Introducing Quantum Machine Learning

Author: Aroosa Ijaz

Reviewer: Patrick Huembeli

Quantum machine learning (QML) is a rapidly emerging field of immense industrial and scientific interest. It beautifully merges the ideas and applications of machine learning with the enigmatic principles of quantum physics. The meaning of learning can be thoroughly redefined due to concepts like interference, entanglement and superposition. Many classical algorithms have already shown promising speed-ups in data and time complexity when quantum systems are harnessed. However, this new field is riddled with unanswered questions and considerable implementation challenges. In these notes, you will:

- get acquainted with the basic terminology and definitions
- have a look at the past, present and future of quantum hardware
- learn about the meaning of learning in the context of quantum machines
- explore the various subcategories of research directions in QML
- look at basics of quantum variational learning

Table of Contents

1.1 Why quantum machine learning?

Let us imagine that one day David Deutsch (the father of quantum computing) drops by the Vector institute of artificial intelligence in Toronto. He runs into Geoffrey Hinton (the father of deep learning) and falls into a passionate discussion about quantum machine learning over coffee. Geoffrey asks David about why this field is attracting so much interest. Let us try to think along Geoffrey's line of curiosity and list some follow-up questions. Some overarching questions are discussed as subsections below.

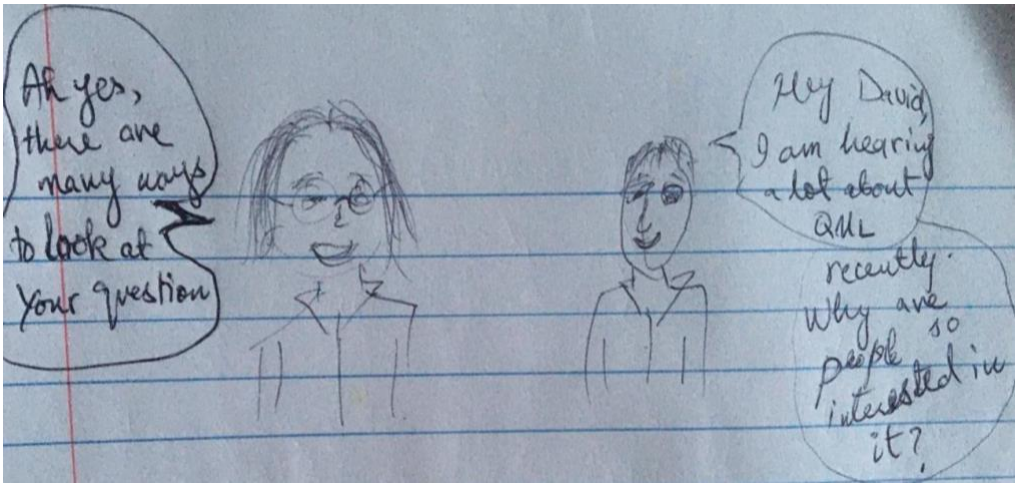

Figure 1.1 – Geoffrey and David spark our curiosity about QML.

Why should we care?

Machine learning has grown rapidly in recent years due to an increase in computational power, data availability and targeted algorithms and applications development. It will continue to play a huge role in shaping technology and human life. With the increasing amount of data and saturation of **Moore's law**, however, improvement and speedups in classical algorithms and computation power will start to saturate. It is important to explore how quantum physics can interact with this growing field, especially as physicists diligently keep working towards realizing a **universal quantum computer**. Many quantum algorithms have already been developed that show exponentially better performance for various problems. Quantum machine learning can have a huge impact on how machine learning evolves over the next decade. It can offer a different model of learning and computation. This does not necessarily imply exponential speedups for all machine learning problems, however.

Moreover, with growing amounts of data, its storage and analysis for classical algorithms will start to consume staggering amounts of energy and resources. It might be cheaper and environmentally more friendly to use quantum memories and quantum routines for certain tasks in the long run. Quantum processors will most likely be used as accelerators with classical computers; just as graphical processing units are used today. This is because classical computers are very cheap and efficient at executing most basic computation tasks. Conditions under which using quantum processors can pay off are under research. Quantum machine learning does not require universal general-purpose quantum computers. Physical hardware that can implement quantum learning algorithms is much

closer than we think, as we will see in section 1.2. Hence, it is important to explore what we can do with these emerging quantum technologies.

How can quantum mechanics help in doing machine learning?

Quantum mechanics offers many "unintuitive" phenomena that are classically unparalleled. Let us try to think of some of the ways in which these principles could potentially affect the capabilities of a learning machine.

- Quantum computing provides a fundamentally different platform for computation. Are quantum learning models computationally more powerful? Can entanglement and interference give a quantum learner access to concept classes that a classical computer cannot? Can quantum complexity and superposition lead to the process of learning a concept with smaller data or query sizes? We will explore quantum learning theory in section 1.3.
- Some quantum algorithms, like **Grover's algorithm** for unstructured search, are known to be more powerful than their classical analogues. Would this also apply to quantum machine learning models? Can parallelization and superposition result in smaller computation time, steps and resource requirements?
- The phenomenon of entanglement is only evident in quantum states and cannot appear in the classical world. Can exploiting this phenomenon help learn different or non-trivial correlations in our data? Can this lead to finding patterns that cannot be replicated on a classical computer?
- Any physical implementation of a quantum computer will have innate noise and realization of comprehensive fault-tolerance capabilities are still decades away. Can we use this innate noise in quantum systems in training quantum machine learning models to get better generalization capabilities just as noise is used in classical machine learning to make models more robust and generalizable?
- Neural networks provided with enough depth and data become universal function learners, i.e. they can learn any function incumbent in the input data. What quantum models act as universal learners?
- Optimization techniques are central to machine learning. What does optimization look like for a quantum device? Can quantum systems work around the requirement of convexity that plagues many classical machine learning methods?
- A crucial question that researchers continue to struggle with is explaining how classical learning models exactly work; explainable models. For example, the theoretical understanding of how neural networks really function and depend on depth and parametrization is still limited. If we use quantum learners dictated by the laws of quantum physics, can these provide explainable learning models?
- One of the leading arguments for the origins of quantum computing was that classical computers are unable to simulate quantum systems. If used for this purpose, can emerging quantum devices help solve existing problems in Physics and help reveal new fundamental laws of nature?

Can we rethink existing machine learning algorithms?

Previously, we saw how new algorithms can be developed when using the physical laws of quantum mechanics. Rather than creating new quantum machine learning algorithms, let us now try to think if we can change only parts of existing classical machine learning algorithms to quantum ones.

- Machine learning and deep learning use linear algebra routines to manipulate and analyse data to learn from it. Can we harness speed ups using the powerful tools of quantum systems with their innate support for linear algebra?
- Quantum states are complex probability distributions and quantum measurements represent sampling from these distributions. Can this naturally help in probabilistic machine learning models where sampling from probability distributions is generally expensive?
- Distances or similarity between quantum states can be easily assessed in their Hilbert spaces using inner product. Can this help in machine learning algorithms where computationally expensive tricks and kernels have to be used to do this?
- Representing classical data as quantum states automatically performs a feature map from the original data space to a high-dimensional Hilbert space. How can we exploit this? What nonlinearities can be used in embedding classical data to quantum states? How do we cluster or classify in these large Hilbert spaces with quantum states? Can this replace classically hard or expensive kernels?
- Topological analysis of large sets of classical data gets increasingly expensive for classical machines. How can we exploit complex topological spaces in quantum mechanics to analyse data?

How can machine learning help in doing Physics?

The applications of classical machine learning algorithms are far-reaching, from genetics, drug discovery and finance to online shopping and social policy. What about Physics? Can we exploit decades of advances in classical algorithms to further Physics research? Can we use data-driven learning techniques to understand complex and elusive problems in physics that cannot be solved analytically or simulated with the current classical computers? For example, exotic phases of matter, particle physics or complex field theories. Recently, more and more physicists have started to think along these lines, as we will see in more detail in section 1.4.

These are just some of the questions that QML scientists are working on to assess what QML can offer. Do any of these questions raise your curiosity?

1.2 What are near-term devices?

Excited to learn about what quantum machine learning research might entail, Geoffrey asks David if implementing these ideas will only become possible with the advent of quantum computers. David explains that this is a popular misconception and that scientists have made great progress in the technological advancement of quantum devices in the recent years.

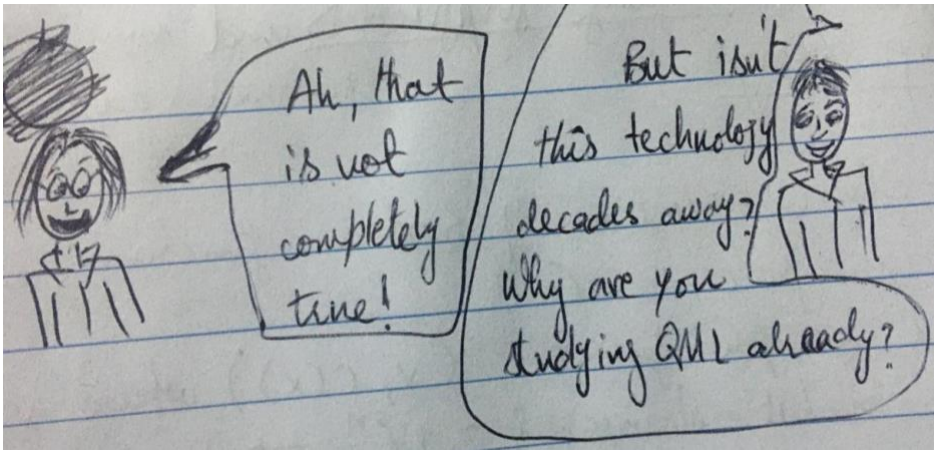

Figure 1.2 – David and Geoffrey talk about the NISQ era.

The past

The principles of quantum mechanics were formalized over the early twentieth century and were mind-boggling enough to bother even Albert Einstein. As physicists continued to try to understand quantum systems, analytic solutions became harder to achieve. The need to simulate these systems became evident. With access to classical computers, limited cases could be simulated. However, the resources required by classical computers for simulating quantum systems grow exponentially with the size of the quantum system. Moreover, the dynamics and correlations in large many-body and highly entangled systems remained elusive. A theoretical and experimental effort was globally started after Yuri Manin and Richard Feynman proposed the idea of **analogue quantum simulation** in the 80s. This entails using a known and controllable quantum system to model the dynamics of an unknown one. For example, if a set of atoms can be trapped and their interactions tuned using external electromagnetic fields, we can use them to study different forms of matter.

This led to the birth of quantum computation; a subfield of Physics that encompasses all problems where a quantum state is manipulated. Let us quickly recall that the fundamental unit of quantum computation is the **qubit** - the analog of the classical binary bit. From quantum mechanics principles, we know that a qubit state can represent a superposition over bit 0 and bit 1; it represents a probability distribution over the two states. For example, consider a simple quantum system with just two energy levels - this is understandably called **Two-Level System (TLS)** - where the ground state can be labelled as state $|0\rangle$ and the excited state as state $|1\rangle$. The system can be in any arbitrary $\alpha|0\rangle + \beta|1\rangle$ state, where α, β are complex numbers and must preserve the state unit-norm condition. This very strange generalization of probability leads to another unique feature in quantum computation: interference of probabilities that can be added **and** subtracted!

A major boost to this field came during the 90s when various new quantum algorithms that could perform better than their classical counterparts were proposed. One of the most impactful examples is the **Shor's algorithm** that factors integers exponentially faster than any known classical algorithm. This attracted a lot of attention - and investment - as the current online security protocols depend on a classical computer's inability to factor large integers in a reasonable amount of time. On the other hand, tremendous technological advances - for example in microscopy, spectroscopy, fabrication and nanotechnology - enabled scientists to isolate and manipulate the first small physical quantum systems through the 2000s and 2010s. Let us not forget that this is a fundamentally challenging problem to solve. It is not easy to perfectly isolate a qubit (remove all interactions between a qubit and its environment) and also apply unitary manipulations to control it. Even a single stray photon can lead to the **wavefunction collapse** by doing an "unintended" measurement.

The basic requirements for realizing a quantum computer were succinctly summarized by Physicist David P. DiVincenzo in 2000. They are now called the **DiVincenzo criteria**:

- The most basic one is the ability to identify a physical quantum system where a qubit can be used to encode quantum information.

- To implement any real computation, thousands of controllable qubits can be needed. So, the choice of the physical system should allow for scalability to a large network.

- Workable characteristic time scales are essential. Let us recall the definitions of these time scales and consider the Two-Level System (TLS) again. Assuming the system is in the ground state, an external electric field can "drive" the system into its excited state. The system decays back to the ground state after a certain time – called **longitudinal relaxation time**. This can be due to spontaneous emission (vacuum fluctuations) or stimulated emission (keeping the external field on). This puts the ultimate time limit on state measurement or readout as after this time both the state and coherence information are lost. The other important timescale is the **transverse coherence time**; the time it takes for the system to lose its phase or coherence information to the environment due to dephasing processes. These could include interactions with stray fields, lattice phonons, spin baths, random charges, strain or thermal fluctuations in the system's environment and so on. This puts a time limit on quantum operations or manipulations that rely on coherent properties of the system.

- The ability to initialize the system in a desired quantum state helps us start any computation with a known state.

- The ability to implement any unitary transformation (universal set of gates) and measure specific qubits are also essential for a universal quantum computer.

In principle, any physical quantum system that supports mutually orthogonal states of a physical property can be used. Some examples that are being explored so far include vacancy centers in diamond, superconducting circuits, trapped ions, semiconductor quantum dots, photons, topological qubits in nanowires and rare-earth ions trapped in crystals. Let us look briefly at some of these candidates:

- Vacancy centres in diamond: These are lattice defect sites in diamond where "vacancy" refers to missing Carbon atoms and "centers" refer to atomic impurities surrounding the vacancy like Nitrogen, Silicon or Germanium. Many of these defects result in electronic states in diamond's band gap that can be used to form qubits. Experimental techniques like fluorescence confocal microscopy can be used to detect and manipulate single defects in the lattice. Diamond provides a scalable architecture, in principle, and can be easily integrated into current silicon-based technology. A major drawback for this system is decoherence and dissimilarity between multiple qubits due to localized noise (lattice phonons, coupling to neighbouring nuclei spins and local impurities).

- Superconducting circuits: LC circuits with a capacitor and an inductor make a harmonic oscillator where energy oscillates between capacitive and inductive forms. This holds even on microscopic levels where the oscillator energy becomes quantized. As all energy levels in this quantum harmonic oscillator are equally spaced, non-linear inductors (Josephson junction) are used to get anharmonic oscillator. The lowest two energy levels are then used to encode qubit states. This usually lies in the microwave frequency regime. To reduce noise and

dissipation, temperatures are reduced to nearly absolute zero and superconducting elements are used. This architecture allows for scalability but suffers from crosstalk and noise in readout electronics with increasing number of qubits. Moreover, using cryogenics makes this system very expensive and non-portable.

- Trapped-ions qubits: Electromagnetically confining potentials are used to trap a group of certain stable ions. These systems come close to the ideal TLS. Moreover, individual ions of the same element are identical (no local irregularities like in solid-state qubits or fabricated superconducting qubits) and can be manipulated using lasers. These qubits provide long coherence times compared to many other qubit systems. Stray fields and trap inhomogeneities lead to decoherence. Scalability is possible but gets limited by trap sizes presently. It will be challenging to make traps large enough for thousands of ions (or millions - as we will see later).

Note that **adiabatic quantum computing** is a different computation model with annealing-based optimization instead of gate application and measurements. A quantum system prepared in the ground state of a simple, known Hamiltonian can be weakly perturbed and slowly driven to the ground state of the problem Hamiltonian. In 2011, this method produced one of the first commercially available quantum computers by the Canadian company D-Wave Systems. Here, however, we will only focus on gate-based quantum computing.

The present

With relentless hard work by academic groups over the last thirty years, the first small networks of qubit systems have started to materialize recently. There is now a growing interest and investment from the commercial sector as well. Aware of the impact quantum technology can have on information and communication technology in the near future, many big technology companies like Google, Microsoft, Intel and IBM have started their own research groups and developed deep collaborations with academic research groups in the last ten years. A large number of start-ups in the field have also recently sprung up and are raising huge investments.

This rapid progress has resulted in a wonderful new development: the number of qubits has grown from less than 10 qubits and a few gates in isolated academic labs to 50 qubits and up to hundred gates in commercial labs. IBM, Rigetti and Google are leading this effort with superconducting qubits. Xanadu uses large coherent cluster states of light to encode qubits and offers room-temperature, continuous-variable quantum computing. IonQ offers trapped-ion quantum computing. Microsoft is working on topological quantum computing.

Many companies are now providing **cloud access** to their hardware due to its bulky and non-portable nature. For example, in the case of superconducting qubits, fabricated chips with nanoscale circuits have to be kept in huge dilution fridges at cryogenic temperatures. These fridges are very costly and cannot be easily setup by anyone anywhere. They require complete mechanical, thermal and electrical insulation support and also depend on the limited Helium-3 and Helium-4 reserves of our planet. Hence, despite the time overhead in communicating over a cloud connection, cloud access makes current devices more accessible, efficient and cheaper to use. Moreover, many companies (especially new start-ups) are offering only quantum algorithms and/or quantum software services instead of building up their own physical quantum computers. They can pay the bigger companies for the cloud access and instead focus on solving new problems with the available devices. For example, Q-CTRL is an Australian company that helps other quantum computing companies improve the quality of their qubits by working on hardware error characterization. Another interesting example is the American

start-up QC Ware that plans to provide a unified cloud platform to connect to the various hardware providers.

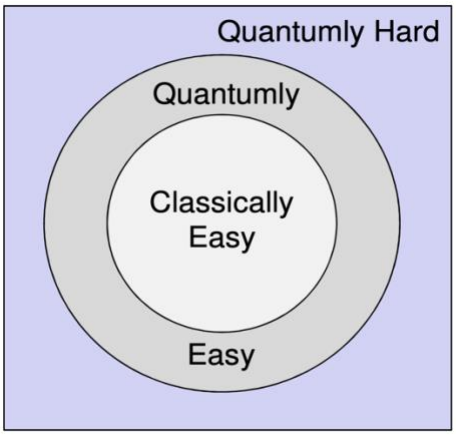

Figure 1.3 – Quantum computers are speculated to solve problems that can be really hard for classical ones. This figure is taken from John Preskill's 2012 paper that introduced the term "quantum supremacy" [5].

With 50 qubits, we start to enter a regime where classical computers cannot catch up anymore. Hence, we are currently entering what is called **Noisy, Intermediate-scale Quantum (NISQ)** era, as the Physicist John Preskill has aptly termed. We have noisy qubits without any error-correction and the quantum system is large enough to not be considered classically "easy". This opens up a whole new frontier of unexplored opportunities! First things first, we can finally start to get the first experimental evidence regarding **quantum supremacy**; the idea that quantum computers can perform **decisively** better than the best classical computer for a certain task, see Figure 1.3. To be more precise, by the term decisive here, we mean **polynomial** or **super-polynomial** (any function whose growth is larger than a polynomially growing function), as we will see in more detail later. This has been theoretically speculated as one of the distinguishing features between quantum and classical computers for a long time. Tasks like simulating quantum systems, factoring and Fourier transforms theoretically benefit from super-polynomial speedups. Aram Harrow and Ashley Montanaro put this very nicely in [10] as:

> *"Supremacy experiments can be thought of as the computational analogue of Bell experiments. Just as Bell experiments refute Local Hidden Variable models, supremacy experiments refute the old **Extended Church-Turing (ECT) thesis**, which asserts that classical computers can simulate any physical process with polynomial overhead".*

Hence, this is not only important to justify the huge investments into building fault-tolerant quantum computers, but it is also an important check to see that we understand quantum computational theory correctly. **If it turns out that we cannot find any task for which quantum supremacy can be proved, we might have to reformulate quantum mechanical theory.**

The physical verification for these claims, however, has been out of reach so far. An encouraging factor is that many computational problems do not depend on the existence of a universal quantum computer. One such example is **boson sampling**. To understand this concept, we can use the Galton board - commonly used in statistics demonstrations. Let us assume bosons (for example photons) are balls incident on a Galton board with multiple input funnels - as shown in Figure 1.4. The number of

collection buckets are assumed to be larger than the number of input funnels/photons. The pegs represent a linear interferometer. Let us say we run our Galton board experiment and record the output pattern in the buckets – for example the output pattern shown in Figure 1.4 is [0, 0, 2, 0, 1, 0, 0,1, 2, 0, 0, 0, 0, 0, 1, 1]. If we run this experiment enough times (maybe millions of times), we can collect the full information about the probability distribution of all possible output patterns. In other words, an output we get from a new run is essentially sampling from this probability distribution. Now, let us say that I give you a certain output pattern and ask you how probable it is that we observe that particular pattern in a new run. It turns out that this is a very hard question to solve for a classical computer and it gets harder as this Galton board becomes larger. In practice, it is not easy to implement boson sampling due to imperfect single-photon sources and photon-counting detectors. A way around using single, indistinguishable photons is using gaussian states of light (like coherent states) - as is done in **gaussian boson sampling**.

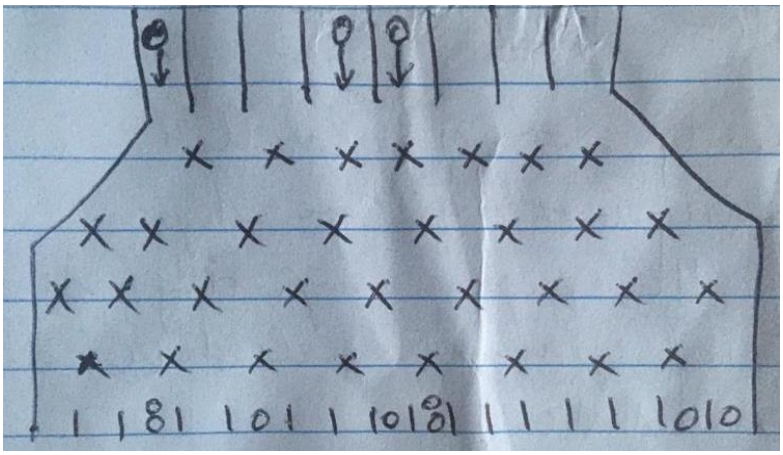

Figure 1.4 – Boson Sampling can be understood using a Galton board with multiple funnels. The crosses represent pegs that scatter the incident balls into various collection buckets.

As we saw before, controlling quantum systems and introducing qubit-qubit interactions while keeping them isolated is very challenging. It will be very encouraging to show that we can execute quantum circuits for a desired computation with reliably low noise. Due to its importance, there is currently a great effort led by early hardware providers to prove quantum supremacy for any task, irrespective of how useful that task is. However, these efforts will also help in answering the question John Preskill asked at that 25th Solvay Conference on Physics in 2012:

"Is controlling large-scale quantum systems merely really, really hard, or is it ridiculously hard?".

Any quantum supremacy experiment requires:

- a clear task
- a corresponding quantum algorithm
- a way to compare it to a classical algorithm (for verification of supremacy)
- verifying results and running the algorithm in the asymptotic limit (for large system sizes). This might require huge time and memory resources (supercomputers, GPUs, aggregate RAMs). This is because a general n-qubit state requires $O(2^n)$ space. Consequently, it is not easy to verify the results of such an experiment - if it is giving the correct results and really is classically hard.

Note that factoring is an exceptional example. If I give you a machine claiming that it is a quantum computer that can implement Shor's algorithm, you can easily verify this. If you run the integer X through the machine and it outputs the factors (a, b), we can quickly check that X = a* b on our laptops. This is because factoring belongs to the **NP (Non-deterministic Polynomial)** complexity class; given an efficient method to implement the problems in this class, their solution can be checked in polynomial time. Other tasks that we can use in quantum supremacy research are not this lucky. Many tasks were proven to be less classically challenging than originally proposed. Researchers from Physics and computer science backgrounds are, hence, currently working on defining well-suited tasks, developing efficient verification methods and identifying the exact quantum/classical boundary - especially for near-term devices. Examples of some verification methods include testing on smaller systems or using statistics. Moreover, techniques like approximate simulation, dynamic programming, Feynman paths (two-qubit gates are decomposed into single-qubit gates and circuit partitioned), rejection sampling and Tensor network contractions can be used to implement bigger classical simulations. These rapid developments are expected to bring about more results in quantum supremacy research in the near future.

The future

The first positive experimental claim to quantum supremacy was reported in 2019 by Google, although its validity was questioned by IBM and others, see [13], [14] in the references. Just as sampling from the output distribution in boson sampling is classically hard, sampling from the output distribution of a random quantum circuit is also hard for large circuits. A random circuit consists of application of gates randomly drawn from the **universal gate set** (for example, any two-qubit entangling gate with arbitrary single-qubit gates is **exactly universal**). This is the task that was implemented on a 53 superconducting-qubit state-of-the-art processor at Google. Great progress was shown in reducing errors in applying one and two-qubit gates. They applied roughly 1000 single-qubit and 400 two-qubit gates in each run, measured all qubits and stored the output pattern. For verification, smaller parts of the same circuit were simulated on classical supercomputers and extrapolated. Many of the tricks mentioned in the last section were employed to enhance classical computer's simulation and memory capabilities. They claimed that their processor took only 200 seconds to run and sample a random quantum circuit a million times and that the same task would take 10,000 years on the finest classical supercomputer! The second supremacy result was reported in December 2020. Researchers and collaborators at University of Science and Technology of China used Gaussian Boson sampling to demonstrate quantum supremacy. 50 indistinguishable gaussian packets of light (squeezed states) with a 100-mode interferometer were used to show that the sampling tasks that their quantum setup could do in seconds would take millions of years on the best classical computer.

So, what now? Where do we go from here? It is expected that the current NISQ devices will start to double in size and power every few years. What can we do with these devices? These are questions that are already starting to attract a lot of attention. With access to actual hardware of up to a hundred qubits and few hundred gates, both young and seasoned researchers have started to play around with these systems. Most likely, the quality and size of these devices is too low to implement the promising applications of drug and material design. However, important insights can be gained about the kind of noise present in the current hardware and how it affects any quantum computation. Hence, we can use NISQ devices to help us in engineering better devices, improving circuit architectures and in optimizing error-correction schemes as these devices scale. Another important thing to note is that the field of quantum computation started with analogue quantum simulation, but these general-purpose devices can essentially be used to implement any computation or simulate the dynamics of any quantum system, **digital quantum simulation**. Noise-resilient applications are being

actively researched. The most relevant one for us is quantum machine learning. In the long term, it is pertinent that the quality of gates and qubits becomes much better and **quantum random access memories** are developed.

Why do we have to work with noisy devices? Why don't we correct for errors already? This is because error identification and error correction are not trivial for quantum states. Due to the **no-cloning theorem**, quantum states cannot be perfectly copied. Additionally, errors cannot be identified without "looking" at the state, which in itself is destructive. Hence, simply using data redundancy like we do in classical computers is not possible. Error-correction protocols that use clever redundancy techniques for both bit and phase errors have been proposed over the last couple of decades. In order to implement fault-tolerance and error-correction, however, we might need millions of quality physical qubits.

1.3 Quantum learning

Quantum machine learning is an application of quantum computing. But what do we exactly mean by learning? What is a learning model and what does it learn? Let us dive a bit deeper into this.

Complexity classes

In the last section, we came across the term NP complexity class. Complexity is a way of studying how different **computational problems** consume resources (time or memory). The harder it is to solve a problem, the more resources it uses. Problems can be categorized into types. Some examples include:

- Decision problems are basically yes/no questions that output a single bit; "is x a prime number". These are most commonly used in deriving complexity theoretic arguments.
- Function problems are similar to decision problems, but the output can be a more complex object; "what is the binary representation of a decimal number x"
- Search problems are similar to function problems but there might be more than one correct answer; "find object y in object x"
- Counting problems are related to questions based on counting the number of answers; "how many cycles are present in graph G"
- Optimization problems are concerned with finding the best answer to a question; "what is the best way to divide a graph G into two smaller graphs"

The other ingredient is the type of **computational model** that is being used to solve the problem. This is the abstract representation of a physical computing device and its underlying working principles. Some examples include:

- Deterministic Turing machine
- Non-deterministic Turing machine
- Probabilistic Turing machine

- Quantum Turing machine

Conventionally, asymptotic behaviour with respect to input size is used to compare various algorithms by using the **Big O notation**. For example, an algorithm that uses a constant amount of resources no matter the size of input, is said to scale as O(1). Figure 1.5 shows some of the popularly used functions.

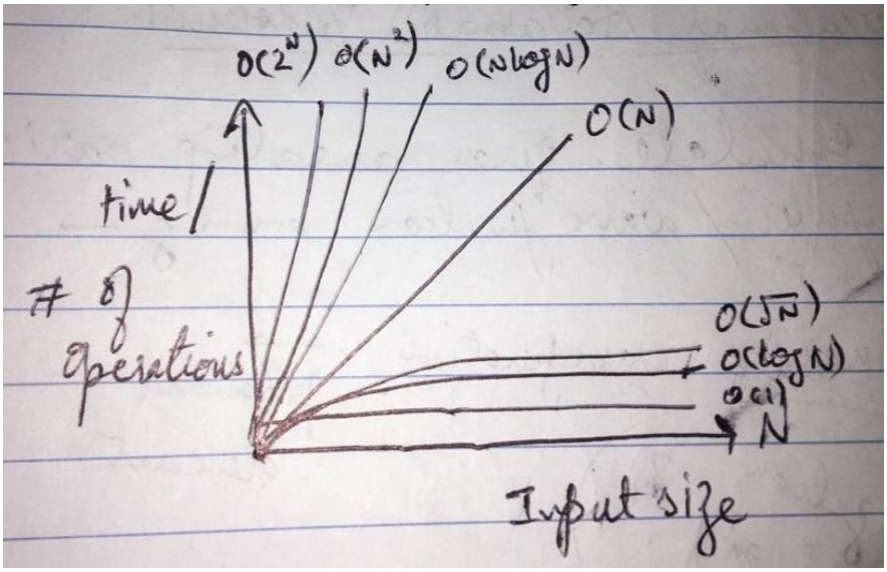

Figure 1.5 – Functions commonly used to derive upper bounds when using the Big O notation.

The complexity of quantum algorithms can similarly be quantized using the number of operations/gates used. To facilitate comparison, a universal gate set can be chosen. It does not matter which set you choose as you can convert one set to the other using a constant amount of resources anyway. All gates can then be decomposed in terms of these "elementary" gates and counted. An algorithm is considered **efficient** if its running time, let us call it T, is a polynomial function of input size N, i.e. $T(N) = f(N^p)$, where f is some polynomial function and p is a constant that represents the degree of the polynomial function. Defining **complexity classes** is a way to group similar problems that might require similar amounts of resources. In Table 1, we describe some of the popular classical and quantum complexity classes. The classes we mostly work with are **P** and **BQP** as they contain the efficient algorithms. Space complexity can also be considered but we will not go into its details here.

Class	**Classical/Quantum**	**Brief Description**
P	Classical	Can be solved by a deterministic Turing machine in polynomial time
EQP	Quantum	Can be solved by a Quantum Turing machine in polynomial time with probability 1
BPP	Classical	Can be solved by a probabilistic Turing machine in polynomial time with probability at least 2/3
BQP	Quantum	Can be solved by a quantum Turing machine in polynomial time with probability at least 2/3

NP	Classical	Solution can be checked by a deterministic Turing machine in polynomial time
MA	Classical	Solution can be checked by a probabilistic Turing machine in polynomial time with probability at least 2/3
QMA	Quantum	Solution can be checked by a quantum Turing machine in polynomial time with probability at least 2/3
EXP	Classical	Can be solved by a deterministic Turing machine in exponential time

Table 1.1 – Some popularly used classical and quantum time complexity classes. Note that P ⊆ BPP ⊆ BQP ⊆ QMA ⊆ EXP, where A ⊆ B means that A is a subset of B.

Classical computational learning

Let us consider the following scenario: David is trying to figure out a certain function **f** that Geoffrey already knows. One way to learn how it looks is to guess it by looking at function values for various inputs **x** (for example, $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, …). As shown in Figure 1.6, Geoffrey can mark the output of the function, **f(x)**, for different values of **x** that David shouts out. After many such back-and-forth rounds between them, David can start to see how the function **f** looks like. How many questions would David have to ask in order to guess the correct **f**? How much time and memory would it take? What functions can he learn? What are his limitations? These are the questions that underlie **learning theory**. This particular method of learning by "querying" a function oracle is called **exact learning**, where in our example Geoffrey is the function oracle.

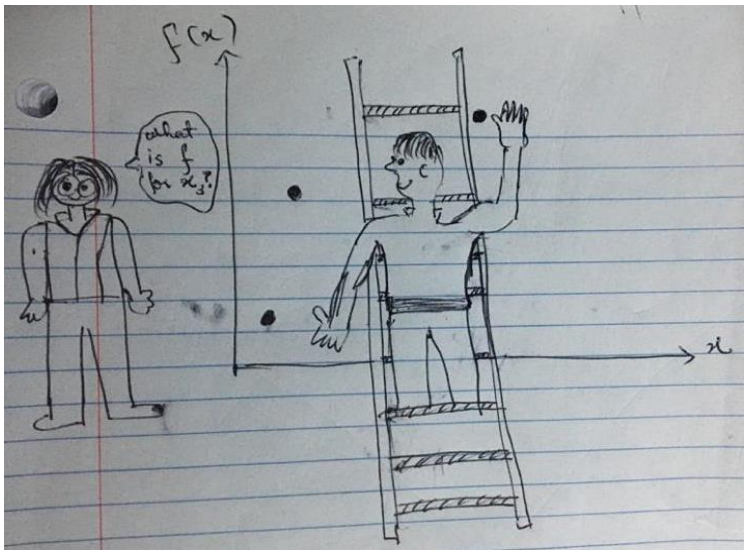

Figure 1.6 – David tries to learn how function f looks like by looking at its output value for different x. For every value of x that David shouts out, Geoffrey - the function oracle - can plot the function output.

If function **f** belongs to a certain known class of functions, then we are essentially querying this class to figure out which member of this class is the target function. These queries are fittingly termed **membership queries**. For example, classes of Boolean functions that map n-bit strings to 1-bit strings (called concept classes) are frequently used in learning theory. A first safe guess is that to get

the complete picture of a **concept** that acts on n-bit inputs, we should query it on all 2^n possible n-bit strings. Can we do any better? Analogous to time complexity that we saw earlier, the study of the required number of queries is termed **query complexity**. A concept is called **efficiently learnable** if it can be learned in a polynomial number of queries (the number of queries grows polynomially with the system size). Examples of concept classes that are polynomial membership-query learnable include some restricted classes of DNF (or CNF) formulas. DNF (CNF) stands for Disjunctive (Conjunctive) normal forms, respectively. This is just a way of writing Boolean functions using only additions and multiplications on grouped Boolean variables. For example, $(b1 \wedge b2) \vee (b3 \wedge b4) \vee (b5 \wedge b6)$ is a DNF formula, where $b1$ to $b6$ are Boolean variables. This is an example of a 2-DNF Boolean expression; when we say k-DNF, it means that each grouped term contains k Boolean variables. General learnability of DNF and CNF classes remains an open question and is widely studied as any Boolean function can be expressed in this form. Examples of exact-learnable DNF subclasses include class of log n-DNF ∩ log n-CNF and monotone DNF (DNF expressions that can not contain any Boolean variable with negation) Boolean formulas. Other examples include monotone decision lists (list of pairs of Boolean functions). Note that, in general, a concept that is **polynomial-query learnable** is not necessarily **polynomial-time learnable**.

The technical term for the function we are trying to learn is **target concept** and it is usually labeled by the letter **c**. In reality, we do not always have access to an expert oracle like Geoffrey who knows the target concept **c** and can help us learn it over an instance space χ (space of input data). The idea of machines learning concepts from data with a deduction procedure was generalized by Leslie Valiant in 1984. He introduced the **Probably Approximately Correct (PAC) learning** method. Assuming a concept class like before, we are given data points according to an unknown distribution **D** and access to an oracle that can tell us their output labels (0 or 1). Let us call these **PAC samples** (x, c(x)). In literature, these are technically called **examples** and the oracle is called a **random example oracle** but let us not use these terms here to avoid any confusion. Next, a hypothesis set of functions $\boldsymbol{H}$ is chosen. This is a set of guess functions and can be finite or infinite. For example, a neural network with infinitely different parameter settings represents infinitely many guess functions. For now, we will assume it is finite. For **h** ∈ $\boldsymbol{H}$, the learner compares **h(x)** to the given label **c(x)** for various instances **x** ∈ χ. Note that we assume that all PAC samples are independent and identically distributed (i.i.d). This is crucial for both learning and generalization:

- we will be learning over a “representative” set of points
- the learned function will only generalize to unseen datapoints (the whole instance space, in principle) if we assume that both seen and unseen data have the same distribution, albeit unknown.

How many data points should we check a hypothesis on? As the name of the learning method suggests, the learned hypothesis (usually called **g**) should approximate the target function up to some tolerated error (of our choice) with high probability. The probability that any hypothesis **h** and **c** differ by more than our tolerance is upper bounded by **Hoeffding’s inequality**, which basically says that if we want to set a small tolerance for error, we have to use more PAC samples during the learning procedure. Moreover, the complexity of the hypothesis set (number of hypotheses in $\boldsymbol{H}$) also effects learning; the more hypotheses you have to check, the more errors you can accumulate. This complexity is characterized by the **Vapnik-Chervonenkis (VC) dimension**. This is the largest number of datapoints for which $\boldsymbol{H}$ can find all possible dichotomies (all possible ways of dividing the set of inputs based on their labels). Hence, this parameter directly reflects the complexity of the concept class; if the data looks very irregularly distributed, VC dimension will be higher, and we will need more samples to get to know the target function.

Why not use an infinite $\boldsymbol{H}$ every time? Other than the cost factor, we also do not want to **overfit** the data. As we will always be using a finite subset of data from the instance space to make our guess of the target function, there is no guarantee that any new datapoints will behave the same way as our "training" subset. Hence, the learned hypothesis **g** can "generalize" with high probability by using the right number of PAC samples and the right size of $\boldsymbol{H}$ (using polynomial number of unique dichotomies or a finite VC dimension). The number of PAC samples required to ensure a certain accuracy and confidence over all target concepts and distributions is termed **sample complexity**. We can guess now that sample complexity $\boldsymbol{S} \propto \frac{\boldsymbol{d\delta}}{\boldsymbol{\epsilon}}$, where $\boldsymbol{d}$ is VC dimension, $\boldsymbol{\epsilon}$ is the error tolerance and $\boldsymbol{\delta}$ is the confidence. A polynomial sample complexity implies **PAC-learnability**. This holds for any learning algorithm, target concept, input data distribution and error tolerance. Note that we may actually never know the target concept as $\boldsymbol{\chi}$ can be infinite and it is also possible that $\mathbf{c} \notin \boldsymbol{H}$.

PAC learning lays the foundation for classical machine learning, specifically supervised machine learning. However, the latter lacks unlimited access to the instance space and sample oracle. Moreover, PAC model gives the worst-case sample complexities over all concepts and distributions on the instance space. In practice, we are only given a fixed number of noisy labelled samples drawn from some unknown joint probability distribution $\mathbf{D}_{\mathbf{x,y}}$ without any notion of a target concept, where **y** is commonly used to represent the output/label. Hence, depending on the application, we choose a learning model (hypothesis set + learning algorithm) and use **risk minimization** on custom error functions (instead of using probability of making errors on drawn examples) such that we get a small training and generalization error for the given data. This approach falls along the lines of statistical learning theory. Many other learning models like agnostic model, statistical query model and others based on algorithmic learning theory and Bayesian inference have also been theorized.

Quantum computational learning

The development of the corresponding quantum learning theory is still an active area of research. A popular conception is that QML will be able to provide speed-ups in machine learning by reducing time, sample or query complexity for various learning algorithms. The theoretical and empirical evidence for this claim, however, is still lacking. The first obvious thing we can say is that any concept class that is quantum efficiently learnable must also be classically efficiently learnable. This should hold for any learning model; exact, PAC or any other. Recall that in these models, learnability is defined in terms of polynomial number of queries/samples. Time complexity is a different factor to consider, and we will come to this point in more detail later.

Let us first look at the **quantum exact learning** scenario. Again, we can assume a quantum oracle that knows all target class functions, can answer membership queries if we give it an input and helps us exactly learn the target function with high probability. The inputs are quantum states $|x\rangle$. If we use concept classes like before, 2^n n-bit input strings become 2^n computational n-qubit basis states; $|00\ldots0\rangle, |00\ldots1\rangle$ to $|11\ldots1\rangle$. We can read off the function label **c(x)** when we use quantum input states by using an ancilla qubit (with a known bit value) as shown in Figure 1.7.

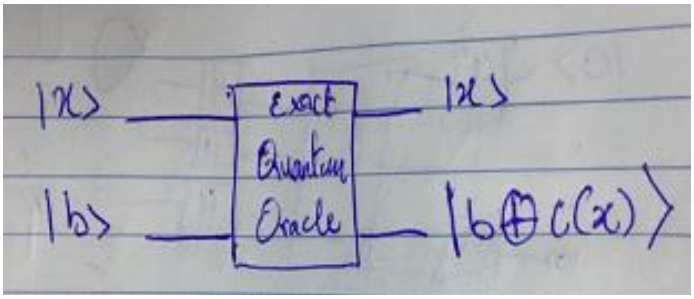

Figure 1.7 – A clever method to get answers from the quantum membership query oracle in the exact learning model, where ⊕ represents bit-wise addition modulo 2 operation (like a XOR gate).

Now, for a quantum oracle in this setting, given a superposition on the input states, we can get an output superposition over all corresponding labels. This might initially suggest that the quantum membership-query complexity can be significantly smaller than the classical one because a single query to the oracle gets us all the information we need. However, let us not forget that, firstly, upon "looking" at the oracle output, we can only "see" one label at a time. Secondly, the query complexity also depends on the complexity and size of the target class. If there are a lot of functions to check or if it is hard to distinguish between the different member functions in the class, we have to make more queries to exactly pinpoint the target one. It was formally proven by Rocco Servedio and Steven Gortler in the early 2000s that quantum learning models cannot achieve better than polynomial speedup in query complexity for the exact learning setting compared to classical learning models. For example, if we convert exact learning problem into an unstructured search problem over a space of functions, we can get polynomial speedup using Grover's search algorithm; we use $O(\sqrt{N})$ instead of $O(N)$ queries, where $N=2^n$ for $\{0,1\}^n$ instance space. Another example is quantum Fourier sampling using Bernstein-Vazirani algorithm on a class of linear functions defined on n-bit strings; we use O(1) instead of $O(N)$ queries. This generalizes to any class of k-Fourier-sparse n-bit Boolean functions. Note that a polynomial speed up might not seem impressive at first but for applications like data science, it can potentially reduce data requirements from billions to millions.

We see similar results in the **quantum PAC learning** model. This was proposed by Nader Bshouty and Jeffrey Jackson soon after the Shor algorithm came out in the late 90s. The **quantum PAC sample** can now be in a superposition over all the PAC-samples **(x, c(x))** in the instance space with the data probability distribution **D**, i.e. $\sum_{x\in\{0,1\}^n}\sqrt{D(x)}|x,c(x)\rangle$**.** However, keep in mind that it can be costly to prepare and measure many copies of input states in superposition from classical data. This is sometimes called the "input problem". It is, however, possible that they are naturally generated from some quantum process or experiment, in which case we might benefit more from quantum learning algorithms. A hypothesis set consists of measurements on these states. Again, we can set custom error tolerance and confidence level for approximately learning the target concept. The sample complexity is now in terms of the number of copies of quantum PAC-samples required. It has been shown that quantum PAC learning models cannot achieve better than constant speedups in sample complexity compared to classical ones.

At first, it might seem that there is no learning problem which can be solved using significantly fewer quantum queries or samples than classical ones. In fact, under certain assumptions on distributions, quantum sample complexity can be further improved. Moreover, learning is not limited to functions in learning theory. Some learning problems are set up to learn the unknown data distribution instead of a target concept over the PAC samples (**density modelling**) and some only want to learn how to produce similar looking samples (**generative modelling**). For example, it was recently shown that for generative modelling, the distribution concept class (class of all probability distributions over the n-bit Boolean instance space) of discrete probability distributions is quantum PAC-efficient learnable

but not classical PAC-efficient learnable. Moreover, many areas and applications of Physics require learning of a quantum state (**quantum state tomography**), a Hamiltonian or a quantum channel (**quantum process tomography**). For example, to exactly learn all the amplitudes of a n-qubit quantum state, we need $O(2^n)$ measurements. This will scale exponentially as the number of qubits grows. In 2007, Scott Aaronson extended the PAC learning model which approximated the state up to an error using O(n) measurements. Here, the hypothesis set was chosen to be a set of two-outcome measurements instead of functions and for every measurement (hypothesis), the expectation value of the state was checked instead of the label, see [28]. In another recent example, models that want to learn to predict the outcome of any physical process (represented by unknown quantum CPTP process $\mathcal{E}$) on a given classical data were considered. It was shown that although both classical and quantum learning models required similar number of circuit runs and had similar *average* error in prediction, exponential reduction in the number of quantum circuit runs was possible for *worst-case* prediction, see [29]. These are theoretical proposals, however, and require functional quantum memories and fault-tolerant quantum computers for their verification. We will look at some practical applications of machine learning in Physics in the next section.

What about time-complexity? The time it takes to run an algorithm can also be used to separate efficient learnability of classical and quantum learners. In other words, it is possible that a class has polynomial sample complexity, but its learning algorithm is highly time-consuming. Also note that, unlike sample complexity, a class that has a polynomial-time quantum learning algorithm does not guarantee a polynomial-time classical analogue. These learning problems belong to the **BQP** complexity class. Hence, this might be one of the promising areas where we can hope to find speedups in quantum machine learning! Over the last decade, a few such algorithms have already been proposed that suggest up to exponential speedups in time complexity. These include QML algorithms based on Grover and HHL algorithms for clustering, classification, principal component analysis and pattern matching. However, these speedups are only promised under certain conditions, some of which are not practically achievable, especially in the near-term. Below we mention a couple of theoretical examples from the few known examples of such classes. For example, quantum PAC-learning the class of k-term DNF Boolean formulas from uniformly distributed data is time-efficient; this is not possible classically without membership queries, as we saw in the last section. Another example is learning classes that depend on factoring – like classes built on factoring Blum integers or classes built on one-way functions popularly used in cryptography - where Shor' algorithm can provide speedups to the quantum PAC learner. For the interested readers, see [23]-[27] and references therein for more examples and details of other learning models like quantum agnostic model and quantum statistical query model. The latter is especially interesting as it better relates to the current implementation of quantum learning algorithms. Many open questions in quantum learning theory remain like extending the stated claims to other function classes, the generalization bounds of quantum learning models, the role of entanglement in learning, the general learnability of NISQ devices and potential advantages in learning when classical data is encoded into quantum states.

1.4 Categories within quantum machine learning

There is no one clear definition of quantum machine learning but we can make our lives much easier by defining subcategories in it. Let us try to explain this by continuing with our story. Let us say that before leaving, David and Geoffrey decide to stay in touch and collaborate in their research. At the Vector institute, Geoffrey works on developing theory and applications of machine learning using classical data. On the other hand, David runs a lab where his students study quantum mechanical systems to understand the laws of nature. Now, they can collaborate in many ways. One way they can collaborate is by exploiting the vast classical machine learning techniques to discover the underlying connections in quantum data (data produced in quantum experiments and theory). This line of work

is generally termed **ML-assisted quantum Physics**. In another way, Geoffrey can call up David and communicate his data to him, who can try if seeing this data through the lens of quantum mechanics can help learning form it in any way. This approach to QML research is generally termed **quantum-enhanced machine learning**. However, If David applies machine learning algorithms based on quantum mechanical principles to quantum data, this comes under the umbrella of what is generally termed **fully-quantum QML**. Let us look at these in a bit more detail.

ML-assisted Quantum Physics

Over the last decade, data driven learning has found a strong foothold in many areas of science including Physics. From decoding data produced at CERN to developing new materials and discovering Black holes, the applications of machine learning in Physics are numerous and fast-increasing. We will mention only a few examples here from quantum Physics that use various supervised, unsupervised and reinforcement machine learning techniques.

- For example, to develop new drugs, we need to know the candidate molecules: from their energy levels to their chemical and thermodynamic properties. However, solving the time-independent Schrodinger equation - that provides us with the energy level diagrams - is analytically possible for only a few small molecules. Experimentally verifying the energy scales and chemical properties like solubility and toxicity of millions of molecules can be very costly. In the last decade, many researchers have employed machine learning models for help in this regard. For example, neural networks have been used with huge success to predict atomization energies of organic molecules by training them on a subset of molecules with known properties. The resulting computational cost and accuracies have been much better compared to traditionally used density-functional theory. This has been extended to **virtual screening** for drug development where chemical space is explored to identify new drugs based on structural and chemical similarity to known drugs. Graphical neural networks have shown a lot of promise here. This can save pharmaceutical companies billions of dollars as the chemical space is extremely large and experimental search for new drugs can get very expensive.

- Other than quantum chemistry, condensed-matter and many-body Physics has also been benefiting from machine learning recently. Many-body quantum systems are systems with multiple interacting quantum particles. We come across these systems in many natural settings; for example, when studying strongly correlated many-body systems (like superconductors and magnets), investigating phases of matter and while designing new materials and drugs. Calculating and even simulating the huge wavefunction and its dynamics becomes exponentially harder with the system size. Note that this can be a hard task even for quantum computers. Exactly simulating the dynamics of a gas of electrons, for example, can require impossible amounts of resources on a quantum computer as well. In 2017, two ground-breaking results came out where neural networks were successfully used to represent many-body wavefunctions, to study their dynamics and to identify their phase transitions, [33-34]. These are highly non-trivial tasks to solve. The nodes in the input layer of a neural network were associated with the many-body system configuration and the weights in the network were allowed to be complex numbers. As an example, let us consider an interacting spin system spread on a 2D grid (2D Ising model). At low temperatures, the spins will interact to align with each other and result in ferromagnetism. If we slowly increase the temperature, we disturb the spins and effect the resulting macroscopic magnetism. A transition point occurs at some critical temperature, where the system becomes paramagnetic. By training the network with data of spin configurations above and below the transition point, neural networks could be taught to identify the phase transition. This can play a huge role in assisting physicists when

it comes to complicated quantum systems for which analytical and/or approximate solutions are very hard to obtain. Consequently, these "self-driving laboratories" are becoming increasingly popular in materials and Physics research. See the review [35] for more details and examples. Note that variational quantum algorithms also offer huge promise for many-body and quantum chemistry problems mentioned above.

- Machine learning has many uses in quantum computing as well. One important application relates to noise. This can take various forms like predicting, correcting and mitigating noise. For example, if we take a near-term quantum device and implement a series of gates, its results will have noise which is hard to track; we cannot know with certainty which qubit and which gates went through what noisy processes, like gate implementation errors or qubit dephasing etc. Moreover, the hardware noise (like stray fields, crosstalk, temperature gradients, charge fluctuations etc.) can vary across both individual runs and devices. Hence, noise models for even a simple circuit can be quite complicated and are hard to formulate mathematically. Machine learning models can be used to learn and predict any device-specific noise models by first gathering data from many experimental runs and comparing it to expected data. The learned noise can then be accounted for in the observed results. These techniques can also be extended to implement *predicted* feedback and correct qubits against decoherence in real-time without having to do any measurements, see [37] as an example. This can play an essential role in improving and scaling near-term quantum devices in the near future. Furthermore, many proposals for using deep learning and reinforcement learning techniques to uplift **quantum error-correction** have come out in the last 4-5 years. Error-correction protocols for quantum information suggest that a single bit of information is encoded into multi-qubit topological states (called **surface codes**). Measurements are done on these qubits (usually on a subsystem) to reveal what errors could have happened. The non-trivial task of learning the error sequence and then correcting it has been shown to be successfully exported to machine learning for various "encodings".

Other applications include automating both the optimization and design of quantum experimental setups (manual quantum control is complicated, often inaccessible and time consuming), state tomography (which needs exponentially many measurements otherwise), assisting in designing optimal measurement strategies and estimating unknown parameters from measurement results (quantum metrology), aiding in classical simulation of quantum systems and discovering new ways of expressing Physical laws. See reviews in [30] and [41] for more details.

Quantum-enhanced machine learning

The term "quantum-enhanced machine learning" is often used when quantum devices run classical data for machine learning. However, simply running classical data through a quantum learning agent does not necessarily guarantee any performance improvements as we already saw in the last section. Hence, under this field, here we will only look at quantum learning algorithms that offer speedups (in computational complexity) to known classical machine learning algorithms. These speedups majorly come from representing $\boldsymbol{N}$ bits of classical information using $\mathbf{log_2}\,\boldsymbol{N}$ qubits and from incorporating **quantum Basic Linear Algebra Subroutines (qBLAS)** like Fourier transform, inner products, matrix diagonalization, exponentiation and inversion that can be implemented exponentially faster on quantum computers. Variants of Grover's search are also popularly used in this regard. Note that these speedups might not be applicable in the near-term as qBLAS generally require a universal fault-tolerant quantum computer, large number of "perfect" qubits and access to a QRAM.

Let us look at some examples:

- **Quantum Principal Component Analysis (QPCA)**: Principal Component Analysis or dimensionality reduction is commonly used to preprocess data in classical machine learning. Rather than using all given features, only the most significant features are chosen. This process involves looking at the correlation between all given features and choosing (or making) ones that carry most information about the data and are least correlated with each other. This requires computing the eigen values of the covariance matrix of all features which scales as $O(d^2)$, where d is the dimension of each feature vector. In 2013, Physicist Seth Lloyd and his colleagues proposed a clever algorithm for QPCA where the quantum ensemble $\boldsymbol{\rho}$ that represents embedded classical data is used as the covariance matrix. Its eigen-decomposition is then done by exponentiating $\boldsymbol{\rho}$ and using this in the quantum phase estimation routine. When measured, the largest eigenvalues are observed with higher probability. QPCA can be performed in time polynomial in log(d) [42].

- Classification and clustering: Given labelled data, Support Vector Machine for binary classification finds two parallel lines (hyperplanes) that mark maximally separating area between the two classes. Given M d-dimensional data vectors, the algorithm essentially converts this into a constrained problem that results in a set of linear equations involving dot products of pairs of all data vectors. This scales polynomially in O(Md). If the data is not linearly separable, it is mapped to higher dimensional spaces using **feature maps**. For **Quantum Support Vector Machine (QSVM)**, given we can efficiently prepare, store and retrieve classical data as quantum states in a QRAM, **HHL algorithm** can provide exponential speedup in solving these linear equations [43]. A QSVM variant based on Grover's algorithm has also been proposed. A similar speed up is seen for unsupervised clustering algorithms. On identifying a centroid for each cluster, we assign each data points by assessing how close or similar it is to the centroid. Here, we can exploit quantum computers for their faster inner products and distance measures (scales as O(logd) [44].

- **Quantum Boltzmann machine (QBM)**: An unsupervised probabilistic learning model that benefits from quantum principles is the Boltzmann machine (BM). It is basically an undirected graph/neural network of random binary variables whose output mimics Boltzmann probability distribution and whose energy mimics that of an Ising model in thermal equilibrium. Optimizing this algorithm corresponds to learning edge weights that produces low energy outputs for the desired task, which is usually to estimate the data distribution or to generate samples similar to input data distribution. Training BMs is generally quite hard as calculating partition functions can be very expensive. Sampling methods are often used for its estimation. In QBMs, the nodes become Pauli operators resulting in the network representing a many-body quantum Hamiltonian with a density matrix based on the corresponding partition function. QBMs can only be exactly evaluated on quantum computers, simulating QBMs on classical computers is very hard, even with sampling methods. Hence, variational approaches are used to solved these. These models have been proposed to be more expressive than their classical counterparts and are an active area of research.

- Gradient-based optimization: this technique is vital to the working of various classical machine learning models like deep neural networks. Moreover, as stated before, most VQAs also use gradient descent/ascent optimization and export their parameter optimization to classical computers. Quantum gradient algorithms do exist but generally require fault-tolerant quantum computers. Given access to a large number of qubits and assuming smooth functions, a **Quantum gradient-descent algorithm** that uses quantum Fourier transform and Grover's search to compute gradients was recently proposed [46]. For d-dimensional data

points, it offers quadratic speedup both in d and the number of random initializations of the algorithm required to check for local optima.

- One of the most significant QML algorithms are **Variational Quantum Algorithms (VQAs)** (sometimes also termed Hybrid Quantum-Classical algorithms). These directly borrow from the working principle of classical machine learning where a parametrized model is optimized using data to learn a certain task. Here, the model is a **parametrized quantum circuit (PQC)**, and the task is to generally construct/learn a desired quantum state or Hamiltonian. Parameter optimization tasks can be exported to classical computers - although the usefulness of this export and possible speedups is still under research. This has led to the birth of **quantum variational learning**. The elements of variational learning include quantum circuits, quantum embeddings (how to represent classical data as quantum states) and quantum. Some notable VQAs include Variational Quantum Eigensolver (VQE), Variational Quantum Simulators (VQS) and Quantum Approximate Optimization Algorithm (QAOA). Let us also note that machine learning is not limited to inspiring PQCs. Interestingly, machine learning techniques can also be directly employed to improve the training of VQAs. For example, classical recurrent neural networks were recently used in helping find good initialization strategies for quantum neural networks [40].

Other examples include efficient data fitting for "Big data" [47] and quadratic improvements in learning efficiency - the number of interaction steps between the learning agent and environment needed to learn to obtain the rewards with high probability - in reinforcement learning [48].

1.5 References and Further Reading

[0] M.A. Nielsen and I.L. Chuang. Quantum Computation and Information. Cambridge University Press, second edition, 2000.

QML reviews and basics

[1] P. Wittek, Quantum machine learning: what quantum computing means to data mining (Academic Press, 2014)

[2] M. Schuld and F. Petruccione, Supervised Learning with Quantum Computers, Vol. 17 (Springer, 2018)

[3] Biamonte J. et al. 2017. Quantum machine learning. Nature 549, 195–202. (doi:10.1038/nature23474)

[4] Carlo C. et al, 2018. Quantum machine learning: a classical perspective. Proc. R. Soc. A.47420170551 (http://doi.org/10.1098/rspa.2017.0551)

NISQ devices

[5] J. Preskill, Quantum computing and the entanglement frontier, 25th Solvay Conference on Physics (2011), arXiv:1203.5813

[6] Preskill, J. Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018) (https://doi.org/10.22331/q-2018-08-06-79)

[7] Shor, P.W. (1994). "Algorithms for quantum computation: discrete logarithms and factoring". Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press: 124–134(doi:10.1109/sfcs.1994.365700)

[8] L. Grover, "Quantum mechanics helps in searching for a needle in a haystack", Phys. Rev. Lett. 79, 325 (1997), arXiv:quant-ph/9706033, https://doi.org/10.1103/PhysRevLett.79.325.

[9] DiVincenzo, David P., "The Physical Implementation of Quantum Computation". Fortschritte der Physik. **48** (9–11): 771–783 (doi:10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E)

[10] Aram W. Harrow and Ashley Montanaro, Quantum Computational Supremacy, https://arxiv.org/pdf/1809.07442.pdf

[11] S. Aaronson and A. Arkhipov, The computational complexity of linear optics, arXiv:1011.3245 (2010).

[12] Arute, F., Arya, K., Babbush, R. *et al.* Quantum supremacy using a programmable superconducting processor. Nature 574, 2019. https://doi.org/10.1038/s41586-019-1666-5

[13] https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/

[14] Edwin Pednault et. al, Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits. https://arxiv.org/pdf/1910.09534.pdf

[15] Sergio Boixo et. al, Characterizing Quantum Supremacy in Near-Term Devices, https://arxiv.org/pdf/1608.00263.pdf

[16] Igor L. Markov et. al, Quantum Supremacy Is Both Closer and Farther than It Appears, https://arxiv.org/pdf/1807.10749.pdf

[17] Riling Li et. al, Quantum Supremacy Circuit Simulation on Sunway TaihuLight, https://arxiv.org/pdf/1804.04797.pdf

[18] Han-Sen Zhong et. Al, Quantum computational advantage using photons, Science, Vol 370, 2020

Quantum learning

[19] Bernstein, E., and Vazirani, U. Quantum complexity theory. SIAM Journal on Computing 26, 5 (1997), 1411–1473

[20] L. Valiant. A theory of the learnable. Communications of the ACM, 27(11):1134–1142, 1984

[21] Michael Kearns and Umesh Vazirani, An introduction to computational learning theory, MIT Press, 1994

[22] Abu-Mostafa, Yaser S., Learning from Data: A Short Course.

[23] Hellerstein, L., Raghavan, V., Pillaipakkamnatt, K., & Wilkins, D., How many queries are needed to learn? Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 1995.

[24] Srinivasan Arunachalam, Quantum Algorithms and Learning Theory, PhD thesis, Universiteit van Amsterdam, 2018.

[25] Nader Bshouty and Jeffrey Jackson, Learning DNF over the uniform distribution using a Quantum example oracle, COLT, 1995

[26] Servedio, R. A., and Gortler, S. J., Equivalences and Separations Between Quantum and Classical Learnability. SIAM Journal on Computing, 33(5), 1067–1092, 2004

[27] Srinivasan Arunachalam et. al, Quantum statistical query learning, 2020. https://arxiv.org/pdf/2002.08240.pdf

[28] Scott Aaronson, The Learnability of Quantum States, 2007 https://arxiv.org/pdf/quant-ph/0608142.pdf

[29] Hsin-Yuan Huang, Richard Kueng, and John Preskill, Information-theoretic bounds on quantum advantage in machine learning, 2021 https://arxiv.org/pdf/2101.02464v1.pdf

Categories in QML

[30] Giuseppe Carleo et al., Machine learning and the physical sciences, Rev. Mod. Phys. 91, 2019

[31] M Rupp, A Tkatchenko, KR Müller, OA Von Lilienfeld, Fast and accurate modeling of molecular atomization energies with machine learning, Physical review letters, 2012.

[32] Wen Torng and Russ B. Altman, Graph Convolutional Neural Networks for Predicting Drug-Target Interactions, J. Chem. Inf. Model, 2019

[33] Giuseppe Carleo and Matthias Troyer, Solving the quantum many-body problem with artificial neural networks, *Science,* Vol. 355, 2017

[34] Juan Carrasquilla and Roger G. Melko, Machine learning phases of matter, Nature Physics volume 13, 2017.

[35] Juan Carrasquilla, Machine Learning for Quantum Matter, 2020, https://arxiv.org/pdf/2003.11040.pdf

[36] Alexander Zlokapa and Alexandru Gheorghiu, A deep learning model for noise prediction on near-term quantum devices, 2020, https://arxiv.org/pdf/2005.10811.pdf

[37] Sandeep Mavadia et al., Prediction and real-time compensation of qubit decoherence via machine learning, Nature communications, Vol. 8, 2017.

[38] R Iten, T Metger, H Wilming, L Del Rio, R Renner Discovering physical concepts with neural networks, Physical Review Letters, 2020.

[39] William J. Huggins, *et al.* "Efficient and noise resilient measurements for quantum chemistry on near-term quantum computers." arXiv:1907.13117 (2019).

[40] Guillaume Verdon et al., Learning to learn with quantum neural networks via classical neural networks, arXiv:1907.05415, 2019

[41] Kishor Bharti et. al., Noisy intermediate-scale quantum (NISQ) algorithms, arxiv:2101.08448, 2021.

[42] Lloyd, S., Mohseni, M. & Rebentrost, P., Quantum principal component analysis, Nature Physics 10, 2014.

[43] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. 113, 2014.

[44] Seth Lloyd, Masoud Mohseni, Patrick Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arxiv:1307.0411, 2013.

[45] Nathan Wiebe, Leonard Wossnig, Generative training of quantum Boltzmann machines with hidden units, arxiv: 1905.09902, 2019.

[46] András Gilyén, Srinivasan Arunachalam, Nathan Wiebe, Optimizing quantum optimization algorithms via faster quantum gradient computation, SODA '19: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, January 2019.

[47] Nathan Wiebe, Daniel Braun, and Seth Lloyd, Quantum Algorithm for Data Fitting, Phys. Rev. Lett. 109, 2012

[48] Vedran Dunjko, Jacob M. Taylor and Hans J. Briegel, Quantum-enhanced machine learning, Phys. Rev. Lett. 117, 2016

[49] Nathan Wiebe, Key questions for the quantum machine learner to ask themselves, New J. Phys. 22, 091001, 2020