Quantum Machine Learning Algorithms: Read the Fine Print

Scott Aaronson

For twenty years, quantum computing has been catnip to science journalists. Not only would a quantum computer harness the notorious weirdness of quantum mechanics, but it would do so for a *practical* purpose: namely, to solve certain problems exponentially faster than we know how to solve them with any existing computer. But there's always been a catch, and I'm not even talking about the difficulty of building practical quantum computers. Supposing we had a quantum computer, what would we use it for? The "killer apps"—the problems for which a quantum computer would promise huge speed advantages over classical computers—have struck some people as inconveniently narrow. By using a quantum computer, one could dramatically accelerate the simulation of quantum physics and chemistry (the original application advocated by Richard Feynman in the 1980s); break almost all of the public-key cryptography currently used on the Internet (for example, by quickly factoring large numbers, with the famous Shor's algorithm [14]); and *maybe* achieve a modest speedup for solving optimization problems in the infamous "NP-hard" class (but no one is sure about the last one). Alas, as interesting as that list might be, it's hard to argue that it would transform civilization in anything like the way classical computing did in the previous century.

Recently, however, a new family of quantum algorithms has come along to challenge this relatively-narrow view of what a quantum computer would be useful for. Not only do these new algorithms promise exponential speedups over classical algorithms, but they do so for eminently-practical problems, involving machine learning, clustering, classification, and finding patterns in huge amounts of data. So, do these algorithms live up to the claims? That's a simple question with a complicated answer.

The algorithm at the center of the "quantum machine learning" mini-revolution is called HHL [9], after my colleagues Aram Harrow, Avinatan Hassidim, and Seth Lloyd, who invented it in 2008. Many of the subsequent quantum learning algorithms extend HHL or use it as a subroutine, so it's important to understand HHL first. (See also a *Nature* News & Views piece by Childs [5].)

HHL attacks one of the most basic problems in all of science: namely, solving a system of linear equations. Given an $n \times n$ real matrix A and a vector b, the goal of HHL is to (approximately) solve the system

$$Ax = b$$

for x, and to do so in an amount of time that scales only *logarithmically* with n, the number of equations and unknowns. Classically, this goal seems hopeless, since n^2 steps would be needed even to examine all the entries of A, and n steps would be needed even to write down the solution vector x. By contrast, by exploiting the exponential character of the wave function, HHL promises to solve a system of n equations in only about $\log n$ steps. But does it really do so? This is one case where it's important to read the fine print.

Briefly, the HHL algorithm "solves $Ax = b$ in logarithmic time," but it does so only with the following four caveats, each of which can be crucial in practice.

(1) The vector $b = (b_1, \ldots, b_n)$ somehow needs to be loaded quickly into the quantum computer's memory, so that we can prepare a quantum state $|b\rangle = \sum_{i=1}^{n} b_i |i\rangle$, of $\log_2 n$ quantum bits (or qubits), whose n amplitudes encode b's entries. (Here, I assume for simplicity that b is a unit vector.) At least in theory, this can be done using a "quantum RAM": that is, a memory that stores the classical values b_i, and that allows them all to be read at once, in quantum superposition. Even then, however, it's essential either that b is relatively uniform, without a few b_i's that are vastly larger than the others,[1]

[1] If the HHL algorithm could be adapted to the case where (say) a single b_i is 1 and all the others are 0, then it would contradict the impossibility of an exponential speedup for black-box quantum search, which was proved by Bennett, Bernstein, Brassard, and Vazirani [3].

or else that the quantum RAM contains (say) the partial sums $\sum_{i=1}^{j} b_i^2$, or pointers to b's large entries, rather than just the b_i's themselves. Alternatively, if b is described by a simple, explicit formula, then the quantum computer might be able to prepare $|b\rangle$ quickly for itself, without needing to consult a quantum RAM. Either way, though, if preparing $|b\rangle$ already takes n^c steps for some constant c, then the exponential speedup of HHL vanishes in the very first step.[2]

(2) The quantum computer also needs to be able to apply unitary transformations of the form e^{-iAt}, for various values of t. If the matrix A is *sparse*—that is, if it contains at most s nonzero entries per row, for some $s \ll n$—and if there's a quantum RAM that conveniently stores, for each i, the locations and values of row i's nonzero entries—then it's known that one can apply e^{-iAt} in an amount of time that grows nearly linearly with s [4]. There are other special classes of matrix A for which a quantum computer could efficiently apply e^{-iAt}. Again, though, if applying e^{-iAt} takes n^c time for some constant c, then the exponential speedup of HHL disappears.

(3) The matrix A needs to be not merely invertible, but *robustly* invertible, or "well-conditioned." More precisely, let $\kappa = |\lambda_{\max}/\lambda_{\min}|$ be the ratio in magnitude between A's largest and smallest eigenvalues. Then the amount of time needed by HHL grows nearly linearly with κ. If κ grows like n^c, then the exponential speedup is gone.

(4) Finally, the limitation noted earlier—that even writing down the solution vector $x = (x_1, \ldots, x_n)$ already requires n steps—also applies in the quantum world. When HHL is finished, its output is not x itself, but rather a quantum state $|x\rangle$ of $\log_2 n$ qubits, which (approximately) encodes the entries of x in its amplitudes. The quantum computer user can then measure $|x\rangle$, in a basis of her choice, to reveal some limited statistical information about x: for example, the locations of any extremely large entries of x, or the approximate value of an inner product $\langle x|z\rangle$, where z is some fixed vector. However, learning the value of any *specific* entry x_i will, in general, require repeating the algorithm roughly n times, which would once again kill the exponential speedup.

To summarize, HHL is not *exactly* an algorithm for solving a system of linear equations in logarithmic time. Rather, it's an algorithm for approximately preparing a quantum superposition of the form $|x\rangle$, where x is the solution to a linear system $Ax = b$, assuming the rapid ability to prepare the state $|b\rangle$ and to apply the unitary transformation e^{-iAt}, and using an amount of time that grows roughly like $\log(n) \cdot \kappa s/\varepsilon$, where n is the system size, κ is the system's condition number, s is its sparsity, and ε (which I didn't mention before) is the desired error.

For all that, couldn't the HHL algorithm *still* be useful for something? Absolutely—as long as one can address all the caveats, and explain why they're not fatal for one's desired application. To put it differently, perhaps the best way to see HHL is as a *template* for other quantum algorithms. One puts flesh onto the template by showing how to prepare $|b\rangle$, apply e^{-iAt}, and measure $|x\rangle$ in a specific case of interest, and then carefully analyzing the resulting performance against that of the best known classical algorithm for that case.

To my knowledge, so far there has been one attempt to work out a potential application of the HHL template from start to finish. Namely, Clader, Jacobs, and Sprouse [6] argued that HHL could be used to speed up the calculation of electromagnetic scattering cross-sections, for systems involving smooth geometric figures in 3-dimensional space. To do this, Clader et al. needed to verify, not merely that solving Maxwell's equations can be reduced to solving a linear system $Ax = b$, but also that (1) the appropriate matrix A is sparse; (2) at least from numerical data, the condition number κ appears to be bounded, at least if careful "preconditioning" is done; (3) provided the object is regular, one can calculate the entries of A and prepare the state $|b\rangle$ explicitly, avoiding the need for a quantum RAM; and (4) one really is interested in specific observables of the solution vector x; one doesn't need the entire vector $(x_1, \ldots, x_n)$.

[2] One reason why preparing $|b\rangle$ could take n^c steps, is memory latency in the quantum RAM. The strongest argument for ignoring this concern is that people also typically ignore memory latency when analyzing *classical* algorithms.

Note also that, for the speedup to be genuine, the quantum RAM needs to be "passive": that is, it must not require a separate parallel processing element for each number that it stores. For if it did, then by dividing the work among the parallel processors, we could solve a system of the form $Ax = b$ in $O(\log^2 n)$ steps purely classically [7], with no quantum computer needed.

Crucially, Clader et al. could not rule out the possibility that, once the problem of solving a linear system has been restricted in all these ways, there's also a *classical* algorithm that provides the answer in nearly the same amount of time as HHL. The most they could say was that they couldn't find such a classical algorithm. The difficulty here is a general one: in quantum algorithms research, we always want to compare against the fastest possible classical algorithm that performs the same task. But if we are honest, the "task" here cannot be defined as solving an arbitrary linear system $Ax = b$, because *HHL can't do that either.* The task, rather, needs to be defined as estimating certain observables of x, for certain special systems $Ax = b$ such that HHL can efficiently estimate the same observables. And that makes it far less obvious whether there might be a clever classical solution as well.[3]

Harrow et al. [9] did prove one important result indicating that there are at least some cases where HHL runs in logarithmic time, whereas any possible classical algorithm requires polynomial time. Namely, they showed that HHL is "universal for quantum computation." What this means is that we can encode *any* quantum algorithm—say, Shor's factoring algorithm—into a system of roughly 2^n linear equations in 2^n unknowns, and then use HHL to "solve" the system (i.e., simulate the algorithm) in polynomial time. Thus, provided we believe *any* quantum algorithm achieves an exponential speedup over the best possible classical algorithm, HHL can in principle achieve such a speedup as well. On the other hand, the linear systems produced by this reduction will be extremely artificial. The essential insight behind the exponential speedup, one might argue, is provided by Shor's algorithm or whatever other quantum algorithm we are simulating, rather than by HHL itself.

In the years since HHL, quantum algorithms achieving "exponential speedups over classical algorithms" have been proposed for other major application areas, including k-means clustering [11], support vector machines [13], data fitting [16], and even computing certain properties of Google PageRank vectors [8]. It's fair to say that all of these algorithms inherit many of the caveats of HHL. With each of them, one faces the problem of how to load a large amount of classical data into a quantum computer (or else compute the data "on-the-fly"), in a way that is efficient enough to preserve the quantum speedup. With each, one faces the problem that, even if the output state $|\psi\rangle$ implicitly encodes all the data one wants in its amplitudes, a measurement of $|\psi\rangle$ reveals only a tiny probabilistic sketch of the information. Most importantly, with each algorithm, one faces uncertainty about whether, after one has properly accounted for the other caveats, one could then find a classical sampling algorithm that achieves similar performance to the quantum algorithm's.

To illustrate these issues, let's consider a 2013 algorithm of Lloyd, Mohseni, and Rebentrost [11] for supervised machine learning. In this task, we are given a quantum state $|u\rangle$ of $\log_2 n$ qubits—or equivalently, a vector of n amplitudes—as well as two "clusters" of other states, $|v_1\rangle, \ldots, |v_m\rangle$ and $|w_1\rangle, \ldots, |w_m\rangle$. The problem is to classify $|u\rangle$ into one of the two clusters, by deciding which mean it is closer to: $|v\rangle = (|v_1\rangle + \cdots + |v_m\rangle)/m$ or $|w\rangle = (|w_1\rangle + \cdots + |w_m\rangle)/m$. Lloyd et al. [11] give an extremely fast quantum algorithm for this problem: in particular, they show how to estimate the inner products $\langle u|v\rangle$ and $\langle u|w\rangle$, to accuracy ε, using only about $\log(mn)/\varepsilon$ steps. They argue that this represents an exponential speedup over classical inner-product computation.

The trouble is, where did we get the states $|u\rangle, |v_k\rangle, |w_k\rangle$ in the first place? The obvious possibility would be that we prepared the states ourselves, using lists of amplitudes written in a quantum RAM. In that case, however, a straightforward preparation strategy would work only if the amplitudes are relatively uniform: if there aren't a few amplitudes that dominate all the others in magnitude. But if the amplitudes are uniform in the required way, then it's not hard to show that we can also estimate the inner products $\langle u|v\rangle$ and $\langle u|w\rangle$ using a *classical* random sampling algorithm, in about $\log(mn)/\varepsilon^2$ steps! In other words, once we carefully spell out conditions under which the quantum algorithm would be useful, we find—at least in this example—that a classical algorithm exists that is at most quadratically slower.

Admittedly, the above argument doesn't rule out that there could be *other* ways to generate the states $|u\rangle, |v_k\rangle, |w_k\rangle$, and that with those other ways, the exponential quantum speedup would stand. And indeed, Lloyd et al. cite a 2009 result of mine [1], which shows that if our task was to estimate $\langle u|F|v\rangle$, where F is

[3] Similarly, Wang [15] has used HHL to give a quantum algorithm for approximating effective resistances in electrical networks. Again, Wang's algorithm *could* give an exponential speedup over classical algorithms, but only under special conditions: for example, that the electrical network has small degree but a large amount of interconnectivity (technically, "large expansion"); and that a description of the electrical network can be quickly loaded into the quantum computer (for example, because the network has a regular pattern). And it is not yet clear whether there are real-world examples where (a) these conditions are satisfied, but (b) there isn't also a fast classical algorithm to approximate the effective resistances.

an $n \times n$ unitary matrix that applies a Fourier transform, then any classical algorithm would need at least about $n^{1/4}$ steps. (In subsequent work by myself and Ambainis [2], we improved this lower bound to at least about $\sqrt{n}/\log n$ steps, nearly matching an upper bound of $\sqrt{n}$ steps.) On the other hand, a quantum computer can estimate $\langle u|F|v\rangle$ just as easily as it can estimate $\langle u|v\rangle$, in only a logarithmic number of steps. To put it differently, if one of two vectors that we want to compare is given to us not directly, but only via its Fourier transform, then we really *do* get an exponential quantum speedup in comparing the vectors.

Even here, however, care is needed. The theorem that Ambainis and I proved—which rules out a superfast classical algorithm to compare one vector u written in memory, against the Fourier transform of a second vector v written in memory—applies only if u and v are both random when considered individually. Furthermore, we still have the issue mentioned before: that a quantum computer can estimate $\langle u|F|v\rangle$ in logarithmic time only if it can quickly prepare the states $|u\rangle$ and $|v\rangle$. And to do that, starting from lists of entries $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_n)$ in a quantum RAM, we need the vectors to be reasonably uniform; neither u nor v can have sharp "peaks." Take away either of these conditions, and the exponential quantum speedup can once again disappear.

The general rule here is: if we want an exponential quantum speedup, then we need there to be an extremely fast way to create a certain kind of superposition, and we also need there *not* to be a fast way to create an analogous probability distribution using a classical computer. The Fourier transform is one convenient way to achieve both of these goals: it can foil classical sampling methods because of its "global" nature, while causing no problems for a quantum computer. But quantum algorithm designers have other tricks up their sleeve.

This past summer, Lloyd, Garnerone, and Zanardi [10] proposed a polynomial-time quantum algorithm for estimating certain topological features of data: most notably *Betti numbers*, which count the number of holes and voids of various dimensions in a scatterplot. What is notable about their algorithm is that its input consists of only $n(n-1)/2$ real numbers—namely, the distances between each pair of points out of n—but using that data, the algorithm constructs a quantum superposition over up to 2^n simplices in a simplicial complex. In other words, much like the algorithm of Clader et al. [6] for electromagnetic cross sections, the new algorithm avoids the expensive need to load in a huge amount of data from a quantum RAM; instead it computes the data as needed from a much smaller input. The Betti number algorithm still has a significant drawback: preparing a superposition over simplices takes a number of steps that grows like $\sqrt{2^n/S}$, where S is the total number of simplices in the complex. So if S is much smaller than 2^n, as it will often be in practice, the algorithm can require exponential time. Even so, the ability to estimate Betti numbers efficiently when $S \approx 2^n$ provides a good example of how the limitations of HHL-type algorithms can sometimes be overcome.

So in summary, how excited should we be about the new quantum machine learning algorithms? To whatever extent we care about quantum computing at all, I'd say we should be excited indeed: HHL and its offshoots represent real advances in the theory of quantum algorithms, and in a world with quantum computers, they'd probably find practical uses. But along with the excitement, we ought to maintain a sober understanding of what these algorithms would and wouldn't do: an understanding that the original papers typically convey, but that often gets lost in secondhand accounts.

The new algorithms provide a general template, showing how quantum computers *might* be used to provide exponential speedups for central problems like clustering, pattern-matching, and principal component analysis. But for each intended application of the template, one still needs to invest a lot of work to see whether (a) the application satisfies all of the algorithm's "fine print," and (b) once we include the fine print, there's *also* a fast classical algorithm that provides the same information. This makes the quantum machine learning algorithms quite different from (say) Shor's factoring algorithm. Having spent half my life in quantum computing research, I still find it miraculous that the laws of quantum physics let us solve *any* classical problems exponentially faster than today's computers seem able to solve them. So maybe it shouldn't surprise us that, in machine learning like anywhere else, Nature will still make us work for those speedups.

Acknowledgments: Thanks to Andrew Childs, Aram Harrow, Greg Kuperberg, Daniel Lidar, Seth Lloyd, and Masoud Mohseni for helpful comments on this piece.

References

[1] S. Aaronson. BQP and the polynomial hierarchy. In *Proc. ACM STOC*, 2010. arXiv:0910.4698.

[2] S. Aaronson and A. Ambainis. Forrelation: a problem that optimally separates quantum from classical computing. arXiv:1411.5729, 2014.

[3] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. quant-ph/9701001.

[4] D. W. Berry, A. M. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. arXiv:1501.01715, 2015.

[5] A. M. Childs. Quantum algorithms: Equation solving by simulation. *Nature Physics*, 5(12):861, 2009.

[6] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.*, 110(250504), 2013. arXiv:1301.2340v4.

[7] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5(4):618–623, 1976.

[8] S. Garnerone, P. Zanardi, and D. A. Lidar. Adiabatic quantum algorithm for search engine ranking. *Phys. Rev. Lett.*, 108(230506), 2012. arXiv:1109.6546.

[9] A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. *Phys. Rev. Lett.*, 15(150502), 2009. arXiv:0811.3171.

[10] S. Lloyd, S. Garnerone, and P. Zanardi. Quantum algorithms for topological and geometric analysis of big data. arXiv:1408.3106, 2014.

[11] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. arXiv:1307.0411, 2013.

[12] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9), 2014. arXiv:1307.0401.

[13] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113(130503), 2014. arXiv:1307.0471.

[14] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. quant-ph/9508027.

[15] G. Wang. Quantum algorithms for approximating the effective resistances in electrical networks. arXiv:1311.1851, 2013.

[16] N. Wiebe, D. Braun, and S. Lloyd. Quantum algorithm for data fitting. *Phys. Rev. Lett.*, 109(050505), 2012. arXiv:1204.5242.