

So given any \(x\) the probability that \(x\) is uncovered is, just \((1-p)^{d+1}\) because I’ve got my \(x\) here, I’ve got my \(d\) neighbors, and the only time that \(x\) is uncovered is if all \(d+1\) of these vertices were not chosen (this sort of calculation is why you maximize independence!). The first thought might be to choose \(p\) so that with pretty high probability nothing is uncovered. But there’s a problem here. The problem is that if \(n\) is large, I need \(p\) pretty close to 1 before these events are all unlikely. If \(d\) is small and \(n >> d\), then I’ve got a bunch of areas of the graph where I could fail. And if every area of the graph fails with probability 1% and there’s enough areas of the graph, I’m probably going to fail. So if I want to make sure I don’t fail, \(p\) had better be really close to 1. But if \(p\) is really close to 1, I’ve essentially picked everything. And from an efficiency point of view, that’s awful. So this sort of simple pick a set at random, hope it works, is not the way to go about here. But maybe there’s a way we can improve this.

Suppose that \(d\) is large and \(p\) not too small. Then most vertices are probably going to be covered (the probability that any vertex is uncovered is small). So say I’ve already covered 99.9% of the vertices in my graph. At this point I don’t want to pick vertices at random anymore, because a randomly placed sensor stands a good chance of missing all of the vertices I still have left to cover. There’s no point in placing a sensor where I can already see everything!

So instead, let’s do a two-step process. Step 1 is going to be the same as before. I’m going to place each vertex in \(S\) with probability \(p\). And again, I still haven’t decided what \(p\) is yet. At this point, we still have some subset \(S'\) of the vertices which is uncovered. For step 2, I’ll just be lazy and stick all of \(S'\) in \(S\).

Step 2 is not perfectly efficient, if you think about it. It might be that I have two adjacent vertices, and I stick both of them in when I could have just used one. But we already didn’t know much about \(G\) beyond its regularity, and with all the randomness we’ve done already we really have almost no understanding of what \(S'\) looks like and what vertices may or may not be adjacent. Besides, if we pick \(p\) right then \(S'\) will only be a very tiny set. And if we only have 1% of the graph left, who cares if we’re not too efficient on that tiny fraction? I just want to make sure I have a dominating set. And by construction, we’ve found one. The real question is, is it small?

Well let’s see. The expected size of \(S\) after step 1 is just \(np\). \(n\) vertices, I include each one with probability \(p\). In step 2, well, for any given vertex, the probability I need to add that vertex to \(S\) in step 2 is just: \((1-p)^{d+1}\).
So the expected number of added vertices is \( n(1 - p)^{d+1} \). By linearity of expectation, the expected size of \( S \) after step 2 is \( n(p + (1 - p)^{d+1}) \).

The key thing here is that, again, there must be a dominating set at least as small as the average from our construction. And this is true no matter what \( p \) is. So you should think about this as sort of a large family of bounds on the size of the smallest dominating set. We can just choose the value of \( p \) that gives us the best bound. Here’s a bound which isn’t the tightest possible, but looks nice and is pretty close for large \( d \).

What we use is the basic calculus identity

\[
1 + x \leq e^x
\]

This bound will show up a LOT in the future, because it’s so useful in handling expressions like \((1 - p)^{d+1}\).

In our case it tells us

\[
\mathbb{E}(|S|) \leq n(p + e^{-p(d+1)})
\]

To optimize over \( p \), I take the derivative, \( 1 - (d+1)e^{-p(d+1)} = 0 \). Solve it, you get \( p = \frac{\ln(d+1)}{d+1} \). Plugging this in to our upper bound, we get that every \( d \)-regular graph has a dominating set of size at most \( n(1 + \frac{\ln(d+1)}{d+1}) \).

As it turns out, for large \( d \) I believe this bound is essentially the correct bound (up to a constant factor at worst). I haven’t actually seen the full proof of this, but there is a reference to it given in the Alon and Spencer Probabilistic Methods book that I linked on the iLearn.

So questions on any of this?

**Question:** Is there a derandomization method where you kind of order the vertices by their valencies and you start picking them up in that order?

**Answer:** Probably. I’d have to actually work through what would happen, but it’s essentially a question question of how large the expected valency is at each step. Sooner or later you’re not going to be able to pick vertices of valency \( d + 1 \) any more, but if the average valency of the graph is large enough, then, the bound might work.

### 3. Sum-Free Sets

Here’s one more example of this sort of just pick the right probability space and everything pops out type of proof. Here, actually, the big trick is figuring out what probability space to work with. Once again we start with a definition

**Definition 4.** A set \( A \) is sum-free if there are no solutions to \( x + y = z \) with \( x, y, z \in A \).

So, examples of sum-free sets are the odd numbers. Or all all the numbers from 11 up to 21. So a question - I’m not sure who originally asked it - given a set \( B \), does it necessarily contain (I’m going to be vague here) a large sum-free \( A \)? So you give me a \( B \), I need to find \( A \) in it that is both large and sum-free.

**Theorem 4.** (Erdős, 1965) Any subset \( B \) of integers contains a sum-free \( A \) with \( |A| > \frac{|B|}{3} \).

For many \( B \) you can find sum-free subsets which are much larger. For example, if half the numbers in \( B \) are odd, and I’ll just say, take the odd numbers and be done. But we don’t know anything about \( B \) here. The proof sort of relies on two really nifty tricks. One trick is to work in, well, I’m not going to work in the integers any more, I’m going to work in the integers mod \( p \), where \( p \) is prime and \( p \geq n \), the largest member of \( B \). It’s harder to be sum-free mod \( p \) than it is to be sum-free in general. So if I can find a sum-free set mod \( p \), I can certainly find a sum-free set in general.

And the idea is that in \( \mathbb{Z}_p \), there is one very large sum-free set. Specifically, if I take the \( x \) such that \( \frac{p}{3} \leq x < \frac{2p}{3} \). Because if I take two things in the lower part of this set, the sum is bigger than \( \frac{2p}{3} \), and if I take...
two things in the upper part, the sum isn’t large enough to wrap back around to \(\frac{p}{3}\). So if \(\frac{1}{3}\) of the numbers in my set happen to fall in this range, I’m done. But there’s no reason to expect me to be that lucky.

Here’s where the second trick comes in. What we’re going to do is we’re going to look at the set \(m \cdot B\), where \(m\) is random in \(\mathbb{Z}_p^*\) (here the notation \(m \cdot B\) means take each element of \(B\) and multiply it by \(m\)). So it’s sort of a completely different probability space to what we’ve seen so far. Instead of taking random things in \(B\) to put in \(A\), I’m taking \(B\) and I’m multiplying it by a random number. And I’m going to say, let \(S_m\) be the \(x \in B\) such that \(\frac{p}{3} \leq mx < \frac{2p}{3}\) mod \(p\).

My claim is that \(S_m\) is sum-free, because if \(x, y, z \in S_m\) and \(x + y = z\), well that tells me that \(mx + my = mz\). And \(mx, my,\) and \(mz\) are all in this \(\frac{p}{3}\) to \(\frac{2p}{3}\) range. So I have a whole collection of these \(S_m\)s, and if I can show that any of them are large, I’m done. Let \(x_{i,m}\) equal 1 if \(i \in S_m\), and 0 otherwise. We have

\[
\sum_m |S_m| = \sum_m \sum_{i \in B} x_{i,m} = \sum_{i \in B} \sum_m x_{i,m} = \sum_{i \in B} \left\lceil \frac{p}{3} \right\rceil = |B| \left\lceil \frac{p}{3} \right\rceil
\]

There are \(p - 1\) terms in the sum, so some \(S_m\) must have size larger than \(|B|/3\).

Essentially the hard part is coming up with the idea of working in a field and the set \(S_m\).

4. Splitting Vectors Evenly

To continue with the idea of constructing a distribution with a small/large expectation to show a variable can be made small/large, let’s consider another example: Given \(n\) unit vectors, \(x_1, \ldots, x_n\) in \(\mathbb{R}^m\), can we divide the \(x_i\) into two sets, \(A\) and \(B\) so that \(\sum_{x_i \in A} x_i \approx \sum_{x_i \in B} x_i\)? More precisely, we wish to minimize the Euclidean norm, \(|\sum_{x_i \in A} x_i - \sum_{x_i \in B} x_i|\). Algorithmically this is a very hard problem, even if \(x_1, \ldots, x_n\) are just real numbers. So instead of trying to get the best possible bound for each set, let’s try and get some sort of upper bound which holds for all sets.

A bad example, in the sense that there is no good split, would be taking \(m > n\) and \(x_1, \ldots, x_n\) mutually orthogonal. By the Pythagorean theorem, \(|\sum_{x_i \in A} x_i - \sum_{x_i \in B} x_i| = \sqrt{n}\). So knowing nothing about the dimension of the space or directions the vectors point, there are some cases where the minimum is at least \(\sqrt{n}\).

Claim 1. There is always a split such that \(|\sum_{x_i \in A} x_i - \sum_{x_i \in B} x_i| \leq \sqrt{n}\)

Together with what we have already observed, this would give a sharp upper bound on the minimum distance. To prove this we will consider a random partition with each vector assigned independently to \(A\) or \(B\) with probabilities \(p(x_i \in A) = p(x_i \in B) = \frac{1}{2}\). Then \(\sum_{x_i \in A} x_i - \sum_{x_i \in B} x_i = |\sum_{i=1}^n \epsilon_i x_i|\), where \(\epsilon_i \in \{-1, 1\}\) with equal probability of being 1 or \(-1\). The idea again is that if we can show this sum is on average small, there must be a partition which makes it small. However, it is very inconvenient to work directly with the expectation because the square roots get ugly to deal with. To compensate, let \(Y = |\sum_{i=1}^n \epsilon_i x_i|^2\). It’s enough
to show that \( E(Y) \) is small, and now we have

\[
E(Y) = E(\sum_i \epsilon_i x_i, \sum_i \epsilon_i x_i) = E(\sum_i \sum_j \epsilon_i \epsilon_j \langle x_i, x_j \rangle) = \sum_i \sum_j E(\epsilon_i \epsilon_j) \langle x_i, x_j \rangle = \sum_i 1 \langle x_i, x_i \rangle = n.
\]

Here we used that inner products and expectations are both linear, that 

\[
E(\epsilon_i \epsilon_j) = \begin{cases} 
0 & i \neq j \\
1 & i = j 
\end{cases}
\]

by the independence of \( \epsilon_i \) and \( \epsilon_i^2 = 1 \), and that \( x_i \) is a unit vector.

No matter what the vectors are, the expected square of the distance is the same: \( n \). In the one extreme when all the vectors are orthogonal, the distance is always \( \sqrt{n} \). In the other extreme, if all vectors are in the same direction, this is saying that the expected square distance of a random walk from 0 after \( n \) steps is \( n \). So there must be a partition where the square distance is at most this large.

The tricks here involved building a random partition, squaring the distances, and exploiting the fact that the expectation of a squared sum is the expectation of a double sum, from which we can use linearity to make things work nicely.

5. The First Moment Method (Markov’s Inequality) and Balls in Bins

Another way to use probabilistic methods is to consider a random process instead of inserting randomness into the calculations of deterministic processes. Now we want to understand how the process “usually” behaves, in the sense that often there is a parameter \( n \) that the probability might depend on. In the above example, the number of vectors could be increased, perhaps. Our hope is that the randomness becomes less random in the limit.

An event is said to happen “almost surely” if its probability approaches 1. If a coin is flipped \( n \) times, almost surely there will be at least one time it lands on heads. Assuming the law of large numbers, we could also say that almost surely there will be around \( \frac{n}{2} \) heads. Maybe for some random variable, \( x \), one might wish to show \( \frac{\mathbb{E}(x)}{\mathbb{E}(x)} \to 1 \) almost surely. Taking \( x \) to be the number of heads, and \( f(x) = \frac{n}{x} \), we might try to show this happens as \( n \) tends to \( \infty \), or at least that it’s unlikely for that ratio to be larger than 1.5 as \( n \to \infty \). And then how unlikely is unlikely?

Example: Given \( m \) balls, \( n \) bins, \( n \) large, where each ball is tossed independently and randomly into a bin (that is, \( p(\text{ball } i \text{ in bin } j) = \frac{1}{n} \)). There are many things we might consider. For example, you might care about load-balancing issues: How many balls are in the most crowded bin? For this example we will instead consider the ”coupon collector’s problem”: how likely is it that all the bins have a ball? This will clearly depend on \( m \), as \( m < n \) immediately means there will certainly be some empty bins, whereas if \( m >> n \), it is intuitively very unlikely there will be any empty bins. There should be a boundary between these extremes, so to reword the question, how large does \( m \) have to be before it becomes likely all \( n \) bins are full? An easier question is, what is \( p(\text{bin } 1 \text{ empty}) \)? \( p(\text{bin } 1 \text{ empty}) = (1 - \frac{1}{n})^m \).
Let $X$ be the number of empty bins.

$$X = x_1 + \cdots + x_n : x_i = \begin{cases} 1 & \text{if bin } i \text{ empty} \\ 0 & \text{else} \end{cases}.$$ 

Thus $E(X) = \sum_{i=1}^n E(x_i) = n \left( 1 - \frac{1}{n} \right)^m$. We can asymptotically analyse this expectation to see when it is large or small, but we need to be able to say two things before we can translate from the asymptotics on $E(X)$ to asymptotics on $X$. Recall that what we really care about is $p(X = 0)$. If $E(X)$ is small, does that mean $X = 0$, or at least is small most of the time? If $E(X)$ is large, does that mean $X$ is large most of the time?

An example that was used during the Cold War to illustrate this difficulty was, let $X$ equal the number of people dead in the next decade due to nuclear war. $p(X = 0)$ is very close to 1 due to the very low probability of nuclear war actually breaking out, but $E(X)$ could be fairly significant due to the very large number of people that would die to nuclear war if nuclear war did happen. So $E(X)$ can be large despite $X$ being small most of the time.

This nuclear war example shows $E(X)$ large $\neq X$ large, however, $E(X)$ small $\Rightarrow X$ small if we add one extra criterion.

**Theorem 5.** Markov’s Inequality: If $X$ is a non-negative random variable, then $p(X \geq \lambda E(X)) \leq \frac{1}{\lambda}$, or with a slight change of variables, $p(X \geq \lambda) \leq \frac{E(X)}{\lambda}$.

The non-negative assumption is necessary, as there can be big positive and negative values that cancel out if that assumption is not made. The $\epsilon_i$ indicator variables in the previous example demonstrate that concept: $E(\epsilon_i) = 0$, and Markov’s inequality clearly fails.

**Proof.** Define a new variable, $Y$, where

$$Y = \begin{cases} \lambda & \text{if } X \geq \lambda \\ 0 & \text{else} \end{cases}.$$ 

By the construction of $Y$, $X \geq Y$, so it follows that $E(X) \geq E(Y) = \lambda \cdot p(X \geq \lambda)$. Because $\lambda > 0$, it is trivial to obtain the second form of Markov’s inequality. □

A special case that frequently shows up in combinatorics is when $X$ is a non-negative integer-valued random variable, in which case $p(X > 0) = p(X \geq 1)$. Applying Markov’s inequality, we get $p(X > 0) \leq E(X)$. If a random variable can be defined in such a way that its expectation tends to zero, the variable will almost always equal zero, if it is integer-valued.

Returning to the balls and bins, when will the expectation for at least one empty bin, found above, tend to zero? If we take $m = n \cdot \ln n + t$, where $t$ is an as yet undetermined variable, then

$$E(X) = n \left( 1 - \frac{1}{n} \right)^{n \ln n + t} \leq n \cdot e^{(-n \ln n + t)/n} = e^{-t/n},$$

where we used the inequality $1 - x \leq e^{-x}$. So if $m = n \ln n + t$ and $\frac{t}{n} \rightarrow \infty$, then $E(X) \rightarrow 0$, so $P(\text{all bins have a ball}) \rightarrow 1$.

The short version of this argument is, find the expected number of empty bins and determine what $m$ has to be for the expected number of empty bins to tend to zero. That gives an upper bound for how large $m$ has to be before the bins are almost surely full. The other half of what we want to argue here is that if
$m << n \ln n$, the number of empty bins is probably bigger than zero. This is the implication that initially is not true just by looking at the expectation. It is true that the expected number of empty bins tends to $\infty$, but an argument is still needed to show that the number of empty bins is close to the expectation: that it is not 0 with large probability, yet huge with tiny probability. The usual trick used to make this argument is considering the second moment instead of just the first moment. In the language of statistics, the mean no longer suffices, so we consider the variance.

6. The Second Moment, and Chebyshev’s Inequality

We define the **Variance** $\text{Var}(X)$ of a variable $X$ to be $\text{Var}(X) = \mathbb{E}((X - \mathbb{E}(X))^2) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$. Other notation frequently used for variance and expectation is $\text{Var}(X) := \sigma^2$ and $\mathbb{E}(X) = \mu$.

The intuition is that if the variance is small, $x$ is usually close to its expectation, $\mathbb{E}(x)$. A way to make this intuition precise is by using another inequality, called Chebyshev’s Inequality. It basically says that most of the time a random variable will be within a small number of standard deviations of the mean.

**Theorem 6.** (Chebyshev’s Inequality:) $\mathbb{P}(|X - \mu| \geq \lambda \sigma) \leq \frac{1}{\lambda^2}$

**Proof.** Let $Y = (X - \mu)^2$. Then $\mathbb{P}((X - \mu) \geq \lambda \sigma) = \mathbb{P}((x - \mu)^2 \geq \lambda^2 \sigma^2)$ using the same sort of trick as above with the distances squared. By Markov’s inequality, applied to $Y$, $\mathbb{P}(x - \mu)^2 \geq \lambda^2 \sigma^2) \leq \frac{\mathbb{E}((x - \mu)^2)}{\lambda^2 \sigma^2} = \frac{1}{\lambda^2}$, since $\mathbb{E}((x - \mu)^2) = \mathbb{E}((x - \mathbb{E}(x))^2) = \text{Var}(x) = \sigma^2$.

A special case of Chebyshev’s inequality that shows up frequently in practice: if $\sigma << \mu$, then with high probability $X \approx \mu$. More precisely, if $\frac{\sigma}{\mu} \rightarrow 0$, then $\mathbb{P}((1 - \epsilon)\mu \leq X \leq (1 + \epsilon)\mu) \rightarrow 1$ for any $\epsilon$. We can see this by plugging $\epsilon \mu$ into Chebyshev’s inequality and noting that $\lambda$ tends to $\infty$, so the probability it is not in this range tends to 0.

To show a random variable is usually large, one way is to first show the expectation is large, and then show the variance is small. That means the variable is usually close to its expectation, which was already shown to be large. Unfortunately, as can be seen below, second moment calculations are usually a bit messier than first moment calculations.

7. Back to Balls in Bins

Returning to balls and bins, where we have the same random variable as before and the same expectation as before, but now we care about the expectation of $X^2$. The following is one way to calculate $\mathbb{E}(X^2)$ that may not be the most efficient, but is the kind of calculation that shows up frequently.
\[
E(X - E(X))^2 = E \left( \sum_{i=1}^{n} x_i - E(x_i) \right)^2 \quad \text{(by linearity)}
\]

\[
= E \left( \sum_i \sum_j [x_i - E(x_i)][x_j - E(x_j)] \right)
\]

\[
= \sum_{i=1}^{n} E([x_i - E(x_i)]^2) + \sum_{i \neq j} E([x_i - E(x_i)][x_j - E(x_j)])
\]

\[
= n \cdot E([x_1 - E(x_1)]^2) + (n^2 - n) \cdot E([x_1 - E(x_1)][x_2 - E(x_2)]),
\]

by symmetry. Although this keeps getting messier, we need to keep in mind that we don’t care about the exact value of the variance so much as an upper bound that shows it is much smaller than the mean. With that in mind, consider the two terms separately, and bound each one of them. In the first term, \(E((x_i - E(x_i))^2) = \text{Var}(x_1)\). Because \(x_1\) is an indicator variable, it is either 0 or 1, and \(x_1 = 1\) with probability \(p = (1 - \frac{1}{n})^m\), so

\[
\text{Var}(x_1) = p(1 - p) \leq p.
\]

This upper bound should be close to the actual amount whenever \(p\) is close to zero. Thus the first term is at most \(n \cdot (1 - \frac{1}{n})^m\). For the second term,

\[
E((x_1 - E(x_1))(x_2 - E(x_2))) = E(x_1x_2 - x_1E(x_2) - E(x_1)x_2 + E(x_1)E(x_2))
\]

\[
= E(x_1x_2) - E(x_1)E(x_2) - E(x_1)x_2 + E(x_1)E(x_2)
\]

\[
= E(x_1x_2) - E(x_1)E(x_2) - E(x_1)x_2 + E(x_1)E(x_2)
\]

\[
= E(x_1x_2) - E(x_1)E(x_2),
\]

where \(E(x_1x_2) = P(\text{bins 1,2 empty}) = (1 - \frac{2}{n})^m\). So \(E(x_1x_2) - E(x_1)E(x_2) = (1 - \frac{2}{n})^m - (1 - \frac{1}{n})^{2m}\). We cannot use our usual asymptotic trick of replacing each of these with exponentials, since they will be the same exponential in both cases. Consider, then, the lower order terms:

\[
E(x_1x_2) - E(x_1)E(x_2) = \left(1 - \frac{2}{n}\right)^m - \left(1 - \frac{2}{n} + \frac{1}{n^2}\right)^m \leq \sum_{k \geq 1} \left(1 - \frac{2}{n}\right)^{m-k} \left(\frac{1}{n^2}\right)^k \binom{m}{k}.
\]

This is an upper bound because aside from the terms that cancelled immediately, all the terms are taken here to be positive. Since \(\binom{m}{k} \leq m^k\), the covariance is at most

\[
\sum_{k \geq 1} \left(1 - \frac{2}{n}\right)^{m-k} \left(\frac{m}{n^2}\right)^k.
\]

We know what range of \(m\) to consider, since if \(m > n \ln n\) we already know there will probably be no empty bins. If \(m \leq n \ln n\), this sum is dominated by the \(k = 1\) term, so the sum is \((1 + o(1)) \cdot \left(\frac{m}{n^2}\right)(1 - \frac{2}{n})^m\).

Combining everything, the first term is at most \(n \cdot (1 - \frac{1}{n})^m\), and the second term is bounded by \(n \cdot (1 - \frac{1}{n})^m\). The whole thing is

\[
\text{Var}(X) \leq n \left(1 - \frac{1}{n}\right)^m + (n^2 - n) \frac{m}{n^2} \left(1 - \frac{2}{n}\right)^m (1 + o(1)) \leq ne^{-m/n} + me^{-2m/n}(1 + o(1)).
\]
Since the mean is $ne^{-m/n}$, we can now apply Chebyshev’s inequality:

$$p(X = 0) \leq p(|X - \mu| \geq \mu) \leq \frac{\sigma^2}{\mu^2} \leq \frac{ne^{-m/n} + me^{-2m/n}}{n^2e^{-2m/n}} = \frac{e^{m/n}}{n} + \frac{m}{n^2}.$$ 

We take $m = n \ln n - t$, so $P(X = 0) \leq \frac{e^{\ln n - t/n}}{n} + \frac{n \ln n}{n^2} = e^{-t/n} + \frac{\ln n}{n},$

which tends to 0 as $t$ tends to infinity. In summary, by the first moment, if we take $t >> n$ and put $n \ln n + t$ balls in $n$ bins: $p($no empty bins$) \to 1$.

By the second moment, if we put $n \ln n - t$ balls in $n$ bins: $p($no empty bins$) \to 0$.

In other words, there is a sharp threshold. If the balls are placed in bins one at a time, we expect the last bin to be filled to get its first ball after about $n \ln n$ balls. That is, randomly dropping the balls costs the efficiency a factor of $\ln n$. A similar argument can be made to show the threshold for the number of balls in the most full bin when filling $n$ bins randomly with $n$ balls is about $\frac{\log n}{\log \log n}$.

The idea to take away here is this two-stage argument. We’re trying to find the threshold for $m$ where the bins became full. One side is really easy; look at the first moment and show the expectation is small. The other side was a bit of a pain. We looked at the expectation of the square and showed that it was not too big. In practice, sometimes the expectation of the square is really long and captures all of the work. You can find 20-25 page papers where one page has the first moment calculations, and the rest is to estimate the variance.

Next week we will look at a few more examples, and then look for a better method. Chebyshev’s inequality only falls off as $\frac{1}{\lambda^2}$, whereas something like the normal distribution falls off exponentially. We should be able to get that exponential kind of tail if things behave nicely enough.

---

1There is an interesting tradeoff in efficiency when filling bins randomly versus checking each bin for an empty bin to fill. The latter involves checking a large number of bins every time a new ball arrives (to see which bin has the fewest balls), while the former leads to a (slow, but growing) cluster of balls in the same bin. An intermediate kind of bin filling protocol would be to have each ball choose two bins randomly, going into the least full of the two. With that protocol, the most full bin is expected to only have about $\ln \ln n$ balls in it. For a fuller overview of this idea that two choices dramatically improves efficiency, take a look at a thing called “The Power of Two Choices.”