Learning with Quantum Computers

Soham Joshi
Computer Science and Engineering
IIT Bombay (of Aff.)
Mumbai, India
sohamjoshi@cse.iitb.ac.in

***Abstract*—Quantum Computing (QC), is a developing field with counter-intuitive and surprising results. Manipulating hidden information caused by the probabilistic nature of quantum information has enabled researchers to formulate algorithms with lesser asymptotic time complexity [1]. Moreover, the combination of hidden information and entanglement at the quantum level has led to the reformulation of several machine learning algorithms in the language of quantum computing [2] [5] [7]. This survey presents some of the basic concepts needed to understand quantum computing and quantum information, and covers some of the machine learning algorithms which have been optimized due to this field.**

***Index Terms*—Quantum Machine Learning, Quantum Computing, Quantum Information, QuGANs**

I. Introduction

This report is a part of the project, "Machine Learning with Quantum Computers" [3], under the guidance of Siddhant Midha and Aditya Sriram. The project is a part of a program named "Winter in Data Science" (WiDS) conducted by Analytics Club, IIT Bombay. In this paper, basic concepts of quantum computing and quantum information will be covered in a way that is accessible to the reader with a knowledge of basic linear algebra and some quantum mechanics. In this report I will also survey some research papers pertaining to the fields of quantum machine learning (QML). Additionally the project includes implementation of quantum algorithms using qiskit and pennylane, the code for which can be found here. This paper is meant to be an overview of QC, so some parts of this paper may leave facts without proof. In such cases, kindly refer [2] for more details.

II. Quantum bits

The bit is the fundamental concept of classical computation and classical information. Quantum computation and quantum information are built upon an analogous concept, the quantum bit, or qubit for short. Note that a bit is just a mathematical entity, in physical circuits high/low voltages can be modelled by a bit taking value 1 or 0. Analogously a qubit is just another abstract mathematical entity. Does this mean that the qubit isn't "real"? Yes, a qubit is a tool used for modelling quantum mechanical systems. But studying the properties of the qubit is still worth our time since this helps us to develop a theory independent of physical constraints and which helps model reality to a good approximation.

A. States of a qubit

Just as a bit can take values 0, 1; a qubit can be in states $|0\rangle$, $|1\rangle$. Moreover, a qubit can be in a linear combination of these states given by :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$.
The state of a qubit is a vector in a two-dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as the computational basis states [1], form an orthonormal basis for this vector space. But, what makes a qubit different from a bit? It is the fact that the coefficients of linear combination α, β cannot be measured. Quantum mechanics tells us that we can obtain much more restricted information about this state, particularly, upon measurement, we can obtain $|0\rangle$ with probability $|\alpha|^2$, and $|1\rangle$ with probability $|\beta|^2$. Since the probabilites must sum to 1, the restriction $|\alpha|^2 + |\beta|^2 = 1$ is imposed on the coefficients of linear combination.

B. Bloch sphere

A tool which helps to visualise actions on a qubit is known as the *Bloch sphere*. Because of the restriction imposed on $|\alpha|, |\beta|$, we can rewrite the state of a qubit as :

$$|\psi\rangle = e^{i\gamma}\left(cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle\right)$$

In the next section we will see that global phase factors have no effect so this can be re-written as :

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle$$

C. Multiple qubits

Suppose we have two qubits. If these were two classical bits, then there would be four possible states, 00, 01, 10, and 11. Correspondingly, a two qubit system has four computational basis states [2] denoted $|00\rangle, |10\rangle, |10\rangle, |11\rangle$. A pair of qubits can also exist in superpositions of these four states,

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

[1]Measurement can be done with respect to a basis other than this. For instance, measurements can be done with respect to the basis $|+\rangle \equiv \frac{|0\rangle+|1\rangle}{\sqrt{2}}, |-\rangle \equiv \frac{|0\rangle-|1\rangle}{\sqrt{2}}$

[2]Another example of commonly used basis are called the "bell states/EPR pairs", given by $|\beta_{00}\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}; |\beta_{01}\rangle = \frac{|01\rangle+|10\rangle}{\sqrt{2}}; |\beta_{10}\rangle = \frac{|00\rangle-|11\rangle}{\sqrt{2}}; |\beta_{11}\rangle = \frac{|01\rangle-|10\rangle}{\sqrt{2}}$

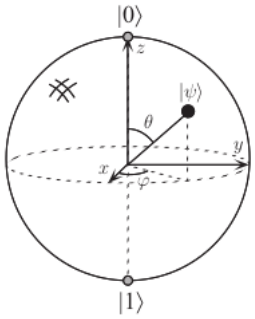

Fig. 1. Bloch sphere representation of a qubit

Similar to the case for a single qubit, the measurement result x (= 00, 01, 10 or 11) occurs with probability $|\alpha_x|^2$, with the state of the qubits after the measurement being $|x\rangle$. More generally, we may consider a system of n qubits. The computational basis states of this system are of the form $|x_1x_2...x_n\rangle$, and so a quantum state of such a system is specified by 2^n amplitudes.

III. Quantum Gates

Classical gates are responsible for manipulating bits, essentially representing a boolean function from bits to bits. Similarly, quantum gates are just a function from qubits to qubits. In this section, we shall explore the different classes of quantum gates. We will not delve into proofs but offer an operational description of how quantum gates work. Further details for this can be found in V

A. Single-Qubit gates

A qubit can be written as a 2×1 unit vector in the space spanned by $|0\rangle, |1\rangle$. So, a transformation from a single qubit to another qubit transforms a unit vector to another unit vector. Hence, it follows that all quantum gates are unitary. Moreover, it turns out that any unitary operation is a valid quantum gate and can be constructed from some "universal" quantum gates (will be discussed later).

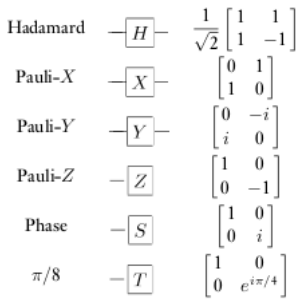

Fig. 2. Single qubit gates

Some single qubit quantum gates are shown in figure 2. Since we are dealing with the qubit as an abstraction, a quantum gate is just a black box operator that performs a unitary transformation to a qubit. For now, we will not deal with the question of the physical realisation of these gates.

B. Multiple qubit gates

In classical computation, there are five notable quantum gates, namely NOT, AND, OR, NAND, NOR. As it turns out, all of these gates can be simulated using quantum gates as well. But, as a first step, consider the controlled-NOT, or the CNOT gate (figure 3). Now, we go into an important point when dealing with quantum gates. Even though the circuit diagram shows two input lines in the circuit, the CNOT gate doesn't act on a single qubit, but a system of two "entangled" qubits. Hence, the only correct way to apply the circuit is to take the tensor product of the two input vectors, apply the gate and obtain another tensor product. The notation $|a\rangle \otimes |b\rangle$ is often abbreviated as $|a\rangle|b\rangle$ or $|ab\rangle$. So the action of a gate U will be given as :

$$|a\rangle|b\rangle \xrightarrow{U} U|a\rangle|b\rangle$$

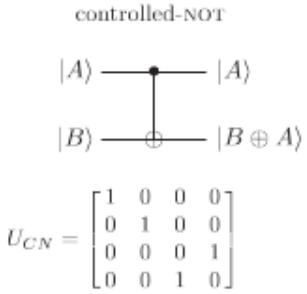

Fig. 3. CNOT gate

Moreover, just because the control qubit does not change values when it is a computational basis state doesn't mean that it will remain the same for all vectors. In fact, the control qubit often changes and measuring it's state is a key step in several quantum algorithms. Another important fact is that any multiple qubit quantum gate can be constructed using CNOT and single qubit quantum gates. Hence, they form a set of universal quantum gates. So, the CNOT gate is analogous to the XOR gate, but not exactly equivalent. Can gates be constructed exactly equivalent to NAND, XOR gates? Turns out this is not possible because quantum gates must be reversible whereas XOR and NAND gates are irreversible, i.e., given the output of these gates, the input cannot be uniquely identified.

C. Quantum circuits

We have already seen some quantum circuits, but let us examine quantum ciruits in some more detail. Each line in the circuit is represented by a wire, but this wire is not necessarily physical, it may as well represent the passage of time, or a physical particle such as a photon. There are a few features allowed in classical circuits that are not usually present in quantum circuits. Loops in a circuit are not allowed, FANOUT and FANIN operations are not allowed. This is because of the

"no-cloning"[3] theorem. An example of a quantum circuit is the "quantum teleportation circuit"(figure 4). The setting of the problem is that Alice and Bob met long ago but now live far apart. While together they generated an EPR pair, each taking one qubit of the EPR pair when they separated. Alice now wants deliver a qubit $|\psi\rangle$ to Bob. She does not know the state of the qubit, and moreover can only send classical information to Bob. This can be achieved using the circuit given.

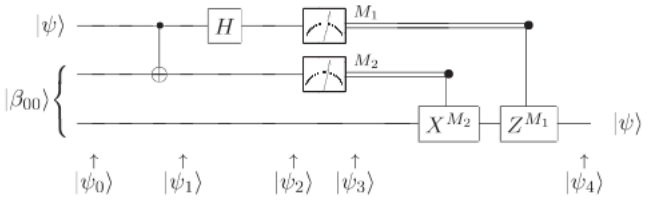

Fig. 4. Quantum circuit for teleporting a qubit. The two top lines represent Alice's system, while the bottom line is Bob's system. The meters represent measurement, and the double lines coming out of them carry classical bits (recall that single lines denote qubits)

D. The Deutsch-Jozsa algorithm

The problem is as follows. Alice, selects a number x from 0 to $2^n - 1$, and mails it in a letter to Bob. Bob calculates some function $f(x)$ and replies with the result, which is either 0 or 1. Now, Bob has promised to use a function f which is of one of two kinds; either $f(x)$ is constant for all values of x, or else $f(x)$ is balanced, that is, equal to 1 for exactly half of all the possible x, and 0 for the other half. Alice's goal is to determine with certainty whether Bob has chosen a constant or a balanced function, corresponding with him as little as possible. In order to solve this we use the algorithm described (figure 5). The circuit for the same is shown in figure 6

Algorithm: Deutsch–Jozsa

Inputs: (1) A black box U_f which performs the transformation $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$, for $x \in \{0, \ldots, 2^n - 1\}$ and $f(x) \in \{0, 1\}$. It is promised that $f(x)$ is either *constant* for all values of x, or else $f(x)$ is *balanced*, that is, equal to 1 for exactly half of all the possible x, and 0 for the other half.

Outputs: 0 if and only if f is constant.

Runtime: One evaluation of U_f. Always succeeds.

Procedure:

1. $|0\rangle^{\otimes n}|1\rangle$ — initialize state
2. $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$ — create superposition using Hadamard gates
3. $\rightarrow \sum_x (-1)^{f(x)}|x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$ — calculate function f using U_f
4. $\rightarrow \sum_z \sum_x \frac{(-1)^{x \cdot z + f(x)}|z\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$ — perform Hadamard transform
5. $\rightarrow z$ — measure to obtain final output z

Fig. 5. Deutsch-Jozsa algorithm

[3]The no-cloning theorem states that "qubit-copying" is not permitted. Hence, junctions of wires are not permitted in a quantum circuit. The no-cloning theorem proves to be useful in quantum encryption protocols like BB84 [4]

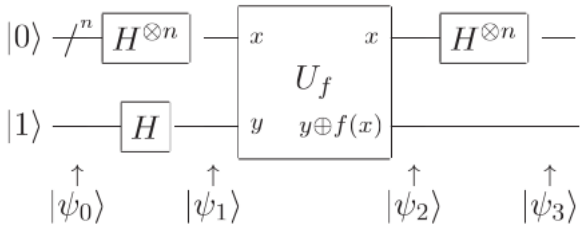

Fig. 6. Deutsch-Jozsa algorithm's circuit

Finally, if the measurement yields all 0s then the function is constant, otherwise balanced. The reader can try verifying this as an exercise.

IV. The postulates of quantum mechanics

Quantum mechanics is a mathematical framework for the development of physical theories. Just as newton's laws are axioms for classical mechanics, the postulates of quantum mechanics are not "derived", but assumed. The motivations of these postulates(axioms) may not always be clear, since they have been guessed after a lot of experimentation. So, this part of the report aims at knowing the postulates and gaining an operational understanding of the same. Understanding these axioms will help us realise what happens in physical systems, and not the abstract vectors we have been dealing with till now.

A. State space

Postulate 1 : Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

Quantum mechanics does *not* tell us, for a given physical system, what the state space of that system is, nor does it tell us what the state vector of the system is. Figuring that out for a specific system is a difficult problem for which physicists have developed many intricate and beautiful rules. We will take the qubit as our fundamental quantum mechanical system.

B. Evolution

Postulate 2 : The evolution of a *closed* quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,

$$|\psi^{'}\rangle = U|\psi\rangle$$

Just as quantum mechanics does not tell us the state space or quantum state of a particular quantum system, it does not tell us which unitary operators U describe real-world quantum dynamics. Quantum mechanics merely assures us that the evolution of any closed quantum system may be described in such a way. Another interesting result follows from this postulate. If t_1 and t_2 are "close", then we can talk about derivative of functions,

Postulate 2' : The time evolution of the state of a closed quantum system is described by the Schrödinger equation

$$i\hbar\frac{d}{dt}|\psi\rangle = H|\psi\rangle$$

In this equation, $\hbar$ is a physical constant known as Planck's constant whose value must be experimentally determined. The exact value is not important to us. In practice, it is common to absorb the factor $\hbar$ into H, effectively setting $\hbar = 1$. H is a fixed Hermitian operator known as the Hamiltonian of the closed system[4].

C. Quantum measurement

Even though the internal evolution of a quantum system can be worked out, there are times when we need to measure the quantum system, i.e. interact it with the experimental set-up which leads to the system not being closed anymore. This postulate addresses this development of the system.

Postulate 3 : Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|psi\rangle$ immediately before the measurement then the probability that result m occurs is given by,

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}$$

The measurement operators satisfy the completeness equation,

$$\sum_m M_m^\dagger M_m = I$$

The completeness equation expresses the fact that probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle$$

A simple but important example of a measurement is the measurement of a qubit in the computational basis. This is a measurement on a single qubit with two outcomes defined by the two measurement operators $M_0 = |0\rangle\langle 0|$, $M_1 = |1\rangle\langle 1|$. The probability of measured outcomes can be verified using this postulate.

An important application of Postulate 3 is to the problem of distinguishing quantum states. In the classical world, distinct states of an object are usually distinguishable, at least in principle. For example, we can always identify whether a coin has landed heads or tails. An important result follows from postulate 3 that only only *orthogonal states* can be distinguished.

[4]Earlier, we spoke about applying a unitary operator to a system. Doesn't this contradict the fact that postulate 2 only holds for closed systems? Yes! We should consider the experimentation set-up as a part of our system if we want to use the hamiltonian to describe it. But, the unitary operator is a good enough approximation that we can push this nuance under the rug.

D. Composite systems

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle\otimes|\psi_2\rangle\otimes\cdots\otimes|\psi_n\rangle$.

Further details about density operators, EPR can be found in [2].

V. Quantum circuits

In this section, we will examine some more circuits and gates, theorems that help us to construct more complicated gates from simple building blocks.

The Pauli matrices give rise to three useful classes of unitary matrices when they are exponentiated, the rotation operators about the $\hat{x}, \hat{y}$, and $\hat{z}$ axes, defined by the equations:

$$R_x(\theta) \equiv e^{\frac{-i\theta X}{2}} = cos\frac{\theta}{2}I - isin\frac{\theta}{2}X$$
$$R_y(\theta) \equiv e^{\frac{-i\theta Y}{2}} = cos\frac{\theta}{2}I - isin\frac{\theta}{2}Y$$
$$R_z(\theta) \equiv e^{\frac{-i\theta Z}{2}} = cos\frac{\theta}{2}I - isin\frac{\theta}{2}Z$$

Theorem : (**Z-Y decomposition for a single qubit**) Suppose U is a unitary operation on a single qubit. Then there exist real numbers α, β, γ and δ such that

$$U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_x(\delta)$$

A. Controlled operation

Suppose U is an arbitrary single qubit unitary operation. A controlled-U operation is a two qubit operation, again with a control and a target qubit. If the control qubit is set then U is applied to the target qubit, otherwise the target qubit is left alone; that is,

$$|c\rangle|t\rangle \rightarrow |c\rangle U^c|t\rangle$$

where $|c\rangle, |t\rangle$ are computational basis states. The general operation is then done via tensor products. The controlled-U operation is represented by the circuit shown in figure 7

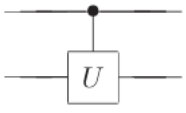

Fig. 7. Controlled-U operation. The top line is the control qubit, and the bottom line is the target qubit. If the control qubit is set then U is applied to the target, otherwise it is left alone

Now, our goal is to construct a generalised controlled-U operation. In order to do this, the following theorem is useful :

Theorem : Suppose U is a unitary gate on a single qubit. Then there exist unitary operators A, B, C on a single qubit such that $ABC = I$ and $U = e^{i\alpha}AXBXC$, where α is some overall phase factor.

First, let us consider the controlled phase shift operation. This operation does, $|00\rangle \rightarrow |00\rangle, |01\rangle \rightarrow |01\rangle, |10\rangle \rightarrow e^{i\alpha}|10\rangle, |11\rangle \rightarrow e^{i\alpha}|11\rangle$. The equivalent circuits for the same are given in figure 8

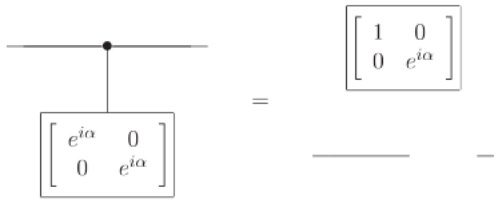

Fig. 8. Controlled phase shift gate and an equivalent circuit for two qubits

Now, we have enough to implement a controlled phase operation. The circuit for the same utilizes black boxes for A, B, C, and the CNOT gate, along with controlled phase shift. The circuit is shown in figure 9

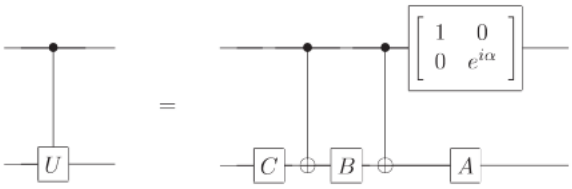

Fig. 9. Circuit implementing the controlled-U operation for single qubit U

B. Multi-qubit controlled operations

A well-known example of a multi-qubit control circuit is known as the *toffoli gate*. This gate acts on the target qubit only if both the control qubits are in the $|1\rangle$ state. Hence, it is an extended CNOT gate in some sense. The circuit for the same is shown in figure 10

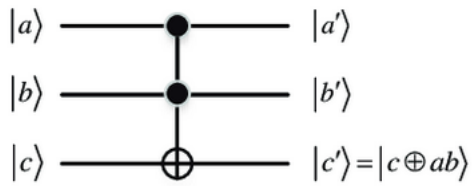

Fig. 10. Toffoli gate

We would further like to generalise this idea just as we did for a single control qubit. So, we introduce the generalised controlled-U gate, with n control qubits and k target qubits. This operator acts on a system of qubits as,

$$C^n(U)|x_1x_2...x_n\rangle|\psi\rangle = |x_1x_2...x_n\rangle U^{x_1x_2...x_n}|\psi\rangle$$

where $x_1x_2...x_n$ in the exponent of U means the product of the bits $x_1, x_2, ..., x_n$

Now, our task is to construct this generalised controlled-U gate using the gates we know how to construct. As it turns out, we can construct we can construct this gate using the single control qubit controlled-U gate, and the toffoli gates. The circuit for the same has been shown in figure 12.

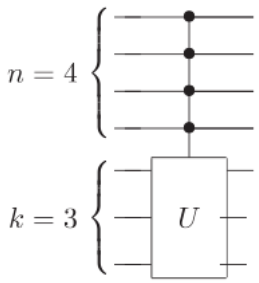

Fig. 11. Sample circuit representation for the $C^n(U)$ operation, where U is a unitary operator on k qubits, for $n = 4$ and $k = 3$.

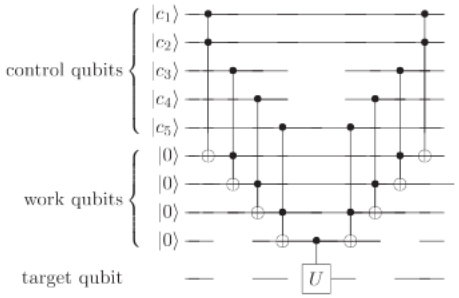

Fig. 12. Network implementing the $C^n(U)$ operation, for the case $n = 5$.

C. Circuit Identities

Here are some important circuit identities. Let subscripts denote which qubit an operator acts on, and let C be a CNOT with qubit 1 the control qubit and qubit 2 the target qubit.

$$CX_1C = X_1X_2$$
$$CY_1C = Y_1X_2$$
$$CZ_1C = Z_1$$
$$CX_2C = X_2$$
$$CY_2C = Z_1Y_2$$
$$CZ_2C = Z_1Z_2$$
$$R_{z,1}(\theta)C = CR_{z,1}(\theta)$$
$$R_{x,2}(\theta)C = CR_{x,2}(\theta)$$

D. Measurement & Universal circuits

There are really only two things worth knowing here :

1) The first principle is that classically conditioned operations can be replaced by quantum conditioned operations
2) **Principle of deferred measurement**: Measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit; if the measurement results are used at any stage of the circuit then the classically controlled operations can be replaced by conditional quantum operations.

Now, let's talk about universal quantum circuits. In classical circuits, NOT, AND and OR are enough to describe any boolean function. Similarly for quantum circuits, there are three constructions we can make :

1) The first construction shows that an arbitrary unitary operator may be expressed ex- actly as a product of unitary operators that each acts non-trivially only on a subspace spanned by two computational basis states.
2) The second construction combines the first construction with the results of the previous section to show that an arbitrary unitary operator may be expressed exactly using single qubit and CNOT gates.
3) The third construction combines the second construction with a proof that single qubit operation may be approximated to arbitrary accuracy using the Hadamard, phase, and $\pi/8$ gates.

For more on universal quantum gates, one can refer [2].

VI. Quantum Fourier Transform

The quantum Fourier transform is an efficient quantum algorithm for performing a Fourier transform of quantum mechanical amplitudes. This idea is useful in algorithms like Grover's search. Let's explore the discrete fourier transform. Let $|0\rangle, |1\rangle, \cdots, |N-1\rangle$ be an orthonormal basis. Let the scalar coefficients corresponding to these basis states be $x_1, x_2, \cdots, x_{N-1}$. Then, the transformation is defined by;

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i j k}{N}}$$

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0} N-1 e^{\frac{2\pi i j k}{N}} |k\rangle$$

Equivalently, an arbitrary state

$$\sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{U} \sum_{k=0}^{N-1} y_k |k\rangle$$

where U is a unitary operator.
In the following, we take $N = 2^n$, where n is some integer, and the basis $|0\rangle, |1\rangle, \cdots, |2^n - 1\rangle$ is the computational basis for an n qubit quantum computer. It is helpful to write the state $|j\rangle$ using the binary representation $j = j_1 j_2 ... j_n$. More formally, $j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$. It is also convenient to adopt the notation $0.j_l j_{l+1} ... j_m$ to represent the binary fraction $j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$.

With a little algebra the quantum Fourier transform can be given the following useful *product representation*:

$$|j_1, ..., j_n\rangle \rightarrow \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_1 \cdots j_{n-1} j_n}|1\rangle)}{2^{n/2}}$$

This product representation is so useful that you may even wish to consider this to be the definition of the quantum Fourier transform. This allows us to construct an efficient quantum circuit computing the Fourier transform, a proof that the quantum Fourier transform is unitary, and provides insight into algorithms based upon the quantum Fourier transform. Let the gate R_k denotes the unitary transformation;

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

Given this, the quantum fourier transform circuit, i.e. an efficient algorithm for the same is shown in figure 13.

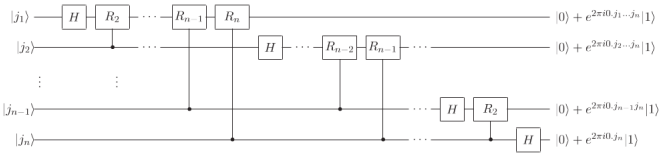

Fig. 13. Efficient circuit for the quantum Fourier transform. This circuit is easily derived from the *product representation* for the quantum Fourier transform. Not shown are swap gates at the end of the circuit which reverse the order of the qubits, or normalization factors of $1/\sqrt{2}$ in the output.

A. Phase estimation

The Fourier transform is the key to a general procedure known as *phase estimation*, which in turn is the key for many quantum algorithms. Suppose a unitary operator U has an eigenvector $|u\rangle$ with eigenvalue $e^{2\pi i \phi}$, where the value of $|\phi\rangle$ is unknown. The goal of the phase estimation algorithm is to estimate $|\phi\rangle$. To perform the estimation we assume that we have available black boxes (sometimes known as oracles) capable of preparing the j state $|u\rangle$ and performing the controlled-U^{2^j} operation, for suitable non-negative integers j.

The quantum phase estimation procedure uses two registers. The first register contains t qubits initially in the state $|0\rangle$. How we choose t depends on two things: the number of digits of accuracy we wish to have in our estimate for ϕ, and with what probability we wish the phase estimation procedure to be successful.

The second register begins in the state $|u\rangle$, and contains as many qubits as is necessary to store $|u\rangle$. Phase estimation is performed in two stages. First, we apply the circuit shown in Figure 14. The second stage of phase estimation, shown in

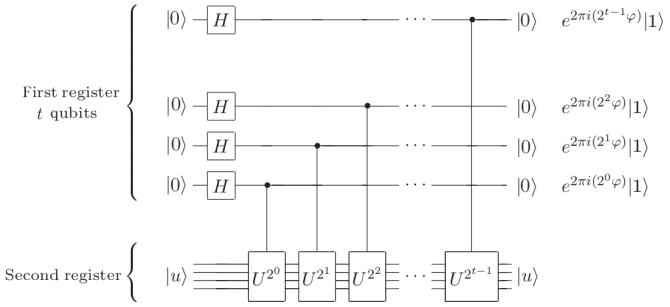

Fig. 14. The first stage of the phase estimation procedure. Normalization factors of $1/\sqrt{2}$ have been omitted, on the right.

figure 15, is to apply the inverse quantum Fourier transform on the first register. This is obtained by reversing the circuit for the quantum Fourier transform in the previous section,

and can be done in $\Omega(t^2)$ steps. The third and final stage of phase estimation is to read out the state of the first register by doing a measurement in the computational basis. Summarizing, the phase estimation algorithm allows one to

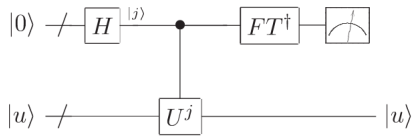

Fig. 15. Schematic of the overall phase estimation procedure. The top t qubits (the '/' denotes a bundle of wires, as usual) are the first register, and the bottom qubits are the second register, numbering as many as required to perform U. The state u is an eigenstate of U with eigenvalue $e^{2i\pi\phi}$. The output of the measurement is an approximation to ϕ accurate to $t - \lceil log(2 + \frac{1}{2\epsilon})\rceil$ bits, with probability of success at least $1 - \epsilon$.

estimate the phase ϕ of an eigenvalue of a unitary operator U, given the corresponding eigenvector $|u\rangle$. An essential feature at the heart of this procedure is the ability of the inverse Fourier transform to perform the transformation

$$\frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i\phi j}|j\rangle|u\rangle \rightarrow |\tilde{\phi}\rangle|u\rangle$$

where $\tilde{\phi}$ denotes a state which is a good estimator for ϕ when measured.

B. Application : Order finding

As it turns out the problems of order finding and factoring are equivalent, that is, the problem of factoring can be expressed as an order finding problem. So, if we solve order finding, we get factoring for free. Now, what is the order finding problem? For positive integers x and N, $x < N$, with no common factors, the order of x modulo N is defined to be the least positive integer, r, such that $x^r \equiv 1(\text{mod } N)$.

The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator

$$U|y\rangle = |xy(\text{mod N})\rangle$$

with $y \in \{0,1\}^L$. (Note that here and below, when $N \leq y \leq 2^L - 1$, we use the convention that xy(mod N) is just y again. That is, U only acts non-trivially when $0 \leq y \leq N - 1$). A simple calculation shows that the states defined by

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi isk}{r}}|x^k \text{ mod N}\rangle$$

for integer $0 \leq s \leq r - 1$ are eigenstates of U , since

$$\begin{aligned} U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi isk}{r}}|x^{k+1} \text{ mod N}\rangle \\ &= e^{\frac{2\pi is}{r}}|u_s\rangle \end{aligned}$$

Using the phase estimation procedure allows us to obtain, with high accuracy, the corresponding eigenvalues $exp(2\pi is/r)$, from which we can obtain the order r with a little bit more work.

There are two important requirements for us to be able to use the phase estimation procedure: we must have efficient procedures to implement a controlled-U^{2^j} operation for any integer j, and we must be able to efficiently prepare an eigenstate $|u_s\rangle$ with a non-trivial eigenvalue, or at least a superposition of such eigenstates. The first requirement is satisfied by using a procedure known as modular exponentiation, with which we can implement the entire sequence of controlled-U^{2^j} operations applied by the phase estimation procedure using $O(L^3)$ gates.
The second requirement is a bit trickier. preparing $|u_s\rangle$ requires that we know r, so this is out of the question. Fortunately, there is a clever observation which allows us to circumvent the problem of preparing $|u_s\rangle$, which is that :

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

Hence, preparing $|1\rangle$ in the second register suffices, which is trivial. Hence, the circuit is complete and is shown in figure 16

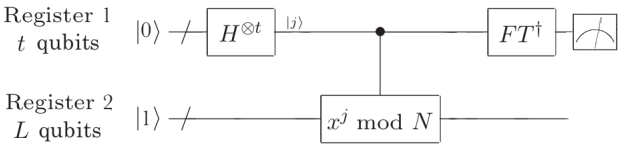

Fig. 16. Quantum circuit for the order-finding algorithm.

Even after doing this, we are not done yet. The measurement will give us an estimate of s/r. We need to determine the value of r from this. This can be done using the continued fractions expansion since we know the number of bits upto which these numbers upto a good accuracy. The process of continued fractions finally yields us r. The algorithm has been summarised in figure 17

Algorithm: Quantum order-finding

Inputs: (1) A black box $U_{x,N}$ which performs the transformation $|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L-bit number N, (2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon})\rceil$ qubits initialized to $|0\rangle$, and (3) L qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1 \pmod N$.

Runtime: $O(L^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

1.	$\|0\rangle\|1\rangle$	initial state
2.	$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|1\rangle$	create superposition
3.	$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|x^j \bmod N\rangle$	apply $U_{x,N}$
	$\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi isj/r}\|j\rangle\|u_s\rangle$	
4.	$\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \|\widetilde{s/r}\rangle\|u_s\rangle$	apply inverse Fourier transform to first register
5.	$\rightarrow \widetilde{s/r}$	measure first register
6.	$\rightarrow r$	apply continued fractions algorithm

Fig. 17. Algorithm for the order-finding problem

C. Application : Factoring

The factoring problem is given a natural number $N \in \mathbb{N}$, to return a non-trivial factor of N, that is, other than 1 and N itself. As described earlier, we will simply reduce this problem into order-finding. This is how we do it.

1) If N is even, return the factor 2.
2) Determine whether $N = a^b$ for integers $a \geq 1$ and $b \geq 2$, and if so return the factor a (classical algorithm described in [6]).
3) Randomly choose x in the range 1 to $N - 1$. If $gcd(x, N) > 1$ then return the factor $gcd(x, N)$.
4) Use the order-finding subroutine to find the order r of x modulo N .
5) If r is even and $x^{r/2} \neq -1(\text{mod N})$ then compute $gcd(x^{r/2}-1, N)$ and $gcd(x^{r/2}+1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

That is it.

VII. Quantum Search Algorithms

This section deals with searching for solutions through a set of some inputs. The algorithm we explore here is called the *grover search algorithm*. But first, how do we know which input is a solution? For this, we need something called an *oracle*

A. Oracle

Suppose we wish to search through a search space of N elements. Rather than search the elements directly, we concentrate on the index to those elements, which is just a number in the range 0 to $N - 1$. For convenience we assume $N = 2^n$, so the index can be stored in n bits, and that the search problem has exactly M solutions, with $1 \leq M \leq N$. A particular instance of the search problem can conveniently be represented by a function f , which takes as input an integer x, in the range 0 to $N - 1$. By definition, $f(x) = 1$ if x is a solution to the search problem, and $f(x) = 0$ if x is not a solution to the search problem.

Given this framework, a quantum oracle is just a black box. It is a unitary operator whose action is described as :

$$|x\rangle|q\rangle \xrightarrow{O} |x\rangle|q \oplus f(x)\rangle$$

We can check whether x is a solution to our search problem by preparing $|x\rangle|0\rangle$, applying the oracle, and checking to see if the oracle qubit has been flipped to $|1\rangle$. In the quantum search algorithm it is useful to apply the oracle with the oracle qubit initially in the state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, The action of the oracle is :

$$|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \xrightarrow{O} (-1)^{f(x)}|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Since the second qubit doesn't change here, it can be omitted :

$$|0\rangle \xrightarrow{O} (-1)^{f(x)}|0\rangle$$

We say that the oracle marks the solutions to the search problem, by shifting the phase of the solution. As it turns out, the grover search algorithm can find a solution in $O(\sqrt{N/M})$ time, where N is the cardinality of the search space, M is the number of solutions.

B. The procedure

The algorithm proper makes use of a single n qubit register. The internal workings of the oracle, in- cluding the possibility of it needing extra work qubits, are not important to the description of the quantum search algorithm proper. The goal of the algorithm is to find a solution to the search problem, using the smallest possible number of applications of the oracle.

The algorithm begins with the computer in the state $|0\rangle^{\otimes n}$. The Hadamard transform is used to put the computer in the equal superposition state,

$$|\psi\rangle = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle$$

The quantum search algorithm then consists of repeated application of a quantum subroutine, know as the Grover iteration or Grover operator, which we denote G. The Grover iteration, whose quantum circuit is illustrated in Figure 18, may be broken up into four steps:

1) Apply the oracle O.
2) Apply the Hadamard transform $H^{\otimes n}$.
3) Perform a conditional phase shift on the computer, with every computational basis state except $|0\rangle$ receiving a phase shift of -1,

$$|x\rangle \rightarrow -(-1)^{\delta_{x0}}|x\rangle$$

4) Apply the Hadamard transform $H^{\otimes n}$.

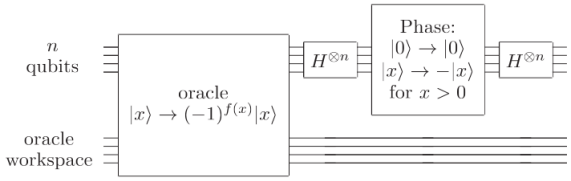

Fig. 18. Circuit for the Grover iteration, G

It is useful to note that the combined effect of steps 2, 3, and 4 is,

$$H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I$$

where $|\psi\rangle$ is the equally weighted superposition of states. Thus the Grover iteration, G, may be written $G = (2|\psi\rangle\langle\psi| - I)O$.

C. Geometric visualisation

The Grover iteration can be regarded as a rotation in the two-dimensional space spanned by the starting vector $|\psi\rangle$ and the state consisting of a uniform superposition of solutions to the search problem. We adopt the convention that, $\sum_x'$, indicates a sum over all x which are solutions to the search

problem, and $\sum_x''$ indicates a sum over all x which are not solutions to the search problem. Define normalized states,

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}} \sum_x'' |x\rangle$$

$$|\beta\rangle \equiv \frac{1}{\sqrt{M}} \sum_x' |x\rangle$$

Simple algebra shows that the initial state $|\psi\rangle$ may be re-expressed as,

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle$$

so the initial state of the quantum computer is in the space spanned by $|\alpha\rangle$, $|\beta\rangle$.

The effect of G can be understood in a beautiful way by realizing that the oracle operation O performs a reflection about the vector $|\alpha\rangle$ in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$. Similarly, $2|\psi\rangle\langle\psi| - I$ also performs a reflection in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$, about the vector $|\psi\rangle$. And the product of two reflections is a rotation. Let $cos\frac{\theta}{2} = (N-M)/N$, so that $|\psi\rangle = cos\frac{\theta}{2}|\alpha\rangle + sin\frac{\theta}{2}|\beta\rangle$. As Figure 19 shows, the two reflections which comprise G take $|\psi\rangle$ to,

$$G|\psi\rangle = cos\frac{3\theta}{2}|\alpha\rangle + sin\frac{3\theta}{2}|\beta\rangle$$

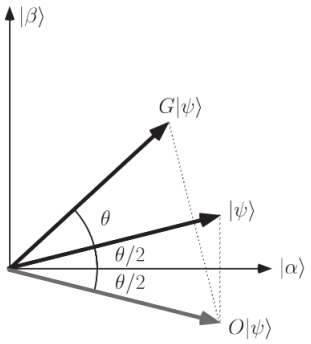

Fig. 19. Grover's iteration visualised

D. Performance

Now, the question boils down to how many grover's iterations/rotations are required to convert $|\psi\rangle$ to $|\beta\rangle$? As it turns out the upper bound on the number of rotations R required after some calculations is given by,

$$R \leq \left[\frac{\pi}{4}\sqrt{\frac{N}{M}}\right]$$

Finally, the algorithm is summarised in Figure 20.

Algorithm: Quantum search

Inputs: (1) a black box oracle O which performs the transformation $O|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle$, where $f(x) = 0$ for all $0 \leq x < 2^n$ except x_0, for which $f(x_0) = 1$; (2) $n+1$ qubits in the state $|0\rangle$.

Outputs: x_0.

Runtime: $O(\sqrt{2^n})$ operations. Succeeds with probability $O(1)$.

Procedure:

1. $|0\rangle^{\otimes n}|0\rangle$ — initial state
2. $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$ — apply $H^{\otimes n}$ to the first n qubits, and HX to the last qubit
3. $\rightarrow \left[(2|\psi\rangle\langle\psi| - I)O\right]^R \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$ — apply the Grover iteration $R \approx \lceil \pi\sqrt{2^n}/4 \rceil$ times.
   $\approx |x_0\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$
4. $\rightarrow x_0$ — measure the first n qubits

Fig. 20. Grover's Algorithm

VIII. Quantum Machine Learning

In this section, we will primarily focused on the studies described in [1], [5], [7] ([8] is a good resource on how to read papers). Coming up with machine learning algorithms using quantum computation is particularly difficult since the algorithm has to perform better than the classical version, is often unintuitive.

A. k-nearest neighbour methods

This is a standard algorithm for unsupervised learning. The procedure of the algorithm relies upon clustering the data-points into some groups depending on the distances of the points from the centroids of the groups. This is illustrated in Figure 21. This is one of the many machine learning methods that rely on the notion of a *distance metric*. There are a lot of distance metrics available, but measuring the inner product between two points or states is a good start since the notion of overlap is natural in the field of quantum mechanics.

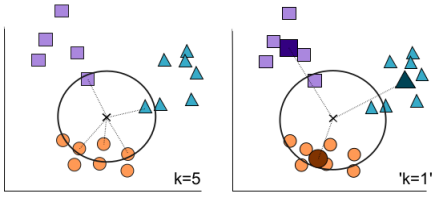

Fig. 21. a: Illustration of the kNN method of pattern classification. The new vector (black cross) gets assigned to the class that the majority of its k closest neighbours have (in this case it would be the orange circle shape). b: A variation is the nearest-centroid method in which the closest mean vector of a class of vectors defines the classification of a new input. This can be understood as a k-nearest neighbour method with preprocessed data and k = 1.

There is a neat method to find the overlap of two states known as the *swap test*. Given a quantum state $|a, b, 0_{anc}\rangle$ containing the two wavefunctions as well as an ancilla register initially set to 0, a Hadamard transformation sets the ancilla into a superposition $1/\sqrt{2}(|0\rangle + |1\rangle)$ followed by a controlled SWAP-gate on a and b which swaps the two states under the condition that the ancilla is in state $|1\rangle$. A second Hadamard gate on the ancilla results in state $|\psi_{SW}\rangle = \frac{1}{2}|0\rangle(|a, b\rangle +$

$|b,a\rangle) + \frac{1}{2}|1\rangle(|a,b\rangle - |b,a\rangle)$ for which the probability of measuring the ground state is given by

$$P(|0_{anc}\rangle) = \frac{1}{2}(1 + \langle a|b\rangle^2)$$

the distance between two real-valued n-dimensional vectors a and b is computed through a quantum measurement. The following states are prepared :

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0,a\rangle + |1,b\rangle)$$
$$|\phi\rangle = \frac{1}{\sqrt{Z}}(\|a\||0\rangle - \|b\||1\rangle)$$
$$\Rightarrow \frac{1}{4}Z\| \langle\phi|\psi\rangle \|^2 = \|a-b\|^2$$

where, $Z = \|a\|^2 + \|b\|^2$. Hence, we can compute the distance between two vectors using this method. Quantum clustering algorithms work along similar lines so will not be covered here.

B. Quantum pattern recognition

The aim of this model is to measure the hamming distance betwen two binary quantum states. Say we are given two binary strings as $|a_1a_2...a_n\rangle$ and $|b_1b_2...b_n\rangle$ with entries $a_i, b_i \in \{0,1\}$. Now construct,

$$|\psi\rangle = |a_1a_2...a_n, b_1b_2...b_n\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Now, denote

$$\bar{d_k} = \begin{cases} 0, \text{if } |a_k\rangle = |b_k\rangle \\ 1, \text{else} \end{cases}$$

Now, over-write qubits of b using `XOR` gate.

$$|\psi^{'}\rangle = |a_1...a_n, \bar{d_1}, ..., \bar{d_n}\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

To write the total Hamming distance $\bar{d_H}(\bar{a},\bar{b})$, we use the unitary operator $U = exp(-i\frac{\pi}{2n}H)$ with $H = 1 \otimes \sum_k (\frac{1}{2}(\sigma_z + 1))_{d_k} \otimes \sigma_z$. Note that this adds a negative sign to the ancilla. Now, apply $H_{anc} = 1 \otimes 1 \otimes H$ consequently resulting in,

$$|\psi^{''}\rangle = cos[\frac{\pi}{2n}\bar{d_H}(\bar{a},\bar{b})]|a_1...a_n, \bar{d_1}, ..., \bar{d_n}, 0\rangle + sin[\frac{\pi}{2n}\bar{d_H}(\bar{a},\bar{b})]|a_1...a_n, \bar{d_1}, ..., \bar{d_n}, 1\rangle$$

Measuring the ancilla in $|0\rangle$ leads to a state in whichthe amplitude scales with the Hamming distance between a and b.

C. Decision Trees

Decision trees (Figure 22) are important models which deal with functions that split dataset into "most-organised" sub-datasets. This can be measured in terms of Shannon's entropy.

However, the quantum analogue of this is not very clear. So, we won't go into further depth here.

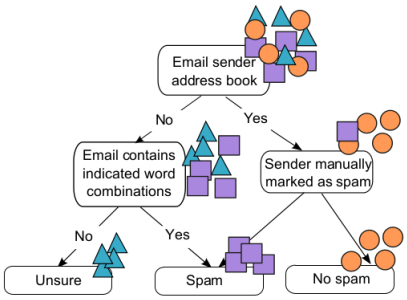

Fig. 22. A simple example of a decision tree for the classification of emails. The geometric shapes symbolise feature vectors from different classes that are devided according to decision functions along the tree structure

D. Hidden quantum Markov Models

Hidden Markov models (Figure 23) are another important method of machine learning that have been investigated from the perspective of quantum information. In a (first order discrete and static) Markov model, a system has a countable set of states $S = \{s_m\}_{m=1,...,M}$ and the transition between these states are governed by a stochastic process in such a way that given a set of transition probabilities $\{a_{ml}\}_{m,l=1,\cdots,M}$, the system's state at time $t+1$ only depends on the previous state at time t In a hidden markov model, the state of the system is only accessible through observations at time $t(o_t)$ that can take one set of symbols, and an observation again has a certain probability to be invoked by a specific state.

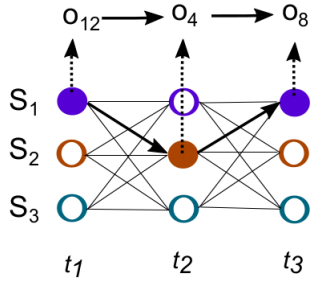

Fig. 23. A hidden Markov model is a stochastic process of state transitions. In this sketch, the three states s_1, s_2, s_3 are connected with lines symbolising transition probabilities. A deterministic realisation is a sequence of states, here the transition $s_1 \to s_2 \to s_1$ that give rise to observations $o_{12} \to o_4 \to o_8$. A task for hidden Markov models is to guess the most likely state sequence given an observation sequence.

The state of a system is given by a density matrix ρ and transitions between states are governed by completely positive trace-nonincreasing superoperators A^i acting on these matrices. These operations can always be represented by a set of Kraus operators $\{K_1^i, ..., K_q^i\}$ fulfilling the probability conservation condition $\sum_q K_q^{i\dagger} K_q^i \leq 1$, (don't know what Kraus operators are? Just think of them as operators for now)

$$\rho^{'} = A^i\rho = \sum_q K_q^{i\dagger}\rho K_q^i$$

The probability of obtaining state $\rho_s = P(\rho_s)^{-1}A^s\rho$ is given by

$$P(\rho_s) = tr[A^s\rho]$$

IX. Quantum GANs (Literature Review)

A. Classical GANs

First, let's explore what classical GANs are. We suppose that the real-world data comes from some fixed distribution $p_R(x)$, generated by some process R. Classical GANs are based on training two functions, a generator(G) and a discriminator(D). G transforms this noise source into data samples $x = G(\theta_G, z)$, creating the generator distribution $p_G(x)$. In the ideal case of a perfectly trained generator G, the discriminator would not be able to decide whether a given sample x came from $p_G(x)$ or from $p_R(x)$. Therefore, the task of training G corresponds to the task of maximizing the probability that D misclassifies a generated sample as an element of the real data. On the other hand, D's goal is to discriminate between these two classes, outputting a binary random variable. Training D thus corresponds to maximizing the probability of successfully classifying real data, while minimizing the probability of misclassifying fake data.

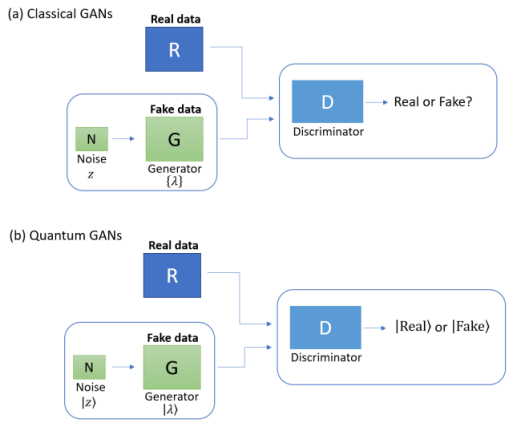

Fig. 24. The figure shows a high-level overview of classical and quantum GANs

B. Quantum GANs

For the quantum case, suppose we are given a data source R which, given a label $|\lambda\rangle$, outputs a density matrix ρ_λ^R into a register containing n subsystems, i.e.,

$$R(|\lambda\rangle) = \rho_\lambda^R$$

The general aim of training a GAN is to find a generator G which mimics the real data source R. The generator takes as input the label $|\lambda\rangle$ and an additional state (noise) $|z\rangle$, and produces a quantum state as a density matrix. Moreover, the gates of the ciruit of G are parametrised by a vector $\vec{\theta_G}$

$$G(\vec{\theta_G}, |\lambda, z\rangle) = \rho_\lambda^G(\vec{\theta_G}, z)$$

Why do we use an extra parameter z?

1) It can be seen as a source of unstructured noise which provides entropy within the generated data. So instead of just getting one output, you get mutliple outputs for the same label.
2) It can be seen as a control for the generator, by tuning $|z\rangle$, we can transform the output state prepared by the generator, varying properties of the generated data which are not captured by the labels λ.

Now, how do we train the functions D and G?

The task of D is to determine whether a given input state was created by the real data source R or the generator G, whereas the task of G is to fool D into accepting its output as being real. If the input was created by R, then D should output $|real\rangle$ in its output register, otherwise it should output $|fake\rangle$.

The optimization objective for QuGAN training (based on the above discussion) can be formalised as the adversarial task :

$$\min_{\vec{\theta_G}} \max_{\vec{\theta_D}} \frac{1}{S}\sum_{\lambda=1}^{S} Pr[(D(\vec{\theta_D}, |\lambda\rangle, R(|\lambda\rangle)) = |real\rangle) \cap (D(\vec{\theta_D}, |\lambda\rangle, G(\vec{\theta_G}, |\lambda, z\rangle)) = |fake\rangle)]$$

The function D takes three arguements, the parameter vector $\vec{\theta_D}$, and a label, image.

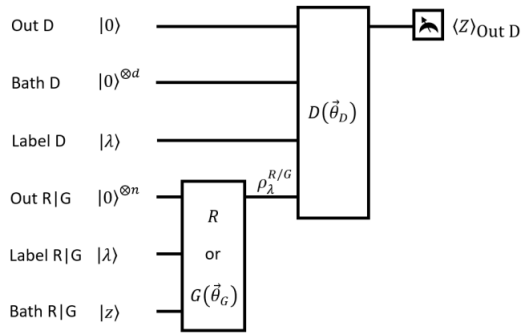

Fig. 25. The general structure of QuGANs

The real source R or the parametrized generator $G(\vec{\theta_G})$ is applied on an initial state $|0, \lambda, z\rangle$ respectively defined on the Label $R|G$ (s-qubit system), Out $R|G$ and Bath $R|G$ registers. The discriminator $D(\vec{\theta_D})$ uses the information $\rho_\lambda^{R/G}$ from the source and an initial resource state $|0, 0, \lambda\rangle$ defined on the Out D, Bath D and Label D registers. D outputs its answer $|real\rangle$ or $|fake\rangle$ in the Out D register. Now, let Z be defined as,

$$Z \equiv |real\rangle\langle real| - |fake\rangle\langle fake|$$

The expectation value $\langle Z\rangle_{\text{Out D}}$ is proportional to the probability that D outputs $|real\rangle$. Now that we have a concrete output to our quantum circuit, we need to design a quantum loss function to update the parameters of the QuGAN.

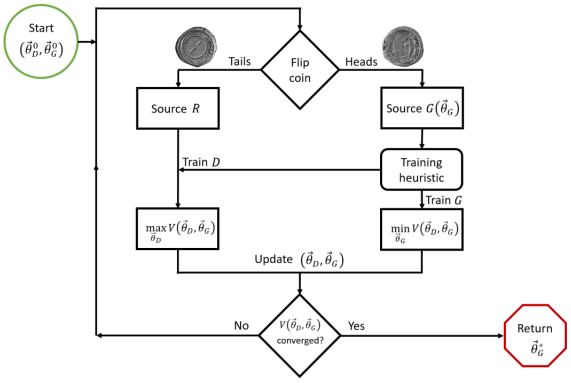

Fig. 26. Algorithmic flow of the training of a QuGAN

C. Quantum loss function

The schematic of training a GAN is given in figure 26.

For keeping this report breif, I will not go into the details of how the below expressions are derived. The reader can refer [7] for the same. The quantum cost function is given as,

$$V(\vec{\theta_D}, \vec{\theta_G}) = \frac{1}{2} + \frac{1}{2S}\sum_{\lambda=1}^{S}(cos^2(\phi)tr(Z\rho_\lambda^{DR}(\theta_D)) - sin^2(\phi)tr(Z\rho_\lambda^{DG}(\vec{\theta_D}, \vec{\theta_G}, z)))$$

If the coin used in figure 26 is fair, then we put $\phi = \frac{\pi}{4}$ and the optimisation problem reduces to,

$$\min_{\vec{\theta_G}} \max_{\vec{\theta_D}} V(\vec{\theta_D}, \vec{\theta_G})$$

Now that we have the cost function, the simplest thing to train our circuit is using gradient descent. So the update step at each iteration will be given as,

$$\vec{\theta_D^{k+1}} = \vec{\theta_D^k} + \chi_D^k \nabla_{\vec{\theta_D}} V(\vec{\theta_D^k}, \vec{\theta_G^k})$$
$$\vec{\theta_G^{k+1}} = \vec{\theta_G^k} + \chi_G^k \nabla_{\vec{\theta_G}} V(\vec{\theta_D^k}, \vec{\theta_G^k})$$

where χ_D^k, χ_G^k are learning rates for the model.

D. Quantum Gradient

Now, we need a quantum gradient circuit to compute the gradients required for gradient descent. Turns out, we can design a circuit for this as well! As with the last section, we shall not derive the circuit here, but I'll show the schematic of the circuit. But first, some notation (fig 27)

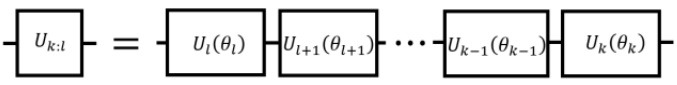

Fig. 27. This notation is used to signify the decomposition of a unitary transformation in its elementary parametrized gates.

A unitary transformation U parametrized by a vector $\vec{\theta}$ with N components is denoted as,

$$U(\vec{\theta}) = U_N(\theta_N)...U_1\theta_1$$

Assuming each element is generated by a Hamiltonian $h_j = h_j^\dagger$, an individual gate has the form,

$$U_j(\theta_j) = e^{-\frac{i}{2}\theta_j h_j}$$

Under this premise, the gradient calculating circuits are given by fig 28

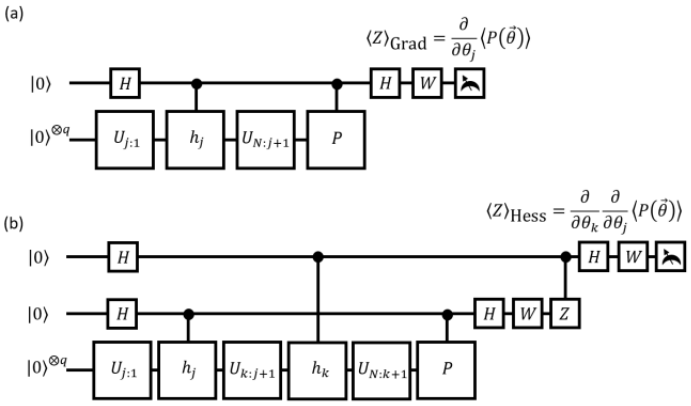

Fig. 28. In (a), we show the general structure of quantum gradients and the structure of quantum Hessians is shown in (b).

Hence, we can use these gradients in gradient descent calculations and we're done!

Conclusion and Remarks

I would like to thank the mentors of this project Siddhant Midha and Aditya Sriram for giving me the opportunity to study and appreciate this subject. I would suggest anyone who is interested in coding problems, ciphers, cryptography, quantum mechanics or just mathematics to try out quantum computing. Personally, I enjoyed QC-QI eventhough I didn't particularly enjoy my first year quantum physics course. As far as resources are concerned, the reader can refer [3] (the project repo), or one can refer my personal repository as well.

References

[1] M. Schuld, I. Sinayskiy and F. Petruccione, "An introduction to quantum machine learning," September 2014.
[2] M. Neilson and I. Chuang, "Quantum Computing and Quantum Information, 10th edition"
[3] S. Midha and A. Sriram "Learning with Quantum Computers" *Github repository*, January 2023.
[4] C. H. Bennett, G. Brassard "Quantum cryptography: Public key distribution and coin tossing" March 2020.
[5] O. Simeone "An Introduction to Quantum Machine Learning for Engineers" May 2022.
[6] D. J. Bernstein. "Detecting perfect powers in essentially linear time". Mathematics of Computation 67 (1998), 1253-1283 Date: 1997.11.06. AMS version: 31pp.
[7] Pierre-Luc Dallaire-Demers, N. Killoran "Quantum generative adversarial networks" April 2018
[8] S. Keshav "How to Read a Paper", David R. Cheriton School of Computer Science, University of Waterloo