The School of Mathematics

THE UNIVERSITY
of EDINBURGH

A Survey on Training Algorithms for Quantum Neural Networks

by

Kwok Chi Long Jason

Dissertation Presented for the Degree of
MSc in Operational Research with Data Science

August 2023

Supervised by
Dr Stefano Cipolla

Abstract

This dissertation surveys Quantum Neural Networks and their training methods. A Quantum Neural Network is a hybrid quantum-classical model devised in the Noisy intermediate-scale quantum era to leverage the principles of quantum computing and machine learning. It is a highly expressive model which has an analogy to classical neural networks. The model's training involves the parameter shift rule to find the analytic gradient used to make an update step in each classical optimisation iteration. In addition to typical classical optimisation algorithms, advanced gradient-based optimisation methods for training quantum neural networks are surveyed, including natural gradient, quantum natural gradient and simultaneous perturbation stochastic approximation algorithms. The behaviour between optimisation algorithms is compared through experiments, outlining the strengths and weaknesses of each method.

Keywords: Quantum Neural Networks, Quantum Machine Learning, Parameter Shift Rule, Quantum Fisher Information, Quantum Natural Gradient

Acknowledgments

I want to take this precious opportunity to express my gratitude to those who supported me in completing the dissertation. I am very grateful to Dr Cipolla, for giving me advice on the direction of the topic and the organisation of my dissertation. I have learnt a lot from him and without his help, this project will not be made possible. I would like to express my thanks to my family and friends in Edinburgh, who accompany me always and give me emotional support. Words cannot express how I am thankful.

Own Work Declaration

I, Kwok Chi Long Jason, declare that this dissertation, titled ”A Survey on Training Algorithms for Quantum Neural Networks” are my own work. I confirm that any assistance received in preparing this dissertation has been acknowledged and that all references have been appropriately cited and referenced.

I understand any act of plagiarism can lead to serious consequences and I have adhered to the university’s policy on academic integrity.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

Originally, *Quantum Computing* is a discipline which aims at solving problems intractable in classical computing by harnessing the fundamental principles of quantum mechanics. With the evolution of modern technology, quantum algorithms gradually become realisable and the community of quantum computing has been growing rapidly, leading to thoughts of whether one could use quantum computers for machine learning tasks. The fusion of the two domains, machine learning and quantum computing, has given rise to an emerging field known as *Quantum Machine Learning.* The core of quantum machine learning is the concept of *Quantum Neural Networks*, a fancy name referring to using parameterised quantum models for machine learning. The idea of Quantum neural networks offers a different perspective to process information and learning from data in contrast to classical learning models, namely using quantum information and entanglement, thereby holding the potential to revolutionise industries when quantum resources become more accessible.

Despite its name being similar to the well-known feed-forward neural networks, the nature of quantum neural networks and their training methods are quite different. Therefore, for helping those who wanted to grasp the idea of quantum neural networks but do not have any background in quantum computing or quantum machine learning, a comprehensive survey of training algorithms for quantum neural networks becomes imperative. This survey aims to throw light on the current research in quantum neural networks and to inspire readers to continue their journey in comprehending quantum machine learning. In this survey, we will delve into introducing the basic ideas of quantum computing and quantum machine learning, and the training algorithms for quantum neural networks. We aspire to provide a comprehensive understanding of the current advancements in training quantum neural networks.

The contents of this survey are structured as follows. Chapter 2 provides a summary of the notions of quantum computing, emphasising elements that are the heart of quantum machine learning. In Chapter 3, we review the foundations of classical machine learning and its development into quantum neural networks. We will give an example of a typical architecture of a quantum neural network, how a quantum neural network can be applied in classification tasks with code examples and explain how quantum neural networks learn from data. In Chapter 4, we survey common training algorithms tailored for Quantum Neural Networks, this includes how gradients of a quantum neural network are estimated, how we can utilise quantum information to optimise parameters in a quantum neural network, and tackling issues when training quantum neural networks.

In summary, this survey endeavours to provide a self-explanatory review of training methods for quantum neural networks. After walking through, readers are expected to have a comprehensive understanding of the synergies between machine learning and quantum computing. All in all, this survey aims to pave the way for innovative applications that unleash the potential of quantum machine learning models.

2 Background: Quantum Computing

Quantum Computing is a discipline of computing that applies the principles of quantum mechanics to perform computations. The attention to the need for quantum computation was warranted by Feynman in the 80s, who discovered that some quantum mechanical phenomena could not be efficiently simulated by a classical computer [13]. This sparked the idea of creating a new kind of computer that utilises quantum effects. But in fact, the excitement of quantum computation among scientists stemmed from the fact that the computational space grows exponentially as the size of a quantum system, expands and such exponential parallelism can yield potentially exponential speed-ups of quantum algorithms than their classical counterparts [28]. Shor, in 1994, amazed academia by proposing a quantum computer algorithm to find an integer's prime factors that only runs in polylogarithmic times [34]; Deutsch and Jozsa, in 1992, described a quantum algorithm to determine a function to be either balanced or constant with exponential speed up, comparing to those solving the same problem classically [10]; Grover, in 1996, devised a quantum search algorithm using merely $O\left(\sqrt{N}\right)$ evaluations, while the classical approach requires $O\left(N\right)$ evaluations [16]. These giants fostered the quest for finding other quantum algorithms among scientists and this includes exploring machine learning algorithms in a quantum computer setting. Hence, before introducing the concepts of a quantum neural network, a typical quantum machine learning approach to be covered in the later sections, it is particularly crucial to illustrate the basic concepts of quantum computing. Therefore, we will explain the idea of quantum computing in this whole section with reference to [18]. The postulates of quantum mechanics will be first introduced in Section 2.1, followed by an outline of quantum computing in Section 2.2. As quantum computing itself is a very broad and advanced topic, we will not cover all of them but the essential parts that suffice to learn the idea of quantum machine learning.

2.1 Postulates of Quantum Mechanics

2.1.1 State Space

A *quantum state* resides in a complex inner product space $\mathcal{H}$, which is called a *Hilbert Space*. According to *Dirac notation* [11], we use a *ket* vector $|\psi\rangle$ to denote a quantum state, where $|\psi\rangle$ is written as a column vector. The conjugate transpose of $|\psi\rangle$, also known as a *bra* vector, is represented as $\langle\psi|$ and it is written as a row vector. It is defined that in this Hilbert Space, the inner product of two vectors, $|\psi_1\rangle$ and $|\psi_2\rangle$, denoted as $\langle\psi_1|\,|\psi_2\rangle$, is the complex conjugate row vector $\langle\psi_1|$ multiplied by $|\psi_2\rangle$. Thus we can see that $\langle\psi_2|\,|\psi_1\rangle^* = \langle\psi_1|\,|\psi_2\rangle$, meaning that the conjugate transpose of the inner product between $|\psi_1\rangle$ and $|\psi_2\rangle$ is the inner product. For any vector $|\psi\rangle \in \mathcal{H}$, $\sqrt{\langle\psi|\,|\psi\rangle}$ is then the norm of $|\psi\rangle$, $\|\psi\|$.

Suppose there is a set of orthonormal basis $\{|e_i\rangle\}$ $\forall\ i = 1, ...N$ which spans $\mathcal{H}$. Any quantum state $|\psi\rangle \in \mathcal{H}$, we can express it as a linear combination of these basis states

$$|\psi\rangle = \sum_i |e_i\rangle\,\langle e_i|\,|\psi\rangle = \sum_i \langle e_i|\,|\psi\rangle\,|e_i\rangle\ . \tag{2.1}$$

Furthermore, the summation of the outer product of all basis states is the identity matrix because

$$|\psi\rangle = \sum_i |e_i\rangle\,\langle e_i|\,|\psi\rangle = \left(\sum_i |e_i\rangle\,\langle e_i|\right)|\psi\rangle$$
$$\therefore \sum_i |e_i\rangle\,\langle e_i| = \mathbb{I}. \tag{2.2}$$

A *pure state* is a quantum state represented by a state vector $|\psi\rangle$. When the system is in a pure state represented by $|\psi\rangle$, we are certain that this quantum state is in $|\psi\rangle$. A *mixed state* implies that the quantum state is not certain and hence is described by a combination of pure states with different probabilities. For example, suppose there are pure quantum states $|\psi_1\rangle$ and $|\psi_2\rangle$. In a quantum system, if there is a probability of 0.5 that the quantum state is in $|\psi_1\rangle$ and a probability of 0.5 that the quantum state is in $|\psi_2\rangle$, then this state is a mixed state. In this instance, it is suitable to use the density operator ρ to characterise any mixed states as the density operator represents a combination of pure states with different probabilities:

$$\rho = \sum_{j=1}^{J} p_j \left|\psi_j\right\rangle \left\langle\psi_j\right| \tag{2.3}$$

where:

- p_j is the probability of the system in $\left|\psi_j\right\rangle$ $\forall j = 1, ..., J$
- $p_j \geqslant 0$ and $\sum\limits_{j=1}^{J} p_j = 1$ $\forall j = 1, ..., J$.

When the quantum system is the pure state $|\psi\rangle$, ρ will just be the outer product $|\psi\rangle \langle\psi|$.

2.1.2 Observables

Observables are physical properties of a quantum system which are quantified through measurement and they mathematically correspond to a linear operator $\mathcal{M}$ in $\mathcal{H}$ acting on the quantum state $|\psi\rangle$. Each observable is associated with a physical property such as energy or spin and has a set of real eigenvalues, which are the possible values that can be measured for that property. To put it in another way, measuring an observable produces a value, specifically one of the eigenvalues linked to the corresponding operator.. The linear operator $\mathcal{M}$ an observable corresponds to should be ensured to be a Hermitian matrix, i.e., $\mathcal{M} = (\mathcal{M}^*)^T = \mathcal{M}^\dagger$ because the measurement result should always be a real number, i.e., $\mathcal{M}$ should have real eigenvalues; and the observable should be measured independently while not interfering with each other, i.e., $\mathcal{M}$ should have orthogonal eigenvectors associated with distinct values. Therefore, $\mathcal{M}$ has a spectral decomposition

$$\mathcal{M} = \sum_i \lambda_i \left|v_i\right\rangle \left\langle v_i\right| \tag{2.4}$$

with a set of eigenvectors $\{|v_i\rangle\}$ corresponding to eigenvalues λ_i.

The expected value of an observable is expressed as $\langle\mathcal{M}\rangle = Tr\{\rho\mathcal{M}\}$, where Tr is a trace of $\rho\mathcal{M}$. Considering a pure state $\rho = |\psi\rangle \langle\psi|$, the expected value of an observable concerning the pure state will be

$$\langle\mathcal{M}\rangle = Tr\{\rho\mathcal{M}\} = Tr\left\{|\psi\rangle \langle\psi| \mathcal{M}\right\} = Tr\left\{\langle\psi| \mathcal{M} |\psi\rangle\right\} = \langle\psi| \mathcal{M} |\psi\rangle . \tag{2.5}$$

2.1.3 Projective Measurement

Consider a set of *projective measurement* operators $\{\mathcal{M}_m\}_{m=1}^{n}$, where the index m corresponds to the measurement outcome. $\mathcal{M}_m$ is the projector satisfying $\mathcal{M}_m^2 = \mathcal{M}_m$. For a pure quantum state $|\psi\rangle$,

1. The probability of measuring an outcome m is

$$p(m) = \langle\psi| \mathcal{M}_m^\dagger \mathcal{M}_m |\psi\rangle . \tag{2.6}$$

2. Because $\sum_m p(m) = 1$, this means that

$$\begin{gathered}\langle\psi| \textstyle\sum_m \mathcal{M}_m^\dagger \mathcal{M}_m |\psi\rangle = 1, \\ \textstyle\sum_m \mathcal{M}_m^\dagger \mathcal{M}_m = \mathbb{I}.\end{gathered} \tag{2.7}$$

3. After observing the outcome m, the quantum state of the system becomes a state consistent with the measurement outcome (also known as the wave function collapse), and should then be normalised as

$$|\psi\rangle \rightarrow \frac{\mathcal{M}_m|\psi\rangle}{\sqrt{\langle\psi|\mathcal{M}_m^\dagger \mathcal{M}_m|\psi\rangle}}. \tag{2.8}$$

2.1.4 Time Evolution

Time Evolution in quantum mechanics refers to a quantum state's transition due to the passage of time. All quantum mechanical time evolutions are controlled by the Schrödinger equation

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

where

- H is the Hamiltonian of a quantum system, a hermitian operator representing the system's total energy
- $\hbar$ is a physical constant. By assumption, it is usually 1 for simplicity

The unique solution to the Schrödinger equation satisfying the initial condition $\left|\psi(0)\right\rangle = \left|\psi_0\right\rangle$ is

$$\left|\psi(t)\right\rangle = U(t)\left|\psi_0\right\rangle,\ U(t) = \exp\left(-\frac{it}{\hbar}H\right) \tag{2.9}$$

in which the time evolution operator $U(t)$ must be a unitary matrix, i.e., $U^\dagger(t) = U^{-1}(t)$. thanks to its orthogonality, $U(t)$ ensures the preservation of the norm in quantum mechanical time evolution.

2.1.5 Composite System

Consider Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, where $\mathcal{H}_i$ corresponds to the quantum system i. The joint space $\mathcal{H}$ is $\mathcal{H}_1 \otimes \mathcal{H}_2$ with dimensions $\dim(\mathcal{H}) = \dim(\mathcal{H}_1) \times \dim(\mathcal{H}_2)$, where the sign $\otimes$ refers to the tensor product. With $\left|\psi_1\right\rangle \in \mathcal{H}_1$ and $\left|\psi_2\right\rangle \in \mathcal{H}_2$, the joint quantum state $\left|\psi\right\rangle$ comprised of $\left|\psi_1\right\rangle$ and $\left|\psi_2\right\rangle$ is

$$\left|\psi\right\rangle = \left|\psi_1\right\rangle \otimes \left|\psi_2\right\rangle.$$

This expression also means the joint quantum state $\left|\psi\right\rangle$ is separable. Any joint quantum state which cannot be expressed in this representation is called an *entangled state*. The formal definition of *entanglement* is that the whole quantum system cannot be expressed as a simple product of its individual parts' states. In other words, if a quantum system is entangled, without considering other local constituents, we cannot fully describe one constituent of this quantum system. A quantum computing example of an entangled state is illustrated in Equation (2.26). In addition, the phenomenon of entanglement has profound implications for quantum physics, quantum teleportation and quantum machine learning, the latter of which will be explored in later sections.

Besides, an operator O which acts on the joint space $\mathcal{H}$ can be reformulated as

$$O = O_1 \otimes O_2.$$

where operator O_i acts on quantum system i

2.2 Introduction to Quantum Computing

2.2.1 Qubit

A *qubit* is the fundamental information unit in quantum computing and is realised as a two-level quantum system. We can express a state of a qubit by a linear combination of the following *computational bases*, which is the orthonormal basis of a two-dimensional Hilbert space, namely

$$\left|0\right\rangle = \begin{pmatrix}1\\0\end{pmatrix},\ \left|1\right\rangle = \begin{pmatrix}0\\1\end{pmatrix}. \tag{2.10}$$

Any qubit state thus can be written as

$$\left|\psi\right\rangle = \alpha_0\left|0\right\rangle + \alpha_1\left|1\right\rangle = \begin{pmatrix}\alpha_0\\\alpha_1\end{pmatrix},\ \alpha_i \in \mathbb{C}\ \forall i = 1,2 \tag{2.11}$$

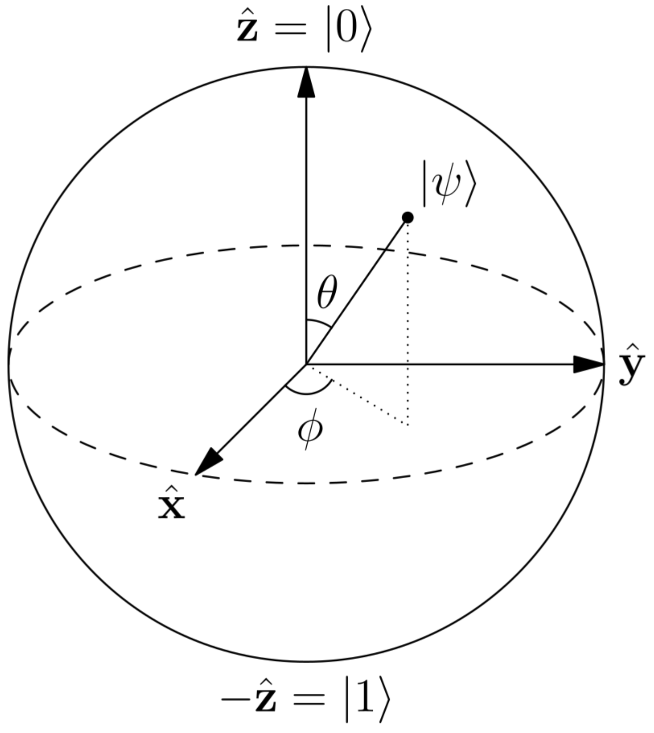


Figure 1: Bloch Sphere as a Geometric Representation of a Single Qubit
[2]

such that $|\psi\rangle$ is a superposition of the computational bases $|0\rangle$ and $|1\rangle$. The Born Rule tells us that the probabilities of finding the qubit in state $|0\rangle$ and $|1\rangle$ are

$$p\left(|\psi\rangle = |0\rangle\right) = |\alpha_0|^2, p\left(|\psi\rangle = |1\rangle\right) = |\alpha_1|^2. \tag{2.12}$$

And the sum of those probabilities in any state should sum up to one

$$|\alpha_0|^2 + |\alpha_1|^2 = 1. \tag{2.13}$$

One can also utilise the *Bloch sphere* to describe a qubit geometrically. A qubit is reformulated as

$$|\psi\rangle = e^{(i\gamma)}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right), \tag{2.14}$$

where $\theta = [0, \pi]$ and $\phi = [0, 2\pi]$ are the spherical coordinates such that $|\psi\rangle$ is visualised in a Bloch sphere, shown in Figure 1. Notice that $e^{(i\gamma)}$ is the global phase that does not have any observable effect.

On top of that, a system of n unentangled qubits can be written as tensor products of single qubits $|q_i\rangle$

$$|\psi\rangle = |q_1\rangle \otimes |q_2\rangle \otimes ... \otimes |q_n\rangle = |q_1 q_2 ... q_n\rangle. \tag{2.15}$$

To represent entangled states (and also unentangled states), it is common to write

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = \alpha_0 |0...00\rangle + \alpha_1 |0...01\rangle + ... + \alpha_{2^n-1} |1...11\rangle, \tag{2.16}$$

$$\alpha_i \in \mathbb{C},\ \textstyle\sum_i |\alpha_i|^2 = 1\ \forall i = 1, ..., 2^n - 1. \tag{2.17}$$

Note $|0...00\rangle, ..., |1...11\rangle$ are the n-qubit computational basis, and as such each binary string can be represented by an integer i for simplicity.

2.2.2 Quantum Gates

Quantum logic gates are unitary transformations operated in a quantum computer. Being a unitary transformation means the norm of quantum states is preserved when quantum logic gates are applied and all quantum gates are reversible, i.e., any quantum logic gate U acting on n qubits corresponds to a reverse operation $U^\dagger$, where $U, U^\dagger \in \mathbb{C}^{2^n} \times \mathbb{C}^{2^n}$ and $UU^\dagger = \mathbb{I}$.

The most common single-qubit gates will be introduced in the following. First of all, a quantum equivalent of the NOT gate is known as an X gate, represented as

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.18}$$

It is straightforward to show that $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. There are also Y and Z gates,

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.19}$$

Gates X,Y and Z have specific names, known as *Pauli Gates*, and also can be denoted by σ_x, σ_y and σ_z. Each Pauli Gate X, Y, and Z acting on a qubit is the same as the single qubit state rotations about the x, y, and z axes of the Bloch Sphere by the angle π respectively. Furthermore, rotations with angle θ about the Bloch Sphere's x,y, and z axes can be generalised as these *Pauli rotations* respectively:

$$R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\cdot\sin\frac{\theta}{2} \\ -i\cdot\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \; R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}. \tag{2.20}$$

A *Hadarmard gate H* transforms a single qubit into a superposition state, such that

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.21}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{2.22}$$

If we want to apply each gate U_i to ith qubit q_i respectively, the resulting operation matrix U can be written as a tensor product $U = U_1 \otimes \ldots \otimes U_n$ so that

$$U|q_1...q_n\rangle = (U_1 \otimes \ldots \otimes U_n)|q_1...q_n\rangle. \tag{2.23}$$

An instance of a 2-qubit gate is the CNOT gate that set the one qubit state conditioned on another qubit state. It performs the NOT gate on the second qubit when the first qubit is state $|1\rangle$, else performs the Identity Gate $\mathbb{I}$ on the second qubit (Performing the identity gate $\mathbb{I}$ does not result in any changes, and $\mathbb{I}$ is of the same matrix form as the identity matrix) when the first qubit is state $|0\rangle$. CNOT gate has a form

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.24}$$

To illustrate how CNOT gate works, the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are equivalent to the canonical basis vectors $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ respectively when expressed in tensor products. Therefore, the ith column of the CNOT gate is the outcome acted on the ith basis vectors. An example is when performing the CNOT gate on $|10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$, as the first qubit state is $|1\rangle$, we flip the second qubit. The resulting state will $|11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$, which is the third column of the CNOT gate. Hence we can conclude that

$$CNOT\,|00\rangle = |00\rangle\,, CNOT\,|01\rangle = |01\rangle\,, CNOT\,|10\rangle = |11\rangle\,, CNOT\,|11\rangle = |10\rangle\,. \tag{2.25}$$

CNOT gate is crucial for creating an entangling circuit and is extremely useful for creating entangling layers mentioned in Section 3.2.1. For instance, this operation

$$CNOT\left((H\otimes\mathbb{I})\,|00\rangle\right) = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle \tag{2.26}$$

creates an entangled state, called a *Bell state.*

In essence, Table 1 summarises the lists of typical quantum logic gates by the operator's name, the circuit representation, as well as the matrix form of each quantum logic gate.

2.2.3 Measurement of Qubit(s)

Usually, the measurements of qubits is a computational basis measurement to measure if each qubit state is $|0\rangle$ or $|1\rangle$. Such measurement of the qubit state is called Z-basis measurement and is represented by $\mathcal{M}_0 = |0\rangle\langle 0|$, $\mathcal{M}_1 = |1\rangle\langle 1|$. Given that $|\psi\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle\,, \alpha_0, \alpha_1 \in \mathbb{C}$, the probability that we observe outcome 0 is

$$\langle\psi|\,M_0\,|\psi\rangle = \langle\psi|\,|0\rangle\,\langle 0|\,|\psi\rangle = |\,\langle 0|\,|\psi\rangle\,|^2 = |\alpha_0|^2 \tag{2.27}$$

and that of 1 is

$$\langle\psi|\,\mathcal{M}_1\,|\psi\rangle = \langle\psi|\,|1\rangle\,\langle 1|\,|\psi\rangle = |\,\langle 1|\,|\psi\rangle\,|^2 = |\alpha_1|^2 \tag{2.28}$$

If the measurement outcome is 0, $|\psi\rangle$ is collapsed into the state $|\psi\rangle \leftarrow \frac{\mathcal{M}_0|\psi\rangle}{\sqrt{\langle\psi|\mathcal{M}_0|\psi\rangle}} = |0\rangle$.

To recap, the full observable of Z-basis measurement is

$$Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \tag{2.29}$$

that has eigenvectors $|0\rangle$ and $|1\rangle$, respectively with eigenvalues $+1, -1$. Thus upon measurement, when we read off $+1$ (-1), we know that we have observed $|0\rangle$ $(|1\rangle)$ as the measurement outcome. If the measurement is not on Z-basis, (i.e., it is not measured in terms of computational basis), we will then create an observable matrix $A = U^\dagger Z U$ to define the outcomes of the measurement in ± 1 eigenvectors. For example, measurement on Y-basis is the same as applying gates $H\mathcal{S}^\dagger$ and measuring it on Z-basis, where

$$\mathcal{S} = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix} \tag{2.30}$$

is called a phase gate. The value of expectation of a single-qubit computational basis measurement, $\langle\sigma_z\rangle$, is between $[-1, 1]$ and practically speaking, such expectation is estimated by sampling the outcomes S times and calculating the average, where S is the shot number.

Similarly, the multi-qubit computational basis measurement's observable is defined as $Z\otimes Z$ for jointly measuring two qubits. But the details of multi-qubit measurement are out of the scope of this survey and thus will be omitted.

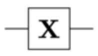
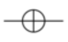
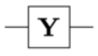
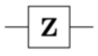
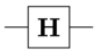
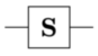
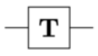
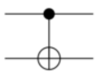
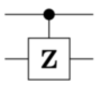
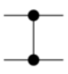
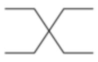
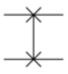
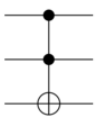
Operator	**Gate(s)**		**Matrix**
Pauli-X (X)			$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)			$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)			$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)			$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)			$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)			$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$
Controlled Z (CZ)			$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & -1\end{bmatrix}$
SWAP			$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$

Table 1: A List of Some Commonly Used Quantum Logic Gates [3]

2.2.4 Data Encoding

In order to learn from original data, it is important to know the methods to encode data in the quantum computer, particularly for quantum machine learning, because the procedures of inputting data into the quantum device are not just preparing the initial state for the quantum algorithm to be executed (and this will be further elaborated in Section 3.2.2). For instance, in classification tasks, data encoding methods transform the original data into a Hilbert Space with higher dimensions so that the classifier can linearly distinguish the sophisticated data in the Hilbert Space which could not be done in the original input data space. Here, several noteworthy data encoding methods for quantum computing will be introduced.

1. **Basis encoding** relates each classical bit of the binary data inputs to each qubit of a quantum system. Considering $x \in \mathbb{R}$, x is expressed as a τ-bit binary string b, where the precision of x depends on τ,
$$b := b_s b_{\tau_l - 1} \cdots b_1 b_0 . b_{-1} b_{-2} \cdots b_{-\tau_r}, \tau = (1 + \tau_l + \tau_r). \tag{2.31}$$
Note that b_s represents the sign, while τ_l and τ_r represent the integer and the proper fraction respectively. To convert b to x, we can use this formula
$$x \;=\; (-1)^{b_s} \left(b_{\tau_l - 1} 2^{\tau_l - 1} + .. + b_0 2^0 + b_{-1} 2^{-1} + b_{-2} 2^{-2} + ... + b_{-\tau_r} 2^{-\tau_r} \right). \tag{2.32}$$
If the N-dimensional data to be input is a vector, we can convert each element to a τ-bit binary string and then concatenate all strings together, with a total length of τN. The basis encoding algorithm is to invert some qubits to represent bits which are non-zero. To give an example, when we want to encode a string 0101 into the quantum system by basis encoding, we first initialise the quantum state $|0000\rangle$ and on the second and fourth qubit, we perform the NOT gate. The resulting matrix operation will be $(\mathbb{I} \otimes X \otimes \mathbb{I} \otimes X) |0000\rangle = |0101\rangle$. We should also bear in mind that if we hope to represent a feature vector accurately, this in turn requires a large number of qubits to do so. Hence, it is not a typical data encoding strategy.

2. **Amplitude encoding** is a strategy which encodes a input vector $x \in \mathbb{C}^N$ into a n-qubit quantum state $|\psi_x\rangle$'s amplitudes,
$$|\psi_x\rangle = \textstyle\sum_{i=0}^{N-1} x_i |i\rangle = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}, \tag{2.33}$$
where $n = log_2 N$ and the set $\{|i\rangle\}$ is the Hilbert Space's computational basis. As we are representing the classical information as amplitudes of a quantum state $|\psi_x\rangle$, we need to preprocess the input by normalising it such that $\sum_i |x_i|^2$. Amplitude encoding could be sophisticated so the details will be omitted here and we will assume that an efficient amplitude encoding method is given, but the idea is to use sequences of multi-controlled rotations to map the ground state $|0...0\rangle$ to $|\psi_x\rangle$.

3. **Time Evolution Encoding** uses a unitary evolution by a Hamiltonian H to encode a scalar $x \in R$, where the encoding operator is
$$U(x) = \exp(-ixH). \tag{2.34}$$
In fact, using the Pauli rotation gates to encode the data is a special case of time evolution encoding, where $H = \frac{1}{2}\sigma_a, a \in \{x, y, z\}$. This strategy is also named angle encoding or rotation encoding. For example, if we want to encode a feature vector $\boldsymbol{x}$ with N dimensions by angle

encoding where $\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$,$\{\vec{x} \mid x_i \in [0, 2\pi]\}$, the number of qubits required is N, and we can implement each Pauli X rotation on each qubit (see Equation (2.20)) such that the data encoder is in form

$$U(\vec{x}) = R_x(x_1) \otimes ... \otimes R_x(x_N). \tag{2.35}$$

For instance, let's say when $\vec{x} = \begin{bmatrix} \pi \\ \pi \\ \pi \end{bmatrix}$, a rotation encoding encodes $\vec{x}$ by rotating the three-qubit ground state around Y-axis by π, this results in the quantum state $|111\rangle$. Rotation encoding is typically applied in variational quantum algorithms for machine learning.

4. **Hamiltonian Encoding**, unlike the aforementioned encoding strategy which encodes the data in the states of qubits, encodes the data into the operator matrix instead by

$$|\psi'\rangle = exp(-iAt)\,|\psi\rangle\,. \tag{2.36}$$

This strategy is used in the Harrow-Hassidim-Lloyd algorithm to solve linear systems [17]. Essentially, Hamiltonian Encoding associates a Hamiltonian H with a Hermitian matrix A. If A is not a Hermitian, we can encode A into

$$H_A = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}. \tag{2.37}$$

Notice that any hermitian matrix A can be decomposed into some Pauli Matrics, i.e.,

$$A = \alpha + \beta\sigma_x + \delta\sigma_y + \gamma\sigma_z. \tag{2.38}$$

Hence, the unitary evolution operator $\exp(-iAt)$ can be expressed in matrix form by

$$\exp(-iAt) = R_x(\beta t)R_y(\delta t)R_z(\gamma t) \tag{2.39}$$

The characteristic of Hamiltonian encoding is that this encoding enables us to process the eigenvalues of A. One example is to multiply A^{-1} with a vector which is amplitude-encoded.

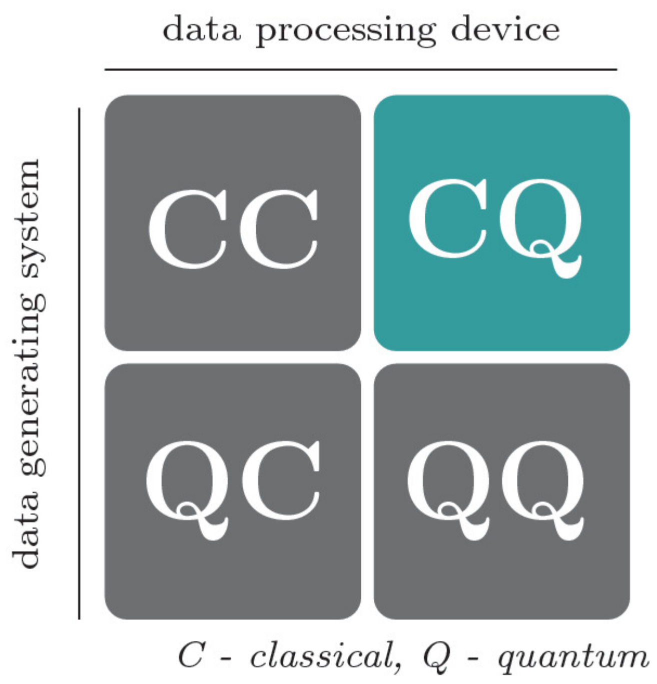


Table 2: Four Approaches to Quantum Machine Learning
[1]

3 Quantum Machine Learning

As discussed, quantum computing describes ways to process information based on quantum theory. Meanwhile, Machine learning is a subfield of artificial intelligence which helps systems to gain knowledge and improve their performance by learning from experience in order to solve problems, without requiring explicit programming. One natural question to ask is how these two disciplines could be related and merged together. In other words, what is quantum machine learning's nature and how quantum machine learning models can be designed using the knowledge of quantum computing and machine learning? As suggested in [1], there exist several approaches to quantum machine learning, in a total of four. These four methods come from two fundamental criteria: The first is whether the data originates from a quantum (Q) or classical system (C); The second is whether we process information in the quantum (Q) or classical (C) device. These four approaches are visualised as a table in Table 2.

The CC approach means that classical data is processed in a classical device. This may sound puzzling but the CC approach is in fact a quantum-inspired machine learning model which does not involve using any quantum computing devices. One example is tensor networks which are employed for the study of many-body quantum systems [27]; The QC approach is to use machine learning for assisting quantum computing. For instance, one can use supervised learning techniques such as neural networks to approximate the quantum states; The QQ approach uses a quantum device to process quantum data without any aids from a classical device. Currently, there is not much research on this topic and how quantum-quantum machine-learning algorithms can be implemented is still questionable and debatable. The last approach, which is the CQ approach, is the main focus of the survey. In the CQ approach, we are interested in investigating how we can use quantum computing to assist machine learning tasks: First, there are classical data observed, such as images and audio signals. Then, this data is fed into a quantum computing device for prediction or analysis. This approach is synonymous with quantum machine learning in many works of literature so we will assume the CQ approach to be equivalent to quantum machine learning from now on.

Quantum machine learning with respect to the CQ approach can be further divided into fault-tolerant quantum machine learning and near-term quantum machine learning. Devising fault-tolerant quantum machine learning algorithms aims to achieve significant speed-ups while achieving the same result as its classical counterpart. However, it can be hardly implemented due to the fact that it requires a substantial amount of quantum resources at an extremely high cost to correct quantum errors with the current technology and would not be a deployable solution in the near term until a breakthrough of large-scale quantum computers that is fault-tolerant. Therefore, as algorithms that require a fault-tolerant quantum device are impractical nowadays, another paradigm called near-term quantum machine learning has emerged, where this approach does not target at speed up but views the quantum device itself to be a machine learning model (Quantum neural networks is exactly the case). Comparing to fault-tolerant quantum machine learning, near-term quantum machine learning methods require far fewer computing resources and hence it is relatively not prone to quantum errors. This means that it is suitable to deploy near-term quantum machine learning right away. This section is going to present the essential concepts of near-term quantum machine learning, starting from reviewing machine learning theory in Section 3.1, and then to a comprehensive explanation of quantum neural networks in Section 3.2.

3.1 Machine Learning Theory Revisited

Machine Learning approaches are categorised into unsupervised learning, supervised learning, and reinforcement learning:

Definition 3.1. ***Supervised Learning***
Given a dataset $D := \{(x^{(m)}, y^{(m)})\}_{m=1}^{M}$ drawn from a probability density function $p(x, y)$, a supervised learning process's objective is to learn to map inputs x_m to outputs y_m. Such a model should generalise to observations outside of the training dataset and can be applied to making predictions given new data point x^*.

Definition 3.2. ***Unsupervised Learning***
Given a dataset $D := \{x^{(m)}\}_{m=1}^{M}$ drawn from a probability density function $p(x)$, an unsupervised learning process's objective is to learn to find patterns (or the underlying probability distribution $p(x)$) of D.

Definition 3.3. ***Reinforcement Learning***
The objective of a reinforcement learning task is to instruct an agent in making decisions in an environment and eventually learn an optimal policy, mapping states to actions, to maximise the total award over time.

Here supervised learning problem is focused. Firstly, a model $f_{\boldsymbol{\theta}}$ is parameterised by a parameter vector $\boldsymbol{\theta}$ and we define loss as a quantification of the model's quality. A loss refers to a measure of the error between the model's prediction and the training data's ground truth. Table 3 shows the common loss function for supervised learning tasks.

Loss Functions		
	$L(f_{\boldsymbol{\theta}}(x), y)$	Used for
ℓ_2 loss	$(f_{\boldsymbol{\theta}}(x) - y)^2$	Regression
ℓ_1 loss	$\lvert f_{\boldsymbol{\theta}}(x) - y \rvert$	Regression
hinge loss	$\max(0, 1 - y f_{\boldsymbol{\theta}}(x))$	Binary classification
logistic loss	$\log(1 + \exp(-y f_{\boldsymbol{\theta}}(x))$	Binary classification
cross-entropy loss	$-\sum_{d=1}^{D} y_d \log(p_d)$	Multi-label classification

Table 3: Lists of Common Loss Functions

We can use the notion of *risk minimisation* to solve a supervised learning task.

Definition 3.4. ***Risk***
Given inputs $x \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$ drawn from a probability density function $p(x, y)$, The risk associated with a model $f_{\boldsymbol{\theta}}$ using the loss function L, which is referred to as the expected loss, is

$$\mathcal{R}_{f_{\boldsymbol{\theta}}} = \mathbb{E}[L_{f_{\boldsymbol{\theta}}}] = \int_{\mathcal{X}} \int_{\mathcal{Y}} p(x, y) L(f_{\boldsymbol{\theta}}(x), y) dx dy \tag{3.1}$$

The model's performance on the whole data domain improves when the risk decreases.

However, in real life, the integrals are intractable and instead, we estimate $\mathcal{R}_{f_{\boldsymbol{\theta}}}$ using the *empirical risk*, which is defined as

$$\hat{\mathcal{R}}_{f_{\boldsymbol{\theta}}} = \hat{\mathbb{E}}[L_{f_{\boldsymbol{\theta}}}] = \frac{\sum_{m=1}^{M} L(f(x^{(m)}, y^{(m)}))}{M} \tag{3.2}$$

where M is the number of samples of a dataset D.

Definition 3.5. ***Minimising Empirical Risk***
Finding the optimal parameters of the model means minimising the cost function $C(\boldsymbol{\theta})$,

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} (\hat{\mathcal{R}}_{f_{\boldsymbol{\theta}}} + g(f_\theta)) \tag{3.3}$$

where $g(f_{\boldsymbol{\theta}})$ is the *regularisation term* (if any) to mitigate over-fitting. So the learning task is formulated as a problem of optimisation. The most popular and general optimisation technique would be the *gradient descent* [29], which updates the parameters $\boldsymbol{\theta}$ of the cost by

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \cdot \nabla C(\boldsymbol{\theta}) \tag{3.4}$$

in which η is the *learning rate* and t is the integer number of the current iteration. Such an iterative update rule uses gradient, which is the vector field indicating the direction where the value of the function increases most quickly in, as the information to optimise the objective function. Apart from the vanilla gradient descent, there are also variants such as *stochastic gradient descent* [6] and *Adam optimisers* [19] to avoid being trapped in local minima and improve convergence speed.

3.2 Quantum Neural Networks as Machine Learning Model

Currently, there is a long way for our technology to achieve mature fault-tolerant quantum computers and only non-error-corrected near-term quantum devices can be deployed in this stage. This long period is called the Noisy-Intermediate Scale Quantum (NISQ) era, where at most thousands of elementary operations on tens of qubits are allowed without an extra expensive cost of error correction. This shows that it is far-fetched to implement large-scale pure quantum algorithms like the Harrow-Hassidim-Lloyd algorithm and Shor's Algorithm to gain substantial speed-ups and it could be possible that we have still waited decades for the promise of quantum computing to be realised. That said, near-quantum computing can still be utilised and one of the widely applicable approaches is near-term quantum machine learning, in the sense that the machine learning approach of near-quantum computing can require just a handful of qubits and quantum gates. This approach is known as hybrid quantum-classical algorithms or also referred to as variational quantum algorithms. Its core concept is to use a quantum algorithm to implement a machine learning model and training of the machine learning model will then be processed by a classical algorithm in the sense that the quantum model is analogous to a black box neural network model.

The most popular way to achieve this is by parameterised quantum circuits, also known as variational circuits. They are named quantum neural networks in the specific case of machine learning. Quantum neural networks contain three components. The first part is to prepare a zero initial state and the second is a quantum circuit $U(\boldsymbol{\theta})$, which is parameterised by a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^k$. The architecture of the quantum circuit is defined by an ansatz, consisting of a sequence of parameterised and fixed gates being applied to the initial state. The third is the measurement of the final quantum state using an observable $\hat{M}$, with expectation values $f(\boldsymbol{\theta}) = \langle 0| U^\dagger(\boldsymbol{\theta}) \hat{M} U(\boldsymbol{\theta}) |0\rangle$. $f(\boldsymbol{\theta})$ is the model output on which a certain cost function depends. Like neural networks, by minimising the cost function, we can optimise the parameters $\boldsymbol{\theta}$. Note that the optimisation of variational circuits is done by a classical query loop, where we evaluate informative properties of the black-box quantum model, and search for better parameters in each iteration. For example, with reference to Figure 2, a common quantum circuit $U(\boldsymbol{\theta})$ is usually made up of a data encoding block $S(x)$ followed by a processing block

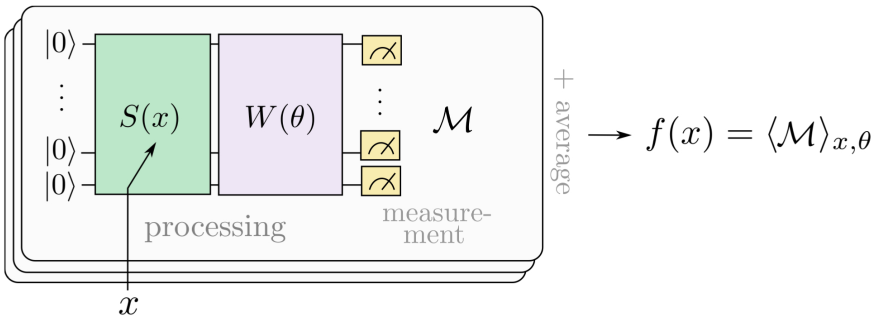


Figure 2: Variational Quantum Circuit's Basic Structure
[32]

$W(\boldsymbol{\theta})$ parameterised by a parameter vector $\boldsymbol{\theta}$, i.e.,

$$U(x;\boldsymbol{\theta}) = W(\boldsymbol{\theta})S(x) \tag{3.5}$$

Notice that such a structure of the quantum variational quantum circuit is presented as a basic example and many other structures are allowed (we will discuss this in Section 3.2.2). We define $|0\rangle := |0...0\rangle$ as the initial quantum circuit state. In the quantum circuit, $U(x;\boldsymbol{\theta})$ acts on $|0\rangle$ so that the final quantum circuit state $|\psi(x;\boldsymbol{\theta})\rangle := U(x;\boldsymbol{\theta})\,|0\rangle$. The output of the model is the expectation of a measurement $\mathcal{M}$ acting $|\psi(x;\boldsymbol{\theta})\rangle$,

$$f(x;\boldsymbol{\theta}) = \langle\psi(x;\boldsymbol{\theta})|\,\mathcal{M}\,|\psi(x;\boldsymbol{\theta})\rangle = \langle 0|\,U^{\dagger}(x;\boldsymbol{\theta})\mathcal{M}U(x;\boldsymbol{\theta})\,|0\rangle := \langle\mathcal{M}\rangle_{x.\boldsymbol{\theta}}. \tag{3.6}$$

Having the output of the model, we can formulate the cost function, and then find the optimal parameters where they minimise the cost function by a classical optimisation method (In Section 4, the techniques to train the quantum neural networks will be discussed).

3.2.1 Example: Variational Classifier

In this example, we will demonstrate how a parameterised quantum model can be used for classification tasks. The aim of this demonstration is to provide readers with intuitions of how a quantum neural network works with visual examples without being bothered by the mathematics behind it. Therefore, there are some caveats when reading this section. Firstly, quantum circuit designs can be arbitrary and experimental so there is a lot of freedom to structure it, hence the choice of the gates used in a parameterised quantum circuit can be exploratory. As a result, one should not see the choice of gate sequence as a law or a rule. Secondly, one may be interested in knowing the mathematically equivalent form of the final quantum state after being processed by the parameterised quantum circuit. However, it should be stressed that the mathematical form of the model (let's say the matrix expression of the final quantum state or the tensor product form of some gate sequences) is very hard to follow and we actually do not need this to train the model or to make predictions. Only when we want to investigate special properties of the quantum circuit, we will make use of the mathematical form of the quantum model (See Sections 3.2.2 and 4.1) but this approach will not be pursued in this example. Instead, we will provide some generalisations to help readers intuitively understand how a quantum neural network works with ease.

The quantum variational classifier that we are going to illustrate is inspired by [31]. Let's start with how a quantum variational classifier does a binary classification. Assume we want to use a quantum variational classifier parameterised by a parameter vector $\boldsymbol{\theta} := \begin{pmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_k\end{pmatrix}$ to classify ith data point $\boldsymbol{x}^{(i)} := \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_{N-1}^{(i)} \end{pmatrix} \in \mathbb{R}^N$ in the training dataset D, where

$$D = \left\{ \left(x^{(1)}, y^{(1)}\right), ..., \left(x^{(M)}, y^{(M)}\right) | \boldsymbol{x}^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y} = \{0, 1\} \; \forall i = \{1, ..., M\} \right\} \tag{3.7}$$

with $\mathcal{X}, \mathcal{Y}$ being the input and output sets respectively. First and foremost, the quantum variational classifier is an N-qubit quantum circuit that matches the dimension of $\boldsymbol{x}^{(i)}$. The big picture of this model is that it transforms the initial quantum circuit state $|\psi_{\text{initial}}\rangle$ from

$$|\psi_{\text{initial}}\rangle = \underbrace{|0\rangle \otimes |0\rangle \otimes ... \otimes |0\rangle}_{\text{N-qubit}} = |00...0\rangle \tag{3.8}$$

to the final quantum state $|\psi^{(i)}_{\text{final}}\rangle$ depending on each point $\boldsymbol{x}^{(i)}$,

$$|\psi^{(i)}_{\text{final}}\rangle = c_{0,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} |00...0\rangle + c_{1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} |00...1\rangle + ... + c_{2^N-1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} |11...1\rangle \,, \tag{3.9}$$

where $|00...0\rangle, |00...1\rangle, ..., |11...1\rangle$ are the computational basis of the quantum system with respective amplitudes $c_{0,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}, c_{1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}, ..., c_{2^N-1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}$ determined by a data point $\boldsymbol{x}^{(i)}$ as well as the parameter vector $\boldsymbol{\theta}$ in some gate sequences of the model subject to $\sum_{j=0}^{2^N-1} |c_{j,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}|^2 = 1, c_{j,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} \in \mathbb{C} \; \forall j$ (by Equation (2.17)). In the case of our model, the first qubit is used to decide the prediction label, in which the probability that the first qubit q_0 is $|1\rangle$, $p(q_0 = |1\rangle)$, is the probability to map the datapoint to label 1, where

$$p(q_0 = |1\rangle \, ; \boldsymbol{x}^{(i)}, \boldsymbol{\theta}) = |c_{2^{N-1},\boldsymbol{x}^{(i)},\boldsymbol{\theta}}|^2 + |c_{2^{N-1}+1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}|^2 + ... + |c_{2^N-1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}|^2 \tag{3.10}$$

where $c_{2^{N-1},\boldsymbol{x}^{(i)},\boldsymbol{\theta}}, c_{2^{N-1}+1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}, ..., c_{2^N-1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}$ correspond to amplitudes of quantum states $|10...0\rangle$,$|10...1\rangle$,...,$|11...1\rangle$. Here is how things come tricky: We will not analytically compute the probability using Equation 3.10 (If we do so, we lose the meaning to use the quantum model as everything will be calculated classically without using the quantum model as the black box model to speed up inference). Instead, we will use quantum measurement as discussed in Section 2.2.3 to find $p(q_0 = |1\rangle \, ; \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$. Consider a one-time Z-basis measurement of the first qubit q_0 and suppose we read off -1. This means we observe $q_0 = |1\rangle$, the quantum system will be collapsed into an $(N-1)$-qubit system (as the first qubit state becomes fixed), where

$$|\psi^{(i)}_{\text{after first qubit measurement}}\rangle = \hat{c}_{2^{N-1},\boldsymbol{x}^{(i)},\boldsymbol{\theta}} |10...0\rangle + \hat{c}_{2^{N-1}+1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} |10...1\rangle + ... + \hat{c}_{2^N-1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} |11...1\rangle \tag{3.11}$$

is the state of the system with amplitudes normalised by $\hat{c}_{j,\boldsymbol{x}^{(i)},\boldsymbol{\theta}} = \frac{c_{j,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}}{\sqrt{p(q_0 = |1\rangle \, ; \boldsymbol{x}^{(i)}, \boldsymbol{\theta})}}$, $j = 2^{N-1}, 2^{N-1}+1, ..., 2^N - 1$. (Note if we want, we can continue to measure the second qubit and so on to obtain the collapsed quantum state after measuring all qubits, with a dimension of N, but we will stick with measuring the first qubit in our model). But essentially, after measurement, we are not interested in the quantum state. To obtain the probability $p(q_0 = |1\rangle \, ; \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$, we initialise the quantum circuit and execute the first qubit Pauli Z measurement, in a total of S times. Each time, we read off either +1 or -1 (corresponding to the collapsed state of the first qubit) and we average the results to obtain an expectation value with a range of $[-1, 1]$. This expectation value of the measurement is represented as $f(x; \boldsymbol{\theta})$, i.e.

$$f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = \langle 0| \, U^\dagger(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \mathcal{M} U(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \, |0\rangle \,, \tag{3.12}$$

where $|0\rangle := |00...0\rangle$ is the quantum device's initial state, $\mathcal{M} = \sigma_z \otimes \mathbb{I} \otimes ... \otimes \mathbb{I}$ is the observable which

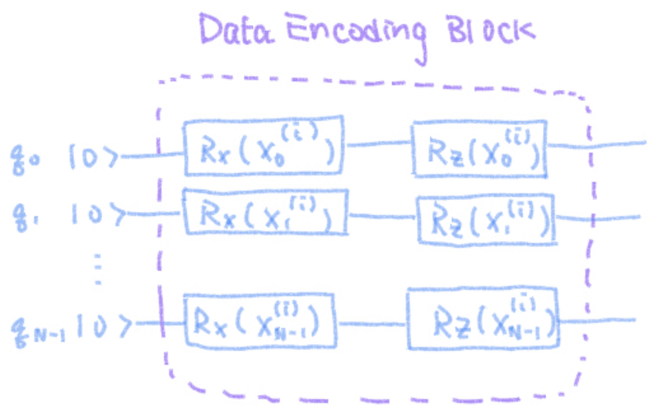


Figure 3: Data Encoding Block

corresponds to the measurement of the first qubit and $U(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})$ is the full model circuit which is parameterised by a parameter vector $\boldsymbol{\theta}$ and takes x as input. By the fact that $f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \in [-1, 1]$, we can transform $f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ into the probability function by

$$p(q_0 = |1\rangle\,; \boldsymbol{x}^{(i)}, \boldsymbol{\theta}) := \frac{-f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + 1}{2} \text{ s.t. } p(q_0 = |1\rangle\,; \boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \in [0, 1]. \tag{3.13}$$

Finally, to classify binary class, the classifier of the model $\mathcal{P}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ is

$$\mathcal{P}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) = \begin{cases} 1 & \text{if } p(q_0 = |1\rangle\,; \boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \geqslant 0.5 \\ 0 & \text{else.} \end{cases} \tag{3.14}$$

This is how a quantum variational classifier that predicts the labels from a data point. To formulate the expected loss from the training dataset D with M training data points, we can write the cost function of the variational classifier using the binary cross-entropy loss as

$$C(\boldsymbol{\theta}; D) = -\frac{1}{M}\sum_{i=1}^{M} y^{(i)} \log p(q_0 = |1\rangle\,; \boldsymbol{x}^{(i)}, \boldsymbol{\theta}) + (1 - y^{(i)}) \log(1 - p(q_0 = |1\rangle\,; \boldsymbol{x}^{(i)}, \boldsymbol{\theta})) \tag{3.15}$$

Consequently, the optimal parameters are found by

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}; D). \tag{3.16}$$

The details of the training algorithms for finding the optimal parameters will be discussed in Section 4.

So far, we have not yet discussed the circuit design for the variational classifier. That said, having reviewed all the essential concepts, the circuit design strategies will be outlined. The whole circuit model roughly comprises the data encoding block $S(x)$, parameterised unitary operation block $W(\boldsymbol{\theta})$, and measurement. Suppose we have normalised the data such that the input set $\mathcal{X} \in [0, \pi]$ (which is an arbitrary range as long as it is in $[0, 2\pi]$) and recall that the data dimension matches the qubit number in the circuit. For the data encoding block, given the ith data point $\boldsymbol{x}^{(i)}$ in dataset D, we want to first encode $x_j^{(i)}$ on the jth qubit. Here, we will use an arbitrary encoding strategy which extends the use of rotation encoding (Please review Time Evolution Encoding in Section 2.2.4): On each jth qubit q_j, we implement a Puali X rotation $R_x(x_j^{(i)})$ followed by Puali Z rotation $R_z(x_j^{(i)})$. See Figure 3.

Next, we will implement the parameterised unitary operation block after the data encoding block. This parameterised unitary operation block consists of several parameterised entangling layers. As we already knew that when for an entangled quantum system, one qubit is not independent of the other, the motivation of stacking entangling layers aim to explore all the computational basis of the quantum system by creating entanglement to effectively prepare the amplitudes $c_{0,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}, c_{1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}, \ldots, c_{2^N-1,\boldsymbol{x}^{(i)},\boldsymbol{\theta}}$

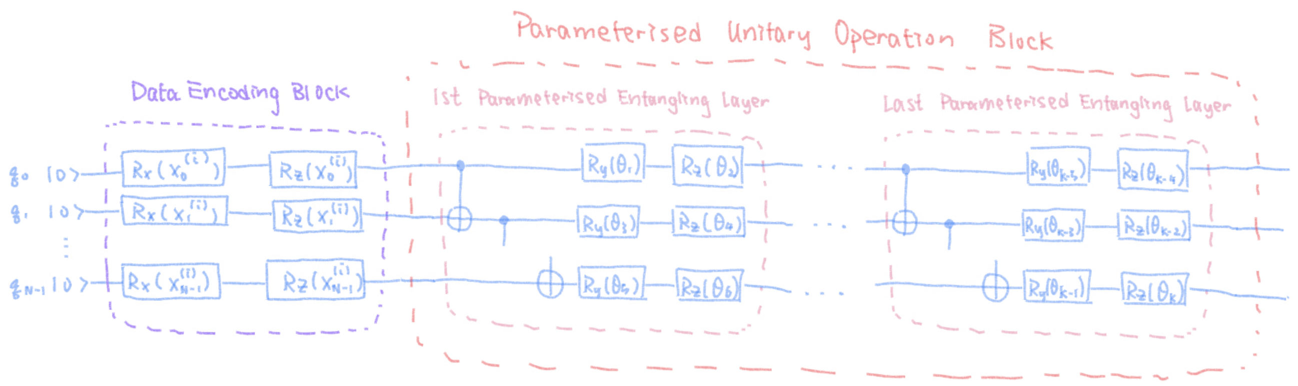


Figure 4: Implementation of the Parameterised Unitary Operation Block after the Data Encoding Block

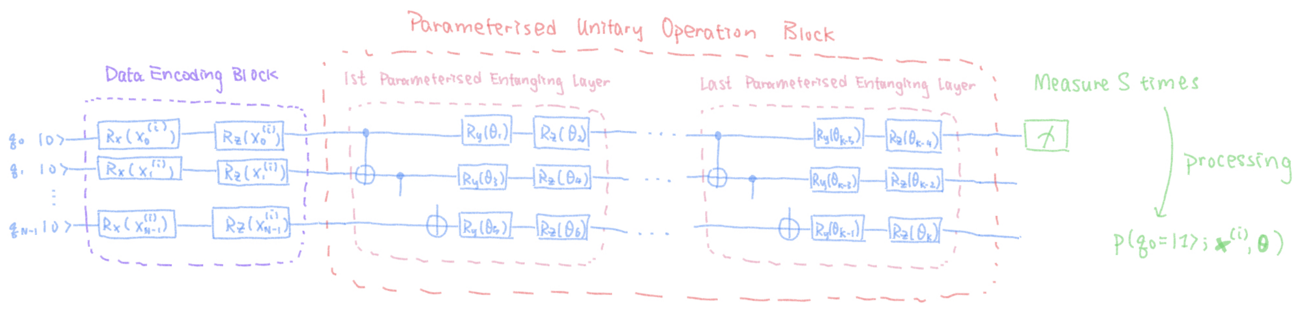

Parameterised Unitary Operation Block
Data Encoding Block
1st Parameterised Entangling Layer
Last Parameterised Entangling Layer
Measure S times
processing
$p(q_0=|1\rangle; x^{(i)}, \theta)$

Figure 5: Full Quantum Circuit of Variational Classifier

in Equation (3.9) (As mentioned, not knowing what these amplitudes analytically are does not matter a lot to understand the intuition of quantum neural networks). In each entangling layer, we will arbitrarily implement $(N-1)$ CNOT gates with jth qubit as control qubit, $(j+1)$th as target qubit for $j = 0, ..., N-2$, followed by implementing a parameterised Pauli Y rotation and parameterised Pauli Z rotation on each qubit, in which each rotation gate corresponds to a parameter in $\boldsymbol{\theta}$, so there are k parameterised rotation gates in total. Figure 4 shows how this block is implemented. After the parameterised unitary operation block, we will have the final quantum state as in Equation (3.9). Eventually, we will extract the information from the final quantum state by first qubit measurement with S measurement shots and processing it to evaluate $p(q_0 = |1\rangle; \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$ depicted in Equation (3.13). Figure 5 is the whole circuit implementation.

To give a simple code example, we implement the variational classifier by using a quantum machine learning library called Pennylane which provides the necessary tools to create parameterised quantum circuits [5]. We tested a 10-entangling-layer variational classifier with the aforementioned routines and circuit structure on two binary class datasets, namely the circle and the moon data set, shown in Figures 6 and 7. These data sets have two continuous value inputs as features and are challenging non-linear datasets to test the capability of the variational classifier.

Variational Classifier	Training Accuracy	Testing Accuracy
Moon Data Set	0.87	0.80
Circle Data Set	0.71	0.65

Table 4: Summary of Performance of the 10-layer Variational Classifier

Using Pennylane's 'qml' module, we can write codes with high readability to set up the parameterised quantum circuit and train it. Figure 8 is the snapshot of the part corresponding to the circuit implementation. The links to the full code are provided in the Appendix. Table 4 summarises the performance of Variational Classifier. After training, the variational classifier achieves a decent performance on the moon data set, getting an accuracy of 0.87 on the training data set and that of 0.80 on the test data set but does not perform well to capture the non-linear pattern on the circle dataset,

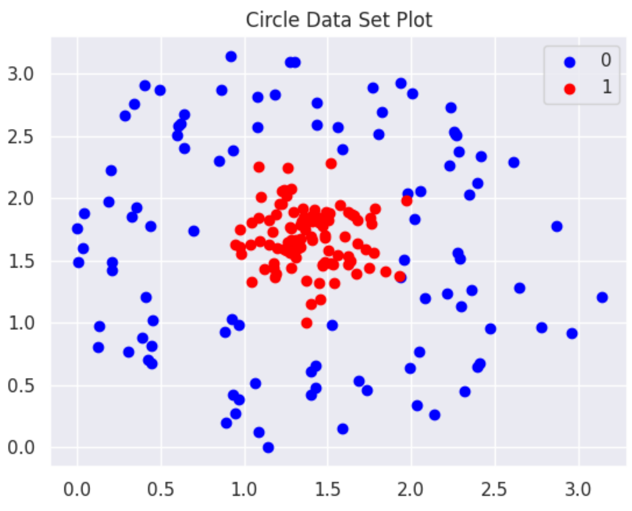


Figure 6: Circle Data Set

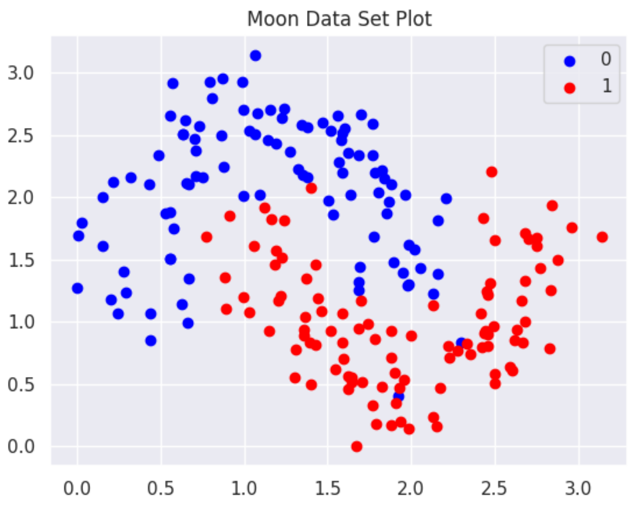


Figure 7: Moon Data Set

only reaching an accuracy of 0.71 on the training data set and that of 0.65 on the test data set for the moon data set. However, there is definitely room for improvement. As such, we will explain how to enhance the accuracy of the variational classifier in Section 3.2.2 and provide the rationale behind it.

3.2.2 What does a Quantum Neural Network Learn?

The variational classifier as an example of quantum neural networks in Section 3.2.1 demonstrates that we can use quantum devices for machine learning tasks by using a parameterised quantum circuit to make predictions in supervised learning. There have been different approaches and strategies to design the architecture of quantum neural networks. However, it is crucial to know how quantum neural networks learn from data and the capability of them to express any possible functions given the input. If the explainability of such a model is not investigated or if the quantum neural network's performance is inferior to other commonly used machine learning models, it may not be meaningful at all. The authors of [33] presented that the expressiveness of the quantum neural network highly depends on the data encoding strategies, namely repeated encoding, and showed that a quantum neural network can be a universal function approximator. We will review their work in this part and provide a computational experiment that extends the idea in Section 3.2.1.

To start with, we constrain the quantum neural network model by considering a one-qubit system to learn a regression task with only one feature. We also pay attention that the circuit architecture to be described is quite different compared to those introduced already and this is very important to what we are going to discuss. Let a quantum model function be $f(x;\boldsymbol{\theta})$, where $f(x;\boldsymbol{\theta})$ is parameterised by weights $\boldsymbol{\theta}$ and maps one-dimensional data inputs $x \in \mathbb{R}$ to predictions. The quantum neural network's model circuit $U(x,\boldsymbol{\theta})$, is constructed by a stack of L *circuit segments*, each of which consists of a circuit block $S(x)$ to encode data and a parameterised entangling layer $W(\vec{\theta_i}), i = 1, ..., L$ s.t. the vertical stack of vectors $\vec{\theta_i}$ is the parameter vector, i.e., $\begin{pmatrix} \vec{\theta_1} \\ \vec{\theta_2} \\ \vdots \\ \vec{\theta_L} \end{pmatrix} := \boldsymbol{\theta}$ (or a whole unitary operation block composed of stacked entangling layers. We will abuse the notation here by assuming $W(\vec{\theta_i})$ is an entangling layer instead of a big chunk of unitary operation block. We will see whichever one we choose, it does not matter at all and we will see why in just a bit. It is fine to just think of it as some parameterised gate sequences). Quantum gates of the form $\mathcal{G}(x)$, which have the time evolution unitary form $\exp(-ixH)$ with H as the Hamiltonian, are used to encode the data input x. For instance, Rotation Encoding satisfies this requirement. (Note this statement is also true that in a multi-qubit system, the tensor product of single rotation gates will result in the form $\exp(-ixH)$ as in Equation (3.35)).$f(x;\boldsymbol{\theta})$ is defined as an observable $\mathcal{M}$'s expectation value,

$$f(x;\boldsymbol{\theta}) = \langle 0| U^\dagger(x,\boldsymbol{\theta})\mathcal{M}U(x,\boldsymbol{\theta}) |0\rangle , \tag{3.17}$$

where $|0\rangle$ is the initial quantum state in the quantum device and $U(x,\boldsymbol{\theta})$ is the model circuit which is parameterised by a parameter vector $\boldsymbol{\theta}$ and takes x as input. The model prediction is calculated by running and measuring the quantum circuit several times and then averaging the results. To solely investigate to effect of data encoding, at this moment $\boldsymbol{\theta}$ is assumed to be fixed, i.e., $W(\vec{\theta_i}) := W$ and $f(x;\boldsymbol{\theta}) = f(x)$. Hence, the structure of the quantum circuit is

$$U(x) = W^{(L+1)}S(x)W^{(L)}...W^{(2)}S(x)W^{(1)}. \tag{3.18}$$

where $S(x) = \mathcal{G}(x)$ with $H = \frac{1}{2}\sigma,\ \sigma \in \{\sigma_x, \sigma_y, \sigma_z\}$. As the data encoding block is repeated in equation (3.18), this data encoding strategy is known as the data re-uploading strategy.

The goal is to express $f(x)$ as a partial Fourier series

$$f(x) = \textstyle\sum_{n\in\Omega} c_n e^{inx}, \tag{3.19}$$

where $\Omega = \{-K, .., K\}$ is a set a integer frequencies. By spectral decomposition, we can decompose

```python
# data encoding block
def data_encoding(x):
    qml.RX(x[0], wires=0)
    qml.RZ(x[0], wires=0)
    qml.RX(x[1], wires=1)
    qml.RZ(x[1], wires=1)

# entangling layer
def entangling_layer(W):
    qml.CNOT(wires=[0, 1])
    qml.RY(W[0], wires=0)
    qml.RZ(W[1], wires=0)
    qml.RY(W[2], wires=1)
    qml.RZ(W[3], wires=1)

# set up quantum device with 2 qubits
dev = qml.device('default.qubit', wires=2)

# set up the circuit
@qml.qnode(dev, interface='torch')
def circuit(weights, x):
    """
    Input: weights (2d array of parameters), x (a datapoint with 2 features)
    Output: expectation of Pauli Z measurement with a range of [-1,1]
    """
    # data encoding
    data_encoding(x)

    # unitary operation block with W.shape[0] layers
    for W in weights:
      entangling_layer(W)
    return qml.expval(qml.PauliZ(0))

# binary cross entropy loss
def cost(weights,y_train,X_train):
    loss = 0
    for y, x in zip(y_train, X_train):
      prob = (-circuit(weights,x)+1)/2
      loss -= y*torch.log(prob) + (1-y)*torch.log(1-prob)
    return loss/len(X_train)
```

Figure 8: Main Part of Variational Classifier Circuit Implementation (Please refer to the link in the Appendix for the full code implementation)

any Hamiltonian H as

$$H = V^{\dagger}\Sigma V \tag{3.20}$$

with Σ being the diagonal matrix of H's eigenvalues $\lambda_1, ..., \lambda_d$ and V being the square matrix with ith column as the eigenvector $\boldsymbol{v_i}$ of H. Therefore, data encoding block $S(x)$ can be rewritten as

$$S(x) = e^{ixH} = V^{\dagger}e^{-ix\Sigma}V \tag{3.21}$$

and $U(x)$ becomes

$$U(x) = W^{(L+1)}V^{\dagger}e^{-ix\Sigma}VW^{(L)}...W^{(2)}V^{\dagger}e^{-ix\Sigma}VW^{(1)}. \tag{3.22}$$

As V and $V^{\dagger}$ can be absorbed into W, letting $\mathcal{W} = VWV^{\dagger}$, we have

$$U(x) = \mathcal{W}^{(L+1)}e^{-ix\Sigma}\mathcal{W}^{(L)}...\mathcal{W}^{(2)}e^{-ix\Sigma}\mathcal{W}^{(1)}. \tag{3.23}$$

In this way, we can assume H to be diagonal without loss of generality. If not we can write $U(x)$ as in Equation (3.23) and arbitrarily choose the diagonal matrix Σ to be the Hamiltonian. This enables us to write each ith component of the quantum state $U(x)\left|0\right\rangle$ separately,

$$\left[U(x)\left|0\right\rangle\right]_i = \textstyle\sum_{j_1,...,j_L=1}^{d} e^{-i(\lambda_{j1}+...+\lambda_{jL})x}W_{ij_L}^{(L+1)}...W_{j_2j_1}^{(2)}W_{j_11}^{(1)}. \tag{3.24}$$

Let $\boldsymbol{j} = \{j_1, ..., j_L\} \in [d]^L$ and $\Lambda_{\boldsymbol{j}} = \lambda_{j1} + ... + \lambda_{jL}$, then

$$\left[U(x)\left|0\right\rangle\right]_i = \textstyle\sum_{\boldsymbol{j}\in[d]^L} e^{-i\Lambda_{\boldsymbol{j}}x}W_{ij_L}^{(L+1)}...W_{j_2j_1}^{(2)}W_{j_11}^{(1)}. \tag{3.25}$$

Therefore, with the complex conjugate of Equation (3.25) and measurement being taken into account, the full quantum model is

$$f(x) = \textstyle\sum_{k,\boldsymbol{j}\in[d]^L} e^{i(\Lambda_k-\Lambda_{\boldsymbol{j}})x}a_{k,j}, \tag{3.26}$$

$$a_{k,j} = \sum_{i,i'} (W^*)_{1k_1}^{(1)}(W^*)_{j_1j_2}^{(2)}...(W^*)_{j_Li}^{(L+1)}M_{i,i'}W_{i'j_L}^{(L+1)}...W_{j_2j_1}^{(2)}W_{j_11}^{(1)}. \tag{3.27}$$

If we define the frequency spectrum Ω as $\Omega := \left\{\Lambda_k - \Lambda_{\boldsymbol{j}}, k, \boldsymbol{j} \in [d]^L\right\}$ and let $c_\omega = \sum\limits_{\substack{k, \boldsymbol{j} \in [d]^L \\ \Lambda_k - \Lambda_{\boldsymbol{j}} = \omega}} a_{k,j}$,

we recover the form of Equation (3.19)

$$f(x) = \textstyle\sum_{\omega\in\Omega} c_n e^{i\omega x}. \tag{3.28}$$

Note that the $0 \in \Omega$ and $-\omega \in \Omega$ for any $\omega \in \Omega$. Also, $f(x)$ is a real-valued function since c_ω=c_ω^*. To measure the count of frequencies with non-zero values in the model, the size of the spectrum can be denoted by $K = \dfrac{(|\Omega| - 1)}{2}$. On the other hand, the degree of the spectrum $D = \max(\Omega)$ is the largest available spectrum frequency. All in all, from the above equations, it can be seen that a quantum neural network's frequency spectrum depends on the data encoding gates' eigenvalues only. On the other hand, the Fourier coefficients are determined by the whole quantum circuit. $f(x)$ is a partial Fourier series since Ω is a set of integer frequencies determined by integer-valued eigenvalues of the data encoding gates.

Consequently, we know that a quantum neural network's expressive power of is depends on the size and degree of its frequency spectrum, and the model's Fourier coefficients. Furthermore, from the Fourier series perspective, the data encoding strategy of using single-qubit Pauli rotations can be analysed.

To start with, a quantum neural network $f_1(x)$ with layer $L = 1$ is considered, i.e.,

$$U(x) = W^{(2)}S(x)W^{(1)}, S(x) = e^{ixH}. \tag{3.29}$$

(If $W^{(1)}$ is the identity gate, we obtain the form same as the variational classifier in Section 3.2.1) For this model, Hamiltonian $H = \frac{1}{2}\sigma,\ \sigma \in \{\sigma_x, \sigma_y, \sigma_z\}$ only contains two distinct eigenvalues λ_1 and λ_2 with energy spectrum $(-\gamma, \gamma) = \left(\frac{1}{2}, \frac{1}{2}\right)$. Our goal is to show the model function is of the form $f_1(x) = A\sin(2\gamma x + B) + C,\ A, B, C \in \mathbb{R}$. First of all, without any loss of generality, eigenvalues are $(\lambda_1, \lambda_2) = (-1, 1)$ can be assumed since rescaling x only results in a change in the global phase and thus we can rescale x to $\tilde{x} = \gamma x$ such that the eigenvalues are the same as the Pauli matrices. This follows that the quantum neural network spectrum is $\Omega = \{-2, 0, 2\}$ by $\Omega = \left\{\Lambda_k - \Lambda_{\boldsymbol{j}}, k, \boldsymbol{j} \in [d]^L\right\}$ and the Fourier coefficients, by Equation (3.27), are

$$c_0 = \sum_{i,i'} (W^*)^{(1)}_{12}\, (W^*)^{(2)}_{2i}\, M_{ii'} W^{(2)}_{i'1} W^{(1)}_{11} \tag{3.30}$$

$$c_2 = \sum_{i,i'} (W^*)^{(1)}_{11}\, (W^*)^{(2)}_{1i}\, M_{ii'} W^{(2)}_{i'2} W^{(1)}_{21} \tag{3.31}$$

$$c_{-2} = c_2^* \tag{3.32}$$

Hence, the model $f_1(x)$ can expressed as

$$f(x) = c_{-2}e^{i2\tilde{x}} + c_0 + c_{-2}e^{-i2\tilde{x}} = c_0 + 2|c_2| \cos\left(2\tilde{x} - \arg(c_2)\right) \tag{3.33}$$

With some algebra manipulations, we recovers the form $f_1(x) = A\sin(2\gamma x + B) + C$, where

$$A = 2|c_2|, B = -\tfrac{\pi}{2} - \arg(c_2), C = c_0. \tag{3.34}$$

This highlights a crucial point that however deep a quantum neural network architecture is constructed, if we encode the data only once, we are constrained by being able to fit a single-frequency Fourier series, given such a data encoding strategy.

That said, we can actually expand the quantum neural network's frequency spectrum by considering the situation when single-qubit Puali rotations gates are repeated in parallel with $L = 1$, in which the encoding gate is repeated r times in parallel (This means the model has a system of r qubits). The data-encoding circuit block $S(x)$ is

$$S(x) = e^{-i\frac{x}{2}\sigma_r} \otimes ... \otimes e^{-i\frac{x}{2}\sigma_1}, \text{ for } \sigma_1, ..., \sigma_r \in \{\sigma_x, \sigma_y, \sigma_z\}. \tag{3.35}$$

Since all rotation gates commute (That is, $AB - BA = 0$ for any operators A and B), we can diagonalise H,

$$S(x) = V_r e^{-i\frac{x}{2}\sigma_z} V_r^\dagger \otimes ... \otimes V_1 e^{-i\frac{x}{2}\sigma_z} V_1^\dagger = V \exp\left(-i\frac{x}{2}\sum_{q=1}^{r} \sigma_z^{(q)}\right) V^\dagger = V e^{-ix\Sigma} V^\dagger, \tag{3.36}$$

where $\sigma_z^{(q)}$ is the non-trivial operator acting on the qth qubit and $\Sigma = \text{diag}(\lambda_1, ..., \lambda_{2^r})$, with $\lambda_p = \left(\frac{p}{2} - \frac{r-p}{2}\right) = p - \frac{r}{2},\ p \in \{0, ..., r\}$. The frequency spectrum of parallel repeated encoding circuit Ω_{par} is thus

$$\begin{aligned}
\Omega_{par} &:= \{\lambda_{k1} - \lambda_{j1} \mid k_1, j_1 \in \{1, ..., 2^r\}\} \\
&= \left\{\left(p - \frac{r}{2}\right) - \left(p' - \frac{r}{2}\right) \mid p, p' \in \{0, ..., r\}\right\} \\
&= \{p - p' \mid p, p' \in \{0, ..., r\}\} \\
&= \{-r, -(r-1), ..., 0, ..., r-1, r\}
\end{aligned} \tag{3.37}$$

This shows that a quantum neural network with data encoded by r parallel Pauli-rotations is a degree r Fourier series. Surprisingly, the same result can be obtained by layer-wise Pauli rotation encoding with $L = r > 1$, where

$$S(x) = e^{-i(\frac{x}{2})\sigma_j}, \sigma_j \in \{\sigma_x, \sigma_y, \sigma_z\}. \tag{3.38}$$

Then the circuit $U(x)$ is

$$U(x) = W^{(L+1)} e^{-i(\frac{x}{2})\sigma_j} W^{(L)} ... W^{(2)} e^{-i(\frac{x}{2})\sigma_j} W^{(1)}. \tag{3.39}$$

Diagonalising σ_j by $\Sigma = \frac{1}{2}\sigma_z$ and by $\Omega = \left\{\Lambda_k - \Lambda_{\boldsymbol{j}}, k, \boldsymbol{j} \in [d]^L\right\}$, accordingly, the frequency spectrum of layer-wise sequential data encoding circuit Ω_{seq} is

$$\Omega_{seq} := \{(\lambda_{k_1} + ... + \lambda_{k_r}) - (\lambda_{j_1} + ... + \lambda_{k_{jr}}) \mid k_1, ..., k_r, j_1, ..., j_r \in \{1, 2\}\} \tag{3.40}$$

and in fact $\Omega_{seq} = \Omega_{par}$. To conclude, a quantum neural network's frequency spectrum is linearly extended by repeated Pauli encodings of data.

Besides, Equation (3.28) can be extended to describe an N-dimensional input case by replacing x by an input vector $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$ and ω by a frequency vector $\boldsymbol{\omega}$ where the difference of the Hamiltonian eigenvalues, used to encode x_i, determines the frequency ω_i. We can intuitively think of this equation as a neural network $f_{\boldsymbol{w},\boldsymbol{W}}(\boldsymbol{x})$ which is composed of a J-unit hidden layer with an activation function φ, which is a complex exponential function, in each hidden unit, that is,

$$f_{\boldsymbol{w},\boldsymbol{W}}(\boldsymbol{x}) := \textstyle\sum_{j=1}^{J} w_{1j} \varphi(\boldsymbol{w}_{2j}^T \boldsymbol{x}). \tag{3.41}$$

where $\boldsymbol{w}_{2j}$ is the N-dimensional weights linking the input to jth hidden units and $\boldsymbol{w_1} = \begin{pmatrix} w_{11} \\ w_{12} \\ \vdots \\ w_{1J} \end{pmatrix}$ is the weight vector which links the hidden units to the output unit. This shows that data reuploading is critical as it can be interpreted as the quantum version of non-linear activations in a neural network and data reuploading determines hidden unit number J that affects the model's flexibility to learn the complex structure of training data.

In view of the above results, if we have sufficient repetitions of the data-encoding gates, a quantum neural network with arbitrary frequency spectra can be realised. All in all, it is important to highlight that just like a classical neural network, a sufficiently wide and deep quantum neural networks are capable to be universal function approximators, though the derivation will not be expounded in this survey. Such a result is interesting and meaningful as we can quantify how the data-encoding strategies affect the quantum neural network's expressivity and it is obvious that to unleash the potential of quantum neural networks we can consider the data-reuploading strategy for the example in Section 3.2.1. Besides, because the quantum neural network learns a truncated Fourier series, it is very relevant to apply it to machine learning tasks for modelling time series or signal processing, serving as alternatives to other machine learning models.

Data Re-uploading Variational Classifier	Training Accuracy	Testing Accuracy
Moon Data Set	0.96	0.82
Circle Data Set	0.98	0.85

Table 5: Summary of Performance of the 10-layer Data Re-uploading Variational Classifier

To demonstrate the power of data reuploading as an analogy of adding hidden units with non-linear activations in neural networks, we will re-implement the variational classifier in Section 3.2.1 but with data re-uploading strategy. We only need to change one line of code by putting the data encoding block inside the for-loop that implements the entangling layers so that it is one after the

```python
# data encoding block
def data_encoding(x):
    qml.RX(x[0], wires=0)
    qml.RZ(x[0], wires=0)
    qml.RX(x[1], wires=1)
    qml.RZ(x[1], wires=1)

# entangling layer
def entangling_layer(W):
    qml.CNOT(wires=[0, 1])
    qml.RY(W[0], wires=0)
    qml.RZ(W[1], wires=0)
    qml.RY(W[2], wires=1)
    qml.RZ(W[3], wires=1)

# set up quantum device with 2 qubits
dev = qml.device('default.qubit', wires=2)

# set up the circuit
@qml.qnode(dev, interface='torch')
def circuit(weights, x):
    """
    Input: weights (2d array of parameters), x (a datapoint with 2 features)
    Output: expectation of Pauli Z measurement with a range of [-1,1]
    """
    for W in weights:
      # data reupload
      data_encoding(x)
      # entangling layer
      entangling_layer(W)
    return qml.expval(qml.PauliZ(0))
```

Figure 9: Main Part of Data Reuploading Variational Classifier Circuit Implementation (Please refer to the link in the Appendix for the full code implementation)

other. In other words, we have 10 data encoding blocks and 10 entangling layers in turn, as shown in Figure 9. From Table 5, after training, the data re-uploading variational classifier achieves an improved performance on the moon data set, getting an accuracy of 0.96 on the training data set and that of 0.82 on the test data set. More importantly, it significantly enhances the performance of the circle data set, getting an accuracy of 0.98 on the training data set and that of 0.85 on the test data set for the circle data set. This shows that the data re-uploading variational classifier is really doing a much better job of capturing the highly non-linear pattern of this challenging dataset than the basic variational classifier.

4 Training Quantum Neural Networks

Training the quantum neural network means involves finding a parameter vector $\boldsymbol{\theta}$ which minimises the cost function of the model. There are several ways to optimise a variational quantum circuit, such as using gradient-free methods like the Nelder-Mead method [26]. Nevertheless, for over-parameterised quantum neural networks, it is often preferred to use gradient-based methods given a large parameter space [24]. The cost function depends on the quantum model's predictions, where the model's behaviour relies on the parameters $\boldsymbol{\theta}$. In order to find the optimal parameters, we have to compute the partial derivatives of the cost function with respect to the parameters. For example, a cost function $C(\boldsymbol{\theta})$ depends on a differential model $f(\boldsymbol{\theta})$ while $f(\boldsymbol{\theta})$ depends on parameters $\boldsymbol{\theta}$. The partial derivative $\partial_{\theta_j} C$ is

$$\partial_{\theta_j} C = \frac{\partial C}{\partial f(\boldsymbol{\theta})} \cdot \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}, \tag{4.1}$$

where θ_j is the jth element of $\boldsymbol{\theta}$. If we know the way to calculate each $\dfrac{\partial C}{\partial f(\boldsymbol{\theta})}$ and $\dfrac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}$, we can find the cost function gradient and can formulate the gradient-based optimisation algorithm to minimise the cost function. This is also known as differentiable programming and is very common in popular machine learning models like deep neural networks. However, when dealing with quantum neural networks, computing the gradients directly is challenging. We should observe that only $\dfrac{\partial C}{\partial f(\boldsymbol{\theta})}$ can be computed classically, using derivative rules. In contrast, $\dfrac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}$ depends on the quantum oracle and we simply cannot use a classical way to measure such a partial derivative. The reason is that a parameterised quantum model is a black box which does not store internal computations and as such, we cannot compute the gradients by using techniques like back-propagation, unlike classical deep learning. In this part, we will explain the parameter shift rule, a method to find the gradients of the quantum neural network model. We will also show why this method is preferred over common numerical methods such as finite difference in the setting of quantum machine learning. Last but not least, knowing how to evaluate the gradients, we will give a comprehensive review of gradient-based optimisations in quantum neural networks and how we can make more out of a classical optimiser.

4.1 The Parameter Shift Rule

In order to apply gradient-descent-based optimisation to train quantum neural networks, it is inevitable to calculate its gradients and such derivatives can be computed by the parameter shift rule [30].

A quantum circuit is composed of $U(\boldsymbol{\theta})$, a gate sequence with $\boldsymbol{\theta}$ being a vector of k parameters, and $U(\boldsymbol{\theta})$ is followed by measuring an observable $\mathcal{M}$. It is assumed that $\mathcal{M}$ is a Pauli-Z observable σ_z which measures exactly one qubit in the quantum system, resulting in a value ± 1. To find $\langle \mathcal{M} \rangle$, the expected value of $\mathcal{M}$, we should first interpret the quantum neural network as a function $f(\boldsymbol{\theta}) : \mathbb{R}^k \to \mathbb{R}$ and relate it to $\langle \mathcal{M} \rangle$

$$f(\boldsymbol{\theta}) = \langle \mathcal{M} \rangle = \langle 0| U^\dagger(\boldsymbol{\theta}) \mathcal{M} U(\boldsymbol{\theta}) |0\rangle \tag{4.2}$$

This is the analytic definition of $f(\boldsymbol{\theta})$. $f(\boldsymbol{\theta})$, when physically implemented, is estimated by sampling the measurement n times and computing the average. Our focus lies in finding the partial derivative $\partial_{\theta_j} f(\boldsymbol{\theta})$, denoting θ_j is the jth element of $\boldsymbol{\theta}$. Before popularising the parameter shift rule, numerical differentiation, such as finite differences, was commonly used to approximate the gradient because evaluating the analytic gradient is prohibited. Yet this comes with the cost of encountering high errors in near-term devices. In contrast, the parameter shift rule is much more feasible and favourable. The rule will be illustrated in the following.

Assume that the entry θ_j affects a specific single gate $\mathcal{G}(\theta_j)$ only out of the whole sequence of gates. By this assumption, θ_j neither affects the gates before $\mathcal{G}(\theta_j)$ nor the gates after $\mathcal{G}(\theta_j)$. We thus decompose $U(\boldsymbol{\theta})$ by $U(\boldsymbol{\theta}) \equiv U(\theta_j) := V\mathcal{G}(\theta_j)W$, for V and W are the gate sequences before and after $\mathcal{G}(\theta_j)$ respectively. We define $f(\theta_j) := \langle 0| W^\dagger \mathcal{G}^\dagger(\theta_j) V^\dagger \mathcal{M} V \mathcal{G}(\theta_j) W |0\rangle$ to be the shorthand of quantum function $f(\boldsymbol{\theta})$ so that all parameters of $f(\boldsymbol{\theta})$ except θ_j are assumed constant. Hence the partial derivative with respect to θ_j, $\partial_{\theta_j} f$, is

$$\partial_{\theta_j} f = \partial \langle 0| W^\dagger \mathcal{G}^\dagger(\theta_j) V^\dagger \mathcal{M} V \mathcal{G}(\theta_j) W |0\rangle . \tag{4.3}$$

Letting $|\psi\rangle = W|0\rangle$ and $\hat{Q} = V^\dagger \mathcal{M} V$, we have

$$\begin{aligned} \partial_{\theta_j} f &= \partial_{\theta_j} \langle\psi| \mathcal{G}^\dagger(\theta_j) \hat{Q} \mathcal{G}(\theta_j) |\psi\rangle \\ &= \langle\psi| \mathcal{G}^\dagger(\theta_j) \hat{Q} \partial_{\theta_|}[\mathcal{G}(\theta_j)] |\psi\rangle + \text{h.c.}, \end{aligned} \tag{4.4}$$

where + h.c. is the abbreviation of "plus its Hermitian conjugate". By the fact that for any pair of operators B and C,

$$\langle\psi| B^\dagger \hat{Q} C |\psi\rangle + h.c. = \frac{1}{2}\Big(\langle\psi| (B+C)^\dagger \hat{Q} (B+C) |\psi\rangle - \langle\psi| (B-C)^\dagger \hat{Q} (B-C) |\psi\rangle\Big) \tag{4.5}$$

we can see that Equation (4.4) is

$$\begin{aligned} \partial_{\theta_j} f =& \frac{1}{2}(\langle\psi| (\mathcal{G}(\theta_j) + \partial_{\theta_|}[\mathcal{G}(\theta_j)])^\dagger \hat{Q} (\mathcal{G}(\theta_j) + \partial_{\theta_|}[\mathcal{G}(\theta_j)]) |\psi\rangle \\ &- \langle\psi| (\mathcal{G}(\theta_j) - \partial_{\theta_|}[\mathcal{G}(\theta_j)])^\dagger \hat{Q} (\mathcal{G}(\theta_j) - \partial_{\theta_|}[\mathcal{G}(\theta_j)]) |\psi\rangle). \end{aligned} \tag{4.6}$$

Therefore, by applying $\mathcal{G}(\theta_j) \pm \partial_{\theta_|}[\mathcal{G}(\theta_j)]$, $\partial_{\theta_j} f$ can be evaluated.

Considering that $\mathcal{G}(\theta_j) = \exp(-i\theta_j G)$ where G is a Hermitian, $\partial_{\theta_|}[\mathcal{G}(\theta_j)]$ is

$$\partial_{\theta_|}[\mathcal{G}(\theta_j)] = -iG \cdot \exp(-i\theta_j G) = -iG\mathcal{G}(\theta_j). \tag{4.7}$$

Substituting Equation (4.7) into Equation (4.4), this results in

$$\begin{aligned} \partial_{\theta_j} f &= \langle\psi| \mathcal{G}^\dagger(\theta_j) \hat{Q} (-iG\mathcal{G}(\theta_j)) |\psi\rangle + h.c. \\ &= \langle\psi'| \hat{Q}(-iG) |\psi^{'}\rangle + h.c. \end{aligned} \tag{4.8}$$

by letting $|\psi'\rangle := \mathcal{G}(\theta_j)|\psi\rangle$. Observe that when G has two distinct eigenvalues, and since the global phase is unobservable, we can shift the eigenvalues to $\pm r$. Using Equation (4.5), we substitue $B = \mathbb{I}$ and $C = -ir^{-1}G$, so the derivative of f is

$$\begin{aligned} \partial_{\theta_j} f =& \ \langle\psi'| \hat{Q}(-iG) |\psi'\rangle + h.c. \\ &= r\langle\psi'| \mathbb{I}\hat{Q}(-ir^{-1}G) |\psi'\rangle + h.c. \\ &= \frac{r}{2}(\langle\psi'| \left(\mathbb{I} - ir^{-1}G\right)^\dagger \hat{Q} \left(\mathbb{I} - ir^{-1}G\right) |\psi'\rangle \\ &\quad - \langle\psi'| \left(\mathbb{I} + ir^{-1}G\right)^\dagger \hat{Q} \left(\mathbb{I} + ir^{-1}G\right) |\psi'\rangle) \end{aligned} \tag{4.9}$$

Theorem 4.1. *Given $\mathcal{G}(\theta_j) = exp(-i\theta_j G)$, where G has two distinct eigenvalues at most, this following identity*

$$\mathcal{G}\left(\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}\left(\mathbb{I} - ir^{-1}G\right) \ and\ \mathcal{G}\left(-\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}\left(\mathbb{I} + ir^{-1}G\right) \tag{4.10}$$

holds.

Proof. G has the spectrum $\{\pm r\} \Longrightarrow G^2 = r^2\mathbb{I}$. Therefore, the Taylor series of $\mathcal{G}(\theta_j)$ is

$$\begin{aligned}
\mathcal{G}(\theta_j) &= \exp(-i\theta_j G) \\
&= \sum_{k=0}^{\infty} \frac{(-i\theta_j)^k G^k}{k!} \\
&= \sum_{k=0}^{\infty} \frac{(-i\theta_j)^{2k} G^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{(-i\theta_j)^{2k+1} G^{2k+1}}{(2k+1)!} \\
&\left[G^{2k} = r^{2k}\mathbb{I}, G^{2k+1} = r^{2k}\mathbb{I}G\right] \\
&= \sum_{k=0}^{\infty} \frac{(-i\theta_j)^{2k}\left(r^{2k}\mathbb{I}\right)}{(2k)!} + \sum_{k=0}^{\infty} \frac{(-i\theta_j)^{2k+1}\left(\left(r^{2k}\mathbb{I}\right)G\right)}{(2k+1)!} \\
&= \mathbb{I}\sum_{k=0}^{\infty} \frac{(-1)^k (r\theta_j)^{2k}}{(2k)!} - ir^{-1}G\sum_{k=0}^{\infty} \frac{(-1)^k (r\theta_j)^{2k+1}}{(2k+1)!} \\
&= \mathbb{I}\cdot\cos(r\theta_j) - ir^{-1}G\cdot\sin(r\theta_j)
\end{aligned} \tag{4.11}$$

and hence $\mathcal{G}\left(\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}\left(\mathbb{I} - ir^{-1}G\right)$ and $\mathcal{G}\left(-\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}\left(\mathbb{I} + ir^{-1}G\right)$

□

In such case, we can estimate $\partial_{\theta_j} f$ using

$$\begin{aligned}
\partial_{\theta_j} f &= \frac{r}{2}\Big(\langle\psi'|\left(\mathbb{I} - ir^{-1}G\right)^\dagger \hat{Q}\left(\mathbb{I} - ir^{-1}G\right)|\psi'\rangle\Big) \\
&\quad - \langle\psi'|\left(\mathbb{I} + ir^{-1}G\right)^\dagger \hat{Q}\left(\mathbb{I} + ir^{-1}G\right)|\psi'\rangle \\
&= r\left(\langle\psi'|\mathcal{G}\left(\frac{\pi}{4r}\right)^\dagger \hat{Q}\mathcal{G}\left(\frac{\pi}{4r}\right)|\psi'\rangle - \langle\psi'|\mathcal{G}\left(-\frac{\pi}{4r}\right)^\dagger \hat{Q}\mathcal{G}\left(-\frac{\pi}{4r}\right)|\psi'\rangle\right) \\
&\quad [\because |\psi'\rangle = \mathcal{G}(\theta_j)|\psi\rangle] \\
&= r\left(\langle\psi|\mathcal{G}(\theta_j)^\dagger\mathcal{G}\left(\frac{\pi}{4r}\right)^\dagger \hat{Q}\mathcal{G}\left(\frac{\pi}{4r}\right)\mathcal{G}(\theta_j)|\psi\rangle - \langle\psi|\mathcal{G}(\theta_j)^\dagger\mathcal{G}\left(-\frac{\pi}{4r}\right)^\dagger \hat{Q}\mathcal{G}\left(-\frac{\pi}{4r}\right)\mathcal{G}(\theta_j)|\psi\rangle\right) \\
&\quad [\because \mathcal{G}(a)\mathcal{G}(b) = \mathcal{G}(a+b)\ \forall a, b \in \mathbb{R}] \\
&= r\left(\langle\psi|\mathcal{G}\left(\theta_j + \frac{\pi}{4r}\right)^\dagger \hat{Q}\mathcal{G}\left(\theta_j + \frac{\pi}{4r}\right)|\psi\rangle - \langle\psi|\mathcal{G}\left(\theta_j - \frac{\pi}{4r}\right)^\dagger \hat{Q}\mathcal{G}\left(\theta_j - \frac{\pi}{4r}\right)|\psi\rangle\right)
\end{aligned} \tag{4.12}$$

Substituting $s = \pi/(4r)$, we finally reach the parameter shift rule

$$\begin{aligned}
\partial_{\theta_j} f &= \nabla\Big(\langle\psi|G(\theta_j + s)^\dagger\hat{Q}\mathcal{G}(\theta_j + s)|\psi\rangle - \langle\psi|G(\theta_j - s)^\dagger\hat{Q}\mathcal{G}(\theta_j - s)|\psi\rangle\Big) \\
&\quad \left[\because \hat{Q} = V^\dagger\mathcal{M}V \text{ and } |\psi\rangle = W|0\rangle\right] \\
&= r\Big(\langle 0|W^\dagger\mathcal{G}(\theta_j + s)^\dagger V^\dagger\mathcal{M}V\mathcal{G}(\theta_j + s)W|0\rangle - \langle 0|W^\dagger\mathcal{G}(\theta_j - s)^\dagger V^\dagger\mathcal{M}V\mathcal{G}(\theta_j - s)W|0\rangle\Big) \\
&\quad \left[\because f(\theta_j) = \langle 0|W^\dagger\mathcal{G}^\dagger(\theta_j)V^\dagger\mathcal{M}V\mathcal{G}(\theta_j)W|0\rangle\right] \\
&= r(f(\theta_j + s) - f(\theta_j - s))
\end{aligned} \tag{4.13}$$

Practically, s is often opted as $\frac{\pi}{2}$ to so that $\partial_{\theta_j} f = \frac{1}{2}\left(f\left(\theta_j + \frac{\pi}{2}\right) - f\left(\theta_j - \frac{\pi}{2}\right)\right)$. In some literature such as in [36], the parameter shift rule has an alternative form of

$$\partial_{\theta_j} f := \frac{f(\theta_j + s) - f(\theta_j - s)}{2\sin(s)} \tag{4.14}$$

A sketch of the proof of Equation (4.14) will be provided in the Appendix. Furthermore, notice that we can chain this rule to calculate the Hessian if needed (See Section 4.2.1).

Besides, we immediately observe that we can only apply the parameter shift rule to the gates with two distinct eigenvalues. To evaluate general gates with more than two distinct eigenvalues, we should either decompose them into linear combinations of gates with two distinct eigenvalues and then use the parameter shift rule to compute the gradient for every gate and accumulate the gradient using the total derivative, or fall back to using finite differences to estimate the gradients. Luckily, it suffices to use the basic parameter shift rule in most of the gates used in quantum machine learning (namely rotation gates) thus the case of the differentiating general gates will not be emphasised. As such, the statistical analysis in the next part will focus on the basic parameter shift rule.

4.1.1 Statistical Analysis of Gradient Estimators

In this section, we will explain the rationale that we prefer the parameter shift rule to finite differences to compute the gradients of the quantum circuit. Here we analyse their statistical behaviour with reference to [21]. Recall that an observable $\mathcal{M}$'s expectation when a parameterised quantum state $U(\boldsymbol{\theta})$ is measured is $f(\boldsymbol{\theta}) = \langle 0| U^\dagger(\boldsymbol{\theta})\mathcal{M}U(\theta) |0\rangle$. Yet this is a theoretical quantity and in reality, we estimate $f(\boldsymbol{\theta})$ with S measurement shots that results in statistical uncertainty. The estimation of S measurement shots, $\hat{f}(\boldsymbol{\theta})$, is

$$\hat{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \hat{\epsilon}, \tag{4.15}$$

where we let $\hat{\epsilon}$ be the estimation error with a mean of 0 and variance $\sigma^2 = \dfrac{\sigma_0^2}{S}$ with σ_0^2 being the variance for a single shot measurement.

To represent first-order derivatives with regard to the jth parameter, we define a function

$$\partial_{\theta_j} f = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}. \tag{4.16}$$

We want to study how good the estimator $\hat{\partial}_{\theta_j} f$ with respect to the number of shots S is. We can use the means square error $\Delta\partial_{\theta_j} f$ to measure the goodness of the estimator, that is,

$$\Delta\hat{\partial}_{\theta_j} f = \mathbb{E}\left[(\hat{\partial}_{\theta_j} f - \partial_{\theta_j} f)^2\right] \tag{4.17}$$

In addition, the bias and variance of the estimator $\hat{\partial}_{\theta_j} f$ is:

$$\text{Bias}(\hat{\partial}_{\theta_j} f) := \mathbb{E}(\hat{\partial}_{\theta_j} f) - \partial_{\theta_j} f, \tag{4.18}$$

$$\text{Var}(\hat{\partial}_{\theta_j} f) := \mathbb{E}\left((\hat{\partial}_{\theta_j} f)^2\right) - \mathbb{E}(\hat{\partial}_{\theta_j} f)^2. \tag{4.19}$$

With some algebras, we can decompose the means square error $\Delta\partial_{\theta_j} f$ into the bias and variance terms, i.e.,

$$\Delta\hat{\partial}_{\theta_j} f = \text{Var}(\hat{\partial}_{\theta_j} f) + \text{Bias}(\hat{\partial}_{\theta_j} f)^2. \tag{4.20}$$

Let a finite-difference gradient estimator be $\hat{\partial}_{\theta_j}^{(h)} f$, where $\hat{\partial}_{\theta_j}^{(h)} f$ is a central difference estimator. The central difference estimator that approximates the jth entry of the gradient for a fixed step size $h > 0$ is

$$\hat{\partial}_{\theta_j}^{(h)} f = \frac{\hat{f}(\theta_j + h) - \hat{f}(\theta_j - h)}{2h}. \tag{4.21}$$

By Equation (4.15), we let $\hat{f}(\theta_j \pm h) = f(\theta_j \pm h) + \hat{\epsilon}_\pm$, so Equation (4.21) becomes

$$
\begin{aligned}
\hat{\partial}^{(h)}_{\theta_j} f &= \frac{f(\theta_j + h) - f(\theta_j - h)}{2h} + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} \\
&= \frac{\left(f(\theta_j) + hf'(\theta_j) + h^2\frac{f''(\theta_j)}{2!} + h^3\frac{f'''(\xi_1)}{3!}\right) - \left(f(\theta_j) - hf'(\theta_j) + h^2\frac{f''(\theta_j)}{2!} - h^3\frac{f'''(\xi_2)}{3!}\right)}{2h} + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} \\
&\quad (\text{where } \xi_1 \in (\theta_j, \theta_j + h),\ \xi_2 \in (\theta_j - h, \theta_j) \\
&= \partial^{(h)}_{\theta_j} f + h^2 \frac{f'''(\xi_1) - f'''(\xi_2)}{12} + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} \\
&= \partial^{(h)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} + O\left(h^2\right)
\end{aligned}
$$

The bias and variance of the central difference estimators respectively are

$$
\begin{aligned}
\text{Bias}\left(\hat{\partial}^{(h)}_{\theta_j} f\right) &= \mathbb{E}\left[\partial^{(h)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} + O\left(h^2\right)\right] - \partial^{(h)}_{\theta_j} f \\
&= \underbrace{\mathbb{E}\left[\partial^{(h)}_{\theta_j} f\right]}_{=\partial^{(h)}_{\theta_j} f} + \underbrace{\mathbb{E}\left[\frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h}\right]}_{=0} + \underbrace{\mathbb{E}\left[O\left(h^2\right)\right]}_{=O(h^2)} - \partial^{(h)}_{\theta_j} f \\
&= O\left(h^2\right)
\end{aligned}
\tag{4.22}
$$

$$
\begin{aligned}
\text{Var}\left(\hat{\partial}^{(h)}_{\theta_j} f\right) &= \mathbb{E}\left(\left(\partial^{(h)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} + O\left(h^2\right)\right)^2\right) - \mathbb{E}\left(\partial^{(h)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h} + O\left(h^2\right)\right)^2 \\
&\quad \left[\text{As } \partial^{(h)}_{\theta_j} f, \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h}, O\left(h^2\right) \text{ are independent, all covariances of two different terms are zero}\right] \\
&= \mathbb{E}\left(\left(\frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h}\right)^2\right) - \underbrace{\mathbb{E}\left(\frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2h}\right)^2}_{=0} \\
&= \mathbb{E}\overbrace{\left(\frac{\hat{\epsilon}_+^2 - 2\hat{\epsilon}_+\hat{\epsilon}_- + \hat{\epsilon}_-^2}{4h^2}\right)}^{\mathbb{E}(\hat{\epsilon}_+\hat{\epsilon}_-)=0\ \because \text{cov}((\hat{\epsilon}_+,\hat{\epsilon}_-)=0} \\
&\quad \left[\text{Let var}\left(\hat{\epsilon}_\pm^2\right) := \frac{\sigma_0^2(\theta_j \pm h)}{S} = \mathbb{E}\left(\hat{\epsilon}_\pm^2\right) - \ \mathbb{E}(\hat{\epsilon}_\pm)^2 = \mathbb{E}\left(\hat{\epsilon}_\pm^2\right),\ S \text{ be the number of measurement shots}\right] \\
&= \frac{\frac{\sigma_0^2(\theta_j+h)}{S} + \frac{\sigma_0^2(\theta_j-h)}{S}}{4h^2} \\
&= \frac{\sigma_0^2(\theta_j + h) + \sigma_0^2(\theta_j - h)}{4Sh^2} \\
&\approx \frac{\sigma_0^2}{2Sh^2}
\end{aligned}
\tag{4.23}
$$

Next, we consider the gradient estimators using the parameter-shift rule. As discussed in Equation (4.14), we let the parameter-shift estimator be $\hat{\partial}^{(s)}_{\theta_j} f$ defined as follows:

$$
\hat{\partial}^{(s)}_{\theta_j} f := \frac{\hat{f}(\theta_j + s) - \hat{f}(\theta_j - s)}{2\sin(s)} = \partial^{(s)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)} \tag{4.24}
$$

where we let $\hat{\epsilon}_\pm$ be the noise induced from measuring $\hat{f}(\theta_j + s)$, with s being the shift in parameter.

The bias and variance of the parameter shift estimators respectively are

$$\begin{aligned}
\text{Bias}\left(\hat{\partial}^{(s)}_{\theta_j} f\right) &= \mathbb{E}\left[\partial^{(s)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}\right] - \partial^{(s)}_{\theta_j} f \\
&= \mathbb{E}\left[\partial^{(s)}_{\theta_j} f\right] + \mathbb{E}\left[\frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}\right] - \partial^{(s)}_{\theta_j} f \\
&= \partial^{(s)}_{\theta_j} f + 0 - \partial^{(s)}_{\theta_j} f \\
&= 0
\end{aligned} \tag{4.25}$$

$$\begin{aligned}
\text{Var}\left(\hat{\partial}^{(s)}_{\theta_j} f\right) &= \mathbb{E}\left(\left(\partial^{(s)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}\right)^2\right) - \mathbb{E}\left(\partial^{(s)}_{\theta_j} f + \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}\right)^2 \\
&\left[\text{As } \partial^{(s)}_{\theta_j} f, \frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)} \text{ are independent, all covariances of different terms are zero}\right] \\
&= \mathbb{E}\left(\left(\frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}\right)^2\right) - \underbrace{\mathbb{E}\left(\frac{\hat{\epsilon}_+ - \hat{\epsilon}_-}{2\sin(s)}\right)^2}_{=0} \\
&= \overbrace{\mathbb{E}\left(\frac{\hat{\epsilon}_+^2 - 2\hat{\epsilon}_+\hat{\epsilon}_- + \hat{\epsilon}_-^2}{4\sin^2(s)}\right)}^{\mathbb{E}(\hat{\epsilon}_+\hat{\epsilon}_-)=0 \ \because \text{cov}((\hat{\epsilon}_+,\hat{\epsilon}_-)=0} \\
&\left[\text{Let var}\left(\hat{\epsilon}_\pm^2\right) := \frac{\sigma_0^2(\theta_j \pm h)}{S} = \mathbb{E}\left(\hat{\epsilon}_\pm^2\right) - \ \mathbb{E}(\hat{\epsilon}_\pm)^2 = \mathbb{E}\left(\hat{\epsilon}_\pm^2\right),\ S \text{ be the number of measurement shots}\right] \\
&= \frac{\frac{\sigma_0^2(\theta_j+h)}{S} + \frac{\sigma_0^2(\theta_j-h)}{S}}{4\sin^2(s)} \\
&= \frac{\sigma_0^2(\theta_j+h) + \sigma_0^2(\theta_j-h)}{4S\sin^2(s)} \\
&\approx \frac{\sigma_0^2}{2S\sin^2(s)}
\end{aligned} \tag{4.26}$$

When we choose $s = \frac{\pi}{2}$, the variance term becomes

$$\text{Var}\left(\hat{\partial}^{(s)}_{\theta_j} f\right) \approx \frac{\sigma_0^2}{2S} \ll \text{Var}\left(\hat{\partial}^{(h)}_{\theta_j} f\right) \approx \frac{\sigma_0^2}{2Sh^2} \text{ for small } h \text{ and same } S \tag{4.27}$$

As we can see, there is a bias-variance trade-off for the finite difference estimator. To reduce the bias term a small step h size should be chosen. Nevertheless, a reduction value of h will result in an exponential increase in the variance of the estimation. To cope with this dilemma, a significantly large number of measurement shots is required, which in turn causes a large overhead in computation. On the contrary, the parameter shift rule for gradient estimation does not suffer from the bias-variance trade-off. It is an unbiased estimator and for large macroscopic shift $s = \frac{\pi}{2}$ the variance will be substantially smaller than the finite difference method's as shown in Equation (4.27). Therefore, the measurement shot number to be chosen can be notably smaller and this explains that the parameter shift estimator has become the *de facto* standard for estimating gradients in parameterised quantum circuits.

4.2 Optimisers for Training Quantum Neural Networks

As quantum neural networks belong to the category of hybrid quantum-classical algorithms, to train the parameterised quantum models we are going to utilise a classical computing device to improve the performance of the quantum models by updating the parameters, either using iterative gradient-based optimisation or gradient-free optimisation. Mostly in training quantum neural networks, gradient-based optimisation methods are preferred as the parameter space could be large in over-parameterised models and gradient-free methods may struggle to achieve convergence. In gradient-based optimisation, for a quantum neural network, a quantum device is responsible for the forward passing of the model to calculate the expected loss as well as evaluating the components from the parameter shift rule for the classical computer to process, while the classical is responsible for computing the gradients and update the parameter in each step. Here we are going to quickly summarise common classical optimisation methods for training quantum neural networks before we introduced more advanced optimisation methods [25].

First-order methods

First-order optimisation methods are optimisation approaches that use the expected loss function $\mathcal{L}(\boldsymbol{\theta})$'s first-order derivatives, where $\boldsymbol{\theta} \in \mathbb{R}^k$. For example, $l2$ loss is a loss function which measures the expected squared error between the prediction $\hat{f}(\boldsymbol{\theta}, x^{(m)})$ and the target value $y^{(m)}$ for all $(x^{(m)}, y^{(m)})$ in M training data, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{2} \left(\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) - y^{(m)} \right)^2. \tag{4.28}$$

By the chain rule, the gradient of $\mathcal{L}(\boldsymbol{\theta})$ can be calculated :

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \sum_{m=1}^{M} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)} \cdot \nabla_{\boldsymbol{\theta}} \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) = \frac{1}{M} \sum_{m=1}^{M} \left(\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) - y^{(m)} \right) \cdot \nabla_{\boldsymbol{\theta}} \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right), \tag{4.29}$$

where $\nabla \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)$ is the function's gradient that is determined by the parameters and mth data point. The gradient of $\mathcal{L}(\boldsymbol{\theta})$ computes the direction of the descent of the expected loss function (but do not consider the curvature of the expected loss function) and iteratively updates the parameters.

First-order methods have the form:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta^{(t)} \boldsymbol{d}^{(t)}, \tag{4.30}$$

in which at each iteration t, the parameter vector $\boldsymbol{\theta}^{(t)}$ is updated by a chosen learning rate η_t and a descent direction $\boldsymbol{d}^{(t)}$. When $\eta^{(t)} = \eta, \eta \in \mathbb{R}\ \forall t$ and $\boldsymbol{d}^{(t)} = -\nabla \mathcal{L}(\boldsymbol{\theta})|^T_{\boldsymbol{\theta}^{(t)}}$, this is the Vanilla Gradient Descent which updates the parameters by moving in the direction of steepest descent at a constant speed. The learning rate can be chosen to be flexible to encourage faster convergence. For example, we can at first pick a larger learning rate to explore the parameter space and then gradually reduce the learning rate so that the optimisation can converge.

Second-order methods

Second-order optimisation methods, in distinction to first-order methods, model the objective function curvature which may yield faster convergence (but could be more costly to compute). One example is Newton's method, which has the form

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta^{(t)} {H^{(t)}}^{-1} \nabla \mathcal{L}(\boldsymbol{\theta})|^T_{\boldsymbol{\theta}^{(t)}}, \tag{4.31}$$

where

$$H^{(t)} = \nabla^2 \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t)}} \tag{4.32}$$

is the Hessian of the expected loss function evaluated at the current parameter value

For instance, to find the Hessian of the $l2$ loss function with M training data as in Equation (4.28), we can first calculate the Hessian for each data point, $H^{(t)}(x^{(m)})$, and then average them to get the overall Hessian matrix. That is,

$$H^{(t)} = \begin{pmatrix} \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_1^2}|_{\boldsymbol{\theta}^{(t)}} & \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_1 \partial \theta_2}|_{\boldsymbol{\theta}^{(t)}} & \cdots & \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_1 \partial \theta_k}|_{\boldsymbol{\theta}^{(t)}} \\ \vdots & \cdots & \cdots & \vdots \\ \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_k \partial \theta_1}|_{\boldsymbol{\theta}^{(t)}} & \cdots & \cdots & \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_k^2}|_{\boldsymbol{\theta}^{(t)}} \end{pmatrix}, \tag{4.33}$$

$$\begin{aligned} \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}|_{\boldsymbol{\theta}^{(t)}} &= \left(\frac{\partial}{\partial \theta_i} \left(\sum_{m=1}^{M} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)} \cdot \frac{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)}{\partial \theta_j} \right) \right) |_{\boldsymbol{\theta}^{(t)}} \\ &= \left(\frac{1}{M} \cdot \sum_{m=1}^{M} \frac{\partial}{\partial \theta_i} \left(\left(\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) - y^{(m)} \right) \cdot \frac{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)}{\partial \theta_j} \right) \right) |_{\boldsymbol{\theta}^{(t)}} \\ &= \left(\frac{1}{M} \cdot \sum_{m=1}^{M} \left(\left(\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) - y^{(m)} \right) \cdot \frac{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)}{\partial \theta_i \partial \theta_j} + \frac{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)}{\partial \theta_j} \cdot \frac{\partial \left(\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) - y^{(m)} \right)}{\partial \theta_i} \right) \right) |_{\boldsymbol{\theta}^{(t)}} \\ &= \frac{1}{M} \cdot \sum_{m=1}^{M} \underbrace{\left(\left(\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right) - y^{(m)} \right) \cdot \frac{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)}{\partial \theta_i \partial \theta_j} + \frac{\partial \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)}{\partial \theta_j} \cdot \frac{\partial (\hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right))}{\partial \theta_i} \right) |_{\boldsymbol{\theta}^{(t)}}}_{H^{(t)}(x^{(m)})}, \end{aligned} \tag{4.34}$$

where $\frac{\partial \hat{f}(\boldsymbol{\theta}, x^{(m)})}{\partial \theta_i \partial \theta_j}$ is the i, jth entry of the Hessian of prediction $\hat{f}(\boldsymbol{\theta}, x^{(m)})$ and $\frac{\partial \hat{f}(\boldsymbol{\theta}, x^{(m)})}{\partial \theta_j}$ is the jth element of $\nabla_{\boldsymbol{\theta}} \hat{f}\left(\boldsymbol{\theta}, x^{(m)}\right)$. If the loss function is precisely the prediction function, $\frac{\partial \hat{f}(\boldsymbol{\theta}, x^{(m)})}{\partial \theta_i \partial \theta_j}|_{\boldsymbol{\theta}^{(t)}}$ is then the i, jth entry of the Hessian of the loss function evaluated at the value of current parameter (For simplicity, the demonstration of Quantum Natural Gradient and Quantum Natural SPSA in Algorithms 1, 2 and 3 will be presented in this way). Or else the above procedures in Equation (4.34) are needed to evaluate the Hessian of the expected $l2-$loss.

Equation (4.31) can be derived easily. Consider the Taylor series expansion of $\mathcal{L}(\boldsymbol{\theta})$ around $\boldsymbol{\theta}^{(t)}$ to approximate $\mathcal{L}(\boldsymbol{\theta})$, up to 2nd order:

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}^{(t)}) + \nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t)}}^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)})^T H^{(t)} (\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}) \tag{4.35}$$

The argument of the minimum of the Taylor series approximation is

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)} - {H^{(t)}}^{-1} \nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t)}}^T \tag{4.36}$$

and Equation (4.36) can be interpreted as Newton's method with a stepsize of 1. So Newton's method is the generalisation of the method in Equation (4.36).

Stochastic Gradient Descent

Stochastic optimisation aims to minimise the expected loss function's average value:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{q(x)}[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z})] \tag{4.37}$$

with $\boldsymbol{z} \sim q(x)$ being a random noise term that comes from the environment. In the vanilla stochastic gradient descent, this can be interpreted as drawing a mini-batch B from the training set M to approximate the expected loss gradient, that is,

$$\nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^{(t+1)}}^T = \nabla \left(\frac{1}{|B|} \sum_{m \in B} l_m(\boldsymbol{\theta}) \right), \tag{4.38}$$

where $l_m(\boldsymbol{\theta})$ is the loss function associated with the mth datapoint in the minibatch and $|B| \ll |M|$. The stochastic gradient descent is unbiased in estimation. This is because the expectation of the stochastic approximation is the gradient of the expected loss, i.e.,

$$\mathbb{E}\left(\nabla\left(\frac{1}{|B|}\sum_{m\in B} l_m(\boldsymbol{\theta})\right)\right) = \frac{1}{M}\sum_{m\in M} l_m(\boldsymbol{\theta}) = \nabla\mathcal{L}(\boldsymbol{\theta}). \tag{4.39}$$

Therefore, stochastic gradient descent asymptotically converges to the minimum and it is faster than gradient descent in practice. There are variants for stochastic optimisation. For example, there is *ADAM* optimisation that makes use of an exponentially weighted moving average of past squared gradients and momentum with the basic form

$$\boldsymbol{m}^{(t)} = \beta_1\boldsymbol{m}^{(t-1)} + (1-\beta_1)\nabla\mathcal{L}(\boldsymbol{\theta})|^T_{\boldsymbol{\theta}^{(t)}} \tag{4.40}$$

$$\boldsymbol{s}^{(t)} = \beta_2\boldsymbol{s}^{(t-1)} + (1-\beta_2)(\nabla\mathcal{L}(\boldsymbol{\theta})|^T_{\boldsymbol{\theta}^{(t)}})^2 \tag{4.41}$$

where $\boldsymbol{m}^{(t)}$ is the momentum term which is the exponentially weighted moving average of past gradients and $\boldsymbol{s}^{(t)}$ is the exponentially weighted moving average of past squared gradients, with $\beta_1 \in [0,1]$ and $\beta_2 \in [0,1]$ being the constants to control the two terms respectively. ADAM updates the parameters by

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)} - \eta^{(t)}\frac{1}{\sqrt{\boldsymbol{s}^{(t)}}+\epsilon} \tag{4.42}$$

where $\epsilon > 0$ is a small value introduced to prevent the denominator from being zero. All in all, ADAM calculates the rolling average of the gradient's 1^{st} and 2^{nd} moments to encourage faster convergence.

4.2.1 Quantum Natural Gradient

Learning the common optimisers is the foundation towards training quantum neural networks. Of course, we can immediately adopt any of the optimisers above to optimise the expected loss of the quantum neural network (because the quantum neural network is a hybrid quantum-classical which relies on a classical optimisation feedback loop upon training). But if we stop the discussion here, it would be quite uninteresting. Are there any advanced methods that extend some classical optimisation methods to use information in the quantum world to update the parameters? The answer is, yes. In this part, we will present a so-called *Quantum Natural Gradient* specifically used to optimise parameterised quantum model by Stokes [35]. But beforehand, we will explain what a *Natural Gradient* refers to, explained by [22] and [7].

Suppose we have a model that is described by a likelihood, a probability density $p(x|\boldsymbol{\theta})$ parameterised by a parameter vector $\boldsymbol{\theta}$. As the approach to finding the optimal $\boldsymbol{\theta}$ in frequentist statistics is the maximum (log-)likelihood estimation, we can define a score function $s(\boldsymbol{\theta})$ to evaluate the goodness of the estimation by

$$s(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\log p(x|\boldsymbol{\theta}). \tag{4.43}$$

In this context, the score function represents the gradient of the log-likelihood function.

Proposition 1. *The score function $s(\boldsymbol{\theta})$'s expected value with respect to the model $p(x|\boldsymbol{\theta})$ is 0.*

Proof.

$$\begin{aligned}
\mathbb{E}_{p(x|\boldsymbol{\theta})}[s(\boldsymbol{\theta})] &= \mathbb{E}_{p(x|\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log p(x|\boldsymbol{\theta})] \\
&= \int \nabla_{\boldsymbol{\theta}} \log p(x|\boldsymbol{\theta}) p(x|\boldsymbol{\theta}) dx \\
&= \int \frac{\nabla_{\boldsymbol{\theta}} p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})} p(x|\boldsymbol{\theta}) dx \\
&= \int \nabla_{\boldsymbol{\theta}} p(x|\boldsymbol{\theta}) dx \\
&= \nabla_{\boldsymbol{\theta}} \int p(x|\boldsymbol{\theta}) dx \\
&\text{(Because } p(x|\boldsymbol{\theta}) \text{ is continuous in } \boldsymbol{\theta} \text{ and as the limit of this integration is a constant,} \\
&\text{by Leibniz integral rule, interchanging the integral sign and derivative is justified)} \\
&= \nabla_{\boldsymbol{\theta}} 1 \\
&= 0
\end{aligned} \tag{4.44}$$

□

To measure the uncertainty of the estimation, the covariance of the score function can be used, that is

$$\mathbb{E}_{p(x|\boldsymbol{\theta})}\left[(s(\boldsymbol{\theta}) - \mathbb{E}_{p(x|\boldsymbol{\theta})}[s(\boldsymbol{\theta})])(s(\boldsymbol{\theta}) - \mathbb{E}_{p(x|\boldsymbol{\theta})}[s(\boldsymbol{\theta})])^T\right] = \mathbb{E}_{p(x|\boldsymbol{\theta})}[(s(\boldsymbol{\theta}) - 0])(s(\boldsymbol{\theta}) - 0)^T]. \tag{4.45}$$

The uncertainty measure $\mathbb{E}_{p(x|\boldsymbol{\theta})}[(s(\boldsymbol{\theta}) - 0])^T(s(\boldsymbol{\theta}) - 0)]$ defines the Fisher Information. The Fisher Information F in matrix form is

$$F := \mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log p(x|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(x|\boldsymbol{\theta})^T\right]. \tag{4.46}$$

Apart from representing the whole matrix, for a discrete probability distribution, we can write the Fisher information matrix formula in terms of the i, jth entry, i.e.,

$$[F]_{ij} = \sum_x \frac{\partial}{\partial \theta_i} \log p(x|\boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(x|\boldsymbol{\theta}) = \sum_x \frac{\frac{\partial}{\partial \theta_i} p(x|\boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})} \tag{4.47}$$

(which we will use in some of our proofs). Because it is computationally intractable to evaluate the Fisher Information F, one can instead approximate F by a sample average, i.e.,

$$F \approx \frac{\sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(x^{(i)}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(x^{(i)}|\boldsymbol{\theta})^T}{M} \tag{4.48}$$

with $\{x^{(1)}, ..., x^{(M)}\}$ being the training data set.

F is closely associated with the model log-likelihood's expected Hessian.

Proposition 2. *The log-likelihood's expected Hessian is the negative Fisher information F*

Proof. Because the log likelihood's Hessian is the Jacobian of the log likelihood's gradient,

$$\begin{aligned}
H_{\log p(x|\boldsymbol{\theta})} &= J\left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right) \\
&= \frac{H_{p(x|\boldsymbol{\theta})} p(x|\boldsymbol{\theta}) - \nabla p(x|\boldsymbol{\theta}) \nabla p(x|\boldsymbol{\theta})^T}{p(x|\boldsymbol{\theta}) p(x|\boldsymbol{\theta})} \\
&= \frac{H_{p(x|\boldsymbol{\theta})}}{p(x|\boldsymbol{\theta})} - \left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)\left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)^T
\end{aligned} \tag{4.49}$$

Taking the expectation of the log likelihood's Hessian with regard to the model, we have:

$$
\begin{aligned}
\mathbb{E}_{p(x|\boldsymbol{\theta})}[H_{\log p(x|\boldsymbol{\theta})}] &= \mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\frac{H_{p(x|\boldsymbol{\theta})}}{p(x|\boldsymbol{\theta})} - \left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)\left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)^T\right] \\
&= \mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\frac{H_{p(x|\boldsymbol{\theta})}}{p(x|\boldsymbol{\theta})}\right] - \mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)\left(\frac{\nabla p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)^T\right] \\
&= \int \frac{H_{p(x|\boldsymbol{\theta})}}{p(x|\boldsymbol{\theta})} p(x|\boldsymbol{\theta})dx - \mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log p(x|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(x|\boldsymbol{\theta})^T\right] \\
&= H_{\int p(x|\boldsymbol{\theta})dx} - F \\
&= H_1 - F \\
&= -F
\end{aligned}
\tag{4.50}
$$

Therefore $F = -\mathbb{E}_{p(x|\boldsymbol{\theta})}[H_{\log p(x|\boldsymbol{\theta})}]$ □

It shows that the Fisher information matrix F measures the model log-likelihood curvature and we can simply use the Fisher Information matrix to replace the Hessian with in second-order optimisations. But in fact, we can go further with the Fisher information by introducing the Natural Gradient Descent.

In the approach of maximum likelihood estimation, we maximise the likelihood to obtain the optimal parameter vector $\boldsymbol{\theta}$. This is identical to minimising a loss function $\mathcal{L}(\boldsymbol{\theta})$, with $\mathcal{L}(\boldsymbol{\theta})$ being the negative log-likelihood because the logarithmic function is a monotone-increasing function. In each gradient descent step, parameters are updated in the direction of $-\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$, the direction of steepest descent evaluated at the current $\boldsymbol{\theta}$ value. That is equivalent to

$$
\frac{-\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\|} = \lim_{\epsilon\to 0} \frac{1}{\epsilon} \mathop{\arg\min}_{\boldsymbol{d} \text{ s.t. } \|\boldsymbol{d}\| \leqslant \epsilon} \mathcal{L}(\boldsymbol{\theta}+\boldsymbol{d}), \tag{4.51}
$$

in which we find a vector $\boldsymbol{d}$ to minimise the loss function in each gradient descent step and as such the update $\boldsymbol{\theta}+\boldsymbol{d}$ is in the current parameter's ϵ-neighbourhood. As the ϵ-neighbourhood in Equation (4.51) is expressed in terms of Euclidean norm, this optimisation method discussed above is depending on the parameter space's Euclidean geometry. On the other hand, as we want to minimise $\mathcal{L}(\boldsymbol{\theta})$, the negative log-likelihood, it is more sensible to consider taking optimisation steps in the likelihood space than the parameter space. We work in the likelihood space, we can take into account information of distribution that we cannot include in the parameter space, such as variance of distributions. Because the likelihood is a probability distribution parameterised by $\boldsymbol{\theta}$, we can use a distribution space to call this likelihood space and what we want to do is to take the optimisation step in the steepest descent direction in the distribution space instead.

In measuring distance in the distribution space, we will not use the Euclidean distance but use the KL-divergence, also known as relative entropy, to evaluate how far is a distribution from another distribution. One may have already anticipated that the Fisher Information is connected closely to the KL-divergence. Therefore, we are going to prove the fact that the Fisher information matrix F is in fact the KL-divergence's Hessian between two pair of probability density functions $p(x|\boldsymbol{\theta})$ and $p\left(x|\boldsymbol{\theta}'\right)$, with regard to $\boldsymbol{\theta}'$ and computed at $\boldsymbol{\theta}'=\boldsymbol{\theta}$.

Proposition 3. *F is the KL-divergence's Hessian between two probability density functions $p(x|\boldsymbol{\theta})$ and $p\left(x|\boldsymbol{\theta}'\right)$, with regard to $\boldsymbol{\theta}'$ and computed at $\boldsymbol{\theta}'=\boldsymbol{\theta}$.*

Proof. The KL-divergence can be decomposed into the following terms, that is,

$$
\mathrm{KL}\left[p(x|\boldsymbol{\theta}) \parallel p\left(x|\boldsymbol{\theta}'\right)\right] = \underbrace{\mathbb{E}_{p(x|\boldsymbol{\theta})}[\log p(x|\boldsymbol{\theta})]}_{\text{entropy}} \underbrace{-\mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\log p\left(x|\boldsymbol{\theta}'\right)\right]}_{\text{cross-entropy}}. \tag{4.52}
$$

The first derivative of the KL-divergance with regard to $\boldsymbol{\theta}'$ is

$$\begin{aligned}\nabla_{\boldsymbol{\theta}'}\mathrm{KL}\left[p(x|\boldsymbol{\theta}) \parallel p\left(x|\boldsymbol{\theta}'\right)\right] &= \underbrace{\nabla_{\boldsymbol{\theta}'}\mathbb{E}_{p(x|\boldsymbol{\theta})}[\log p(x|\boldsymbol{\theta})]}_{=0} - \nabla_{\boldsymbol{\theta}'}\mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\log p\left(x|\boldsymbol{\theta}'\right)\right] \\ &= -\mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}'}\log p\left(x|\boldsymbol{\theta}'\right)\right] \\ &= -\int p(x|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}'}\log p\left(x|\boldsymbol{\theta}'\right)dx.\end{aligned} \tag{4.53}$$

The second derivative of the KL-divergence with regard to $\boldsymbol{\theta}'$ is

$$\nabla^2_{\boldsymbol{\theta}'}\mathrm{KL}\left[p(x|\boldsymbol{\theta}) \parallel p\left(x|\boldsymbol{\theta}'\right)\right] = -\int p(x|\boldsymbol{\theta})\nabla^2_{\boldsymbol{\theta}'}\log p\left(x|\boldsymbol{\theta}'\right)dx \tag{4.54}$$

Therefore, when evaluated at $\boldsymbol{\theta}' = \boldsymbol{\theta}$, the Hessian of KL-divergence between two probability distributions $p(x|\boldsymbol{\theta})$ and $p\left(x|\boldsymbol{\theta}'\right)$ with respect to $\boldsymbol{\theta}'$ is

$$\begin{aligned}H_{\mathrm{KL}[p(x|\boldsymbol{\theta}) \parallel p(x|\boldsymbol{\theta}')]} &= -\int p(x|\boldsymbol{\theta})\nabla^2_{\boldsymbol{\theta}'}\log p\left(x|\boldsymbol{\theta}'\right)\Big|_{\boldsymbol{\theta}'=\boldsymbol{\theta}}dx \\ &= -\int p(x|\boldsymbol{\theta})H_{\log p(x|\boldsymbol{\theta})}dx \\ &= \mathbb{E}_{p(x|\boldsymbol{\theta})}[H_{\log p(x|\boldsymbol{\theta})}] \\ &= F \text{ by Equation (4.50).}\end{aligned}$$

We also need the Taylor series expansion of KL-divergence up to 2nd order, which is

$$\begin{aligned}\mathrm{KL}[p(x|\boldsymbol{\theta})\|p(x|\boldsymbol{\theta}+\boldsymbol{d})] \approx\; & \underbrace{\mathrm{KL}[p(x|\boldsymbol{\theta})\|p(x|\boldsymbol{\theta})]}_{\substack{=0 \text{ as KL-divergence of} \\ \text{identical distributions is zero}}} + \left(\nabla_{\boldsymbol{\theta}'}\mathrm{KL}\left[p(x|\boldsymbol{\theta})\|p\left(x|\boldsymbol{\theta}'\right)\right]|_{\theta'=\theta}\right)^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^TF\boldsymbol{d} \\ &= 0 - \underbrace{\mathbb{E}_{p(x|\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}'}\log p\left(x|\boldsymbol{\theta}'\right)\right]^T}_{=0 \text{ by Proposition 1}}\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^TF\boldsymbol{d} \\ &= \frac{1}{2}\boldsymbol{d}^TF\boldsymbol{d}\end{aligned} \tag{4.55}$$

Hence, we have $\mathrm{KL}[p(x|\boldsymbol{\theta}) \parallel p(x|\boldsymbol{\theta}+\boldsymbol{d})] \approx \frac{1}{2}\boldsymbol{d}^{\boldsymbol{T}}F\boldsymbol{d}$. □

Having outlined the essential preparations, we can now derive the natural gradient descent algorithm which harnesses the Fisher information. We are interested to search for an update vector $\boldsymbol{d}$ to minimise the loss function $\mathcal{L}(\boldsymbol{\theta})$ within the distribution space with KL-divergence as the distance metric. This can be formally expressed as

$$\boldsymbol{d}^* = \underset{\boldsymbol{d} \text{ s.t. } \mathrm{KL}[p(x|\boldsymbol{\theta})\|p(x|\boldsymbol{\theta}+\boldsymbol{d})]=c}{\arg\min} \mathcal{L}(\boldsymbol{\theta}+\boldsymbol{d}), \tag{4.56}$$

with c being a constant. Why the KL-divergence is chosen to be constant is that we want to ensure a consistent speed when moving along the distribution space. This makes the natural gradient descent algorithm more robust to the model reparameterisation as it concerns the distribution only but not how the model is being parameterised. Now let's move forward to solving Equation (4.56). Equation (4.56) can be written in Lagrangian function approximating the $\mathcal{L}(\boldsymbol{\theta}+\boldsymbol{d})$ by 1st order Taylor series expansion and the KL-divergence by 2nd order Taylor series expansion (shown in Equation (4.55)), i.e.,

$$\begin{aligned} \boldsymbol{d}^* &= \arg\min_{d} \mathcal{L}(\boldsymbol{\theta}+\boldsymbol{d}) + \lambda\left(\mathrm{KL}[p(x|\boldsymbol{\theta})\|p(x|\boldsymbol{\theta}+\boldsymbol{d})] - c\right) \\ &= \arg\min_{\boldsymbol{d}} \mathcal{L}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})^T \boldsymbol{d} + \frac{1}{2}\lambda \boldsymbol{d}^T F \boldsymbol{d} - \lambda c \end{aligned} \tag{4.57}$$

Solving the minimisation, we set $\frac{\partial}{\partial \boldsymbol{d}}\boldsymbol{d}^* = 0$:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{d}}\left(\mathcal{L}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})^T \boldsymbol{d} + \frac{1}{2}\lambda \boldsymbol{d}^T F \boldsymbol{d} - \lambda c\right) &= 0 \\ \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) + \lambda F \boldsymbol{d} &= 0 \\ \boldsymbol{d} &= -\frac{1}{\lambda} F^{-1} \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \end{aligned} \tag{4.58}$$

As $\frac{1}{\lambda}$ is a constant factor, we can absorb it into the user-defined learning rate. The natural gradient is the optimal ascent direction of the loss function while taking its distribution space's local curvature into consideration, that is.

$$\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = F^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \tag{4.59}$$

Algorithm 1 Natural Gradient Descent

```
Require: Parameterised model, loss function $\mathcal{L}$, gradient estimator, learning rate $\alpha$ are provided,
1: Initialise parameters $\boldsymbol{\theta}$ which have slight random deviations from zero
2: while not converged do
3:     Compute the loss function $\mathcal{L}(\boldsymbol{\theta})$ by forward passing of the model
4:     Evaluate the loss function's gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
5:     Estimate the Fisher information matrix $F$
6:     The natural gradient is computed by $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = F^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
7:     The parameters are updated by $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \alpha\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
8: end while
```

As we have seen, the Fisher information captures how much a probability distribution changes when the parameter vector $\boldsymbol{\theta}$ changes. On top of that, what we are discussing can clearly be applied to parameterised quantum circuits. A measurement of the parameterised quantum state in a quantum neural network collapses it into a probability distribution with possible measurement outcomes depending on the observable. From the previous discussion, we know that using Euclidean distance as a measure might not be a sensible choice as this means we are optimising the loss function within the parameter space. Because parameters do not have equal influences on the loss function, we have considered optimising the loss function within the distribution space, which uses the KL-divergence to measure the distance between probability distributions. Optimising in the distribution space serves as a smarter choice as it uses a new distance metric that is more associated with the loss function. Another sensible choice, which we have not discussed, is to take a step backwards and use a metric to measure the distance of the final quantum states before measurement, and this leads to the introduction of quantum Fisher information.

We can use fidelity as the distance metric to measure the distance between quantum states. The fidelity between any two pair of pure final quantum states $|\psi(\boldsymbol{\theta})\rangle$ and $|\psi(\boldsymbol{\theta}')\rangle$ is

$$f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) := \left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}')\rangle\right|^2 \tag{4.60}$$

The distance metric for two pair of pure quantum states, $g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right)$, is then

$$g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) := 1 - f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) = 1 - |\langle\psi(\boldsymbol{\theta})|\,||\psi(\boldsymbol{\theta}')\rangle\rangle|^2. \tag{4.61}$$

$g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right)$ is truly a distance measure as it is always positive, that is,

$$g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) \geqslant 0 \tag{4.62}$$

. Moreover, it satisfies that the distance between identical pure quantum states is zero, i.e.

$$g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta})\rangle\right) = 0. \tag{4.63}$$

We can further derive the information matrix related to fidelity.

Proposition 4. *The information matrix associated with fidelity is described by M_f, in which*

$$[M_f]_{ij} = 2Re\left[\langle\partial_i\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle - \langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\,\langle\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle\right], \tag{4.64}$$

where ∂_i is the shorthand for $\dfrac{\partial}{\partial\theta_i}$.

Proof. Consider a small displacement vector $\boldsymbol{d}$, we have

$$|\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle = |\psi(\boldsymbol{\theta})\rangle + \sum_i d_i\,|\partial_i\psi(\boldsymbol{\theta})\rangle, \tag{4.65}$$

where d_i is the ith element of $\boldsymbol{d}$. Then it follows that,

$$\begin{aligned} f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right) &= |\,\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\,|^2 \\ &= \left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle + \textstyle\sum_i d_i\,\langle\psi(\boldsymbol{\theta})|\,|\partial_i\psi(\boldsymbol{\theta})\rangle\right|^2 \\ &= \left|1 + \textstyle\sum_i d_i\,\langle\psi(\boldsymbol{\theta})|\,|\partial_i\psi(\boldsymbol{\theta})\rangle\right|^2 \end{aligned} \tag{4.66}$$

with f being the fidelity measure. Rewriting Equation (4.66) in absolute values, we have

$$\begin{aligned} f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right) = 1 &+ \sum_i d_i\left(\langle\psi(\boldsymbol{\theta})|\,|\partial_i\psi(\boldsymbol{\theta})\rangle + \langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\right) \\ &+ \sum_{ij} d_id_j\,\langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\,\langle\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle. \end{aligned} \tag{4.67}$$

By the fact that $|\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle$ should be a pure state, this implies that

$$f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right) = \;\; 1 \tag{4.68}$$

We can decompose $f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right)$ into

$$\begin{aligned} f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right) = \underbrace{\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle}_{=1} &+ \sum_i d_i\left(\langle\psi(\boldsymbol{\theta})|\,|\partial_i\psi(\boldsymbol{\theta})\rangle + \langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\right) \\ &+ \sum_{ij} d_id_j\,\langle\partial_i\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle \end{aligned} \tag{4.69}$$

Putting Equation (4.68) into Equation (4.69), we can see that

$$\sum_i d_i\left(\langle\psi(\boldsymbol{\theta})|\,|\partial_i\psi(\boldsymbol{\theta})\rangle + \langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\right) = -\sum_{ij} d_id_j\,\langle\partial_i\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle \tag{4.70}$$

Putting Equation (4.70) into Equation (4.67), we have

$$\begin{aligned} f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right) = 1 &- \sum_{ij} d_id_j\,\langle\partial_i\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle \\ &+ \sum_{ij} d_id_j\,\langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\,\langle\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle. \end{aligned} \tag{4.71}$$

As such, the i, jth entry of the Hessian of $g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right)$ is

$$\begin{aligned}
[M_f]_{ij} &= \frac{\partial^2}{\partial d_i \partial d_j} g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) \\
&= \frac{\partial^2}{\partial d_i \partial d_j}\left[1 - f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right)\right] \\
&= \frac{\partial^2}{\partial d_i \partial d_j}\left[\sum_{ij} d_i d_j \langle \partial_i \psi(\boldsymbol{\theta})| |\partial_j \psi(\boldsymbol{\theta})\rangle - \sum_{ij} d_i d_j \langle \partial_i \psi(\boldsymbol{\theta})| |\psi(\boldsymbol{\theta})\rangle \langle \psi(\boldsymbol{\theta})| |\partial_j \psi(\boldsymbol{\theta})\rangle\right] \\
&= 2\text{Re}\left[\langle \partial_i \psi(\boldsymbol{\theta})| |\partial_j \psi(\boldsymbol{\theta})\rangle - \langle \partial_i \psi(\boldsymbol{\theta})| |\psi(\boldsymbol{\theta})\rangle \langle \psi(\boldsymbol{\theta})| |\partial_j \psi(\boldsymbol{\theta})\rangle\right]
\end{aligned} \tag{4.72}$$

The multiple 2 and the real part comes from the fact that $d_i d_j$ comes with conjugate terms and $z + z^* = 2\text{Re}[z]$ for any complex number z. □

To verify that this information matrix aligns with the classical scenario, consider the state

$$|\psi(\theta)\rangle = \sum_x \sqrt{p(x|\boldsymbol{\theta})}\, |x\rangle, \tag{4.73}$$

where $|x\rangle$ is the xth computational basis state with real amplitudes (it is real as we are considering the classical case) $\sqrt{p(x|\boldsymbol{\theta})}$ which is determined by the model. The measurement of $|\psi(\theta)\rangle$ results in a classical probability density function $p(x|\boldsymbol{\theta})$. the first term $\langle \partial_i \psi(\boldsymbol{\theta})| |\partial_j \psi(\boldsymbol{\theta})\rangle$ in Equation (4.72) is

$$\begin{aligned}
\langle \partial_i \psi(\boldsymbol{\theta})| |\partial_j \psi(\boldsymbol{\theta})\rangle &= \sum_x \left(\partial_i \sqrt{p(x|\boldsymbol{\theta})}\right)\left(\partial_j \sqrt{p(x|\boldsymbol{\theta})}\right) \\
&= \frac{1}{4}\sum_x \frac{\partial_i p(x|\boldsymbol{\theta})}{\sqrt{p(x|\boldsymbol{\theta})}} \frac{\partial_j p(x|\boldsymbol{\theta})}{\sqrt{p(x|\boldsymbol{\theta})}} \\
&= \frac{1}{4}\sum_x \frac{\partial_i p(x|\boldsymbol{\theta}) \partial_j p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}
\end{aligned} \tag{4.74}$$

We can in fact zero out the second term in Equation (4.72):

$$\begin{aligned}
\langle \partial_i \psi(\boldsymbol{\theta})| |\psi(\boldsymbol{\theta})\rangle &= \sum_x \left(\partial_i \sqrt{p(x|\boldsymbol{\theta})}\right) \sqrt{p(x|\boldsymbol{\theta})} \\
&= \frac{1}{2}\sum_x \frac{\partial_i p(x|\boldsymbol{\theta})}{\sqrt{p(x|\boldsymbol{\theta})}} \sqrt{p(x|\boldsymbol{\theta})} \\
&= \frac{1}{2}\sum_x \partial_i p(x|\boldsymbol{\theta}) \\
&= \frac{1}{2}\partial_i \sum_x p(x|\boldsymbol{\theta}) \\
&= \frac{1}{2}\partial_i 1 \\
&= 0
\end{aligned} \tag{4.75}$$

All things considered, the classical case for $[M_f]_{ij}$ in Equation is

$$[M_f]_{ij} = \frac{1}{2}\sum_x \frac{\partial_i p(x|\boldsymbol{\theta}) \partial_j p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})} = \frac{1}{2}\underbrace{\left(\sum_x \frac{\frac{\partial}{\partial \theta_i} p(x|\boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} p(x|\boldsymbol{\theta})}{p(x|\boldsymbol{\theta})}\right)}_{=F \text{ by Equation (4.47)}} = \frac{1}{2}[F]_{ij} \tag{4.76}$$

Therefore, we see that the quantum information matrix M_f is consistent with the classical case and the quantum Fisher information $\mathcal{F}$ can be defined using the correction in Equation (4.76), that is,

$$\begin{aligned} \mathcal{F} &= 2M_f, \\ [\mathcal{F}]_{ij} &= 4\mathrm{Re}\left[\langle\partial_i\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle - \langle\partial_i\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta})\rangle\,\langle\psi(\boldsymbol{\theta})|\,|\partial_j\psi(\boldsymbol{\theta})\rangle\right] \end{aligned} \tag{4.77}$$

From Equation (4.2.1), the Hessian is the second-ordered partial derivative of g_f which depends on the fidelity f. Let $g_f(\boldsymbol{d}) := g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}+\boldsymbol{d})\rangle\right)$ be the distance metric shorthand for measuring the distance between a quantum state with parameters $\boldsymbol{\theta}$ and that changed by any direction $\boldsymbol{d}$. By the parameter shift rule discussed in Section, the function g_f's first order derivative with respect to θ_j is

$$\frac{\partial}{\partial\theta_j}g_f = \frac{g_f(+s\boldsymbol{e}_j) - g_f(-s\boldsymbol{e}_j)}{2\sin(s)}$$

where the jth element of vector $\boldsymbol{e}_j$ is 1 and the other elements are 0 otherwise. This can easily be applied to get higher-order derivatives and thus the Hessian is

$$H_{ij} = \frac{g_f(+s(\boldsymbol{e_i}+\boldsymbol{e}_j)) - g_f(+s(-\boldsymbol{e_i}+\boldsymbol{e}_j)) - g_f(+s(\boldsymbol{e_i}-\boldsymbol{e}_j)) + g_f(-s(\boldsymbol{e_i}+\boldsymbol{e}_j))}{2\sin^2(s)} \tag{4.78}$$

As $g_f = 1 - f_{\text{fidelity}}$, choosing $s = \frac{\pi}{2}$, we have

$$\begin{aligned} H_{ij} = \mathcal{F}_{ij} = -\frac{1}{2}\Big[&\left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}+(\pi/2)\cdot(\boldsymbol{e_i}+\boldsymbol{e}_j))\rangle\right|^2 - \left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}+(\pi/2)\cdot(\boldsymbol{e_i}-\boldsymbol{e}_j))\rangle\right|^2 \\ &- \left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}-(\pi/2)\cdot(\boldsymbol{e_i}-\boldsymbol{e}_j))\rangle\right|^2 + \left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}-(\pi/2)\cdot(\boldsymbol{e_i}+\boldsymbol{e}_j))\rangle\right|^2\Big] \end{aligned} \tag{4.79}$$

In order compute any $\left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}')\rangle\right|^2$, assuming that $U(\boldsymbol{\theta})\,|0\rangle = |\psi(\boldsymbol{\theta})\rangle$, we can prepare the quantum state $U^\dagger(\boldsymbol{\theta})U(\boldsymbol{\theta})\,|0\rangle$ and evaluate the probability of observing the outcome $|0\rangle$, as

$$\begin{aligned} p\left(|0\rangle\right) &= \langle 0|\,U^\dagger(\boldsymbol{\theta})U\left(\boldsymbol{\theta}'\right)|0\rangle\,\langle 0|\,U^\dagger\left(\boldsymbol{\theta}'\right)U(\boldsymbol{\theta})\,|0\rangle \\ &= \langle\psi(\boldsymbol{\theta})|\,|\psi\left(\boldsymbol{\theta}'\right)\rangle\,\langle\psi\left(\boldsymbol{\theta}'\right)|\,|\psi(\boldsymbol{\theta})\rangle \\ &= \left|\langle\psi(\boldsymbol{\theta})|\,|\psi(\boldsymbol{\theta}')\rangle\right|^2. \end{aligned} \tag{4.80}$$

This could be done by measuring the quantum state $U^\dagger(\boldsymbol{\theta})U(\boldsymbol{\theta})\,|0\rangle$ for some measurement shots and computing the expectation of measurement followed by a transformation into the probability value.

As we already have the knowledge to obtain the quantum information matrix, there is no reason not to apply it to the natural gradient descent method in its quantum version. We call this method the *Quantum Natural Gradient Descent.* How it is different from the classical natural gradient descent is that rather than taking optimisation steps in the distribution space, this quantum method updates the parameter in the steepest descent direction with regard to the geometry of quantum information which uses fidelity for measuring the distance between final quantum states. The procedures of the whole algorithm are described in Algorithm 2

4.2.2 Simultaneous Perturbation Stochastic Approximation for Quantum Natural Gradient Descent

In Section 4.2.1, we have explained the quantum natural gradient descent. Unlike gradient descent, it considers the steepest descent of the loss function in the quantum information geometry which potentially leads to a huge speed-up in training the quantum neural network. However, one major drawback of quantum natural gradient descent algorithm is it requires computing the quantum Fisher information at each step. With k parameters, we need $\mathcal{O}\left(k^2\right)$ function evaluations to compute the information matrix and it becomes very expensive to compute with a large number of parameters. That said, there is an approach to approximate the matrix that requires a constant number of function evaluations only. It is known as the *Simultaneous Perturbation Stochastic Approximation (SPSA) of the Quantum Fisher Information.* In the following, we will explain the ideas of SPSA and how it can

Algorithm 2 Quantum Natural Gradient Descent

```
Require: Parameterised Quantum model, loss function $\mathcal{L}$, gradient estimator by parameter shift
    rule, quantum Fisher Information matrix estimator by parameter shift rule, learning rate $\alpha$ are
    provided,
1: Initialise parameters $\boldsymbol{\theta}$ which have slight random deviations from zero
2: while not converged do
3:     Compute the loss function $\mathcal{L}(\boldsymbol{\theta})$ by forward passing of the model
4:     Evaluate the loss function's gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
5:     Estimate the quantum Fisher information matrix $\mathcal{F}$
6:     The quantum natural gradient is computed by $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \mathcal{F}^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
7:     The parameters are updated by $\boldsymbol{\theta} \to \boldsymbol{\theta} - \alpha\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
8: end while
```

be applied in modifying the natural gradient descent method, with reference to [14] .

SPSA

In analytic gradients or finite difference methods, evaluating the derivatives of a function scales linearly with the number of parameters. Meanwhile, SPSA is a simultaneous perturbation method that avoids linearly growing computational costs, where it stochastically approximates the gradient by simultaneously perturbing all parameters in a random direction. It is an unbiased estimator as long as we draw the random directions by a continuous uniform distribution $U(-1,1)$ for each parameter.

Let $\nabla f(\boldsymbol{\theta}) \in \mathbb{R}^k$ be the gradient of f with k-dimensional parameter vector $\boldsymbol{\theta}$. The tth step of SPSA samples a random direction $\boldsymbol{\Delta}^{(t)}$ from the distribution $U\left(\{1,-1\}^k\right)$ and then approximates the gradient $\nabla f\left(\boldsymbol{\theta}^{(t)}\right)$ as follows:

$$\nabla f\left(\boldsymbol{\theta}^{(t)}\right) \approx \frac{f\left(\boldsymbol{\theta}^{(t)} + \epsilon\boldsymbol{\Delta}^{(t)}\right) - f\left(\boldsymbol{\theta}^{(t)} - \epsilon\boldsymbol{\Delta}^{(t)}\right)}{2\epsilon}\boldsymbol{\Delta}^{(t)}, \tag{4.81}$$

where $\epsilon > 0$ is a small displacement term. SPSA uses merely two evaluations of f compared to $\mathcal{O}(d)$ evaluations in finite differences or analytic gradient methods. In addition to the first-order method, we can extend SPSA to approximate the Hessian. At the tth iteration, using two random directions $\boldsymbol{\Delta}_1^{(t)} \sim U\left(\{1,-1\}^k\right)$ and $\boldsymbol{\Delta}_2^{(t)} \sim U\left(\{1,-1\}^k\right)$, SPSA approximates the Hessian $\hat{H}^{(t)}$ by

$$\hat{H}^{(t)} = \frac{\partial\mathfrak{F}}{2\epsilon^2}\frac{\boldsymbol{\Delta}_1^{(t)}\boldsymbol{\Delta}_2^{(t)^T} + \boldsymbol{\Delta}_2^{(t)}\boldsymbol{\Delta}_1^{(t)^T}}{2}, \tag{4.82}$$

$$\partial\mathfrak{F} := f\left(\boldsymbol{\theta}^{(t)} + \epsilon\boldsymbol{\Delta}_1^{(t)} + \epsilon\boldsymbol{\Delta}_2^{(t)}\right) - f\left(\boldsymbol{\theta}^{(t)} + \epsilon\boldsymbol{\Delta}_1^{(t)}\right) - f\left(\boldsymbol{\theta}^{(t)} - \epsilon\boldsymbol{\Delta}_1^{(t)} + \epsilon\boldsymbol{\Delta}_2^{(t)}\right) + f\left(\boldsymbol{\theta}^{(t)} - \epsilon\boldsymbol{\Delta}_1^{(t)}\right). \tag{4.83}$$

Note that Hessian estimation in Equation (4.83) is an unbiased estimator for the full Hessian. However, to ensure the method is numerically stable, we prefer a biased rolling average of Hessians, that is

$$\tilde{H}^{(t)} = \frac{\left(\sum_{i=1}^{t}\hat{H}^{(i)}\right) + \hat{H}^{(0)}}{t+1} \tag{4.84}$$

where the initial guess of the Hessian $\hat{H}^{(0)} = \mathbb{I}_k$ is a $k \times k$ identity matrix. Besides, The first-order SPSA method is used for evaluating gradient. Therefore, each iteration of the second-order method with SPSA thus requires 6 function evaluations compared to k^2+k evaluations in the analytic method. In order to ensure the estimated Hessian is positive semi-definite and invertible, $\tilde{H}^{(k)}$ is replaced with $\sqrt{\tilde{H}^{(t)}\tilde{H}^{(t)}}$ and add a $k \times k$ diagonal matrix $\beta\mathbb{I}_k$, with β being a small positive real number, that is,

$$\overline{H}^{(t)} = \sqrt{\tilde{H}^{(t)}\tilde{H}^{(t)}} + \beta\mathbb{I}_k. \tag{4.85}$$

It is proved in [9] that when a set of conditions is met, SPSA with a biased Hessian estimator can still be guaranteed to converge, but it is out of the scope of this survey.

SPSA of the Quantum Fisher Information Matrix

Now we show how to apply second-order SPSA to estimate the quantum Fisher information matrix. From Equation (4.61), as the distance for two pure quantum states is $g_f\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) = 1 - f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) = 1 - |\langle\psi(\boldsymbol{\theta})| |\psi(\boldsymbol{\theta}')\rangle|^2$, we know that $\partial g_f = -\partial f_{\text{fidelity}}$. Hence, the SPSA method in Equation (4.83) that approximates the quantum information matrix $\hat{\mathcal{F}}^{(t)}_{SPSA}$in the tth iteration is

$$\hat{\mathcal{F}}^{(t)}_{SPSA} = -\frac{\partial \mathfrak{F}}{2\epsilon^2}\frac{\boldsymbol{\Delta}_1^{(t)}\boldsymbol{\Delta}_2^{(t)T} + \boldsymbol{\Delta}_2^{(t)}\boldsymbol{\Delta}_1^{(t)T}}{2}, \tag{4.86}$$

$$\begin{aligned}\partial\mathfrak{F} := \tilde{f}_{\text{fidelity}}\left(\boldsymbol{\theta^{(t)}}, \boldsymbol{\theta}^{(t)} + \epsilon\boldsymbol{\Delta}_1^{(t)} + \epsilon\boldsymbol{\Delta}_2^{(t)}\right) - \tilde{f}_{\text{fidelity}}\left(\boldsymbol{\theta^{(t)}}, \boldsymbol{\theta}^{(t)} + \epsilon\boldsymbol{\Delta}_1^{(t)}\right)\\ - \tilde{f}_{\text{fidelity}}\left(\boldsymbol{\theta^{(t)}}, \boldsymbol{\theta}^{(t)} - \epsilon\boldsymbol{\Delta}_1^{(t)} + \epsilon\boldsymbol{\Delta}_2^{(t)}\right) + \tilde{f}_{\text{fidelity}}\left(\boldsymbol{\theta^{(t)}}, \boldsymbol{\theta}^{(t)} - \epsilon\boldsymbol{\Delta}_1^{(t)}\right),\end{aligned} \tag{4.87}$$

with ϵ being a small positive displacement term and $\tilde{f}_{\text{fidelity}}\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right) := f_{\text{fidelity}}\left(|\psi(\boldsymbol{\theta})\rangle, |\psi(\boldsymbol{\theta}')\rangle\right) = \left|\langle\psi(\boldsymbol{\theta})| |\psi(\boldsymbol{\theta}')\rangle\right|^2$ is the fidelity between two pure quantum states $|\psi(\boldsymbol{\theta})\rangle$ and $|\psi(\boldsymbol{\theta}')\rangle$. $\tilde{f}_{\text{fidelity}}\left(\boldsymbol{\theta}, \boldsymbol{\theta}'\right)$ can be measured by evaluating the state $U^\dagger\left(\boldsymbol{\theta}'\right)U(\boldsymbol{\theta})|0\rangle$ with S measurement shots and computing the probability of observing outcome 0, as mentioned in Equation (4.80). After $\hat{\mathcal{F}}^{(t)}_{SPSA}$ is computed, we can implement the procedures in Equation (4.84) and 4.85 to obtain $\overline{\mathcal{F}}^{(t)}_{SPSA}$. Before implementing the quantum natural gradient descent with SPSA, we also need to evaluate the gradient using SPSA first-order method in Equation (4.81) to obtain $\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ with regard to the loss function $\mathcal{L}(\boldsymbol{\theta})$. Having everything prepared, the full algorithm is illustrated in Algorithm 3:

Algorithm 3 Quantum Natural SPSA Algorithm

```
Require: Parameterised quantum model, loss function $\mathcal{L}$, gradient estimator by first-order SPSA,
    quantum Fisher information matrix estimator by second-order SPSA, learning rate $\alpha$ are provided,
1: Initialise parameters $\boldsymbol{\theta}$ which have slight random deviations from zero
2: while not converged do
3:     Compute the loss function $\mathcal{L}(\boldsymbol{\theta})$ by forward passing of the model
4:     Evaluate the loss function's gradient $\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ by stochastic approximation
5:     Estimate the approximated quantum Fisher information matrix with regularisation $\overline{\mathcal{F}}_{SPSA}$
6:     The approximated quantum natural gradient is computed by $\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \overline{\mathcal{F}}^{-1}_{SPSA}\hat{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
7:     The parameters are updated by $\boldsymbol{\theta} \to \boldsymbol{\theta} - \alpha\tilde{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
8: end while
```

Here, we implement a small experiment which uses a non-data encoded parameterised quantum neural network to find the lowest energy (i.e., the minimum expectation value of measurement) of this toy quantum model. As we know that the minimum energy must be -1, we can compare the convergence of different gradient-based optimisation methods. We tested the vanilla gradient descent, quantum natural gradient descent and quantum natural SPSA algorithm with the same learning rates to optimise a four-qubit non-data encoded quantum neural network composed of 4 parameterised entangling layers, having 48 trainable parameters in total. The optimiser implementation is adopted from [12] for Quantum Natural SPSA, otherwise, it is implemented using Pennylane's optimiser class. We will investigate the loss function's convergence behaviour over the number of optimisation steps between these optimisation algorithms. Since different optimisation algorithm is associated with varying numbers of function evaluations, it is not fair that we solely consider the number of optimisation steps as an indicator of optimisers' performance. We will also plot the change in the loss function against cumulative training time so as to assess their performance. We expect that the quantum natural gradient descent should have the best performance in terms of optimisation steps This is because it optimises the loss function in the quantum information geometry rather than $l2$-geometry, which should enable faster convergence than the vanilla gradient descent, while quantum natural SPSA is

```python
num_wires = 4
num_layers = 4

dev = qml.device("lightning.qubit", wires=num_layers)
circuit_block = qml.StronglyEntanglingLayers

@qml.qnode(dev)
def circuit(params):
  circuit_block(params, wires=list(range(num_wires)))
  return qml.expval(qml.PauliZ(0))

param_shape = circuit_block.shape(num_layers, num_wires)
init_param = np.random.normal(scale=0.1, size=param_shape, requires_grad=True)

cost_function = qml.QNode(circuit, dev)

max_iterations = 200
step_size = 0.04

opt = qml.GradientDescentOptimizer(stepsize=step_size)

params = init_param

gd_cost_history = []
gd_time_history = []

start=time.time()
for n in range(max_iterations):

    # take an optimisation step
    params, prev_energy = opt.step_and_cost(cost_function, params)

    gd_cost_history.append(prev_energy)

    # track the loss
    energy = cost_function(params)
    now = time.time()
    gd_time_history.append(now-start)

    if n % 20 == 0:
        print(
            f"Iteration = {n},  Loss = {energy}"
        )


print()
print(f"Final value of the Loss = {energy}")
print("Number of iterations = ", n)
```

Figure 10: Part of Code Implementation to Track the Convergence Behaviour of Using Different Optimisers to Train a Quantum Neural Network, with Gradient Descent as an Example

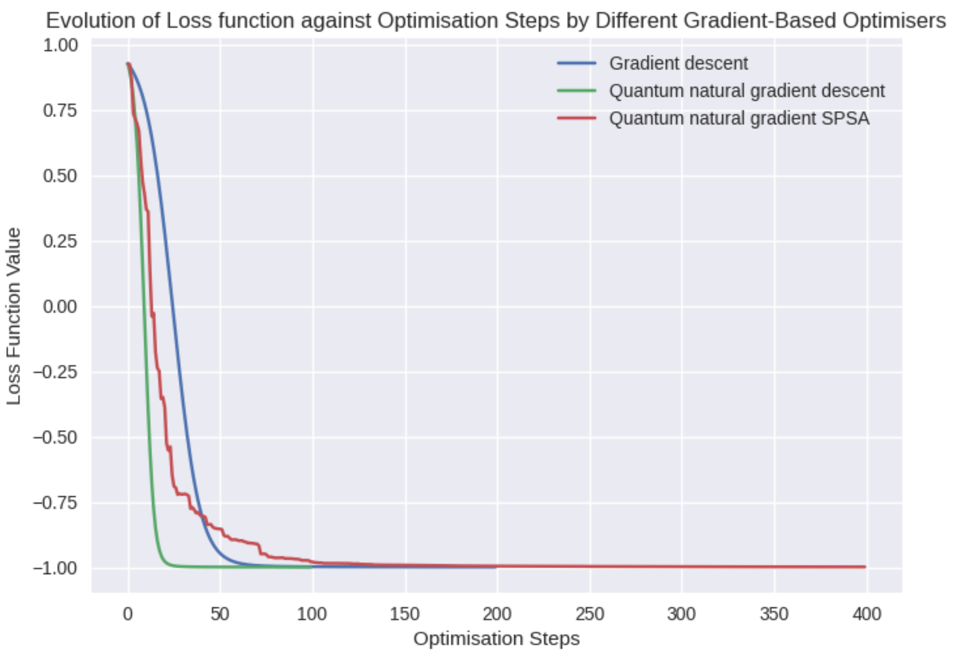


Figure 11: The Evolution of Loss Function against Optimisation Steps by Different Gradient-Based Optimisers

The learning rates of all optimisers are 0.04; A regularisation term of 0.1 is used for quantum natural gradient descent and quantum natural gradient SPSA to ensure the quantum Fisher information matrix invertibility; Training time is tracked in terms of seconds.

a stochastic approximation of the quantum natural gradient which should be slightly worse than the quantum natural gradient descent. We will also investigate their convergence in terms of training time, to account for the influence of the number of an algorithm's function evaluations in each step on the actual performance.

From Figure 12, as expected, the performance of quantum natural gradient descent is the best of all as it has the fastest convergence in terms of optimisation steps. Notice that the curve of quantum natural gradient SPSA's loss function is more angular due to its random nature and surprisingly, its performance is just slightly worse than the quantum natural gradient descent in the beginning. However, when the loss function is approaching its minimum, its stochastic approximations of the quantum Fisher information matrix prevent it from exploiting small solution space and its advantage soon loses to gradient descent.

Nonetheless, what we see could be the tip of the iceberg. Though the execution time is specific to each computer, it is still worthwhile to investigate the evolution of the loss function against cumulative training time in Figure 11. With regard to training time, quantum natural gradient SPSA has the fastest convergence thanks to significantly fewer circuit evaluations as it requires only $\mathcal{O}(1)$ function evaluations and needs not to measure derivatives concerning each parameter one by one compared to the other two methods. On the other side, despite having more function evaluations than gradient descent, quantum natural gradient descent still manages to perform better in time. One can anticipate that as the dimension k of the parameter vector increases (decreases), the advantage of quantum natural gradient descent over vanilla gradient descent becomes smaller (larger) since it requires $\mathcal{O}(k^2)$ evaluations in contrary to $\mathcal{O}(k)$ evaluations in gradient descent and at some point, it would be too costly to compute the quantum Fisher information. So does it means quantum natural gradient SPSA is the most superior? However, there is no definitive answer as to which optimisation method is the most superior. In spite of its fast convergence, one of its cons is that due to its stochastic nature, the argument of minimum is not precise as it oscillates around the minimum. Besides, it is more sensitive to its hyper-parameters, such as the regularisation term and the learning rate, and not carefully choosing it would lead to divergence. It is also true that different models could lead to dissimilar results. That said, the experiment here just serves as a baseline for studying the characteristics of

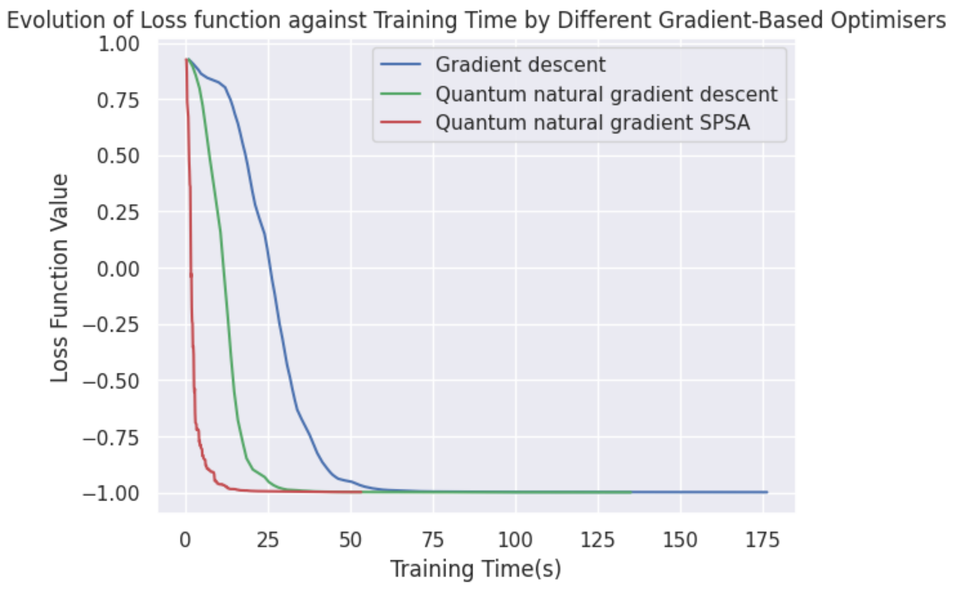


Figure 12: The Loss Function's Evolution against Training Time by Different Gradient-Based Optimisers
(Training time is tracked in terms of seconds and the hyperparameter setting is the same as in Figure 11)

different gradient-based optimisations algorithms. All in all, finding the best optimisation method costs a lot of effort to fine-tune and test. Therefore, it is always encouraged to experiment and try out different optimisation strategies to achieve a better result. For instance, from the above finding, it is motivated to use the quantum natural SPSA in the first training phase and then use quantum natural gradient descent or gradient descent to fine-tune the parameters. Combining the usage of different optimisers can strike a balance between computation time and convergence.

4.2.3 Trouble Shooting Quantum Neural Networks

So far we have discussed much of the gradient-based optimisers for optimising the loss functions associated with quantum neural networks, but have not mentioned the troubles faced when training quantum neural networks. When training classical neural networks, it is possible to encounter the vanishing gradient problem, as there are more layers are added to neural networks where the gradients during backpropagation become very small as they propagate from the output to the input units, causing neural networks very difficult to train. One question to ask is that does this issue also exists in quantum neural network parameters. The author in [23] argues that in some parameterised quantum models, when the number of layers (depth) and qubits (width) grow, the gradients become exponentially small. The implication of this is that the gradients will contain almost no information and make it very hard to update the parameters. This problem is called the Barren Plateau, characterized by a predominantly flat loss function landscape where the direction of gradient descent towards the minimum cannot be determined.

Mitigating barren plateaus efficiently in variational quantum models and quantum neural networks are research problems ongoing. Grant in [15] proposed a heuristic approach for parameter initialisation to address the challenge of the barren plateau problem, which involves random initialisation of certain parameters followed by carefully choosing the remaining parameters. As such, the final circuit becomes a series of shallow unitary blocks where each block is evaluated to the identity. Such a method restricts the effective initial circuit depth so the gradients are not exponentially small at the beginning, allowing the initial circuit to be trained without difficulty. However, this method does not guarantee avoiding the barren plateau problem during training. Another common method, though it is not considered efficient, is to borrow concepts from training classical deep learning which implements greedy layer-wise training of neural networks [4]. This method starts with just optimising a quantum circuit with only one trainable entangling layer and fixing those parameters. Afterwards, we append the second layer, optimising and fixing the parameters in that layer. We continue this process up to the desired layers we want. This strategy can avoid barren plateaus because when we are optimising the model, we are just training a shallow circuit locally. In summary, this provides a baseline for troubleshooting issues in training quantum neural networks.

5 Conclusions

At the culmination of the journey we have taken, we have appreciated quantum neural networks as near-term quantum machine learning applications to combine the subtleties of quantum states with the power of machine learning algorithms. To conclude our quest of exploring quantum neural networks and their training algorithms, we would like to point out several important themes which we have developed.

First of all, a quantum neural network uses a parameterised quantum circuit to transform an initial quantum state to a desired entangled quantum state depending on a data point and parameters. To make predictions from the quantum neural network for machine learning problems, we use measurement to extract statistical properties of the quantum circuit as useful information to predict labels. The measurement is then formulated as a cost function so that we can optimise the parameters in the quantum neural network through a classical feedback loop.

Secondly, if we care about the quantum neural network's performance, the process of data encoding in a quantum circuit cannot be underrated. This is because increasing the repetitions of data encoding gates widens the frequency spectrum of the Fourier series which the quantum neural network learns. Using a variational classifier as an example, we have demonstrated how data re-uploading strategies massively improved the quantum neural network's performance in classifying highly non-linear data sets.

More importantly, we need gradients to train a quantum neural network. But since back-propagation is prohibited when calculating its gradients, we explored a powerful method in quantum gradient estimation known as the parameter shift rule and demonstrated the rationale for preferring it over numerical methods to estimate gradient is that comparing with finite differences, it is a statistically unbiased estimator with much less variance. As such, the parameter shift rule lays the foundation for using a classical optimisation method to train a quantum neural network.

In addition to reviewing commonly used optimisers, we surveyed advanced optimisation methods specifically for training a quantum neural network. A quantum natural gradient algorithm is a second-order optimisation method and is an extension of natural gradient algorithms. Instead of optimising parameters in the parameter space, a quantum natural gradient optimises parameters in the quantum information geometry, involving the use of quantum Fisher information. This results in faster convergence. To reduce overheads in computation, there is an approximated version of the quantum natural gradient, namely Simultaneous Perturbation Stochastic Approximation of the quantum Natural gradient. We have investigated the convergence behaviour of these optimisers, unveiling their strengths and limitations. We also suggested the philosophy of combining optimisation algorithms to make the most of them.

In conclusion, the essence of this survey is to provide a meticulous review of what training a quantum neural network is like so that readers can gain a comprehensive understanding to be prepared for the era of quantum machine learning and quantum computing. Looking forward, with the rapid growth in quantum technology, we will be thrust into the age with accessible quantum computing resources and soon will be able to deploy large quantum neural network models that solve complex problems at speed. Hence, quantum machine learning's future is teeming with countless possibilities and it is undoubtedly exciting to witness such a bright future.

References

[1] E. Aïmeur, G. Brassard, and S. Gambs. Machine learning in a quantum world. In *Advances in Artificial Intelligence: 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Québec City, Québec, Canada, June 7-9, 2006. Proceedings 19*, pages 431–442. Springer, 2006.

[2] Anon. Bloch sphere, Jul 2023.

[3] Anon. Quantum logic gate, Jul 2023.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

[5] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[6] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer, 2010.

[7] L. Bouchard. *Data, Uncertainty and Error Analysis.* UCLA Department of Chemistry and Biochemistry, 2022.

[8] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[9] A. N. Chowdhury, G. H. Low, and N. Wiebe. A variational quantum algorithm for preparing quantum gibbs states. *arXiv preprint arXiv:2002.00055*, 2020.

[10] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[11] P. A. M. Dirac. A new notation for quantum mechanics. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 35, pages 416–418. Cambridge University Press, 1939.

[12] Y. Duan. Quantum natural spsa.

[13] R. P. Feynman. Quantum mechanical computers. *Optics news*, 11(2):11–20, 1985.

[14] J. Gacon, C. Zoufal, G. Carleo, and S. Woerner. Simultaneous perturbation stochastic approximation of the quantum fisher information. *Quantum*, 5:567, 2021.

[15] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[16] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[17] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[18] P. Kaye, R. Laflamme, and M. Mosca. *An introduction to quantum computing.* OUP Oxford, 2006.

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] S. Lloyd and C. Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[21] A. Mari, T. R. Bromley, and N. Killoran. Estimating the gradient and higher-order derivatives on quantum hardware. *Physical Review A*, 103(1):012405, 2021.

[22] J. Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

[23] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

[24] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[25] K. P. Murphy. *Probabilistic machine learning: an introduction.* MIT press, 2022.

[26] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[27] R. Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, 2019.

[28] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys (CSUR)*, 32(3):300–335, 2000.

[29] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[30] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[31] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

[32] M. Schuld and F. Petruccione. *Machine learning with quantum computers.* Springer, 2021.

[33] M. Schuld, R. Sweke, and J. J. Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, 2021.

[34] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[35] J. Stokes, J. Izaac, N. Killoran, and G. Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[36] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin. General parameter-shift rules for quantum gradients. *Quantum*, 6:677, 2022.

Appendices

1. The code is available at Colab

2. Proof of the parameter shift rule as in [36]:

Proof. Assume that there is a general gate $U(x)$:

$$U(x) := e^{i\theta_j G}, \tag{.1}$$

where θ_j is a single parameter of U, and G is a Hermitian matrix. $U(\theta_j)$ is applied to the quantum state $|\psi\rangle$. Given that there is a measurement B acting on $U(\theta_j)|\psi\rangle$, the expectation of the measurement B is thus

$$\begin{aligned} f(\theta_j) &:= \langle\psi| U^\dagger(\theta_j) B U(\theta_j |\psi\rangle \\ &= \sum_{j,k=1}^{d} \overline{\psi_j e^{i\omega_j\theta_j}} b_{jk}\psi_k e^{i\omega_k\theta_j} \\ &= \sum_{\substack{j,k=1\\ j<k}}^{d} \left[\overline{\psi_j} b_{jk}\psi_k e^{i(\omega_k-\omega_j)\theta_j} + \psi_j \overline{b_{jk}\psi_k e^{i(\omega_k-\omega_j)\theta_j}}\right] + \sum_{j=1}^{d} |\psi_j|^2 b_{jj}, \end{aligned} \tag{.2}$$

where $\left\{e^{i\omega_j\theta_j} \mid \omega_j \in \mathbb{R};\ \omega_{i+1} \geqslant \omega_i \forall i, i+1 \in [d]\right\}_{j\in[d]}$ are the eigenvalues of $U(x)$. Letting $r = |\{\omega_j\}_{j\in[d]}|$ be unique eigenvalues of Hermitian matrix G, we define a set of R unique positive differences (or frequencies) $\{\Omega_l\}_{l\in[R]} := \{\omega_k - \omega_j \mid j,k \in [d], \omega_k > \omega_j\}$, where $R \leqslant \binom{r}{2}$ is the maximum number of unique differences. By letting $c_{jk} = \overline{\psi_j} b_{jk}\psi$, we can write the expectation into a truncated Fourier series, i.e.,

$$f(\theta_j) = a_0 + \sum_{l=1}^{R} c_l e^{i\Omega_l\theta_j} + \sum_{l=1}^{R} \overline{c_l} e^{-i\Omega_l\theta_j} = a_0 + \sum_{l=1}^{R} a_l\cos(\Omega_l\theta_j) + b_l\sin(\Omega_l\theta_j), \tag{.3}$$

$$c_l =: \frac{1}{2}(a_l - ib_l)\ \forall l \in [R], \tag{.4}$$

$$a_0 := \sum_{j=1}^{d} |\psi_j|^2 b_{jj}, \tag{.5}$$

with an implicit assumption that a mapping exists between indices j,k and index l s.t. $c_l = c_{l(j,k)}$ is unambiguous. Therefore, as $f(\theta_j)$ is a truncated Fourier series, we can obtain $\{a_l\}$ and $\{b_l\}$ by a discrete Fourier transform when evaluating $f(\theta_j)$.

When G has only two eigenvalues, this means that there is only one frequency $\Omega_1 = \Omega$. $f(\theta_j)$ is then

$$f(\theta_j) = a_0 + a_1\cos(\Omega\theta_j) + b_1\sin(\Omega\theta_j) \tag{.6}$$

and its derivative is

$$f'(\theta_j) = [-a_1\sin(\Omega\theta_j) + b_1\cos(\Omega\theta_j)]\Omega \tag{.7}$$

As such we can derive the parameter shift rule by introducing a shift term $s \in \mathbb{R}$. Consider $f(\theta_j + s)$ and $f(\theta_j - s)$, where

$$\begin{aligned} f(\theta_j + s) &= a_0 + a_1\cos(\Omega\theta_j + \Omega s) + b_1\sin(\Omega\theta_j + \Omega s) \\ &= a_0 + a_1[\cos(\Omega\theta_j)\cos(\Omega s) - \sin(\Omega\theta_j)\sin(\Omega s)] + b_1[\sin(\Omega\theta_j)\cos(\Omega s) + \cos(\Omega\theta_j)\sin(\Omega s)] \end{aligned} \tag{.8}$$

and

$$\begin{aligned}f(\theta_j - s) &= a_0 + a_1\cos(\Omega\theta_j - \Omega s) + b_1\sin(\Omega\theta_j - \Omega s)\\ &= a + a_1[\cos(\Omega\theta_j)\cos(\Omega s) + \sin(\Omega\theta_j)\sin(\Omega s)] + b_1[\sin(\Omega\theta_j)\cos(\Omega s) - \cos(\Omega\theta_j)\sin(\Omega s)].\end{aligned}$$

Consider the difference between the two terms,

$$\begin{aligned}f(\theta_j + s) - f(\theta_j - s) &= a_1[-2\sin(\Omega\theta_j)\sin(\Omega s)] + b_1[2\cos(\Omega\theta_j)\sin(\Omega s)]\\ \frac{f(\theta_j + s) - f(\theta_j - s)}{2\sin(\Omega s)} &= -a_1\sin(\Omega\theta_j) + b_1\cos(\Omega\theta_j).\end{aligned} \tag{.9}$$

Therefore, substituting $\frac{f(\theta_j + s) - f(\theta_j - s)}{2\sin(\Omega s)} = -a_1\sin(\Omega\theta_j) + b_1\cos(\Omega\theta_j)$ into Equation (.7), the parameter shift rule is

$$f'(\theta_j) = \frac{f(\theta_j + s) - f(\theta_j - s)}{2\sin(\Omega s)}\Omega \tag{.10}$$

For Pauli rotation gates, Ω will be one and hence

$$f'(\theta_j) = \frac{f(\theta_j + s) - f(\theta_j - s)}{2\sin(s)}. \tag{.11}$$

□