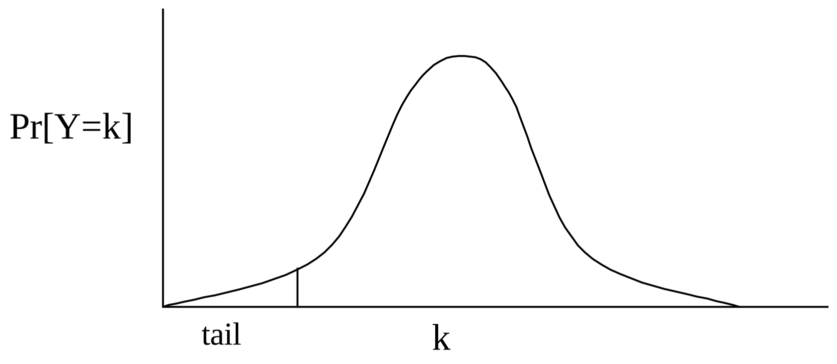
CS174 Spring 98 Lecture 6 Summary

Tail Bounds

Last time we looked at occupancy problems and derived some results on the *distribution* of some random variables. We derived bounds on the probability of a bin containing more than k balls, and the expected number of bins containing exactly k balls for various k. We had to do specific analysis for balls and bins. This lecture gives two bounds that work for any probability distribution. One (Markov) requires almost no knowledge about the distribution but gives very weak bounds. The other (Chebyshev) needs knowledge of standard deviation but gives good bounds.



The bounds we study are called tail bounds because they correspond to the area or total probability of tails under a probability distribution like the one above.

Markov Bounds

Let Y be a non-negative random variable, and t a positive number. The Markov bound is

$$\Pr[Y \geq t] \leq \frac{E[Y]}{t}$$

The bound doesn't depend on any knowledge of the distribution of Y, except that its non-negative. The proof is easy once the bound is rewritten as:

$$t \Pr[Y \geq t] \leq E[Y]$$

That inequality comes from the definition of E[Y]. E[Y] is the sum of all possible values of Y times the probability of each value, which we can write as

$$E[Y] = \sum_{k<t} k \Pr[Y = k] + \sum_{k \geq t} k \Pr[Y = k]$$

The first term is non-negative because Y is, so we can remove it and make the inequality:

$$E[Y] \geq \sum_{k \geq t} k \Pr[Y = k] \geq \sum_{k \geq t} t \Pr[Y = k]$$

And then the last term rewrites as

$$E[Y] \geq t \sum_{k \geq t} \Pr[Y = k] = t \Pr[Y \geq t]$$

Which is the inequality we wanted to prove.

Examples

The Markov bound is usually not very exciting. Consider placing n balls into n bins as per last lecture. Then the number of balls in any bin is a non-negative random variable, whose expected value is 1. Let Y be the number of balls in bin 1. E[Y] = 1, so by Markov

$$\Pr[Y \geq k] \leq \frac{1}{k}$$

Or if k=10, the probability of more than k balls is less than 0.1. But we know from last lecture that that probability is much smaller (about 10^{-6}). So the Markov is not a tight bound. That's not surprising, because it assumes nothing about the distribution of Y.

Chebyshev Bounds

Chebyshev bounds give us a lot more because they use more information about the distribution. Specifically, they use information about the standard deviation of the random variable. Remember that for a random variable Y, the variance V is defined as

$$V = E[(Y - \overline{Y})^2]$$

Where the expected value of Y is

$$\overline{Y} = E[Y]$$

And the standard deviation σ is defined as the square root of the variance.

$$s^2 = V$$

Then the Chebyshev bound for a random variable X with standard deviation σ is:

$$\Pr[| X - \overline{X} | \geq ts] \leq \frac{1}{t^2}$$

The proof is by defining

$$Y = (X - \overline{X})^2$$

And then applying the Markov bound to Y.

Example

In order to apply the Markov bound to the occupancy problem, we need to compute the variance of Y, the number of balls in bin 1. Let Y_j be an indicator random variable that is 1 if ball j goes into bin 1. Then

$$Y = \sum_{j=1}^{n} Y_j$$

If the variance of Y is Var[Y], and because the Y_js are independent:

$$Var[Y] = \sum_{j=1}^{n} Var[Y_j] = nVar[Y_j]$$

By lemma 3.4 (hopefully you saw this in M55), and

$$Var[Y_j] = E[(Y_j - \overline{Y}_j)^2] = \frac{1}{n^2} . \frac{n-1}{n} + \left(1 - \frac{1}{n}\right)^2 \frac{1}{n} = \frac{n^2 - n}{n^3}$$

Because the expected value of Y_j is 1/n. Therefore

$$Var[Y] = (n-1)/n$$

Which is very close to 1. By Chebyshev, if Y>k, then |Y-E[Y]|>k-1, so

$$\Pr[Y > k] = \Pr[|Y - \overline{Y}| > k-1] \leq \Pr[|Y - \overline{Y}| > (k-1)s\ ] \leq \frac{1}{(k-1)^2}$$

So Chebyshev gives us a bound that falls off as the inverse of the square of k.

Randomized Selection

Selection is the problem of finding an element of rank k from some set. The rank of an element x is k if there are k-1 elements in the set that are smaller than x. In practice you might do this for an unsorted array. If the array is sorted, it is of course trivial to find the element of rank k. You might have seen a linear-time selection algorithm in CS170, if not, get ready…

The idea is to randomly pick a "smallish" subset of elements from the set, and to sort them. Then you find a pair of elements a and b from the sorted subset such that [a,b] should contain the element you're looking for. You compare all the elements with [a,b] and create a new subset P of elements in that interval. By knowing how many elements are less than a or greater than b, you figure out x's rank m in P. Then sort P and pick the element in the m^{th} position.