Theorem 6 and the Kolmogorov 0-1 law

Let's remember that the Kolmogorov 0-1 law says that tail events (events that are not affected by flipping any finite set of bits) have probability 0 or 1 (when those bits are sampled independently).

Finiteness of T is not a tail event, yet we've shown an exact characterization of the distributions μ for which it has probability 1. Obviously there should be a connection.

One distinction is that for general μ, the edges under a vertex are not independent. However, the 0-1 law doesn't really depend upon the local choices being bits; mainly it depends on the distribution being an infinite product. So this is only a minor technical distinction. The only fine point is in case infinitely many bits are needed locally, which will be the case if μ has unbounded support, but the question we are posing now makes sense even if μ has bounded support.

So let μ have finite support. Then there are finitely many *potential vertices* L_n at level n. We can simulate our tree process by a different, percolation-type process [2] in which we randomly choose, for every vertex in L_n, how many of its potential children it is actually connected to. Obviously this process simulates the tree process but it also has more happening in it since there are lots of vertices and edges that are not connected to the root.

Let I_v be the event that vertex v belongs to an infinite component. Then

$[\![\text{Percolation-type graph has an infinite component}]\!] = \cup_v [\![I_v]\!]$.

The LHS is a tail event and so when it has probability 0, it forces all the events on the RHS (which are not tail events) to have probability 0. In particular, $\Pr(I_{\text{root}}) = 0$.

This gets a one-sided inequality, the same a.s.-finite side that we derived earlier from Borel-Cantelli #1. What about when we're in the parameter regime where the percolation graph a.s. has an infinite component? As we've seen, $\Pr(\text{LHS}) = 1$ does *not* force $\Pr(I_{\text{root}}) = 1$. However, what if, in a sample from the tree process, we get to examine more and more of the top levels of T? Do we get an increasingly accurate posterior of the probability that T is infinite?

Yes! This depends on another 0-1 law, Levy's, which generalizes Kolmorogov's. I hope to discuss this when (if) we get to the material on martingales.

Back to the complete NAND tree of depth n

Let's see how well random DFS performs here. Let T_n = worst-case expected number of leaves evaluated by Random DFS.

What is a little trickier here than for MAJ3 is that for the 0 output of a NAND gate, there is no "short certificate:" you really need to look at both inputs to certify this output value. This is unlike for MAJ3 where no matter the output, there is always some pair of inputs which are enough to certify the output value. On the other hand for the 1 output of a NAND gate, there is a short certificate of length just 1, and from this we will get some savings.

What we need to show is that the "no short certificate" problem diminishes with depth because the adversary cannot set up a distribution on inputs which forces us always toward this situation.

Formally we do this by keeping track not of one but of two expectations. For $b \in \{0,1\}$ let T_n^b = worst-case # leaves evaluated, on inputs which evaluate to b. Then

$$T_n^0 \leq 2T_{n-1}^1$$

because the only way for a NAND gate to evaluate to 0 is if both inputs evaluate to 1. The upper bound factor of 2 is only the trivial bound, but what is helpful is that the expression is in terms of

[2] I don't want to define percolation here but for those familiar with percolation, this is only slightly more general, in that the edges beneath a vertex are not necessarily independent of each other.

the T^1 side, where we *do* have a short certificate, because NAND is 1 if either input is a 0. So there we have the recursion:

$$T_n^1 \leq \frac{1}{2} \cdot T_{n-1}^0 + \frac{1}{2} \cdot (T_{n-1}^1 + T_{n-1}^0) = T_{n-1}^0 + \frac{1}{2} \cdot T_{n-1}^1$$

because, using Random DFS, there's probability at least half that we find a 0 input first. So we can write the vector recursion

$$\begin{pmatrix} T_n^0 \\ T_n^1 \end{pmatrix} \leq \begin{pmatrix} 0 & 2 \\ 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} T_{n-1}^0 \\ T_{n-1}^1 \end{pmatrix}$$

The eigenvalues of this matrix are $\frac{1 \pm \sqrt{33}}{4}$, and the larger in absolute value is $\frac{1+\sqrt{33}}{4}$, so we find that $T_n^0, T_n^1 \in O((\frac{1+\sqrt{33}}{4})^n) \cong 1.686^n$, which is to say $N^{\lg \frac{1+\sqrt{33}}{4}} \cong N^{0.7537}$, where $N = 2^n$ is the number of leaves.

Of course, our next step should be to see whether there is a lower bound to match this performance, for either the MAJ3 or NAND trees. It turns out that our algorithms are optimal but this takes some work. The first step is a fundamental fact about randomized complexity.

2.3 Randomized vs. distributional complexity, error-free ("Las Vegas")

Let L be a computational problem, restricted to inputs of "size" n (however that is measured). We'll usually suppose this is a finite set (although it doesn't much matter).

M_L = set of correct algorithms for L. (Regardless of efficiency.) We'll suppose this is a countable set.

X_L = set of inputs to L.

$$T : M_L \times X_L \to \mathbb{N}$$

$$T(r, x) = \text{ complexity of algorithm } r \text{ on input } x$$

The quantity we are usually interested in is:

Definition 12. *The* randomized complexity *of L is*

$$\text{Rand}(L) = \inf_{\substack{p \\ \text{dist. on } M_L}} \max_{x \in X_L} E_{r \in p} T(r, x)$$

(We write $r \in p$ to denote that r is sampled from distribution p.)

Giving an upper bound on Rand is, in principle, a straightforward task: devise an algorithm and bound its complexity. Well, it's not always easy, but it's clear "what" you have to do.

Lower bounds are another matter. How do you show that *no* algorithm can beat a certain complexity? Fortunately, there is an interesting way of going about this. We need the following definition.

Definition 13. *The* distributional complexity *of L is*

$$\text{Dist}(L) = \max_{\substack{q \\ \text{dist. on } X_L}} \inf_{r \in M_L} E_{x \in q} T(r, x)$$

Key facts are:

$$\text{Rand}(L) \geq \text{Dist}(L) \qquad \text{Weak LP Duality} \tag{2.1}$$

$$\text{Rand}(L) = \text{Dist}(L) \qquad \text{Strong LP Duality} \tag{2.2}$$

We need here only the easy part, weak LP duality, which is proven as follows: Take optimal distributions p and q for each side. (Or in the case of the inf over infinitely many algorithms, get within ε of optimal for arbitrary ε.) Writing T as an $|M_L| \times |X_L|$ matrix, p and q as column vectors, and denoting a singleton column vector by e_j, we have

$$\text{Rand}(L) = \max_x p^\dagger T e_x \geq p^\dagger T q \geq \min_r e_r^\dagger T q = \text{Dist}(L)$$

What this means is that instead of lower bounding the complexity of randomized algorithms on worst-case inputs, it is enough (the strong version tells us it is actually equivalent) to lower bound the complexity of deterministic algorithms on random inputs. And even if we can't find the ideal distribution on inputs, we still get weaker lower bounds from suboptimal distributions so long as we can analyze them. This is a very powerful fact.