Lecture 25 — Quantum "Supremacy"

"Quantum supremacy" is a term coined by John Preskill in 2012. It refers to a very specific concept: Getting an actual real-life quantum computer to perform some computational task that is believed to be impossible to perform by any real-life classical computer. The term is in the news these days, as it seems there is a chance that quantum supremacy will either be achieved — or at least, will be *claimed* to have been achieved — in the near future. Even — next year? So I want to end the course by talking about quantum "supremacy", so you'll be well-informed to judge potential claims.

By the way, as a term, quantum "supremacy" has its pluses and minuses; it's somehow distasteful sounding, but at the same time, it's kind of catchy, and no other shorthand term for the concept has caught on. So we'll stick with it.

Again, the idea is to actually build a real-world quantum circuit — perhaps with 72 qubits and 1000 gates — that does... *something*. And then to assert that no real-world classical computer can do this *something*. And, we don't care if this *something* is in any way *useful* — it's just something.

Perhaps the most likely team that will claim to demonstrate quantum supremacy in the near future is John Martinis's group, originally from UCSB, now at Google. They are very good at building real-world quantum circuits, and they've published a concrete plan to achieve "quantum supremacy". And what is the *something* that their real-world quantum circuit is going to do? It's... *implement a random quantum circuit.* ("Simulate itself", as some naysayers say. But that's fine with me personally; I'm on board the hype train!)

But first: Why don't we have large-scale quantum computers already? If we did, then implementing Shor's algorithm would be the ideal demonstration of quantum supremacy.

Well, it's hard to isolate and shield a physical qubit so that its state (including entanglement with other qubits) does not collapse/decohere/get corrupted by noise. Another way to put this: implementing the 0-qubit Identity gate is hard! That's one major problem! As far as I know, implementing 1-qubit gates and 1-qubit measurements is not that much harder. But implementing even the simplest 2-qubit gate, CNOT, is apparently *much* harder still (you have to get qubits to interact!). These are all huge engineering challenges.

Also: you have to physically lay out these qubits. It's natural that they're laid out in a 2-d grid. And you can only do CNOTs between grid-adjacent pairs of qubits. Luckily this is fine; circuits like this — 1-qubit gates, plus CNOTs between grid-adjacent qubits — have been proven to be sufficient (in theory) for full quantum computation.

But getting a qubit to "last" for a short spell of time — like, through the application of a dozen gates — is hard. Indeed, for real-world practice, the "parallel" time-complexity of quantum circuits is paramount, as circuits looks like this —

draw circuit that looks like a few success "time-slices",

where each potentially involves each qubit in a 1- or 2-qubit gate.

So getting high-quality qubits that can last, and accurate gate implementations, is the main hard part — moreso than just getting "lots" of qubits. Bear this in mind when you hear groups boasting of 25 or 50 (actually 49 because it's 7×7) or 72 ($= 8 \times 9$) qubits.

Classical fault-tolerance. In the early days of computing, people also used to worry about noise and gate-failure for *classical* circuits, too. Like, suppose you're computing with classical AND/OR/NOT circuits, but

each gate fails (outputs garbage), independently with small probability $\eta > 0$ ("noise rate").

Well, can you still achieve full-fledged classical computation? I'll tell you — though it turned out to be a practically-irrelevant problem. The main reason is that a real-world circuit implements a logical bit, 0 or 1, with, like, a million physical particles (electrons on a wire) — high or low voltage. It's almost like it's using the "repetition error-correcting code" with million-fold repetition! And unlike with qubit-states, a bit is digital — 0 or 1. Today's classical computers apparently do still have a "noise rate", but it's like better than 2^{-64}.

Anyway, von Neumann in 1952 thought about the theoretical problem and sketched a proof (later made rigorous) that with suitable error-correction/fault-tolerance techniques...

$\exists$ universal $\eta_0 > 0$ such that, provided $\eta < \eta_0$,
can make any m-gate circuit into a "fault-tolerant" version, with size $\approx m \log m$

Also, it's not hard to show that a "blowup" factor of $\gtrsim \log m$ is necessary. The exact *threshold* η_0 here depends on your gate set, noise model, and what coding/fault-tolerance scheme you use.

The issue of noise is much more severe with quantum circuits, where it seems you want to use 1 particle for 1 qubit. Shor recognized this soon after his algorithm, and developed the first *quantum error-correcting codes.* It's perhaps surprising that these can even exist — you're protecting quantum amplitudes, you seemingly can't repeat/copy qubits due to No-Cloning, you can't look at your qubits to see if they got corrupted, without spoiling them yourself. But — it's possible! Not too complicated to do Shor's basic ones either — though more sophisticated ones (e.g., "surface codes") now exist.

Quantm error-correcting codes alone are not enough; you need to be able to *compute* on error-corrected qubits — even though your computations also experience η-noise. We assume, by the way, roughly the same error model in the quantum case — each gate outputs garbage (potentially a mixed state, even) independently with probability η. But the necessary *fault-tolerance* theorem was proven around 1996 by Dorit Aharonov and Miki Ben-Or:

Quantum Threshold Theorem: $\exists$ universal $\eta_0 > 0$ such that, provided $\eta < \eta_0$, can make any m-gate quantum circuit into a "fault-tolerant" version, with size $\approx m\text{polylog}(m)$

Their initial estimate was $\eta_0 = 10^{-6}$. Subsequent improvements (more sophisticated ECC's and fault-tolerance algorithms) have led to

$$\eta_0 \approx 10^{-3} \dots 10^{-2},$$

depending on exact model details. Interestingly, results from this year (Gottesman + Fawzi–Grospellier–Leverrier) showed that it's possible to get only a *constant-factor* circuit-size blowup too (as opposed to polylog(m)), albeit with much smaller η_0. Wait — how? Wasn't it shown that blowup $\log m$ is needed even classically? Well, the new results assume noiseless classical computation.

It's a real, numerical threshold: if you can just get your qubit/gate noise relates below η_0, you can bootstrap to arbitrary fault-tolerant quantum computation. You may ask, what kinds of rates are the best engineers achieving these days? Well, it's kinda close to $\eta \approx 10^{-3}$. So why aren't we done? Well, there are many caveats. For one, you'd probably want a few extra factors of 10 for safety, or to overcome the mismatches between theory and practice. For another, the $\eta_0 \approx 10^{-3}$ estimate assume you are using the best possible ECCs and fault-tolerance algorithms, which are very sophisticated; e.g., they use a few hundred physical qubit per logical qubit. So if you want 100 logical qubits…

Yet another problem is that the ECC-decoding/correcting algorithms used — which are completely classical! — are, like, cubic-time algorithms, yet the qubits decay so fast you would need to run them, like, every 10 clock-cycles on your classical computer. Since that's currently impossible, you'd have to use much less sophisticated quantum fault-tolerance algorithms… which in turn might require $\eta_0 = 10^{-8}$… which we can't achieve in practice today.

Indeed: Quantum error correction (let alone fault-tolerance) has never been demonstrated in practice, even at a very small scale. (As far as I know.)

Given this, how are they planning to achieve "quantum supremacy"? Well, it will involve them making their absolute highest-quality $72 = 8 \times 9$ qubits, doing maybe 20 layers of un-error-corrected 1-qubit, CNOT, and measurements on them, and hoping everything comes through okay.

The quantum supremacy plan. Can't do any actually useful computation with such a circuit. So what's the quantum supremacy plan? Here's the one proposed by the Martinis–Google group. As mentioned, it's basically "simulate" a quantum circuit. And to make it hard (hopefully) for classical algorithms, it'll be a *random* quantum circuit.

1. On paper, draw a *random* quantum circuit B (of the aforementioned type). Call it the *Blueprint*. All $n = 72$ input qubits to $|0\rangle$, all qubits finally measured.
2. Output is a (classical, randomly constructed) probability distribution p_B on $\{0,1\}^n$.
3. Declare the computational task to be: Create a machine that generates draws from p_B.
4. (Try to) solve the task by building Q, a physical implementation of B.
5. Hope that Q's output distribution, p_Q, matches p_B.
6. Assert no classical (randomized) circuit C can have output distribution, p_C, matching p_B.
7. Declare Quantum Supremacy. Profit (?). (Perhaps with a Nobel Prize?)

Note that the computational task here is *not* a decision problem, it's a "sampling problem". We'll talk later about the assertion that no real-world classical C can do a good job at generating p_B. First, though, we'll talk about p_Q matching p_B itself. In fact, in the Martinis plan, they foresee that, since the physical Q will experience a lot of noise, it will *also* do a bad job at generating p_B. The plan/hope, though, is to just show that it does a *somewhat* less terrible job than any classical C!

Porter–Thomas distribution. First things first: What will p_B (probably) look like? Remember, this is a probability distribution that is *randomly generated*, via the randomly chosen quantum circuit B. Note that

$$B \text{ induces some } N = 2^{72}\text{-dimensional unitary, } U.$$

It is claimed that, since B is a randomly chosen circuit, U will (at least in some ways) act like a "uniformly random unitary". While this has not been formally proven, it's a mathematically clean statement and recent positive results along these lines *have* been shown (e.g., Arrow–Mehraban '18 showed that a certain kind of random n-qubit matrix, qubits in a 2-d grid, depth $O(\sqrt{n})$, indeed acts in some ways like a truly random unitary).

So let's assume that B induces a truly random unitary. Since B's input is all-$|0\rangle$'s, the final output state is simply the first column of that unitary. In turn, it's known mathematically that the first column of a uniformly random unitary is virtually identical to a random vector where each entry is an independent (complex) *Gaussian* of variance $1/N$. Now the output probabilities for each of the $N = 2^n$ strings are the (magnitude-)squares of these Gaussians. In turn, the square of Gaussian is distributed as an *exponential* random variable. Thus we are fairly confident that the probability distribution p_B will have the so-called...

$$\textit{Porter–Thomas distribution}: \quad \text{model } p_B(x) = \frac{\mathrm{Exp}(1)}{N}, \text{ for each } x \in \{0,1\}^n,$$

where the Exp(1)'s denote independent exponential random variables. (Sketch pdf $f(x) = e^{-x}$.) So..., it's a vaguely uniform distribution, in that each probability is typically $\frac{\text{const.}}{N}$. But... there are some heavier and lighter elements. E.g., whp half of the strings will have probability $\leq .7/N$, half will have probability $\geq .7/N$. (Actually, ".7" is $\ln 2$.)

The question (for the algorithm) is, roughly: *which* strings are heavy and light? Can you generate random strings with the p_B probabilities, given access to B?

Evaluating algorithms. The plan is to physically build a quantum circuit Q that models B. It will generate some distribution p_Q on $\{0,1\}^n$. Due to noise, it's too much to hope that p_Q will exactly be p_B. But perhaps one can fairly declare "quantum supremacy" if it's noticeably closer than any distribution p_C that can be generated by a physical classical computer. But what should "close" mean? A very natural measure, statistically speaking, is the *KL divergence*:

$$d_{KL}(p_B \| p_Q) \text{ vs. } d_{KL}(p_B \| p_C), \quad \text{where } d_{KL}(p\|q) = \sum_x p(x) \ln(p(x)/q(x)).$$

As I'll describe shortly, the Martinis et al. plan kind of gets its stats backwards, IMHO. But let's briefly talk about the above. One might assert (as Martinis et al. do) that a classical algorithm, even knowing B, can't really do better than simply generating the uniform distribution $p_C(x) \equiv 1/N$. A bit of a bold claim, but perhaps roughly true. We'll discuss this as well. But if it's true, it's not hard to show that

$$d_{KL}(p_B \| \text{uniform}) \approx \gamma \approx .577$$

with high probability, where γ is the "Euler–Mascheroni constant" (nth Harmonic number minus $\ln n$, in the limit as $n \to \infty$). So the high-level idea is that if the Google group can show that their Q has, say,

$$d_{KL}(p_B \| p_Q) \leq .56,$$

or ideally, $\leq .01$, they might fairly declare "quantum supremacy".

How will they do that, though?! They say they might be able to heuristically understand the noise rates in their own physical quantum gate implementations, and thereby maybe predict $d_{KL}(p_B \| p_Q)$.

Ideally, though, you'd like to actually build your Q and then *estimate* how well you did; i.e., estimate $d_{KL}(p_B \| p_Q)$. There are several problems here. First of all, let's assume that you even could perfectly compute $p_B(x)$ for any x. (In fact, that's supposed to be classically hard! A Catch-22? Well, they suggest trying the whole story for smaller numbers of qubits, like $n = 16$ or $n = 25$, where you *can* use an exponential-time algorithm to exactly compute $p_B(x)$. Then you can at least get a sense for how things are going for $n \ll 72$, and then maybe extrapolate up.) Still you have some problems, because all you can really do about p_Q is sample from it! And how will you use that ability to estimate

$$d_{KL}(p_B \| p_Q) = \sum_{x \in \{0,1\}^n} p_B(x) \ln(p_B(x)/p_Q(x)).$$

What they do is decide to look at a different (perhaps statistically unsound?) figure of merit, namely

$$\sum_x p_Q(x) \ln(1/p_B(x)).$$

(This is the backwards of what one would call the "cross-entropy".) At *least* this has the virtue that you can empirically estimate it: Draw a bunch of x's from Q, compute $\ln(1/p_B(x))$ for each, and average. The idea again is that if the result is noticeably smaller than what you'd get if you used p_C = uniform in place of p_Q, then you are achieving "quantum supremacy". This is a slightly dicey claim to me, because one can show that there are highly nonuniform distributions that do *better* than the uniform distribution on this new figure of merit. Indeed, you can even do "better than perfect" ($p_C = p_B$) if you can just classically find, say, one of the top .1% most probable strings under p_B...

But I digress. They have a somewhat reasonable claim for achieving quantum supremacy, providing it's really true that classically simulating the output distribution of a random quantum circuit (with good KL-divergence error, say) is hard. But is there actually complexity-theoretic evidence for that?

Time travel. I'm out of time in the course to tell you more! But let me briefly say one thing. Might

$$\mathsf{BQP} = \mathrm{BPP}?$$

We don't think so, but perhaps our best evidence is "we don't think Factoring is doable efficiently, classically". What would be cool is if we could say something like

$$\mathsf{P} \neq \mathsf{NP} \implies \mathsf{BQP} \neq \mathrm{BPP}.$$

Because given that we can't prove *anything* uncoditionally in complexity theory, "$\mathsf{P} \neq \mathsf{NP}$" is like the *gold standard* assumption we're willing to make. Unfortunately, we can't prove the above. But if instead of asking about solving *decision* problems, you ask about solving *sampling* problems (as you do, in the quantum supremacy story!), then we *can* prove something like the above. Specifically, a line of work due to Barbara Terhal, David DeVincenzo, Scott Aaronson, Michael Bremner, Richard Jozsa, Dan Shepherd, Alex Arkhipov, and others shows:

Theorem: "PH doesn't collapse" $\implies$ "classically sampling quantum output distributions is impossible".

Here "PH doesn't collapse" is like the *silver standard* assumption in complexity theory (not as beloved as "$\mathsf{P} \neq \mathsf{NP}$", but still a longstanding and cherished assumption). And "classically sampling quantum output distributions" can be taken to mean, "Given a quantum circuit (even of quite restrictive simple forms) with inputs fixed to $|0\rangle$, create a classical randomized algorithm whose output distribution matches the quantum circuit's output distribution, in the sense that it gets the probability of each string correct up to a factor of, say, 1.4." Granted this is a *bit* stronger than what's assumed hard in quantum supremacy (the quantum supremacy situation is easier in that: a) the circuit is promised to be random; b) you don't have to get *every* probability correct up to a constant multiplicative factor, maybe you just have to be overall good up to constant KL-divergence error). Still.

Finally, how do you prove the above Theorem? Interestingly, the key idea dates back to David Deutsch, the 10^{500} parallel universes guy, and it involves *time travel*... To be super-brief, General Relativity is *consistent* with the existence of "closed time-like curves"; basically, "wormholes" that allow bits to be sent back in time. (Kurt Gödel was the first to prove this! Note that it doesn't mean these wormholes exist; it just means they don't provably *not* exist.)

Just as Deutsch did with quantum computation, he asked: Suppose we find such a closed time-like curve, and it can send 1 bit back in time. What extra computational power would that give you? Turns out it would give a *lot* of power to quantum computers (roughly, you'd get a very large complexity class, containing all of "PH"), but not *so* much power to classical computers (roughly, you'd get a somewhat small complexity, not much bigger than NP). And then just by hypothetically considering this, you can show that *if* classical computers could closely simulate the output distributions of quantum computers, then time-traveling classical computers could simulate time-traveling quantum computers — thereby showing that a "small" complexity class would contain a very "big" one; i.e., the polynomial-time hierarchy would collapse!