Quantum Computing (CST Part II)

Lecture 16: Case Studies in Near-term Quantum Computation

In less than ten years quantum computers will begin to outperform everyday computers, leading to breakthroughs in artificial intelligence, the discovery of new pharmaceuticals and beyond.
Jeremy O'Brien (2016)

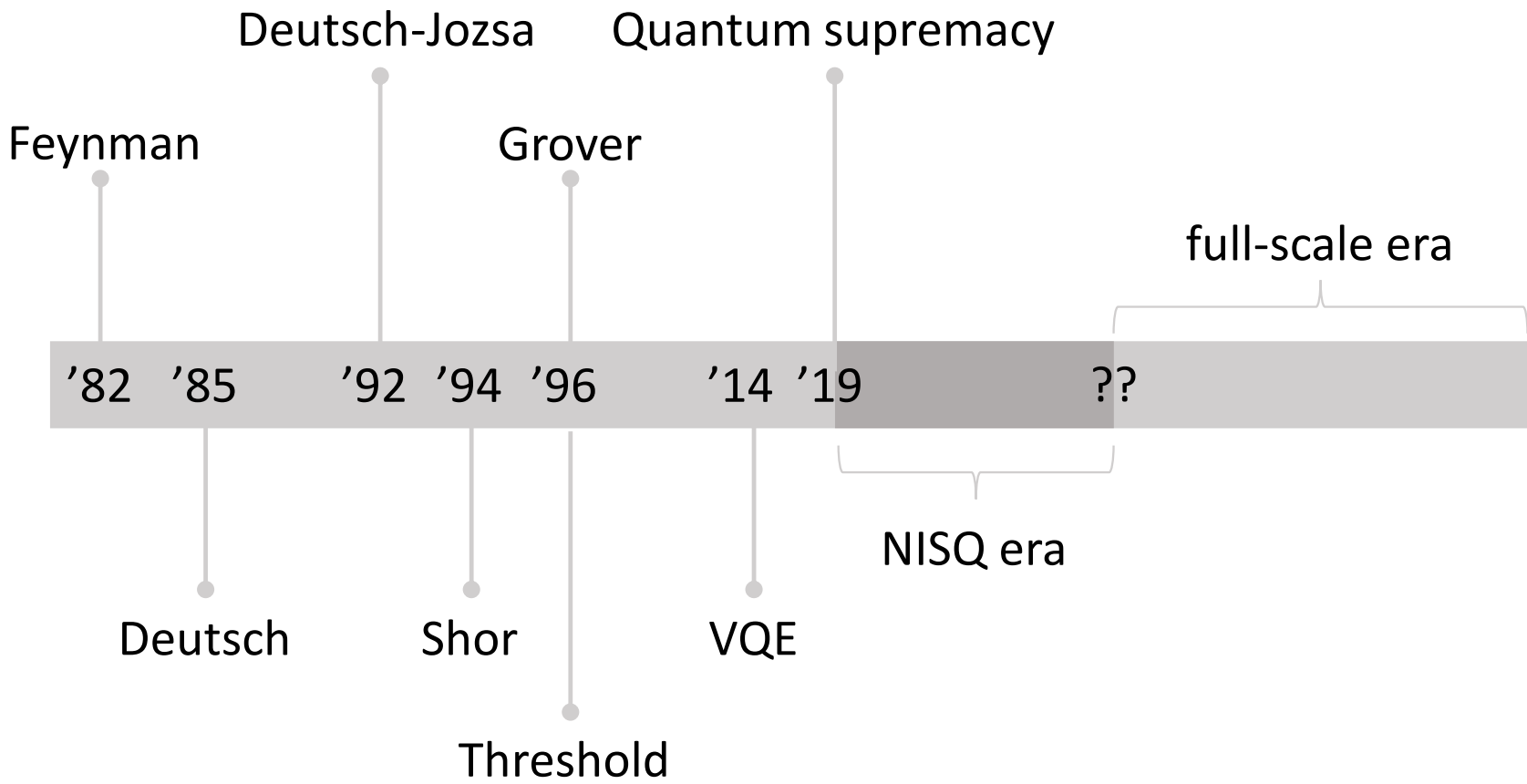
Resources for this lecture

The information disseminated in this lecture is drawn from a wide variety of sources. In particular see **John Preskill's** paper `https://arxiv.org/abs/1801.00862` for a comprehensive overview of the NISQ era.

Nielsen and Chuang chapter 7 gives a good overview of the general principles of quantum hardware design, and introduces the physical details of a number of the types of quantum computers, but some of the technical information is now out of date.

Quantum algorithm zoo can be found at `quantumalgorithmzoo.org`

Timeline of quantum computing



NISQs and full-scale fault-tolerant quantum computers

The successful quantum supremacy experiment, demonstrated by Google in 2019, has heralded the start of the *NISQ era*. "NISQ" stands for *noisy intermediate-scale quantum* (computer), a name coined by John Preskill.

Much of the current focus of the quantum computing community (especially in industry, for obvious reasons) is on finding real-world advantageous applications of quantum computing using NISQs. We have already met VQE, a near-term quantum chemistry algorithm, and quantum annealing (e.g., D-Wave) broadly fits into this category too.

However, many of the quantum algorithms we have studied in this course *do* require full-scale fault-tolerant quantum computers, and the time when such technology exists I have termed the *full-scale era* in this lecture.

Overview of quantum algorithms

Quantum algorithm zoo lists over 60 quantum algorithms, some of the main ones are:

Algorithm	**Function**	**Speed-up**	**Era**
Shor	factoring	super-polynomial	full-scale
Grover	search	polynomial	full-scale
HHL	linear algebra	super-polynomial	full-scale
QPE	chemistry	super-polynomial	full-scale
VQE	chemistry	heuristic*	NISQ
Annealing	optimisation	heuristic	NISQ
QAOA	optimisation	heuristic	NISQ

*There is strong evidence that VQE provides a very good speed-up in practise, which is in some ways commensurate with an exponential speed-up in theory.

HHL and QAOA

During this course, we have studied most of the important quantum algorithms listed on the previous slide. The exceptions being:

- **HHL**, a quantum algorithm invented in 2008 for approximately solving sparse systems of linear equations, known by the initials of its inventors, Aram Harrow, Avinatan Hassidim and Seth Lloyd. From the HHL Wikipedia article:

  "Due to the prevalence of linear systems in virtually all areas of science and engineering, the quantum algorithm for linear systems of equations has the potential for widespread applicability."

- **QAOA**, the *Quantum Approximate Optimization Algorithm*, invented in 2014 by Edward Farhi, Jeffrey Goldstone and Sam Gutmann. After VQE it is arguably the most promising candidate for a quantum algorithm to show quantum advantage on some useful application in the NISQ era.

Overview of quantum algorithms

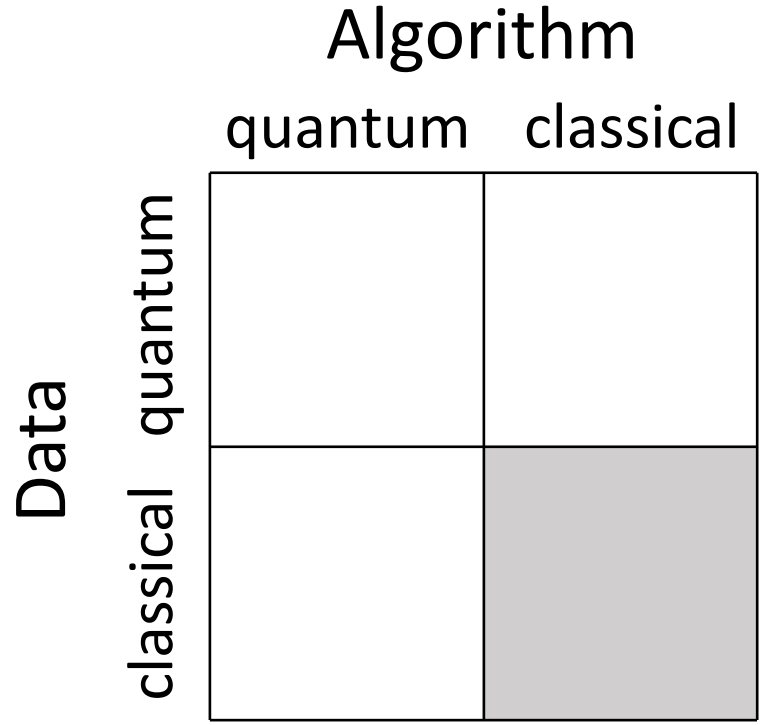
Quantum algorithm zoo lists over 60 quantum algorithms, some of the main ones are:

Algorithm	**Function**	**Speed-up**	**Era**
Shor	factoring	super-polynomial	full-scale
Grover	search	polynomial	full-scale
HHL	linear algebra	super-polynomial	full-scale
QPE	chemistry	super-polynomial	full-scale
VQE	chemistry	heuristic*	NISQ
Annealing	optimisation	heuristic	NISQ
QAOA	optimisation	heuristic	NISQ
Quantum machine learning		various	both?

*There is strong evidence that VQE provides a very good speed-up in practise, which is in some ways commensurate with an exponential speed-up in theory.

Quantum machine learning

"Quantum machine learning" is a buzz-word heavy slide title, but what does it actually mean? Crudely, it can be divided into three categories:



Quantum machine learning (cont.)

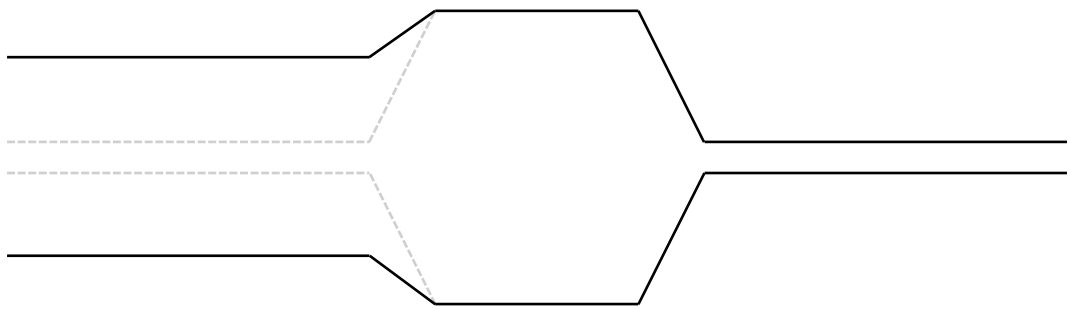
- **Quantum machine learning on classical data** refers to the use of quantum algorithms to enhance the performance of conventional machine learning algorithms. For example, quantum optimisation (annealing or QAOA), search (Grover) and / or linear system solving (HHL) may be called as subroutines by some otherwise classical machine learning algorithm.
- **Classical machine learning on quantum data** refers to the use of conventional, classical learning techniques to learn something about some quantum data. Quantum state tomography is a basic example.
- **Quantum machine learning on quantum data** is, it could be argued, true quantum machine learning, in the sense that we want to discern some information from a quantum data-set, which may not be possible if that quantum data were simply measured and classical learning applied.

In short, the second and third items differ because, in the former the quantum state is measured and thus collapsed into classical data on which classical machine learning is applied, whereas in the latter quantum operations are applied to the quantum data.

Where does quantum data come from?

To understand quantum machine learning in more detail, it is worth addressing where quantum data comes from in the first place. Crudely speaking, this can be grouped into three categories:

- **Quantum data as an output of a quantum process**, e.g., from a quantum sensor. This is the natural set-up for classical or quantum machine learning on quantum data.
- **Quantum data prepared "directly" from classical data**. However, the process of preparing a desired quantum state (i.e., to encode the classical data as appropriate) may not be possible to do efficiently, and we may end up with an algorithm looking like:



  rather than the double-necked bottle structure that we previously identified as crucial for quantum algorithms to be efficient overall.
- **QRAM**, which requires a little further explanation.

QRAM

The problem of bespokely preparing a quantum state from classical data when performing quantum machine learning on classical data may be circumvented if we can efficiently address an arbitrary superposition of the classical data bits. One idea for how to achieve this was proposed by Vittorio Giovannetti, Seth Lloyd and Lorenzo Maccone, who describe their proposal for QRAM as:

> *"A random access memory (RAM) uses n bits to randomly address $N = 2^n$ distinct memory cells. A quantum random access memory (QRAM) uses n qubits to address any quantum superposition of N memory cells."*

Constructing a QRAM (or equivalent thereto) is arguably the most important obstacle that must be overcome in order for quantum machine learning on classical data to find real-world application.

Is quantum winter coming?

In contrast to the nice progression from quantum supremacy to NISQ era to full-scale era depicted, many in the quantum computing community fear that the current bubble will burst and there will be a (possibly long) "quantum winter". A personal view:

Owing to the already-advanced state of quantum chemistry algorithms in particular – which are likely to demonstrate useful quantum advantage in the very near-term – I don't foresee a true quantum winter in which interest evaporates entirely. What may happen, however, is that we get stuck in the NISQ era indefinitely, and quantum computers are seen as purely simulators and optimisors, rather than true universal computational machines.

HBO

From the NISQ era to the full-scale era

Fault-tolerance is the feature that distinguishes the full-scale era from the NISQ era, and this will require an error correction overhead estimated to be in the region 100–1000. That is, it will take 100–1000 physical qubits to make each "clean" logical qubit.

Qubit fidelities and error correcting codes may well improve, bringing this number down, but the fact remains that a serious scaling-up of the number of qubits in a quantum computer (currently approximately 50) is needed to build a fault-tolerant quantum computer.

This in turn has led some in the quantum computing community to talk about the need for a "quantum transistor" – a highly scalable physical realisation for a qubit, that changes the game for quantum computing in the same way that the transistor did for classical computing.

Types of qubit

There are various proposals for physically realising a qubit, of which the most promising are superconducting qubits and trapped-ion qubits.

- At present, superconducting quantum computers have the most qubits, and superconducting qubits offer fast gate times.
- On the other hand, trapped-ion qubits have the highest fidelity, and in some architectures are networked such that they can have much greater (non-planar) connectivity than superconducting quantum computers (whose qubits are typically laid out on a rectangular grid, with nearest-neighbour interactions) – proponents of trapped-ions argue that the networked architecture makes trapped-ion quantum computers more scalable.
- Other technologies include: silicon qubits; nitrogen-vacancy qubits; and optical (photonic) qubits.

Existing quantum computers

These four companies are leading the way in quantum computer hardware (all superconducting):

- **Google**: 53 qubits
- **IBM**: 53 qubits
- **Intel**: 49 qubits
- **Rigetti**: 32 qubits

An honourable mention also goes to **Microsoft** who are taking a completely different approach, and aiming to construct a quantum computer from *topological qubits*, which are intrinsically error-resistant.

In the UK we have:

- **NQIT**: building a networked trapped-ion quantum computer.
- **University of Sussex**: building a planar trapped-ion quantum computer.
- **Oxford Quantum Circuits**: building a 2-plane superconducting quantum computer.

How good is a particular quantum computer?

The total number of qubits tends to grab the headlines, but how good a particular quantum computer is actually depends on three factors:

1. The number of qubits.
2. The quality of those qubits (fidelity).
3. The connectivity (what overhead will be incurred to move the qubits around such that they can interact).

Quantum volume is a measure that has been proposed to quantify how good a given quantum computer is, incorporating these three factors. The quantum volume of a quantum computer is given by:

$$Q_v = (\min(n, d))^2$$

where n is the number of qubits and d is the depth of a random circuit that can be executed before an error is expected to occur.

Quantum volume

- The depth term, d, is a function of both the fidelity and the depth overhead incurred when executing a circuit consisting of random 2-qubit interactions.
- Therefore the quantum volume is increased for quantum computers with higher connectivity, as fewer SWAP gates will be needed to rearrange interacting qubits to be local, and so the depth overhead will be smaller.
- The presence of the $\min$ term in the definition indicates whether the performance of a given quantum computer is limited by a lack of qubits or poor fidelity / connectivity of the qubits – i.e., a more nuanced picture than simply quoting the number of qubits.
- Quantum volume has been conceptualised so that it gives a reasonable benchmark of the general performance of near-term quantum computers.
- However some researchers refute that random circuits are appropriate for this, and instead assert that it part of the role of quantum software design to execute algorithms in an efficient manner, given the physical locality constraints of the hardware.

Quantum software

Quantum software has largely taken its structure from its classical counterpart. That is:

- Quantum algorithms, as studied in this course, are written in (relatively) high-level terms in order to assert their mathematical validity, and significant further work is required to actually execute them in an efficient manner on hardware.
- One element of this is a quantum compiler, the design of which is a non-trivial quantum information processing task in its own right.
- Software development for quantum-classical hybrid algorithms, such as VQE, presents its own challenge, as it is necessary to use the two components (classical and quantum) in such a manner that the quantum computer is fully exploited for those tasks for it offers an advantage, whilst handing off everything else (including overall control) to the classical computer.

Quantum software development is a burgeoning field, with two leading companies based in Cambridge: **Riverlane** and **Cambridge Quantum Computing**.

Quantum-inspired classical algorithms

One of the most important problems in data-science is the construction of *recommendation systems*. Suppose that we have n products and a purchase history of m users, from which we need to make product recommendations:

- Until 2016 only techniques which run in time $\mathcal{O}(\text{poly}(mn))$ were known.
- In 2016 Iordanis Kerenidis and Anupam Prakash published a quantum algorithm to achieve this task in time $\mathcal{O}(\text{poly}\log(mn))$. That is, an exponential speed-up.
- Then in 2018 Ewin Tang published a classical algorithm inspired by Kerenidis and Prakash's quantum algorithm that also achieves the task in time $\mathcal{O}(\text{poly}\log(mn))$.

Therefore, even in a classical world, there is merit in thinking quantumly.

Summary

This lecture has given a general overview of the current state of the field of quantum computing, and its near-term prospects. We have looked at:

- Quantum algorithms
- Quantum hardware
- Quantum software
- Quantum-inspired classical algorithms