Chapter 1
Introduction to the course

Welcome to the course on quantum computing and machine learning. Quantum computing describes information processing with devices based on the laws of quantum theory. On the other hand, machine learning is the science of making computers learn how to make inferences or predictions from data. Both quantum computing and machine learning are anticipated to play an essential role in shaping our society.

In this chapter, we give the forest view of the course without being highly explicit in explaining everything; as anything we deal with here finds another mention in the later chapters, where they are elucidated in their fine details.

The standard way to start this chapter is by explaining quantum computing in section 1 and then following this up with the discussion of machine learning in section 2. Section 3 gives more details about the new field - quantum machine learning. Finally, we review what we will discuss in the course in section 4 and supplementary reading in section 5. So, let us start this exciting journey now!

1 Quantum computing

Currently, quantum computation is one of the hottest scientific topics of the twenty-first century. It involves operating quantum entities[1] (electrons, photons, or single atoms) that are in two states simultaneously – precisely like Schrodinger's legendary 'dead and alive' cat. This quantum entity is the basic unit of quantum computing and is referred to as a qubit (a short for 'quantum bit' and pronounced 'cubit' like the biblical unit of length).

Quantum computing rose to prominence after the development of two quantum algorithms which established the advantages of quantum computation over the classical one. Shor's algorithm (1994) endangered our current encryption methods, leading to a significant interest of the military and big business in quantum computing and a massive surge of research interest in quantum key distribution. Grover's algorithm (1996) then arrived with the potential to speed up our data searches. Another breakthrough came when China teleported a qubit from earth to a satellite in 2016.

Let us see this qubit in more detail now.

1.1 Qubit

Electrons have a property called spin, which is not the same as what we mean by spin in the physical world around us. The spin can mean that the electron is pointing either up or down. Denoting 'up' with 0 and 'down' with 1, we have a binary quantum switch. This switch can exist simultaneously – a hard-to-digest fact - in 'up' and 'down'. This simultaneous existence in multiple states or configurations is known as superposition. A single quantum switch can, thus, store the numbers 0 and 1 simultaneously. Such a quantum switch is called a qubit - a nomenclature that the classical world has inspired.

[1] Super-duper tiny objects on which 'quantum laws' apply

Apart from the 'spin' of the electrons, there are other physical quantities that a qubit can represent. The qubits can represent a single photon's polarization (horizontal vs. vertical), squeezed (amplitude vs. phase) light, atomic spin (up vs. down), and many others.

Though a qubit lives in the superposition of many states, it collapses to one of the states when measured. Thus, this act of measurement changes the qubit. This 'collapse' has been one of the surprising and, at the same time, weird parts of quantum theory. It bewildered even Einstein so much that he seriously debated the underpinning of quantum mechanics with Bohr (one of the founders of quantum mechanics). To enjoy this debate more, you can read [1].

Fun reading 1: What is this measurement and why does it change the behavior of a quantum system? Also, if the measurement changes how the system behaves, how does we know if the quantum mechanics is correct?
These questions are deeply philosophical and very difficult to answer. Though there are many interpretations of it, I can say none is deeply satisfying. There is a Copenhagen (Bohr worked at the University of Copenhagen) interpretation, which says, 'sweep the issue under the rug.' Read more, in Chapter 1 of the book by Aaronson [6].

The existence of a 'qubit' has incredible implications. Two classical bits exist in four combinations - 00, 01, 10, and 11 - and can represent any of the four numbers from 0 to 3. Consequently, we would need four pairs of numbers, one byte, to represent all four numbers. But with qubits, this is different. As a qubit lives in a superposition of two states, the two qubits live in a superposition of four states. This essentially means that all four states are available in one go. More concretely, this means that all four states (00, 01, 10, 11) and their corresponding numerical values (0, 1, 2, 3) are available simultaneously. Thus, what we could achieve by one classical byte is available to us with only two qubits. Now, let us stretch this argument a little more. Assuming n qubits, we will be staring at 2^n numbers at the same time. If $n = 300$, we will have 2×10^{90} numbers at the same time. With only 300 qubits, thus, we have exceeded the total number of atoms in the universe! This, by any imagination, is a super feat, raring to change the face of computation.

In the previous example, we have seen n qubits will create 2^n numbers at the same time. These 2^n numbers will also be processed simultaneously by a quantum computer. Thus, quantum computing has inherent parallelism.

In the following sub-section, we review some of the fundamental properties of quantum computing that make it all so powerful.

1.2 Fundamental properties of quantum computing

Quantum computing uses three fundamental properties of quantum physics: superposition, interference, and entanglement.

Superposition refers to the quantum phenomenon in which a quantum system can exist concurrently in multiple states. We have discussed this in detail in section 1.1 and have seen its impressive implications.

Quantum interference permits us to bias quantum systems toward the desired state. The idea is to construct a pattern of interference where the paths leading to wrong answers interfere destructively, but the paths leading to the correct answer reinforce each other. Believe me, this aspect requires mastery and some deep thoughts over a cup of coffee before you start imagining quantum interference in your grey matter.

Entanglement is an extremely strong correlation between quantum particles. Entangled particles remain perfectly correlated even if separated by great distances. This aspect is heavily exploited in virtually all quantum algorithms, quantum teleportation, and quantum error correction, to name a few.

1.3 The merit of quantum computing

Quantum computing is not powerful due to its speed or memory. Its speed is relatively slow, and its memory is tiny with a few quantum bits. Further, these qubits are prone to errors. We will see more about it in the following sub-section.

Due to the miraculous superposition states, quantum computing can compute multiple states concurrently. A quantum algorithm can then exploit 'interference' and can embolden paths with the desired results. These algorithms, thus, reveal poles apart complexity characteristics than their classical counterparts. Quantum algorithms, like Shor's algorithm, are thus exponentially faster than classical ones. Let us look at how algorithm complexity is calculated to understand what that entails. The algorithm complexity is expressed in terms of the number of computational steps taken to run an algorithm. For example, to add two numbers, expressed in terms of n digits, the number of steps is proportional to n, expressed as $O(n)$. This means that the number of steps in the addition operation increases linearly with n. Similarly, the computational complexity of multiplication is $O(n^2)$, meaning that the number of steps increases by the square of the number of digits. These algorithms are said to be solvable in polynomial time, that is, in n^p steps, where p is an integer. But these problems are comparably simple. For instance, see Figure 1. In this figure, we show the computational complexity of algorithms with polynomial vs. exponential complexity (where the number of steps is proportional to k^n, where k is any number). It is no brainer to see that algorithms with exponential complexity will suddenly go through the roof. And there are classical algorithms with exponential complexity. For example, the algorithm for finding the prime factors of an n-digit number has a complexity of $O(e^{n/3})$. Hence, the Shor's algorithm, which tackles the above problem in a quantum domain and has a complexity of $O((\log n)^3)$, is surely going to be disruptive.

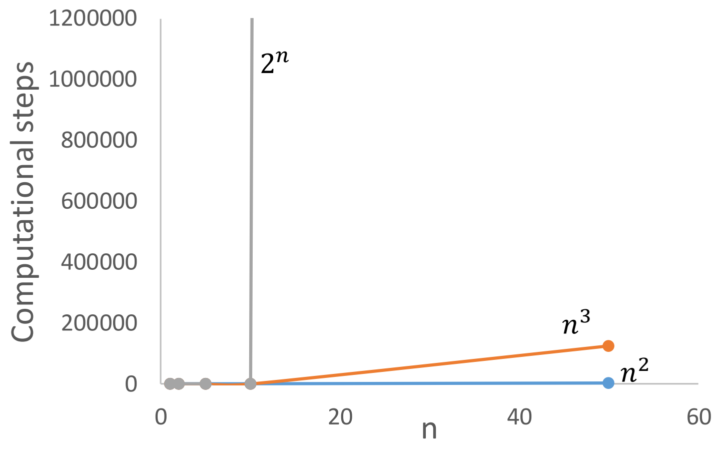

Figure 1: Number of computation steps in algorithms with polynomial and exponential complexity

If the argument with the number of computational steps is not convincing enough for you, then we have yet another reason. Our smartphone can multiply numbers with 800 digits in a few seconds, while a supercomputer takes about 2,000 years to factorize such numbers. Hopefully, now you know why computer scientists hate anything with exponential complexity.

1.4 Three circles of a quantum algorithm

We see that quantum algorithms have exponential improvements in speed over classical algorithms, a phenomenon which is referred to as quantum advantage. But is this all to go gung-ho about quantum algorithms? What about other metrics, like utility and number of quantum bits? We will see in the text below that for any quantum algorithm to be truly disruptive, it needs to fare well on these aspects as well.

A disruptive quantum algorithm should satisfy UAN (utility, advantage, and the number of qubits) criteria, that is,

1. It should have a practical utility.
2. It should provide a quantum advantage.
3. It should manage with a limited number of qubits.

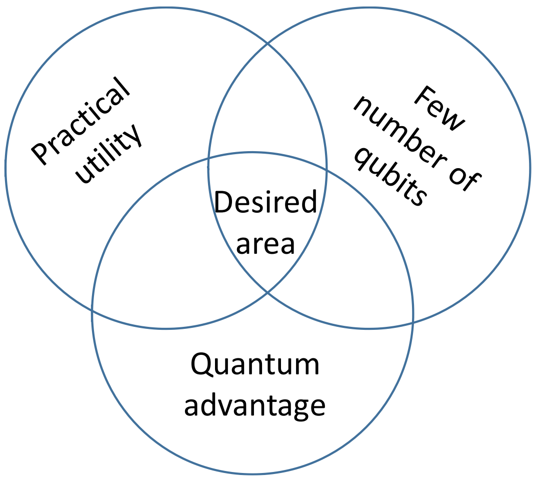

Figure 2: Desired characteristics of a quantum algorithm

When the first two metrics are pretty straightforward, the third objective requires some extra explanation. Quantum algorithms need to run on quantum computers that are prone to noise. The noise effects can build up with the number of qubits and may completely ruin a computation. Thus, the current quantum computers have only a limited number of qubits (127, more about quantum computers in sub-section 1.5). Thus, at this moment, any quantum algorithm which requires many qubits to operate cannot be tested. This requirement will ease out in the future as the number of qubits in a quantum computer increases.

Let us now see how current quantum algorithms fare on the UAN criteria. Shor's algorithm satisfies the first two criteria but requires a million qubits. Thus, Shor's algorithm does not provide us with immediate gains. In 2019, Google established quantum supremacy by announcing that its **Sycamore quantum computer** had completed a task in 200 seconds that would take a conventional computer 10,000 years. But again, the problem that is solved is not of a practical utility. Thus, the quest continues to find the quantum algorithms that satisfy all three criteria. And this is the right time to enter the field of quantum computing.

1.5 Quantum computers today

Current quantum computers involve large pieces of expensive and volatile equipment to solve elementary problems, such as finding the factors of the number 15. But so was the state of conventional computers in the 1950s with vacuum tubes and mercury delay lines. Anyone who has seen the migration of conventional computers from laboratory-sized machines to the PC and the iPad of today can believe that the computing world could be transformed within a decade.

Quantum computers must be very accurate because they deal with a continuous quantum state. Quantum algorithms also deal with controlling continuously varying parameters. In quantum computers, errors are minor, but they can accumulate with the increasing number of qubits ruining a given computation. Many noise sources include heat, stray electromagnetic fields, material defects, and control electronics. Thus, we need error correction, leading to fault-tolerant computing. There are ways to correct errors in quantum computers. However, error corrections require significant overheads.

Error correction and many algorithms so far designed like Shor's algorithm require millions of qubits. In 2019, Google announced its 53-qubit quantum computer, Sycamore. Meanwhile, IBM launched a series of quantum computers named after birds: Falcon, Hummingbird, and Eagle, see Table 1. Eagle has the maximum number of qubits (127) from IBM. Even though IBM endeavors for a 1000-quantum bits computer by 2023, the quantum processors in the near term will have between 50 and 100 quantum bits. Thus, these computers will only be able to run short programs, which become increasingly unstable as the number of qubits required increases due to noise accumulation.

It will still be challenging for classical computers to match what quantum computers with 50 qubits can do. Thus, we are already into a different era of computation, known as the noisy intermediate-scale quantum (NISQ) era. The word 'noisy' is used because error correction is still not used, and the word 'intermediate-scale' denotes that the number of qubits is small but still big enough to show quantum advantage or supremacy.

Table 1: Growth chart of Quantum Computers from IBM (See Qiskit [12])

Quantum Computer	**Falcon**	**Hummingbird**	**Eagle**	**Osprey**	**Condor**	**Flamingo**	**Kookabura**
Number of qubits	27	65	127	433	1121	1386+	4158+
Year	2019	2020	2021	2022	2023	2024	2025

In this NISQ era, variational quantum eigensolver, widely known as VQE, is supposed to be the first to occupy a sweet spot in the desired area in Figure 2. These algorithms use quantum computers and classical computers in a loop. Quantum computers do computation based on the optimized parameters fed by classical computers. And classical computers perform the optimization algorithm on the information they obtain from quantum computers. As these quantum computers do calculations based on the parameters from classical computers, they are called parameterized quantum circuits (PQCs).

2 Machine learning

Machine learning is a broad field where the machine learns aspects of data presented to it to take decisions. It touches on many aspects like statistics, mathematics, network design, computer science, neuroscience, genetics, behavioral economics, etc.

Mathematically, machine learning is about function approximation and optimization. Let us say we have a true function $g(x)$. As an instance, we can assume x to be pictures and $g(x)$ to be a function that can tell whether there is a cat in the picture or not. As a human, this task is very natural and straightforward, but not for computers. Thus, using machine learning techniques, we can approximate function $g(x)$ by another function (also known as a model) $f(\overline{x}, \theta)$, where $\overline{x}$ is the subset (as definitely we will not have access to all the pictures in the world) of the data x and θ is called the function's parameters. θ can be a vector or a matrix. More importantly, we tweak this θ to get the best approximation of the function f. Thus, as machine learning engineers, we have two main tasks. The first task is to pick this model f in the best way; many models exist, and we will see them eventually. The second task is to then pick the optimum values of θ.

2.1 Ingredients of machine learning

Let us now understand how to go about this function approximation and optimization task. Simultaneously, we will add some machine learning vocabulary.

Machine learning has three main ingredients: data, model, and cost function.

Data: Data is required to do anything, right? So, it naturally is the first and the most crucial ingredient. Let us now understand the data in more detail. Data can have many forms – text, audio, pictures, spreadsheet, etc. The data type can also influence the model we choose.

Let us say we have a black and white picture of a flower on a computer. If the picture is on a computer, it has to be a digital picture, meaning it can be broken down into pixels (a tiny area formed by dividing the picture into rows and columns). A pixel can be represented by a numeric number, and let us say we chose a number between [-1,1] to denote the image's brightness at each pixel location. If the image is 28×28, it simply means that there are 784 pixels in the image, and so a **data point (**in this case, a vector**)** $\overrightarrow{X_1}$ will have 784 entries in it. Each entry communicates to us the value of a pixel of the image. See the example of a data point below

$$\overrightarrow{X_1} = \begin{pmatrix} 0.8 \\ 0.2 \\ . \\ . \\ . \\ -0.5 \end{pmatrix}$$

The numbers in $\overrightarrow{X_1}$ are called the features of the data, and the number of features is called the dimension of the data. For example, $\overrightarrow{X_1}$ has a dimension of 784.

The entire **data set** can have n data points, meaning that it has n pictures in total. For example, the below data set contains n samples.

$$\vec{X} = [\overrightarrow{X_1}, \overrightarrow{X_2}, \dots, \overrightarrow{X_n}]$$

It is easy to see that the entire data set can be represented as a matrix, where the number of columns of a matrix is the number of data points, and the number of rows of a matrix is the dimension of the data point.

Model: Given $\vec{X}$ and the set of parameters θ, the model f gives an output $\hat{y}$. The output $\hat{y}$ is known as a label, prediction, or distribution. Note that $\hat{y}$ is similar to $f(\vec{X}, \theta)$. Examples of models are linear models, neural networks, and support vector machines.

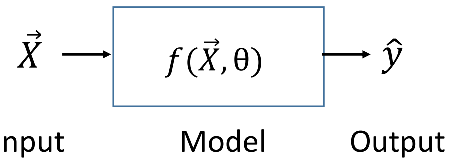

Figure 3: Input, model, and output relation

Cost function: To characterize the performance of a model, we need to define a cost function, which should be a function of the difference between the actual value y (if it is available) and the estimated value $\hat{y}$. One prevalent cost function is the mean-square error cost function, which is

$$MSE = (\hat{y} - y)^2$$

2.2 Examples of problems in machine learning

Machine learning has four main problems: supervised, unsupervised, reinforcement learning, and generation of data.

Supervised learning: In supervised learning, we have predecided labels. So any data we get has to be mapped to one of those labels.

There are various examples of supervised learning:

Classification: In this, we try to predict the discrete label of the data. Classification is similar to hypothesis testing (used in the context of digital communication). For example - when we receive an email, the email client has to categorize this as spam or not a spam email. Another important example is to recognize someone's handwritten digits. A good data set available there is the modified national institute of standards and technology (MNIST) data set. Some popular classification algorithms are Naïve Bayes, support vector machines, and logistic regression.

Regression: We have to find the numerical predictions about future events. Note that the regression is also a type of classification except that its output is an infinite number of numeric numbers. For example - weather prediction. Some popular regression algorithms are linear regression, nonlinear regression, and Bayesian linear regression.

Unsupervised learning: Unsupervised learning has no complete and clean labeled dataset. Its main aim is to explore the underlying patterns and predict the output. Here we provide the machine with data and ask it to look for hidden features and cluster the data in a way that makes sense.

There are various examples of unsupervised learning:

Clustering: Clustering identifies how different types of data are related and creates new segments based on those relationships. Clustering finds the relationship between data points so they can be segmented. A popular clustering algorithm is k −means clustering.

Association: The idea is that given n products and the purchase history of m users, the system has to make a product recommendation. For example, this can be used to recommend which Netflix show you should watch or which Amazon product you are most likely to order next. So, this has an extensive application base. Some popular association algorithms are based on neural networks.

Dimensionality reduction: Dimensionality reduction refers to techniques for reducing the number of input variables in training data. A popular dimensionality reduction algorithm is principal component analysis.

Reinforcement learning: Many popular games are built using reinforcement learning. The problem statement is as follows: There is an agent who can take specific actions in a given environment. Based on the action an agent takes, he can get awards or punishment. The objective of reinforcement learning is to make an agent act so that his awards are maximized. A computer agent to play several games like Go, Dota, and Spacecraft has been developed using reinforcement learning. In fact, the world champion of Go, Lee Se-dol retired from professional play after Google's DeepMinds defeated him. Some popular reinforcement learning algorithms are Q-learning, and Markov decision.

Generation of new data: A generative adversarial network (GAN) is a new model for generating new data. It stands for generative modeling using deep learning methods.

Generative modeling is an example of unsupervised learning that involves learning the regularities in input data so that the model can generate new examples that plausibly mimic one of the data points. GANs use a discriminator model classifier that attempts to classify examples produced from the generative model as either real (from the dataset) or fake (generated). The GANs train the generative model until the discriminator model is deceived about half the time, denoting that the generator model generates credible examples.

To understand what GAN can do, let us look at some of the stunning pictures of human beings that GAN has produced, see Figure 4. It is unbelievable that these human beings do not exist.

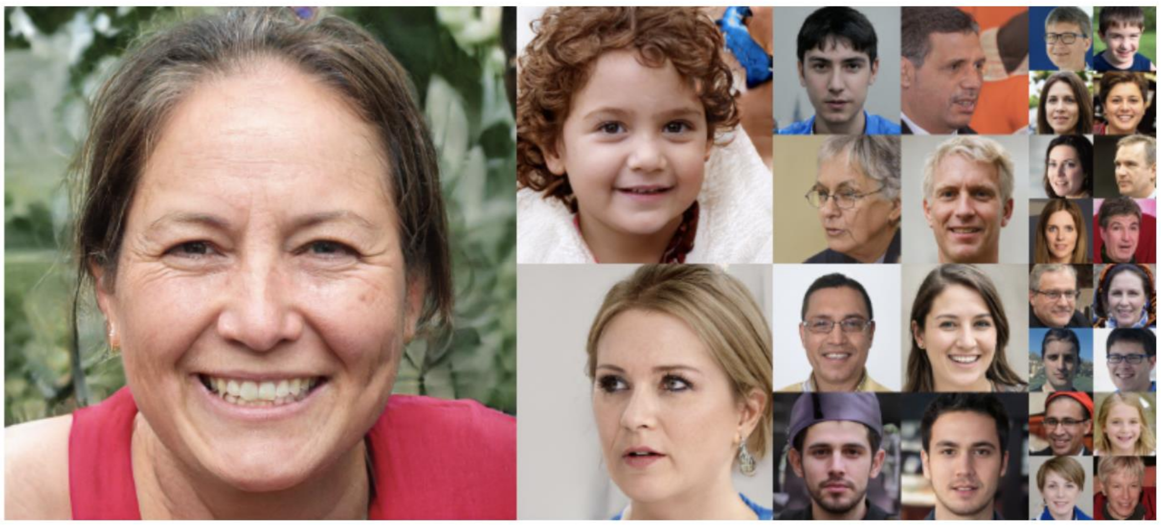

Figure 4: Pictures of humans created by using GAN [This picture has been taken based on 1812.04948.pdf (arxiv.org)]

3 Quantum machine learning

In the previous section, we have seen how machine learning algorithms learned to beat world champions in chess and Go. But these algorithms become progressively hard to train because they consist of billions of parameters. If we look at the generative pre-trained transformer 3 (GPT-3) network, which produces human-like text, has 175 billion parameters. Training the GPT-3 would require 355 years to train on a single graphics processing unit (GPU) and cost $4.6 million. Thus, it is clear that classical computers will not work. Of course, quantum computing seems like a reasonable choice here. But even though quantum computers are limited in their number of qubits today, this amalgamation between quantum computing and machine learning, or quantum machine learning, will play a role in the future. It is thus in our best interest to catch the bus before it leaves.

Quantum machine learning can mean four directions (as is introduced by Aimeur, Brassard, and Gambs [2]), as shown in the figure below:

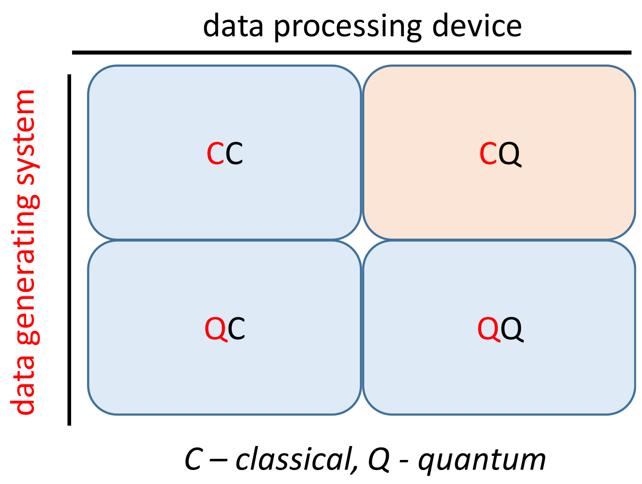

Figure 5: Four approaches to quantum machine learning

3.1 Quantum-inspired machine learning (CC)

This is the *CC* flavor in which classical data is being processed classically. This refers to the classical machine learning algorithms; however, these machine learning algorithms are 'inspired' by quantum computing research and thus are also known as 'quantum-inspired machine learning' approaches. A primary example is considering tensor [2]networks for neural network training.

A famous example of quantum-inspired machine learning comes from the area of recommendation systems. In 2016, Kerenidis and Prakash [3] proposed a quantum recommendation system that provides an exponential speed over the classical algorithms. In 2018, inspired by this quantum algorithm, Ewin Tang [4] proposed a classical randomized algorithm, which could achieve the same speed guarantees as the quantum algorithm proposed by kerenidis and Prakash. Thus, the advantages of the quantum recommendation system disappeared. Note that the algorithm proposed by Ewin Tang is an example of quantum-inspired machine learning. Such a quantum algorithm for which a classical equivalent with similar speed guarantees is available is also referred to as a 'dequantized' algorithm.

3.2 Machine learning for quantum systems (QC)

In this flavor, machine learning is used for improving quantum computing. For example, error correction is a vital requirement of a quantum computer, and classical machine learning can predict the error is in which qubit. The second example is from the neural networks. Neural networks have a high representation power. Thus, we can try to simulate a quantum system with neural networks. For example, neural networks have been used to simulate many-body physics. Another application could be to use machine learning to discriminate between quantum states emitted by a source.

3.3 Quantum for machine learning (CQ)

[2] Tensor product is used a lot in quantum computing, as we will see.

CQ flavor is our favorite in this course, and we mostly stick with it. In this flavor, we use quantum computing to improve current machine learning algorithms. For example, we can utilize quantum computing to create a restricted Boltzmann machine, a popular neural network that tunes parameters to create a joint probability density function between input and output.

Another example comes from using quantum computing to solve a linear system of equations. The Harrow, Hassidim, and Lloyd (HHL) quantum algorithm can solve a linear system of equations within a logarithmic runtime (with respect to data points), as against the classical algorithms, which take a linear time. Thus, the quantum algorithm provides an exponential speed-up compared to the classical algorithm. However, most quantum algorithms suffer from an input-output bottleneck, as even reading and writing the data between the classical-quantum interface takes linear time (as every data point has to be written and read), losing out all the advantages of the quantum algorithm. The input-output bottleneck can be solved by having a quantum RAM (QRAM), which many research groups are now targeting to design.

3.4 Quantum-generalized machine learning (QQ)

In these sets of algorithms, both the input and the output data are quantum. Thus, the input-output bottleneck problem is immediately solved. For example, the Large Hadron Collider (LHC) at Cern produces a significant amount of quantum data that can be analyzed by quantum computing. This set is closely related to the CQ system, as all the techniques developed in the CQ system can be seamlessly ported to the QQ case.

4 Overview of the course

The course begins with an introduction to linear algebra and quantum mechanics. Quantum computing does not require knowledge of Physics because we capture all Physics in postulates (a set of rules). The course discusses these postulates in detail for closed (non-interacting)[3] and open (interacting) quantum systems. Then, we look at some quantum circuits and gates. After covering the basics of quantum mechanics, we look at some "textbook" quantum algorithms, like Deutsch-Jozsa, Bernstein-Vajirani, and Simon algorithms. Subsequently, we see Shor's algorithm, which brought back the world's attention to quantum computing. Shor's algorithm builds on the idea of quantum Fourier transform and its related effects like quantum phase estimation. Finally, we see Grover's search algorithm, the next most cited quantum algorithm after Shor's. These algorithms still do not satisfy all the requirements of a successful quantum algorithm.

After covering all the basic requirements of quantum computing, we explore quantum computing to solve machine learning algorithms more efficiently. We begin by discussing the HHL algorithm, which can solve a linear system of equations. The HHL algorithm facilitates the development of the quantum version of linear regression, least-squares support vector machines, and recommendation systems. Subsequently, we discuss quantum linear regression and least-squares support vector machines. Then, we discuss unsupervised quantum algorithms for clustering and principal component analysis. These algorithms provide benefits under certain caveats (to be explicated later).

Afterward, we review two classes of quantum deep learning networks. The first class of quantum deep learning models contains classical and quantum components and is referred to as hybrid quantum-classical neural networks. The second class use only quantum gates in their structure.

[3] More details later

Finally, we look at various quantum optimization techniques. The two most common techniques are the VQE and the quantum approximate optimization algorithm, also known as QAOA. It is generally believed that these classes of algorithms will be the first to meet the UAN criteria. The principal idea in both these methods is to define the cost function of some quantum quantities. We then derive the maximum or minimum value of the cost function by optimization. The cost function calculation happens in a quantum computer, and optimization happens in a classical computer. All the optimizations are approximate, so they are perfect for NISQ computers.

5 *Supplementary reading*

There are plenty of materials at our hand that will help us in our pursuit of developing a quantum community. First of all, the book, "Quantum computation and quantum information," by Nielsen and Chuang [5], is an excellent book on quantum computing. Though old, this is by far the most comprehensive source of quantum computing to date. Another good text, particularly for quantum algorithms, is by Yanofsky and Mannuc [6]. All quantum algorithms have been discussed in this with much ease. The book by Scott Aaronson [7] also provides excellent and in-depth coverage. While there are many other good books, we will not be able to mention them all.

In the area of quantum machine learning as well, the literature has kept on increasing. For example, one of the first books on this topic is by Peter Wittek [8], titled "Quantum Machine Learning: What Quantum Computing Means to Data Mining." Unfortunately, Peter lost his life in one of the cyclones during his trip to the Himalayas. The book by Schuld and Petruccione [9] is on machine learning on quantum computers. The book by Pattanayak [10] covers many topics we discuss in this course. The reference [11] provides an excellent survey of many other sources in this area. Reference [12] has excellent information on quantum computation chemistry that we will see during our discussions of VQE. Some other interesting references can be found at [13] and [14].

This reference will not be complete if I do not mention some of the sources that are made publically available by IBM and Google. IBM has launched Qiskit [15], an open-source software development kit (SDK) for creating quantum programs. In Python [16], we can easily install Qiskit and use it to run on a local simulator (your laptop, for example) or prototype IBM quantum hardware. There is also a Qiskit textbook available at [17]. Similarly, Google provides many tools like Cirq and TensorFlow [18] for quantum programming.

Let us finish this section by making your interests in quantum computing jacked up. If you are interested in chess, there is a version of quantum chess! You can find the video of quantum chess played between Stephen Hawking and Paul Rudd at [19]. You can also watch the video at [20] to learn more about the game.

6 *Assignment question*

1. Explore the sources available on quantum computing and indicate your preferred choice. Also, mention why do you like it?

References:

[1] https://en.wikipedia.org/wiki/Bohr%E2%80%93Einstein_debates
[2] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. Advances in artificial intelligence, pp.: 431-442. Springer (2006).
[3] Iordanis Kerenidis, and Anupam Prakash. Quantum Recommendation Systems. arXiv, 2016.
[4] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. STOC 2019: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, June 2019.

[5] Michael A. Nielsen, and Issac L. Chuang. Quantum computation and quantum information. Cambridge University Press, 2002.
[6] Noson S. Yanofsky, and Mirco A. Mannuc. Quantum computing for computer scientists. Cambridge University Press, 2008.
[7] Scott Aaronson. Quantum Computing since Democritus. Cambridge University Press, 2013.
[8] Peter Wittek. Quantum Machine Learning: What Quantum Computing Means to Data Mining. Elsevier Science, 2016.
[9] Maria Schuld and Francesco Petruccione. Machine learning with quantum computers. Springer International Publishing, 2021.
[10]Santanu Pattanayak. Quantum machine learning with Python. Apress, 2021.
[11]Vedran Dunjko, and Peter Wittek. A non-review of Quantum Machine Learning: trends and explorations. Quantum Views 4, 32, 2020.
[12]Sam McArdle, Suguru Endo, Al\'an Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan, Quantum computational chemistry, RevModPhys., vol. 92, no. 1, pp.: 5003, 2020.
[13]HTTP Link: https://arcb.csc.ncsu.edu/~mueller/qc/qc18-2/qc18/readings/
[14]HTTP Link: https://github.com/Christophe-pere/Roadmap-to-QML
[15]Qiskit is available at: https://qiskit.org/
[16]Anaconda distribution site: https://www.anaconda.com/products/distribution
[17]Qiskit TextBook is available at: https://qiskit.org/textbook/content/ch-ex/
[18]Google quantum development source is available at: https://quantumai.google/
[19]Youtube video: https://www.youtube.com/watch?v=Hi0BzqV_b44&ab_channel=IQIMCaltech
[20]Youtube video: https://youtu.be/LikdmXfWO2A