Useful Quantum Advantage with an Ising Born Machine

Brian Coyle

Master of Science by Research
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh
2018

Abstract

In this work, we propose a generative quantum machine learning algorithm, which we conjecture is not possible to simulate efficiently by any classical means. We call the algorithm the Ising Born Machine as it generates statistics according to the Born rule of Quantum Mechanics and involves several possible quantum circuit classes which all derive from an Ising Hamiltonian. We recall the currently known proofs of the classical hardness to simulate these circuit classes up to multiplicative error, and make some slight modification such that they are compatible and more useful for the Ising Born Machine construction. Further, we invoke the use of *kernel* methods as part of the algorithm, and incorporate a kernel function which also claims to demonstrate a quantum classical result, thereby strengthening our claim. Finally, we also present numerical simulations using Rigetti's Forest quantum simulation platform to investigate and test the ability of the model to learn a simple toy data distribution, with two cost functions, the Kullback-Leibler Divergence and the Maximum Mean Discrepancy (MMD).

Acknowledgements

Firstly, I would like to thank my primary supervisor, Elham Kashefi, for allowing me to undertake this project and providing much help and guidance. Furthermore for hosting me in Paris for several months, through which I made many useful collaborations. I would also like to thank my co-supervisors, Vincent Danos, Tony Kennedy and Ajitha Rajan for many useful discussions and guidance. I would like to thank some of the people who helped me throughout the project; Rawad Mezher, Ellen Derbyshire, Dan Mills and especially Andru Gheorghiu for correcting many of my mistaken assumptions, fielding stupid questions and for a thorough proofreading of the dissertation. I would also like to acknowledge my fellow PPar Cohort, and the Edinburgh and Paris Quantum Computing groups for making the year interesting and enjoyable.

I would like to thank my parents, Gene and Brenda, without whom none of this would have been possible. A special thanks to Kathryn who supported me enthusiastically throughout the entire thing.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. This work was supported in part by the EPSRC Centre for Doctoral Training in Pervasive Parallelism, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L01503X/1), the University of Edinburgh and Entrapping Machines, (grant FA9550-17-1-0055).

(Brian Coyle)

Table of Contents

List of Figures

Chapter 1

Introduction

There has been an explosion of interest and subsequent rapid development in the field of Quantum Computation/Information since its inception. In particular, one sub-field is swiftly emerging as a potential major application of Quantum Computers, in particular using near term quantum devices. This is the new field of Quantum Machine Learning (QML), the attempt to merge the advances and speed-ups that quantum architectures can provide, with the widespread field of Machine Learning. It is an exciting topic because it is relatively unexplored and its potential is as yet untapped. The spark that initiated this field happened in 2009, with the development of the quantum linear equation solver, the HHL algorithm [39], and progress has been rapid since. In particular, the field has the luxury of a wealth of classical machine learning techniques and tools which can be leveraged to design new algorithms which outperform existing classical ones in some aspect, for example in speed or accuracy.

It is exactly this point that we address in this work, with a particular focus on applications which could be run on Noisy Intermediate Scale Quantum (NISQ) Devices [57]. These quantum devices are those which possess around $50 - 100$ qubits, with the capability of applying quantum circuits up to depth[1] ~ 100, in a noisy fashion. We will take the optimistic viewpoint in this dissertation and assume that this NISQ era will be dawning in the very near future. In particular we will focus on the next milestone in the field of Quantum Computing, which is the demonstration of '*quantum computational supremacy*'[2] This refers to the demonstration of some task which could be run much faster on a quantum computer than on any classical computer. The major focus for demonstrating such a thing falls on so-called '*sampling*' problems, or producing samples

[1] The depth of a quantum circuit is the number of layers in it, where each layer takes one time-step.
[2] Also referred to by the less unfortunate titles of quantum superiority or quantum advantage.

from some probability distribution using a quantum computer which could not be correctly produced (efficiently) on a classical one. Currently leading this charge is Google [53], who are focussing on Random Circuit Sampling (RCS). However, the concentration of this work will be on alternative models which are also capable of demonstrating quantum supremacy, such as Instantaneous Quantum Polynomial Time (IQP), [61], or Quantum Approximate Optimization Algorithm (QAOA), [26], circuits. We will broadly refer to as 'Ising' models, since they are inspired by the Ising model in statistical physics. However, these supremacy models are only capable of generating 'random' samples from a probability distribution, where the efficient generation of these samples is the required demonstration of quantum supremacy. While the problem itself is interesting, since it would demonstrate an experimental separation between a quantum computer and any classical one, there is currently *no use* for these samples once they are produced. It is the goal of this work to make these sampling tasks useful for some purpose, specifically for a machine learning application, while retaining the ability to demonstrate quantum supremacy.

In particular we define an algorithm which uses these circuit classes for a generative machine learning model, called an Ising Born Machine (IBM), to learn a probability distribution. This model also uses kernel methods to aid learning, which are well studied in classical ML literature. We conjecture that this algorithm could not be simulated on any classical device in polynomial time, and we claim it therefore provides good motivation for an algorithm which could outperform any classical algorithm in *accuracy* as well as speed, when learning certain probability distributions. We provide numerical results of the implementation of this algorithm using Rigetti's Forest simulation platform.

To begin, Chapter (2) introduces the required preliminaries and discusses models of quantum computation. Chapter (3) discusses the sub-universal models of quantum computation that will define our generative algorithm, and Section (3.3) details the required complexity theoretic arguments to show these models are hard to simulate classically. Chapter (4) discusses generative quantum machine learning models, including the Quantum Boltzmann Machine, and our own Ising Born Machine model, and links our model to the hardness results of Sections (3.3.2 - 3.3.4). Section (4.3) presents the numerical simulations of our model. Finally, Chapter (5) concludes and introduces future research directions and improvements.

1.1 Contributions

The contributions of this work are the following:

- We define one of the first quantum machine learning algorithms which could demonstrate a provable quantum advantage, motivated by compelling complexity theory arguments, and could also be implemented on near-term quantum devices.
- In doing so, we make a connection between the classical hardness of quantum simulation and application in Machine Learning to learn probability distributions.
- We gave a proof of hardness of classical simulation up to multiplicative error for a new class of quantum computations, denoted $\mathsf{IQP}_{\mathrm{y}}$.
- We introduce the novel idea of using a 'quantum-hard' kernel function in the computation of the MMD cost function in our generative model.
- Finally, we test the model using Rigetti's quantum simulator and produce numerical results in Python to investigate the model's functionality.

Chapter 2

Preliminaries

First of all, a brief review of quantum computation notations and terminology will be presented. There are several possible models of quantum computation that are being studied in current research. In some sense, the only requirement for a given model or physical system to be useful for quantum computation is the ability to encode information in some quantum state. The most common and familiar version is that of *discrete variable* (DV) quantum computation, in which the information is encoded in some discrete parameters of the quantum state. This is the most natural approach as we attempt to map our wealth of experience and knowledge from the world of classical computation, into the quantum one. This is due to the fact that the building block in the classical world is the *bit*, a binary valued variable which can take values $\{0, 1\}$. As such, the *qubit* or quantum bit, which can take values (or states) $\{|0\rangle, |1\rangle\}$, was defined to be the quantum analogue of the classical bit. The DV approach is physically realised, for example, when encoding the qubit into energy levels of an ion or atom, and typically an approximation is made such that only a finite amount of these discrete energy levels are available to the computation. While this is the most widespread, and possibly most natural notation, it is not the only possibility. Alternatively, the information could be encoded in some continuous parameters of the quantum state. This is the approach taken in *continuous variable* (CV) quantum computation. In particular, CV quantum computation is useful for its experimental amenability and is naturally suited to photonic systems, for example, in which the phase and amplitude of the quantum photon state carry the information. The CV approach is only mentioned here for completeness, and does not play a further role in this thesis. From this point on, it will be assumed we are dealing only with discrete variable quantum computation.

As mentioned above, the fundamental building block of quantum computation is the

qubit, which is a vector residing in a finite dimensional Hilbert Space, $\mathcal{H}$. A Hilbert space is a complex valued vector space, which possesses an inner product[1]. The typical approach is to work only in a two dimensional Hilbert Space, in which the quantum state, $|\psi\rangle$ is parametrized by two complex numbers, α, β:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

The notation, $|0\rangle, |1\rangle$, maps over from the classical binary valued bits. In fact, the basis spanned by these *computational basis* vectors is the Pauli-Z basis, i.e. these vectors are exactly the eigenvectors of the the Pauli-Z operator. This can be seen if $|0/1\rangle$ are written in vector notation:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.2}$$

with the Pauli-Z operator given by:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.3}$$

The eigenvalues of this operator are $+1$ for the eigenvector $|0\rangle$ and -1 for $|1\rangle$. The Pauli-X and Pauli-Y operators are defined similarly:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{2.4}$$

with eigenvectors $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, $|\pm i\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm i\,|1\rangle)$ of the X, Y operators respectively. Together, these three operators, (2.3, 2.4), along with the identity form a basis of the Lie Algebra of the unitary group of dimension two, $\mathcal{U}(2)$. They can also be viewed in the extremely useful *Bloch Sphere* picture, illustrated by Figure (2.1)[2]. The eigenvalues of Z, $|0/1\rangle$, sit on the North and South poles of the sphere, while the eigenvalues of X, Y lie orthogonally on the equator.

To preserve probabilities, the quantum state defined above, (2.1), must be normalised to 1. This enforces the condition:

$$\langle\psi|\psi\rangle = [\alpha^* \langle 0| + \beta^* \langle 1|]\,[\alpha\,|0\rangle + \beta\,|1\rangle] = |\alpha|^2 + |\beta|^2 = 1$$

The above state, (2.1), is an example of a *pure* state, and represents the state of maximal knowledge about the system. A more general form of '*mixed*' quantum states

[1] A more formal definition is given in Section (4.2.2.1).

[2] www.clipart-finder.com/clipart/QubitBlochSphere677.html

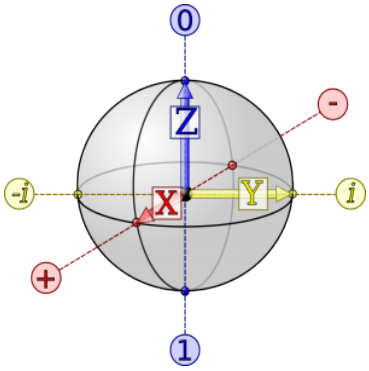


Figure 2.1: Illustration of the eigenvalues of X, Y, Z on the Bloch Sphere.

can be written as so-called 'density matrices':

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{2.5}$$

The system can be in one of the pure states, $|\psi_i\rangle$, with a probability, p_i, and these probabilities represent statistical uncertainty about the system since it is a classical mixture of the possible pure states.

Finally, an important property of general quantum states is that they can exhibit *entanglement*. This is a form of non-local correlation between systems of more than one qubit, that Einstein famously referred to as "*spooky action at a distance*", or the apparent ability for one quantum system to affect the properties of another, even when separated by an arbitrarily large distance. A quantum state is said to be entangled if it is not *separable*, i.e. it cannot be written as a separable tensor product[3] over the subsystems of the qubits. For example, with two qubits the following state, (2.6), is separable:

$$|\psi\rangle_{12} = |0\rangle_1 \otimes |0\rangle_2 = |\phi\rangle_1 \otimes |\chi\rangle_2 \tag{2.6}$$

Whereas the state, (2.7), is entangled[4]:

$$|\psi\rangle_{12} = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right) \tag{2.7}$$

2.1 Models of Quantum Computation

There are three major equivalent models of quantum computation which are studied. These are referred to as:

[3] A tensor product between vector spaces, X, Y is $X \otimes Y$ which is itself a tensor product. It is a mathematical operator that generalizes the outer product.

[4] This is one of the canonical so-called Bell states. In fact it is actually *maximally* entangled.

(1) Circuit Model of Quantum Computation/Quantum Gate Model,

(2) Measurement Based Quantum Computation (MBQC),

(3) Adiabatic Quantum Computation (AQC).

2.1.1 Quantum Gate Model

The most widespread model is called the Circuit Model or Quantum Gate Model due to its analogues to classical circuit based computing. A *Quantum Gate* is defined by a unitary operation[5], U, acting on an n-qubit register. It turns out that *any* unitary matrix from the unitary group, $\mathcal{U}(n)$, is a valid quantum gate. This defines a key feature of quantum circuits; since their operations are unitary, the circuit must be reversible. This requirement arises from the need to conserve the norm of the state throughout the quantum operation, and hence conserve probabilities. For example if a gate, U, acts on the quantum state, $|\psi\rangle$:

$$|\Phi\rangle = U|\psi\rangle \qquad \implies \langle\Phi|\Phi\rangle = \langle\psi|U^\dagger U|\psi\rangle = \langle\psi|\psi\rangle = 1$$

The other fundamental ingredient of a quantum circuit is a quantum measurement. A quantum measurement is defined by a set of m *measurement operators*, $\{M_m\}$. A measurement of a quantum state, $|\psi\rangle$, with outcome m, is equivalent to the operator, M_m, acting on the state. The probability of outcome m occurring is given by the Born rule of quantum mechanics:

$$p(m) = |\langle\psi|M_m^\dagger M_m|\psi\rangle|^2 = \mathrm{Tr}\big(M_m^\dagger M_m|\psi\rangle\langle\psi|\big) = \mathrm{Tr}\big(M_m^\dagger M_m \rho\big) \tag{2.8}$$

The last equality is the more general form, and holds for mixed states also. The state after the measurement given by:

$$|\psi\rangle_m = \frac{M_m|\psi\rangle}{\sqrt{p(m)}} = \frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}} \qquad \rho_m = \frac{M_m \rho M_m^\dagger}{\mathrm{Tr}\Big(M_m^\dagger M_m \rho\Big)} \tag{2.9}$$

The operators satisfy a completeness relation, $\sum_m M_m^\dagger M_m = \mathbb{1}$. A specific form of measurement are *projective* measurements, defined by projection operators, P_m. These projectors are in turn defined by an *observable*, M, where M is an hermitian operator:

$$M = \sum_m m P_m \tag{2.10}$$

[5] An operator, U is *unitary* if $U^\dagger U = \mathbb{1}$

An operator, O, is Hermitian if $O^\dagger = O$, where $\dagger$ represents the complex conjugate transpose. P_m are projectors on the subspace of M, corresponding to the eigenvalue m which satisfy $\sum_m P_m = \mathbb{1}, P_m P_{m'} = \delta_{m,m'} P_m$. For example, the Pauli-Z observable, Z, is defined by two projectors which in turn are defined by the eigenvectors of the Z matrix:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = +1 \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} - 1 \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = +1P_0 - 1P_1$$

$P_0 = |0\rangle \langle 0|, P_1 = |1\rangle \langle 1|$, and ± 1 is the eigenvalue of the $|0/1\rangle$ eigenvector respectively.

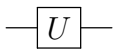
The expectation value of a given operator, M, with respect to a state, $|\psi\rangle$ is:

$$\mathbb{E}(M) = \langle \psi | M | \psi \rangle \tag{2.11}$$

The expectation value gives the average outcome which will be received by a measurement of the observable, M.

It is possible to build circuit diagrams, which will describe a given quantum operation in the gate model, out of the following ingredients:

- Input qubits: n qubits prepared in the standard basis, $|0\rangle^{\otimes n}$[6].
- Quantum Gates, defined by unitary matrices acting on some subset of the qubits:

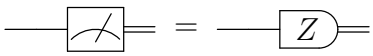


(2.12)

- Quantum/classical Wires, representing the flow of quantum (qubits) or classical (bits) information respectively:

(2.13)

- Measurements in the computational basis:



(2.14)

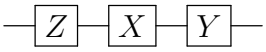
All circuit diagrams in this Dissertation were created using the Q-circuit package in LaTeX [1]. It is standard to assume that the preparations and measurements are in the computational/standard basis, since a preparation (measurement) in any other basis can be obtained through a single qubit rotation unitary into that basis. For instance, to prepare a qubit in the Hadamard/Pauli-X basis ($|\pm\rangle$), one would prepare a single qubit

[6]The tensor product notation is standard: $|\cdot\rangle^{\otimes n} = \underbrace{|\cdot\rangle \otimes \cdots \otimes |\cdot\rangle}_{\text{n times}}$.

in the computational basis, $|0\rangle$, and then apply the single qubit Hadamard gate, H, given by (2.17).

Of particular usefulness are the following unitary gates which will be used throughout this dissertation:

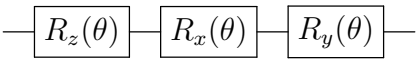
(1) The Pauli Z, X, Y single qubit gates:



(2.15)

where the matrices representing Z, X, Y are given by (2.3, 2.4).

(2) More generally, single qubit rotation gates around the Pauli-X, Y, Z axes on the Bloch Sphere respectively:



(2.16)

where

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z}, \qquad R_x(\theta) = e^{-i\frac{\theta}{2}X}, \qquad R_y(\theta) = e^{-i\frac{\theta}{2}Y}$$

where a single qubit rotation around an axis, $\hat{n}$ ($\hat{n} = (0,0,1) \implies Z$, for example) is given by:

$$R_{\hat{n}}(\theta) = e^{-i\frac{\theta}{2}\hat{n}\cdot\sigma} = \cos(\theta/2)\mathbb{1} - i\sin(\theta/2)(n_z Z + n_y Y + n_x X)$$

where $\sigma = \{Z, X, Y\}$

(3) The Hadamard, H, and $\tilde{H}$ gates:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \tilde{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \tag{2.17}$$

Simple calculation shows $H = \frac{1}{\sqrt{i}}XY^{\frac{1}{2}} = \frac{1}{\sqrt{i}}XR_y(\frac{\pi}{2})$, which will be of use later on.

Any single qubit unitary gate can be decomposed into these rotation gates. However, to create computations which are universal for quantum computation[7], we also need two-qubit gates. It turns out that this is actually sufficient for universality, because any n qubit unitary operation can be decomposed into a sequence of gates which act on at most subsets of two qubits, i.e. single and two-qubit gates.

[7] Universality is the ability to apply any n-qubit unitary operation, more detail will be given in Section (3)

In particular, two of the most important two-qubit gates, are *controlled* versions of the Pauli-Z and Pauli-X gates, called the Controlled-Z (CZ) and Controlled-NOT ($CNOT$) gates respectively:

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \qquad CNOT = CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.18}$$

In general, the order of the qubits on which these gates act is important. The first qubit argument is referred to as the *control* qubit, while the other is the target. This means the gate only has an effect on the target qubit if the control is in the $|1\rangle$ state.

Both of these gates are important as they create entanglement between two qubits in some states. For example, when CZ acts on two qubits initially each prepared in the Hadamard basis, $|\pm\rangle$. the result is the following:

$$CZ_{1,2}(|+\rangle \otimes |+\rangle) = CZ_{1,2}\frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right) \tag{2.19}$$

$$= \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle - |11\rangle\right) \tag{2.20}$$

The subscript $\{i, j\}$ indicates that the gate is acting on qubit i as the control qubit, and qubit j as the target qubit. However, since the Control-Z is symmetric in its arguments, as seen in (2.21), it does not matter which qubit is labelled as the control/target.

$$CZ_{i,j}\,|i\rangle\,|j\rangle = (-1)^{ij}\,|i\rangle\,|j\rangle = CZ_{j,i}\,|i\rangle\,|j\rangle \tag{2.21}$$

A generalisation of the CZ gate which will be of critical importance to this work is the Control-Phase ($CPHASE$) or $CZ(\alpha)$, which simply applies a general phase, θ, on the target qubit, conditional on the control:

$$CZ(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{pmatrix} \tag{2.22}$$

This becomes the standard CZ for $\alpha = \pi$.

2.1.2 Measurement Based Quantum Computation (MBQC)

An alternative model of quantum computation is called Measurement Based Quantum Computation (MBQC) or One-Way Quantum Computation, and was first defined by

[58]. This model is driven not by unitary quantum gates, but instead by preparations of qubits, and measurements in a particular basis (hence the name). By adaptively correcting for measurement outcomes in a feedforward manner, the model becomes universal for quantum computation, and equivalent to the circuit model. This is because measurement outcomes are generically random, so to *deterministically* simulate any gate in the circuit model, the outcomes must be corrected for by adapting future measurements. In the standard form, MBQC requires the creation of *graph states*, or networks of qubits described by a graph, the nodes of which are the qubits, and the edges are given by entangling links.

Typically, the nodes of the graph will be prepared in the $|+\rangle$ state, with the entanglement created by CZ gates. For example, a $1D$ graph state with two qubits is described by the state, (2.20). CZ is typically used as the standard entangling operation because of its symmetric action on qubits, as seen above, whereas in contrast, the $CNOT$ gate is *not* symmetric.

The main ingredient involved in MBQC are measurements. The measurements can be done in any single qubit basis, however they are typically assumed to be only in the Pauli $X-Y$ plane of the Bloch sphere. A measurement in this basis is defined by following operators:

$$M_{\pm\theta} = |\pm_\theta\rangle\langle\pm_\theta|, \qquad |\pm_\theta\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle \pm e^{\pm i\theta}|1\rangle\right) \tag{2.23}$$

It is possible to simulate a Z basis measurement by preparing the qubits in the computational basis, and measuring in the above basis, (2.23) due to the following symmetry:

$$\langle s|\pm_\theta\rangle = \langle s|Z(\theta)|\pm\rangle = e^{i\theta s}\langle s|H|0/1\rangle = \langle\pm_s|0/1\rangle \tag{2.24}$$

so these operations are equivalent up to an unimportant global phase.

In fact, this is flexibility is a crucial ingredient in a particular type of quantum verification, known as FK-verification [30] and universal blind quantum computation (UBQC) [19] which revolves around a client preparing single qubit states, and sending them to a (potentially malicious) quantum-powerful server to carry out some quantum computation. The special structure of MBQC allows the server to randomize the qubit preparation so the server is effectively blind to the actual computation, or even the real input or output. In essence, using this technique, the server is essentially just a computation tool at the whim of the client, who can detect if the server tries to cheat and implement some incorrect computation for its own benefit.

The final ingredient to make MBQC universal is to implement corrections to the system based on the outcomes of the probabilistic measurements, as mentioned above. This allows for a deterministic computation to be applied. However, the work of this dissertation is precisely examining the case where corrections are *not* applied, they will not be discussed in further detail. Specifically, since the outcomes of the measurements on the quantum system are fundamentally probabilistic in nature, they generate statistics according to some probability distribution, which is a feature that will be exploited later for generative machine learning tasks.

2.1.3 Adiabatic Quantum Computation

Finally, the last model of quantum computation we discuss is the adiabatic model (AQC), which is somewhat analogous to analogue classical computing, which was first defined by [27]. In this model, the information is encoded in *Hamiltonians*[8] of quantum systems, and is centred around the adiabatic theorem, [13]. This method involves initially preparing a ground state of a Hamiltonian of a system which is simple to prepare and adiabatically evolving the system to the ground state of a Hamiltonian which is complicated to prepare. Ideally, the solution to a desired problem will be encoded in this final state.

Theorem 2.1: Adiabatic Theorem

A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.

The minimal spectral gap (if it exists) of an instantaneous Hamiltonian, $H(t)$, is the difference between its smallest, and second smallest eigenvalues. This Hamiltonian is designed such that at time $t = 0$, it is exactly the easy to prepare Hamiltonian, and at time $t = T$, it becomes the Hamiltonian which encodes the problem solution. This gap is exactly what stops adiabatic quantum computation from being able to solve much more difficult problems than the circuit model for example, as it is known that the running time of an adiabatic computation is proportional to the inverse gap (of the instantaneous Hamiltonian, $H(t)$) squared [27, 69]:

$$T = O\left(\frac{1}{gap^2}\right) \tag{2.25}$$

[8] Hamiltonians are Hermitian operators that describe the energy of a system.

Intuitively, this implies that problems which are hard to solve can only be encoded into Hamiltonians which have an exponentially small gap, meaning that the runtime becomes exponential. This is because it may take exponential time to pass through the minimal gap *without* a transition to a higher energy level, which would not necessarily give a solution to the problem at hand.

Chapter 3

(Sub) Universal Models of Quantum Computation

All of the above models of quantum computation have the capability to demonstrate *universal* quantum computation, in the sense that they are equivalent to a Quantum Turing Machine, [23], the quantum analogue of the classical Turing machine. However, it is more common nowadays to talk about universality in terms of the circuit model. A set of quantum gates is said to be universal for quantum computing if it is possible to express any unitary in terms of a sequence of these gates. However, since there is an uncountable number of possible unitaries in the unitary group of dimension n, $\mathcal{U}(n)$, not all unitaries could be represented exactly by a finite set of gates. With this in mind, it is sufficient to define universality by the ability to *approximate* any unitary to an arbitrary accuracy.

It turns out this is possible simply with single and two-qubit gates, i.e. by decomposing any unitary, U in $\mathcal{U}(n)$ into a sequence of gates over $\mathcal{U}(1)$ and $\mathcal{U}(2)$ which act only on subsets of one, or two qubits respectively, [55]. Furthermore, it is sufficient to further restrict the two-qubit unitary to be only the $CNOT$ gate, (2.18). Any single qubit gate can be expressed, to an arbitrary accuracy, by the following gate set:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix} \tag{3.1}$$

The T gate is also sometimes referred to as the $\pi/8$ gate for historic reasons, and the P gate is typically called the phase gate. It should be noted that the latter two gates are both diagonal in the computational basis and can be written in the form $R_z(\theta)$ for $\theta = -\pi, \pi/2$ respectively. Furthermore, one can swap out the $CNOT$ gate for the CZ

since these two gates are equivalent up to local H gates acting on the target qubit.

$$CNOT = |1\rangle\langle 1| \otimes X \tag{3.2}$$

$$\implies (\mathbb{1} \otimes H)CNOT(\mathbb{1} \otimes H) = (\mathbb{1} \otimes H)\,|1\rangle\langle 1| \otimes X(\mathbb{1} \otimes H)$$

$$= |1\rangle\langle 1| \otimes HXH = |1\rangle\langle 1| \otimes Z = CZ \tag{3.3}$$

Using the commutation relation, $HX = ZH$. The CZ will be more relevant later in this work so we will use it instead from this point on. An alternative finite set of universal gates is given by $\{H, CNOT, P, \text{Toffoli}\}$, where the Toffoli is exactly the $CNOT$ generalized to three qubits, i.e. the third qubit is flipped if and only if the first two are both in the $|1\rangle$ state.

3.1 Instantaneous Quantum Polynomial Time Computation

In terms of complexity theory, the class BQP is the class of decision problems we expect to be solvable by a quantum computer in polynomial time and polynomial resources (i.e. efficiently) in the input to the problem, informally speaking.

Definition 3.1: BQP [55]

A language, $\mathcal{L}$, is in BQP iff there exists a uniform family of quantum circuits, $\{C_n, n \in \mathbb{N}\}$[a], such that:

(1) $\forall n \in \mathbb{N}, C_n$ takes n qubits as input and outputs one bit.

(2) $\forall x \in \mathcal{L}, \qquad Prob(C_{|x|}(x) = 1) \geq \frac{2}{3}$

(3) $\forall x \notin \mathcal{L}, \qquad Prob(C_{|x|}(x) = 0) \geq \frac{2}{3}$

[a] A uniform family of circuits is a map $w \to C_w$ which is computable in polynomial time, where C_w is a classical description of the circuit that takes input of bitstrings $w = x_1 \dots x_n$ of length n and includes the input, output and circuit compositions.

Due to the probabilistic nature of quantum mechanics, it is the most natural generalisation of the classical class BPP; the class of decision problems solvable by a probabilistic classical Turing machine in polynomial time with bounded error. This definition intuitively implies that a BQP circuit can *decide* if a given input has a particular property, or not, with a probability bounded away from one half by at least a constant, i.e. better than just randomly guessing.

However, it is also possible to define several complexity classes relating to *sub-universal* models of quantum computation. These classes are not thought powerful enough to solve every problem in BQP, but are still considered outside what is possible classically. The first example of a sub-universal class, and one which will play a major role in this dissertation is the class, IQP, which stands for *Instantaneous Quantum Polynomial* time, [61]. This class of computation was originally defined in terms of X-Programs, i.e. gates diagonal in the Pauli-X basis operating on computational basis states[1]. The definition is derived by restricting the uniform circuit families in the above definition of BQP, to be composed only of the gates diagonal in the X basis. However, it is more useful to this dissertation (and indeed more commonplace) to describe them in terms of gates diagonal in the *computational basis*, acting on Hadamard states. Specifically an IQP computation consists of 3 steps:

(1) Hadamard basis preparation, i.e preparing n qubits initially in the state $|+\rangle^{\otimes n}$:

(2) Unitary evolution with m operators described in the computational basis:

$$U(\theta_j, S_j) = \exp\left(i \sum_{S_j \subseteq [n]} \theta_j \bigotimes_{i \in S_j} Z_i\right) \tag{3.4}$$

(3) Followed finally by measurements in the Hadamard basis: $|\pm_{\mathbf{s}}\rangle \langle \pm_{\mathbf{s}}|$ with outcome string $\mathbf{s}$.

This class of computation is called *instantaneous* because of the commuting nature of the intermediate gates in step (2), which can be applied in any order. S_j in step (2) above is the subset of the total n qubits on which each gate, $U(\theta_j, S_j)$, acts. In all the following, we assume that $|S_j| \leq 2 \forall j$, i.e. the computation consists of gates acting on at most subsets of two qubits. This is sufficient for universality as described in the previous section. It is important to note at this point that with this restriction, the term in the exponent of (3.4) can be written as an Ising Model Hamiltonian[2]:

$$i \sum_{S \subseteq [n]} \theta \bigotimes_{i \in S} Z_i = iH_z = i \sum_{i<j} \theta^2_{ij} Z_i Z_j + i \sum_{k=1}^{n} \theta^1_k Z_k \tag{3.5}$$

[1]When we say an operator, A, is 'diagonal in a particular basis', this means the matrix representing that operator can be completely diagonalized in terms of the particular eigenbasis. Specifically this means there exists operators, K, such that $K^{-1}AK$ is diagonal. K will contain the eigenvectors of A as it columns.

[2]The Ising model of statistical physics is a model of ferromagnetic spins and is described in terms of coupling terms, represented by the Z_iZ_j term in (3.5), between neighbouring spins, and local magnetic fields, given by the Z_k term.

The parameters, $\theta = \{\theta^2_{ij}, \theta^1_k\}$, can be viewed as the coupling and local magnetic fields respectively. The evolved state is then:

$$\left|\psi^{\theta}_f\right\rangle = \prod_{j=1}^{m} U_j(\theta_j, S_j) \left|+\right\rangle^{\otimes n} \tag{3.6}$$

Importantly, the resulting output probability distribution which can be derived from this class of computations can be written as follows using the Born rule:

$$P(\mathbf{s}) = \left| \langle \pm_{\mathbf{s}}| \prod_{j=1}^{m} U_j(\theta_j, S_j) \left|+\right\rangle^{\otimes n} \right| \tag{3.7}$$

where, for notational purposes $|\pm_{\mathbf{s}}\rangle = \bigotimes_{i=1}^{n} |\pm_{s_i}\rangle$. $|\pm_{s_i=\pm 1}\rangle = 1/\sqrt{2}(|0\rangle + (-1)^{s_i}|1\rangle)$, so for $s_i = 1$, the outcome of measuring qubit i, is $+1$, with the post-measurement state, $|+\rangle$, and for $s_i = -1$, the result is $|-\rangle$. The initial state has been acted on by m gates, all of which are diagonal in the computational basis. As mentioned above, these types of computations were originally referred to as X-Programs since they consisted of initial state preparation in the computational basis, diagonal gates in the Pauli-X basis, and measurements in the computational basis. The resulting computation is equivalent due to the following fact:

$$\langle +|^{\otimes n} \exp\left(i \sum_{S\subseteq[n]} \theta \bigotimes_{i\in S} Z_i \right) |+\rangle^{\otimes n} = \langle 0|^{\otimes n} H^{\otimes n} \exp\left(i \sum_{S\subseteq[n]} \theta \bigotimes_{i\in S} Z_i \right) H^{\otimes n} |0\rangle^{\otimes n}$$

$$= \langle 0|^{\otimes n} \exp\left(i \sum_{S\subseteq[n]} \theta \bigotimes_{i\in S} HZ_iH \right) |0\rangle^{\otimes n} = \langle 0|^{\otimes n} \exp\left(i \sum_{S\subseteq[n]} \theta \bigotimes_{i\in S} X_i \right) |0\rangle^{\otimes n}$$

using the commutation relation, $ZH = HX$, and $H^2 = \mathbb{1}$.

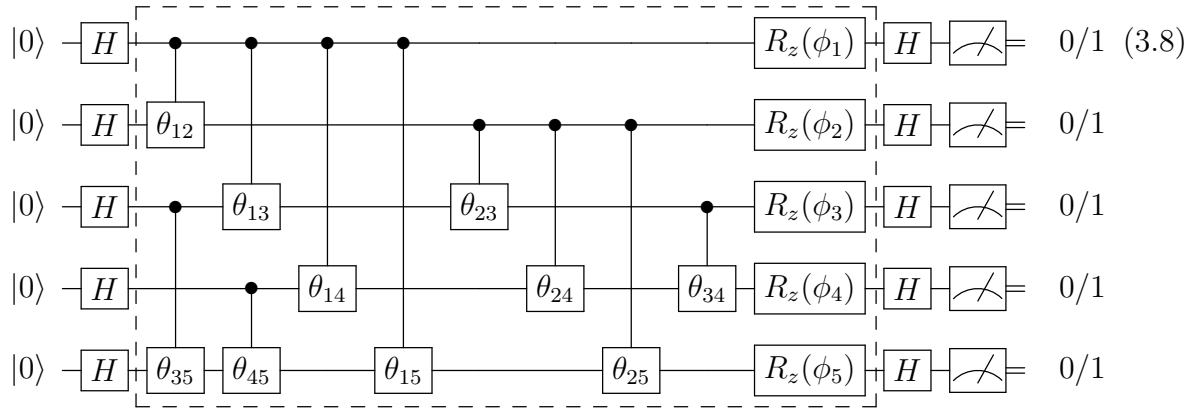
The attempts in the literature to define a class of computation which demonstrates *Quantum Computational Supremacy* are often satisfied with a single generating parameter, $\theta = \frac{\pi}{8}$, for all unitary operations[3].

However, it should be somewhat obvious that to define a machine learning algorithm from these types of computations, which is our goal, we require the model to be as flexible as possible in terms of its parameters, i.e. not fixed. As such, we must investigate whether it is possible to extend the set of values for the parameters, $\theta^2_{ij}, \theta^1_k$, and investigate for which parameter values the system still demonstrates supremacy.

[3]The reason for this, as we shall see in Section (3.3.2), is that taking multiple repetitions of gates parametrized with this angle, one can recover most gates in a set which are required for universality.

3.1.1 IQP circuit

The most general form of an IQP[4] circuit is given by (3.8):



(3.8)

where $\theta_{ij} \equiv R_z(\theta_{ij})$. Any single qubit gate which is diagonal in the computational basis can be absorbed into the gates $R_z(\phi_i)$, resulting in a new gate, $R_z(\phi_i')$, with a shifted angle, ϕ_i'. The section of the circuit in the dashed box is the 'Ising' part of the circuit, which begins and ends with a layer of Hadamard gates. Depending on the specific circuit layout, the contents of this box will change, i.e. how many qubits are entangled with each other, what single qubit rotations are used etc.. It should be noted that since the gates in the dashed box all commute with each other (which is the defining quality of IQP circuits), they can be applied in any order, but the convention is usually that the two-qubit entangling gates are applied first, followed by the single qubit gates. In the above, (3.8), we have effectively an all-to-all connected graph, where the connections (graph weights) are determined by the $CZ(\theta_{ij})$ entangling gates. Unfortunately, this high level of connectivity is not typically not achievable on near term quantum devices, a more common layout uses simply *nearest neighbour* connectivity.

3.2 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) has been defined relatively recently to solve combinatorial optimization, or constraint satisfaction problems (CSP), [26]. This algorithm essentially uses an approach to simulate a non-commuting Hamiltonian on a gate based quantum computing model. In the initial definition, the algorithm was used as a candidate to solve CSP problems such as MaxCut, and find the maximum of some objective function which counts the number of clauses which are satisfied by some assignments of bits to a set of variables. Conveniently, one can think of the situation as

[4]From this point on, when we refer to IQP, we mean this layout with at most two-qubit operations.

a graph, which takes binary values, $z_i \in \{0, 1\}$, on the vertices, and the problem asks how can these binary values be assigned such that the maximum number of constraints are satisfied. For example, a constraint could be: $(z_1) \wedge (z_2) \wedge (z_1 \vee z_2)$, which is satisfied by $z_1 = 1, z_2 = 0$ for example. The problem MaxCut asks for a configuration of the bits for which the objective function (which is given by the number of satisfied constraints) is maximized, whereas the approximate problem looks for a solution assignment for which the objective function is *close* to maximum.

However, we are not interested in this algorithm to solve MaxCut, instead we want to exploit the circuit it implements for machine learning purposes. This is motivated by the recent work, [68], which uses the QAOA as a subroutine to approximately prepare the Gibbs state of a Quantum Boltzmann Machine, defined by [8]. This will be discussed in more detail in Section (4.1.2). Our motivation for choosing this type of circuit is the same as for IQP, both circuits potentially demonstrate quantum computational supremacy as we shall see shortly.

The QAOA is defined in terms of a 'cost' Hamiltonian, $\mathcal{H}_z$, and a 'mixer' Hamiltonian, $\mathcal{H}_x$ (borrowing the terminology of [68]). The mixer Hamiltonian is assumed to be one which has an easily prepared ground state (typically a product state), for example (3.9):

$$\mathcal{H}_x = \sum_{i=1}^{n} X_i \tag{3.9}$$

Since this Hamiltonian acts only on each qubit, i, individually, it can be easily seen that an eigenvector of the entire n qubit system can be decomposed into eigenvectors of each qubit. For example, an eigenvector of one term in the sum above is $|+\rangle$, so we can write one possible eigenvector of the system as $|+\rangle \otimes \cdots \otimes |+\rangle = |s\rangle = 2^{-n/2} \sum_{z \in \{0,1\}^n} |z\rangle$ for example. This state is also be the ground state of the mixer Hamiltonian.

As mentioned above, in the QAOA (and indeed even in the Adiabatic Algorithm) the CSP is encoded into a target or cost Hamiltonian, which has the following form:

$$\mathcal{H}_z = C(z) = \sum_{a=1}^{m} C_a(z) \qquad C_a(z) = \begin{cases} 1 \text{ if } z \text{ satisfies constraint } a \\ 0 \text{ if } z \text{ does not.} \end{cases} \tag{3.10}$$

This function is the sum of all constraints, a, which are satisfied. In other words, minimising the 'Hamiltonian' energy when translating the problem to a quantum architecture, is equivalent to find the maximum number of constraints which are satisfied.

Just like the IQP circuit, if we were to interpret the problem in its most general form, it would contain constraints like $z_1 \vee z_2 \vee z_3$ for example[5]. This equates to a three-

[5]This would be satisfied by a configuration s.t $z_1 z_2 z_3 \neq 000$

(qu)bit operation, or in other words, a quantum gate acting on three qubits. However, we will see in the IQP case that we only need to implement single and two-qubit gates for universal quantum computation. The same conveniently holds for the QAOA. As such, we will restrict to the case where we only have at most two-bit constraints so the 'cost Hamiltonian' again has the form of an Ising one:

$$\mathcal{H}_z = -\sum_{i<j} J_{ij} Z_i Z_j - \sum_k b_k Z_k \tag{3.11}$$

In general, the QAOA depends on a parameter, $p \geq 1$ which effectively controls the depth of the resulting circuit. It has been shown, [28], that the output distribution of even the shallowest depth version of the algorithm, $p = 1$, cannot be sampled from classically, and hence provides a good candidate to demonstrate quantum supremacy. This includes circuits of the form (3.13), and a depth p-QAOA has p layers of these same gate sets acting in an alternating fashion, i.e. it produces a state:

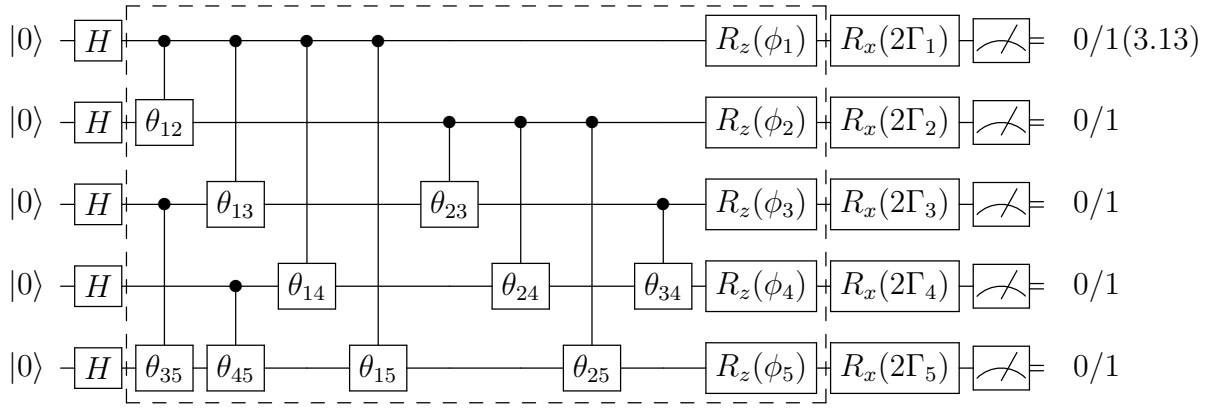
$$|\gamma, \beta\rangle = \prod_{i=1}^{p} \left(e^{-i\beta_i \mathcal{H}_x} e^{-i\gamma_i \mathcal{H}_z}\right)^p |s\rangle \tag{3.12}$$

The $2p$ parameters, $\gamma = \{\gamma_1, \ldots, \gamma_p\}, \beta = \{\beta_1, \ldots, \beta_p\}$ are optimized in the QAOA to produce a ground (or thermal) state of the target cost Hamiltonian, which is assumed to be difficult to prepare directly. Notice, this method of 'switching' from one Hamiltonian to the other is very similar to the Quantum Adiabatic Algorithm (or Quantum Annealing) and is a discrete method to approximately simulate the evolution of the system, as an alternative to the 'analog' methods of quantum annealing or adiabatic quantum computation. As $p \to \infty$, the state produced at the end of the algorithm is the exact state of the cost Hamiltonian which is required.

3.2.1 QAOA circuit

The general circuit for the subclass of QAOA that we are dealing with is exactly the same as the IQP circuit, (3.8) with the alteration of having rotations around the Pauli-X axis,

U_f^X, in place of the final layer of Hadamard gates.



A notable difference between IQP and QAOA is the addition of extra parameters, $\mathbf{\Gamma} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$. These additional gates indicates a potential advantage of the QAOA over an IQP circuit for a machine learning algorithm, simply due to the introduction of n extra trainable parameters. However, in the rest of this work, we will assume for simplicity these extra parameters are set as constant, $\Gamma_i = \frac{\pi}{4}, \forall i$.

3.3 Hardness of Classical Simulation

In this section, we will provide the various results which indicate that neither circuit above, IQP nor QAOA, can be classical simulated up a multiplicative error, in the limit as the size of the input, $n \to \infty$. Before this can be shown however, we must define exactly what we mean by classical simulation and furthermore the types of error we examine. There are typically two types of simulation one can consider. The first is the notion of strong simulation:

Definition 3.2: Strong Classical Simulation [16, 32]

A quantum circuit family is strongly simulable if, given a classical description of the circuit, any output probability, $p(\mathbf{x})$, *and* any marginal probability, $\sum_{\mathbf{x}'} p(\mathbf{x})$, can be computed to m digits of precision in $poly(n, m)$ time, where n is the size of the input to the circuit.

However, for our purposes, the more relevant notion is instead that of *weak simulation*:

Definition 3.3: Weak Classical Simulation

A quantum circuit family is weakly simulable if, given a classical description of the circuit, a classical algorithm can produce samples, $\mathbf{x}$, from the output distribution, $p(\mathbf{x})$, in $poly(n)$ time.

As mentioned in [16], strong simulation implies weak simulation, and it is this weak simulability which we want to rule out as being classically hard, i.e. that there exists no classical probabilistic algorithm which can sample from the output distribution of the classes of sub-universal circuits that we consider. One could ask, however, "What if we do not care about getting samples from the *exact* distribution, and instead an approximation is good enough?". This is a very important and relevant question to ask when discussing quantum supremacy, since it could be the case that even quantum computers cannot produce the exact dynamics that they are supposed to, due to the presence of noise. Noise typically results in decoherence and the destruction of the entanglement and interference properties of the quantum circuit, so in the presence of noise the resulting output distribution could become classically simulable[6]. This is a very hot topic in the field at the minute, given the approaching dawn of the quantum supremacy era, and as such we want to have strong (theoretical) guarantees about the experiments which claim to demonstrate supremacy, even in the presence of reasonable noise. The two major notions of simulation error that are considered are, [16]:

- Multiplicative Error: A circuit family is weakly simulable to multiplicative (relative) error, if there exists a classical probabilistic algorithm, Q, which produces samples, $\mathbf{x}$, according to the distribution, $q(\mathbf{x})$, in time which is polynomial in the input size, such that it differs from the ideal quantum distribution, $p(\mathbf{x})$, by a multiplicative constant, $c < 1$:

$$\frac{1}{c}p(\mathbf{x}) \leq q(\mathbf{x}) \leq cp(\mathbf{x}) \qquad \forall \mathbf{x} \tag{3.14}$$

- Variation Distance Error: A circuit family is weakly simulable to variation distance error, ϵ, if there exists a classical probabilistic algorithm, Q, which produces samples, $\mathbf{x}$, according to the distribution, $q(\mathbf{x})$, in time which is polynomial in the input size, such that it differs from the ideal quantum distribution, $p(\mathbf{x})$ in variation

[6] Indeed this has been shown to be the case for IQP in [18] even for simple depolarising noise before each qubit measurement.

distance, ϵ:

$$\sum_{\mathbf{x}} |p(\mathbf{x}) - q(\mathbf{x})| \leq \epsilon \tag{3.15}$$

As noted in [31], an alternative (almost equivalent) definition of weak multiplicative error sampling is defined in [66]:

$$|p(\mathbf{x}) - q(\mathbf{x})| \leq \gamma p(\mathbf{x}), \qquad \forall \mathbf{x}, \qquad \gamma < 1 \tag{3.16}$$

From this definition, it can be clearly seen that hardness of sampling to within multiplicative error is, in some sense, weaker than sampling to total variation distance error. The former implies the latter, but the converse is not necessarily true. Taking a sum over all possible samples, $\mathbf{x}$, in (3.16), and setting $\gamma = \epsilon$ gives the result of (3.15) immediately. Intuitively, multiplicative error sampling is 'harder' since it must hold *for all* the samples, i.e. the classical algorithm, Q, must capture all the fine features of the target distribution, p. In contrast, variation distance error says that the distributions only have to be similar 'overall'. A third notion of error is also studied:

- Additive polynomial error: A circuit is weakly simulable to additive polynomial error if there exists a classical sampler, $q(\mathbf{x})$, such that:

$$|p(\mathbf{x}) - q(\mathbf{x})| < \frac{1}{f(n)} \tag{3.17}$$

  where $f(n)$ is a polynomial function of the input size, n.

As noted in [31], it would be desirable to have a quantum sampler which could achieve the bound, (3.14), but this is not believed to be an experimentally reachable goal. That is why much effort has been put in trying to find systems for which supremacy could be provably demonstrated according to the error condition, (3.17), which is relatively easier to achieve in the lab. The works of [5, 17], for example, attempt to prove such a scenario, but lean on more complexity theory assumptions than those typically required to prove hardness of sampling to multiplicative error. However, in this work, we will focus primarily on the case of sampling to within multiplicative error, as this is more well studied, and is usually the first step proven in these hardness results.

The final ingredient required is the notion of '*postselection*'. Postselection is a tool used in complexity and probability theory which enables one to "win the lottery". Formally, it is the ability to condition on a particular event in the probability space. If the raw probability of an event, E, occurring, is given by $P(E)$, the 'postselected' distribution

is given by the conditional probability on some other event, F, $P(E|F)$. This is referred to as 'postselecting' on event F occurring.

Intuitively, postselection allows one far more power than would be normally allowed in any probabilistic theory, as it gives the ability to essentially produce an outcome (which could be exponentially unlikely in the standard theory), with probability 1 in the postselected version of the theory. For example, since quantum computers operate probabilistically (the desired solution to an encoded problem occurs with a certain probability), but if equipped with the theoretical power of postselection, they would be able to produce the solution deterministically.

3.3.1 Complexity Theory

In this section we will lay the required foundation of complexity theory results which are required to prove the classical hardness of sampling from the circuit classes we care about, like IQP and QAOA.

Firstly, it is possible to define a postselected model of a probabilistic complexity class, A, given by PostA, which means we can postselect a certain number of the output registers to be the 0 outcome, for example. The relevant results, [16], [26], provide an argument to show the hardness of classically sampling from the output distributions of certain circuit families results in a collapse of the so-called Polynomial Hierarchy.

Definition 3.4: Polynomial Hierarchy (PH) [64]

The polynomial hierarchy (PH) is defined as follows:

$$\mathsf{PH} = \bigcup_{k\in\mathbb{N}} \Sigma_k^P \tag{3.18}$$

where:

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathsf{P} \tag{3.19}$$

The other levels, $i \geq 0$ are defined recursively with respect to oracles[a]:

$$\Delta_{i+1}^{\mathsf{P}} = \mathsf{P}^{\Sigma_i^{\mathsf{P}}}, \qquad \Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}, \qquad \Pi_{i+1}^{\mathsf{P}} = \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}} \tag{3.20}$$

[a] Oracles are like black boxes that can return an answer to a problem in the complexity class in which they are defined in one step (or one query to them).

P is the complexity class of decision problems solved in polynomial time on a deter-

ministic Turing Machine. It is the *zeroth* level of the PH. Also by this definition, NP and coNP sit in the *first* level of the Hierarchy. NP is the class of decision problems where a *yes* instance can be *verified* in polynomial time, but the solution cannot be *produced*[7] in polynomial time. coNP is the complement of NP, in which one can efficiently verify a *no* instance of a problem instead. A *collapse* of the Polynomial Hierarchy is an instance where a higher level is proven to sit inside a lower level, for example, if NP is proven to be equal to P then the PH would collapse to the zeroth level, i.e. complete collapse. A collapse at a higher level is in some sense a generalisation of $\mathsf{P} = \mathsf{NP}$, and while not as unlikely, it is still believed such a collapse does not exist, i.e. the PH is infinite[8]. Another relevant class is called #P, which is a counting complexity class, and is a generalisation of NP , which is a decision class. Briefly, #P is believed to be a much larger class than NP, which can be seen intuitively as follows. Problems in NP are of the form: '*Does there exist a solution to a problem?*'. In contrast, the corresponding #P version would ask: '***How many*** *solutions exist to the problem?*'. Finally, to reiterate, BPP is the classical version of BQP, and is commonly referred to as the class of problems which classical (probabilistic) computers can solve efficiently.

Now, we are trying to show the following in relation to our problem. If there existed a classical algorithm, $\mathcal{P}$, which could produce samples according to the probability distribution outputted from a given *quantum* model, $\mathcal{Q}$ (within a multiplicative error) in polynomial time, this results in a collapse of the polynomial hierarchy. Since we have good reason to believe the latter does not happen in reality, then there must exist no classical algorithm which can achieve the former. This is exactly the argument given in [16], [26] for IQP and QAOA respectively.

Firstly, it is known by Toda's Theorem [67], that a P machine (i.e. a deterministic Turing machine) with access to a #P oracle can solve any problem in the PH:

$$\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}} \tag{3.21}$$

Furthermore, Aaronson, [3], proved that BQP equipped with postselection can also solve #P problems, $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PostBQP}}$[9]. This illustrates the power of postselection which elevates 'standard' quantum computing to be able to solve every problem in the polynomial hierarchy. Now, the final ingredient is to ask where classical computation, i.e. BPP,

[7] If $\mathsf{P} \neq \mathsf{NP}$.

[8] For a nice intuitive explanation of the PH and its collapse, see [26].

[9] The use of the oracle formalism is necessary because #P and BQP are not the same *type* of complexity class; the former is a class of *counting* problems, while the latter are *decision* problems. The oracle formalism captures in some sense the '*power*' of these classes to derive an equivalence. Further, what was actually proved is that $\mathsf{PP} = \mathsf{PostBQP}$, but $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$.

fits into this. It is known that BPP with postselection sits in the third level of the PH, [37]:

$$\mathsf{P}^{\mathsf{PostBPP}} \subseteq \Sigma_3^{\mathsf{P}} \tag{3.22}$$

Now that the relevant ingredients have been collected, we can state the aim of these proofs slightly more formally. Roughly speaking, we want to show that if a given classical sampler (which is in BPP) can produce samples, $\mathbf{z}$, according to a distribution $p(\mathbf{z})$ which is close (in multiplicative error) to the output distribution, $q(\mathbf{z})$, from a universal quantum sampling model, BQP, then we can prove that PostBPP = PostBQP. This implies the following:

$$\begin{aligned} &\mathsf{PH} \subseteq \mathsf{P}^{\#P} = \mathsf{P}^{\mathsf{PostBQP}} \overset{(a)}{=} \mathsf{P}^{\mathsf{PostBPP}} \subseteq \Sigma_3^{\mathsf{P}} \\ &\implies \mathsf{PH} \subseteq \Sigma_3^{\mathsf{P}} \end{aligned} \tag{3.23}$$

The above (3.23) is a collapse of the PH to the *third* level, and is believed to be unlikely. To illustrate how this is true (an argument to say that if a classical sampler existed, (a) would be true, i.e. PostBPP = PostBQP), we repeat the result of [28]. Clearly, PostBPP $\subseteq$ PostBQP since any problem solvable on a classical computer is also solvable on a quantum computer. The reverse inclusion involves proving that a decision problem solvable by a BQP machine with postselection, can also be solved by a BPP machine with postselection, *if* the BPP machine could output strings to within multiplicative error of the BQP circuit. Let $q(z_1, z_2) = |\langle z_1, z_2| U |0, 0^n\rangle|^2$ be the output distribution produced by a general BQP unitary acting on a $n+1$ qubit register. Postselecting on the n qubit register to give the all-zero outcome, 0^n results in the postselected quantum distribution:

$$q_{\text{post}}(z_1) = \frac{q(z_1, 0^n)}{q(0, 0^n) + q(1, 0^n)} \tag{3.24}$$

Now, a decision problem is solvable by a PostBQP machine if $q_{\text{post}}(z_1 = 0) \geq 2/3$ on the Yes instance of the problem, and $q_{\text{post}}(z_1 = 0) \leq 1/3$[10] on the No instance. Similarly, we can define the distribution produced by some classical algorithm equipped with postselection as:

$$p_{\text{post}}(z_1) = \frac{p(z_1, 0^n)}{p(0, 0^n) + p(1, 0^n)} \tag{3.25}$$

Now, if this distribution, $p(z_1, z_2)$, was close to the quantum distribution, $q(z_1, z_2)$, to within multiplicative error, this implies by (3.16), that the distributions can be related

[10]This is equivalent to saying $q_{\text{post}}(z_1 = 1) \geq 2/3$

as follows:

$$|p(z_1, z_2) - q(z_1, z_2)| \leq \gamma q(z_1, z_2) \tag{3.26}$$

$$\implies -\gamma q(z_1, z_2) \leq p(z_1, z_2) - q(z_1, z_2) \leq \gamma q(z_1, z_2)$$
$$(1-\gamma)q(z_1, z_2) \leq p(z_1, z_2) \leq (1+\gamma)q(z_1, z_2)$$
$$\implies p_{\text{post}}(z_1) = \frac{p(z_1, 0^n)}{p(0, 0^n) + p(1, 0^n)} \leq \frac{(1+\gamma)q(z_1, 0^n)}{(1-\gamma)(q(0, 0^n) + q(1, 0^n))} = \frac{1+\gamma}{1-\gamma} q_{\text{post}}(z_1)$$

Similarly,

$$p_{\text{post}}(z_1) \geq \frac{1-\gamma}{1+\gamma} q_{\text{post}}(z_1)$$

So, if $q_{\text{post}}(z_1 = 0) \geq 2/3$ on the No instance, this implies that $p_{\text{post}}(z_1 = 0) \geq \frac{2(1-\gamma)}{3(1+\gamma)} \approx 0.54$ if $\gamma = 0.1$ on the No instance also, and $p_{\text{post}}(z_1 = 1) \leq \frac{1(1+\gamma)}{3(1-\gamma)} \approx 0.41$ on the Yes instance. Hence, if a classical sampler exists with a distribution close in multiplicative error to the quantum one, that same sampler could be leveraged to solve PostBQP decision problems, and so we arrive at the result PostBQP = PostBPP[11] which precipitates the desired PH collapse.

The above is a necessary step, however we are still not done. What we actually care about are these *sub-universal* models, Q, where Q = {IQP, QAOA} for example. As such, we actually want to show that PostQ = PostBQP. If this can be done, then subject to the non-collapse of the PH, we can conclude that there must exist no BPP algorithm to produce samples from the correct output distribution of our sub-universal model of choice. In other words, these models are "as hard" as full universal quantum computation to classically simulate.

The aim of the next section is to illustrate how both the QAOA and IQP, along with another slightly different model which we refer to as IQP_y, can simulate universal quantum computation when given the extra power of postselection.

Firstly, we repeat the main results of [16, 28], which show that specific instances of IQP and QAOA can simulate BQP with the aid of postselection. Then we repeat an argument of [32] to show that these results also hold for many instances of these classes. Finally, we show a similar result for IQP_y and therefore extend the hardness result slightly to include the full class of circuits we will use in the Ising Born Machine later in Section

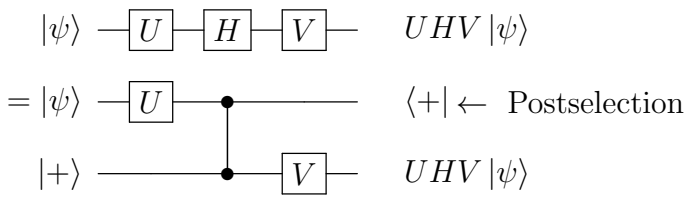
[11] Note, it is not necessary to achieve the exactly the same decision gap as in the quantum case (2/3 for Yes etc.), it is only required to be bounded away from 1/2 by a constant, as mentioned in Section (3.1).

(4.1.3). It should be mentioned for completeness that there also exists a fourth major choice for a model to demonstrate supremacy, (the others being RCS, IQP, QAOA) which is called BosonSampling, [5], but it is not as obvious to implement this in a circuit model since it works best with photonic quantum computing architectures. As such, we do not consider it here.

3.3.2 Hardness of Simulating IQP

As mentioned above, IQP circuits consist of local Z rotations: $D_1^k(b_1) = \exp(ib_k Z_k)$ and two-qubit operations $D_2^{ij}(J_{ij}) = \exp(iJ_{ij}Z_iZ_j)$ acting on $|+\rangle$ states followed by a final layer of H gates and computational basis measurements. As shown in [16], for a single parameter, $b_i = \pi/8$, one can generate the $P = D_1(\pi/8) = e^{i\pi/8Z}$ and $Z \propto D_1(\pi/8)^4$, so four repetitions of the P gate is equivalent to a Z gate. Also, for $J_{ij} = \pi/8$, one can get the CZ gate since $CZ_{ij} \propto [D_2^{ij}(\pi/8)]^2[D_1^i(\pi/8)]^4[D_1^j(\pi/8)]^4$, which will be shown in Section (4.1.3). As such, we now have that three ingredients out of the required four in a universal set, $\{CZ, P, Z, H\}$, are automatically a part of IQP circuits.

The missing ingredient is the Hadamard gate, which is *not* diagonal in the computational basis, and hence is not allowed in the IQP / QAOA model. However, this is where postselection comes to the rescue. If we define PostIQP to be IQP where we now also allow so-called 'postselection registers', [16] shows that PostIQP = PostBQP. One inclusion is straightforward, namely: PostIQP $\subseteq$ PostBQP since IQP is a subclass of BQP. The reverse inclusion requires showing that it is possible to simulate a H gate in the postselected IQP model at any location in the intermediate BQP circuit where one is required[12]. This is done via the following Hadamard Gadget:



(3.27)

U, V are any diagonal gates that proceed or follow the Hadamard. The notation $\langle +|$ indicates a postselection, that we are forcing the first qubit to be in the state, $|+\rangle$. This results in the initial state, with the desired gate sequence, UHV being 'teleported' to the ancilla. Also, it is important to realise that the Hadamard in the original circuit is

[12]This is because, in general, a BQP circuit may have polynomial depth, i.e. polynomially many non-commuting gates. In IQP this is not allowed.

intermediate in between U and V, which are assumed to be diagonal. These intermediate H gates are the ones which we want to remove. Due to the relation $|+\rangle = H|0\rangle$, postselection in the $|\pm\rangle$ basis is equivalent to the circuit beginning with a H gate, and postselecting in the computational basis (which is the standard IQP structure).

It can be seen that this structure now fits in the PostIQP model as we have swapped the intermediate Hadamard gate with a CZ, which is diagonal, and a postselection. The Hadamard Gadget works as follows (setting $U = V = \mathbb{1}$) for clarity. Assume we wish to apply H to the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, so the resulting state should be:

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

The operation of the gadget is as follows:

$$\begin{aligned}
\implies CZ|\psi\rangle|+\rangle &= (\alpha|0+\rangle + \beta|1-\rangle) \\
&\overset{(a)}{=} \frac{1}{\sqrt{2}}(\alpha|++\rangle + \alpha|-+\rangle + \beta|+-\rangle - \beta|--\rangle) \\
&= \frac{1}{\sqrt{2}}\left(|+\rangle \underbrace{[\alpha|+\rangle + \beta|-\rangle]}_{\substack{\text{Desired State} \\ =H|\psi\rangle}} + |-\rangle \underbrace{[\alpha|+\rangle - \beta|-\rangle]}_{\substack{\text{Unwanted Correction} \\ =XH|\psi\rangle}}\right)
\end{aligned}$$

(a) follows because $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle$. Therefore by postselecting the original qubit to be $|+\rangle$, it is possible to project the Hadamard onto the ancilla. In this way, a general BQP circuit can be collapsed to an IQP circuit by adding an ancilla for each Hadamard in the original circuit.

With the above gadget, one can easily see the power of postselection. If we were not able to deterministically select the outcome of the original qubit, it could end up in the $|-\rangle$ state with some probability. Indeed this is exactly what happens in the standard quantum computing model. This outcome would lead to XH being applied to the ancilla, rather than the desired H, and as such would require a correction to be applied (specifically an extra X, since $X^2 = \mathbb{1}$) to get a deterministic outcome. Thus, the above shows that $\mathsf{PostIQP}(\pi/8, \pi/8) = \mathsf{PostBQP}$[13], i.e. IQP with a single parameter for the single and two-qubit operations is sufficient to generate any quantum computation.

3.3.3 Hardness of Simulating QAOA

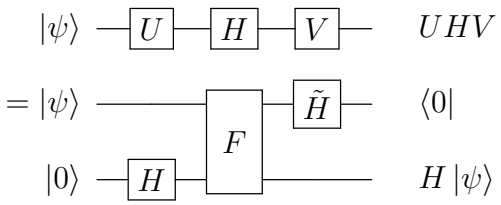
In the above, we have demonstrated how IQP with postselection is equal to BQP with postselection. A very similar result can be shown for QAOA, i.e. that PostQAOA =

[13] See (3.31) for an explanation of this notation.

PostBQP, as proven by [28]. We will now repeat this argument, noting that since the structure of the intermediate part of a QAOA circuit is exactly the same as IQP, the only difference is the final measurement gate, U_f^X. As such, it is clear that CZ, P, Z can be generated in exactly the same way as in IQP. The only non-trivial part is how the intermediate Hadamards can be generated. The above gadget (3.27) no longer fits with the QAOA model, since QAOA circuits end with rotations around the Pauli - X axis on all qubits, rather than Hadamards. Following [28], if $\gamma = \pi/4$, the final measurement gate is in fact given by (2.17, 3.28), $\tilde{H}$:

$$\tilde{H} = e^{-i\frac{\pi}{4}X} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \tag{3.28}$$

Again, this gate can be generated by an integer multiple of $\pi/8$. The gadget required in this case to simulate a Hadamard on the desired qubit is given by:



(3.29)

where:

$$F = e^{-i\frac{\pi}{8}12|01\rangle\langle 01|} e^{-i\frac{\pi}{8}4|11\rangle\langle 11|} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -i \end{pmatrix} \tag{3.30}$$

This gate is added to compensate the addition of $\tilde{H}$ at the end of the circuit, and since it is diagonal in the computational basis, it can be decomposed into CZ gates and local rotations. We will see how this is done for the second exponential term in (3.30) later in Section (4.1.3). The first term can be computed the same way[14]. Thus, just as in IQP, PostQAOA$(\pi/8, \pi/8, \pi/4)$ = PostBQP.

However, all of the above has only assumed very specific values to generate the above circuit classes, i.e. $\pi/8, \pi/4$. We now ask the question, for what parameter values do these models remain classically hard to simulate? This is of vital importance for our machine learning algorithm for which, as we shall see later, we need to *train* these parameters and as such they will need to take on many values. It is essential that for *all* of the parameter

[14]Note: $|ij\rangle\langle ij| = |i\rangle\langle i| \otimes |j\rangle\langle j|$

values we allow the algorithm to take, the resulting circuit class is hard to simulate in order to get a quantum advantage.

Notice, in the above models, it was necessary to generate maximal entanglement between two pairs of qubits for universality (i.e. the particular parameter value which admits a maximally entangling CZ gate). We will see in Section (4.1.3), how different parameters in the two-qubit unitaries ($\exp(i\theta ZZ)$) result in $CZ(\alpha)$[15] gates plus local corrections, which are not maximally entangling for $\alpha < \frac{\pi}{4}$, and hence how the 'Ising gate' we are referring to is related to the gates on our standard universal set.

To answer the question of which parameter values the computation can take, we refer to the work of [32] and [55]. In particular, if we have diagonal gates parametrized by an angle, θ, acting on either one or two qubits:

$$D_1(\theta_1) = e^{i\theta_1 Z} \qquad D_2(\theta_2) = e^{i\theta_2 ZZ} \tag{3.31}$$

Where we denote $\mathsf{IQP}(\theta_1, \theta_2), \mathsf{QAOA}(\theta_1, \theta_2, \Gamma)$ to be the classes of circuits composed of diagonal two-qubit gates with parameter(s), θ_2, single qubit gates with parameter(s), θ_1, and in the case of QAOA, the Γ parameters are those in the final X rotations which determine the measurement basis.

Now, all the gates in our models are of the form $U(\alpha) = e^{i\alpha\Sigma}$, where Σ is some tensor product of Pauli operations, i.e. $\Sigma = \{Z \otimes Z, Z, X, Y\}$ for example. By applying each of these gates m times, where $m = O(1/\epsilon)$ is an integer, we can get a gate which is arbitrarily close in some error measure to the required gate, which has parameter $\pi/8$. We have already seen how diagonal gates with this parameter can generate all the required diagonal gates for universality, and with postselection we get the rest. Therefore, if it is possible to simulate this $\pi/8$ parameter, with an arbitrary parameter, α, (and postselection) we will be able to simulate $\mathsf{PostIQP}(\pi/8, \pi/8)$ (and $\mathsf{QAOA}(\pi/8, \pi/8, \pi/4)$), and hence BQP. The error measure is defined, [55], as the difference between the operations of two gates on a quantum state, when maximized over all possible states:

$$E(U, V) = \max_{|\psi\rangle} ||(U - V)|\psi\rangle|| \tag{3.32}$$

Where $||\cdot||$ is the norm of a vector: $||U|\psi\rangle|| = \sqrt{\langle\psi| U^\dagger U |\psi\rangle}$. If, by m repetitions, the gate is within ϵ of the required gate with parameter $\pi/8$, $U(\pi/8 + \epsilon)$, the error induced

[15] The relationship between α and θ will also be made apparent.

by this extra ϵ factor will be $O(\epsilon)$:

$$\begin{aligned} E\left(U(\pi/8+\epsilon), U(\pi/8)\right) &= |1-e^{i\frac{\epsilon}{2}}| = |1-(1-i\epsilon/2-\epsilon^2/8+O(\epsilon^3))| \\ &= |i\epsilon/2+\epsilon^2/8+O(\epsilon^3))| = \sqrt{(\epsilon^2/8)^2+[\epsilon/2]^2+O(\epsilon^3)} \\ &= \sqrt{\epsilon^4/64+\epsilon^2/4^2+O(\epsilon^3)} = \epsilon/2\sqrt{1+\epsilon^2/16+O(\epsilon^3)} = O(\epsilon) \end{aligned}$$

Specifically, with the two-qubit gate for example:

$$D_2(\theta_2)^m = D_2(m\theta_2) = D_2\left(\frac{\pi}{8}+\epsilon\right) \tag{3.33}$$

so the gate with parameter θ_2 can be made ϵ close to the required $CZ \sim D_2(\frac{\pi}{8})$ with respect to the error, as noted by [32, 55]. The same thing holds for the single qubit gates with angle θ_1 approximating Z or P for example. Furthermore, it also holds for the final rotation gates, $R_x(\Gamma)$, from QAOA and $R_y(\Delta)$ which will appear in $\mathsf{IQP}_{\mathrm{y}}$. However, we have not said for what constraints on θ_1, θ_2 this argument holds. Clearly it cannot hold for *every* value, since as shown in [32], if there is no entanglement in the system (i.e. $\theta_2 = 0$ for all qubits) the system becomes separable, and hence can be written as an Ising partition function which is exactly solvable. Similarly, for a system which has *only* entanglement but no local rotations, the same is true. The key ingredient is that the given instance must be able to simulate universal quantum computation in some sense. As such, for all parameters which are either:

$$\theta = \begin{cases} \frac{(2l+1)\pi}{8d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{3.34}$$

The irrationality of the parameter values allows the above, (3.33), to be true since $2\pi\nu m \mod 2\pi$ is distributed uniformly[16]. If this parameter, ϵ, is lower than the threshold for fault-tolerant quantum computation, as noted in [32], then we can reliably simulate universal quantum computation.

3.3.4 Hardness of Simulating $\mathsf{IQP}_{\mathrm{y}}$

Finally, we extend the above results slightly to include a larger class of computations, namely it is possible include final measurements of the the final qubits not only in the Hadamard basis, (H as in IQP), and in a rotated X basis ($\tilde{H}$, as in QAOA), but also in

[16] See [14] for a proof that any arbitrary phase can be approximated to accuracy, ϵ, with $poly(1/\epsilon)$ repetitions of a phase which is an irrational multiple of 2π. This shows that we it is possible to achieve any gate: $e^{i2\nu\pi\hat{n}\dot{\sigma}}$ using m repetitions of $(e^{i2\nu\pi\hat{n}\dot{\sigma}})^n$ if ν is irrational.

any other basis in the $X-Y$ plane of the Bloch sphere. For example, a final measurement in a Pauli-Y rotated basis:

$$R_y(\Delta) = e^{-i\Delta Y} \tag{3.35}$$

The computation involving diagonal gates acting on $|+\rangle$ basis states, with a final measurement in this (3.35) basis shall be referred to as $\mathsf{IQP}_{\mathrm{y}}(\theta_1, \theta_2, \Delta)$. Replacing this Y rotation with any one in the $X-Y$ plane gives the class of Ising Born Machine circuits that will be used in this work.

If there exists no classical randomized polynomial time algorithm to produce samples, $\mathbf{z}$, in polynomial time according to the IQP distribution:

$$p_{\mathsf{IQP}}(\mathbf{z}) = |\langle \mathbf{z}| H^{\otimes n} U_z H^{\otimes n} |0\rangle^{\otimes n}|^2 \tag{3.36}$$

then no algorithm exists to produce samples, $\mathbf{z}'$, according to the $\mathsf{IQP}_{\mathrm{y}}$ distribution either:

$$p_{\mathsf{IQP}_{\mathrm{y}}}(\mathbf{z}') = |\langle \mathbf{z}'| \bigotimes_{i=1}^{n} e^{i\frac{\Delta_i}{2}Y_i} U_z H^{\otimes n} |0\rangle^{\otimes n}|^2 \tag{3.37}$$

This is due to the relationship: $H = \frac{1}{\sqrt{i}} X Y^{\frac{1}{2}} = \frac{1}{\sqrt{i}} X R_y(\frac{\pi}{2})$. Therefore, choosing $\Delta_i = \pi/4$, we get that the two distributions, (3.36, 3.37) are related as follows:

$$\begin{aligned} p_{\mathsf{IQP}}(\mathbf{z}) &= |i^{-n} \langle \mathbf{z}| X^{\otimes n} \sqrt{Y}^{\otimes n} U_z H^{\otimes n} |0\rangle^{\otimes n}|^2 && (3.38)\\ &= |\langle \mathbf{z}'| \bigotimes_{i=1}^{n} e^{i\frac{\pi}{4}Y_i} U_z H^{\otimes n} |0\rangle^{\otimes n}|^2 && (3.39)\\ &= p_{\mathsf{IQP}_{\mathrm{y}}}(\mathbf{z}') && (3.40) \end{aligned}$$

so $\mathbf{z}'$ is simply $\mathbf{z}$ with every bit flipped. Clearly, if one could produce samples, $\mathbf{z}$, by some classical means efficiently, then the same algorithm with one extra step can be used to produce the samples, $\mathbf{z}'$, and hence $\mathsf{IQP}_{\mathrm{y}}(\theta_1, \theta_2, \pi/4)$ is also classically hard to sample from, where θ_1, θ_2 have the same constraints as in IQP or QAOA.

In Sections (4.2.1, 4.2.2), we will see that the computing the gradient of the Ising Born Machine requires running circuits which are parameter shifted with respect to the original one. Furthermore, as we train the model, the parameters, θ_1, θ_2 will change. We can ensure that the system remains hard to simulate as long as we start with an initial configuration of the parameters that demonstrates supremacy, and update them such that this remains the case. For example, if we choose the initial parameters, $\theta_i = 2\pi\nu$, where ν is irrational, then an update will be of the form $\theta^{(n+1)} = \theta^{(n)} + \mu$. If we choose

$\mu = 2\pi\alpha$, then the new parameters at stage, $n+1$, will be $\theta^{(n+1)} = 2\pi\beta$, $\beta = \nu + \alpha$. Since ν was irrational originally, α will also be irrational, and therefore the new system will be hard to simulate. Although, clearly we cannot allow updates like $\alpha = -\nu$ $(+\delta)$, where δ is rational since this will result in the parameter going to 0 (δ) and could make the model classically simulable. A good example of this, is if we choose the final parameter in QAOA, $\Gamma = 0$. In this case, we create a uniform superposition from the initial Hadamards, $1/\sqrt{2^n}\sum_{\mathbf{z}} |\mathbf{z}\rangle$, and then apply gates in the computational basis. This will only add phases to each contribution $e^{i\theta}|z_j\rangle$. If we have no final 'measurement' gate (i.e. the parameter is zero), this is equivalent to measuring the state in the computational basis at the end. Since the phases do not have the ability to interfere with each other, they will have no effect on the measurement probabilities[17], and the final distribution will be simply the uniform distribution over all n-bit strings, $\mathbf{z}$. This can clearly be classically simulated by a sequence of coin flips[18].

Furthermore, this can be extended to (almost) any final measurement rotation in the $X-Y$ plane. Firstly, if the parameter values, $\mathbf{\Gamma}, \mathbf{\Delta}$ for QAOA or $\mathsf{IQP}_{\mathrm{y}}$ are chosen as in (3.34), then they allow simulation of $\exp(i\pi/4Y_k), \exp(i\pi/4X_k)$ by the same argument as (3.33), i.e. by applying each one $l = O(1/\delta)$ times to simulate the required gates to within δ in the error. This means it is also possible to *train* these parameters, $\{\Gamma_k, \Delta_k\}$ to enhance the model, just as we will do for the Ising parameters, $\{J_{ij}, b_k\}$ in Section (4.1.3). However, we will not do this in this dissertation, instead leave it to future work. Finally, the result for any final rotation in the $X-Y$ plane follows immediately since, for example, to get to the states $|\pm\rangle$ on the Bloch Sphere from $|\pm i\rangle$, this involves a rotation around the Pauli-Z axis, so this extra rotation can be absorbed back into the single qubit rotation parameters, b_k. This can be generalised easily.

[17] This is because $|\langle 0| e^{i\theta}|z_j\rangle|^2 = |e^{i\theta}\langle 0|z_j\rangle|^2 = |\langle 0|z_j\rangle|^2$

[18] Thanks to Andru Gheorghiu for pointing this out.

Chapter 4

Quantum Machine Learning Models

The field of machine learning (ML) has found great application in our modern world and is now ubiquitous. It is the aim of the new and rapidly growing field of *quantum* machine learning (QML) to apply quantum techniques to the wealth of knowledge and resources available in its classical counterpart. This is done in the hope to find a candidate problem which maps naturally into the quantum domain and which leads to a better solution, or to reach a solution more quickly than is possible classically. This question has practical relevance, as currently it may take hours or days to train a specific machine learning model to perform a task, given classical technology. If quantum methods could speed this process up to minutes or seconds, this would be an excellent advantage to have.

The initial step into the field of QML was the discovery of the HHL [39] algorithm for solving systems of linear equations with a potential exponential speedup[1] over the best classical methods to solve the same problem. As Scott Aaronson puts it, [4], while this method does have several technical requirements which may prohibit an actual realisation of this speedup in practice, it forms the "skeleton" of a useful algorithm to be built on by other works addressing these technical limitations. A particular instance is the development of Quantum Random Access Memory (qRAM), [35], which is another hot topic in the field at the current time. In general, these types of algorithm, such as HHL, require the ability to take a given dataset and prepare the data (say a vector $\vec{d} = (d_1, \ldots, d_n)$) in a coherent superposition state:

$$|\Psi\rangle = \sum_j d_j |j\rangle \tag{4.1}$$

This can be done using qRAM [35], and, in some sense, asks for the ability to access all the data values at once. Unfortunately, a major limitation to these algorithms seems to

[1] This exponential speedup comes with several caveats and only applies in certain specific cases.

be how difficult an actual working qRAM is to create in practice. A second potential algorithm which *had* the most promise for a *realizable* implementation was a quantum algorithm for recommendation systems, [41], such as the one used by Netflix to recommend products to customers. This quantum algorithm gave an exponential speedup over the best known classical algorithms to solve the same problem, under more realistic assumptions than the HHL algorithm. However, this algorithm was also very recently *de*-quantized, [65], in the sense that it inspired a *completely classical* algorithm which is exponentially faster than any existing classical one, given the same assumptions as its quantum counterpart. This can be seen as both a win and a loss for the field of QML, since the best hope for an exponential quantum advantage has disappeared for this problem, however it possibly could not have existed without the initial quantum insight.

We mention these above examples for the following reason. The HHL algorithm has a very well defined application: solving systems of linear equations, which has much use in the field of ML, but as mentioned above, the physical realisation of this algorithm is far from straightforward. The recommendation system algorithms *also* have a well defined problem to solve, but the quantum version lacked the theoretical footing of complexity theory results to guarantee a classical-quantum separation, hence its subsequent de-quantization. It is our hope that the model we will present in this work has the best of both worlds. While it may have a less obvious problem to solve, it has a much easier implementation than HHL, particularly for the dawn of the NISQ era, but comes with the relevant complexity theory results to ensure it also does not become de-quantized.

4.1 Generative (Quantum) Machine Learning

The problem we address is that of Generative Machine Learning (GML), for which we propose a quantum solution. Roughly speaking, the problem that is tackled with generative models is the following:

> Given samples from a probability distribution (i.e. a dataset), can we *learn* that probability distribution with a generative model, such that when querying the (trained) model, *new* samples are produced according to the original data distribution (or a distribution which is *close enough* for practical purposes)?

The goal of these types of generative algorithms is to essentially learn some representation of a given dataset, such that it can be reproduced on command. In a specific

example, generative models are used in statistical classification, where one has an observed variable, X, and a target variable, Y. Generative classification models wish to learn the *joint* distribution over both random variables, $P(X, Y)$. Typically Y is a set of 'labels', with which to label the observations, X. Once this joint distribution has been learned, it is applied to the classification problem as follows. Given a point in the support of the observed variable, $X = x$, what label, y, should be applied to that observation? This is given according to the conditional distribution: $P(Y|X = x)$, which can be computed using the joint.

On the other hand, there exists the alternative method of *discriminative* classification algorithms which attempts to learn the conditional directly, i.e. it learns $P(Y|X = x)$ in order to best guess how to best learn label the samples. However, we will adopt a broader view of these approaches without specific reference to the classification problem. As such, 'generative model' will refer to learning some probability distribution, whereas 'discriminative model' will refer to discriminating data into classes. In other words, we are not concerned with how the distribution is used once it has been learned, we only care about learning it such that it allows sampling afterwards.

As a simple example of generative models, imagine one is given a set of images of cats (a dataset). For some reason, we wish to generate *new* images of cats, which we have not seen before. A generative machine learning algorithm would learn some distribution over the images of cats to "learn what a cat looks like". Once it has acquired this internal representation, it should be able to probabilistically produce a new image according to what the algorithm has decided the defining features of a cat should be, given the images it has already seen. Figure (4.1) illustrates the distinction between discriminative and generative ML models; the former is asked to classify a new sample to be either a cat or dog image, based on its features, the latter is required to *generate* a new image of a cat, for example.

As mentioned above, generative models typically require the drawing of samples from the trained model, and it turns out even in the training process, we usually require a method of sampling from these models efficiently. Given a probability distribution over some data (this will be an empirical distribution computed from samples from the training set), we can heuristically describe how the learning procedure works:

(1) Start with a candidate probability distribution, typically given by parameters initialized at random.

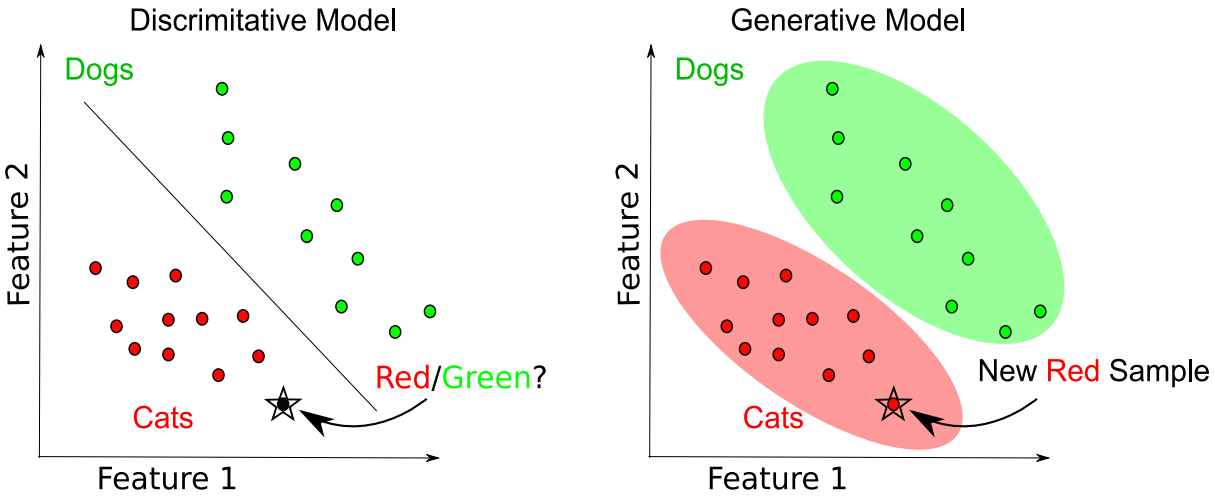


Figure 4.1: Difference between Discriminative and Generative Models.

(2) Compare some 'distance'[2] between the data, and the instantaneous distribution produced by our model at some points during the training.

(3) Update the parameters in the model according to some optimization procedure (e.g. gradient descent) so that the distance is minimized.

(4) Repeat steps (2)-(4) until a minimum of the distance function has been attained.

(5) Sample from the model to produce new models according to the original data distribution

In step (2), it is necessary to compute the instantaneous distribution outputted by our model. To do so, we must sample from the model to build a picture of this distribution. This will be the best approximation of the model to the training data at the current time, and it is the goal of the training algorithm to make the approximation better as time goes on.

Given this need, hopefully the application of quantum computers in this area is becoming apparent. We went to great lengths above in Sections (3.3) to describe probability distributions from which samples can be outputted from quantum devices, but which could not be derived by any classical means. It is the conjecture of this work, that because these distributions cannot be simulated efficiently classically, therefore these types of Supremacy-QML algorithms present opportunities to learn probability distributions *better* than is possible using any purely classical algorithm. This point is discussed further in Chapter (5).

[2]This is a highly important point which will be addressed in detail later.

4.1.1 Training a Machine Learning Algorithm

In step (3) of the training procedure in the last section, we mentioned "training", referring to the updating of parameters in the model. In many machine learning algorithms, the tool of choice to do this is called *gradient descent*. This is simply an iterative algorithm which optimizes some "cost" function with respect to the parameters, in order to find a minimum of the function[3].

As a simple illustration, imagine sitting in a parameter space, defined by the parameters, θ, of the model. The cost function will define some landscape in this parameter space, and the goal is to find the minimum of this landscape by going downhill as fast as possible[4]. Let a cost function (in a generative model for example) be given by $J(p_\theta, \pi)$ encoding some relationship between the model distribution, p_θ, and the data, π, which should be equal to zero when $p_\theta = \pi$, i.e $J(p_\theta, p_\theta) = 0$. For instance, if the parameters in the model are initialized at random, the training process would consist of iteratively updating the parameters at each epoch[5] according to the learning rule:

$$\theta^{(n+1)} = \theta^{(n)} - \eta \nabla_\theta J(p_\theta, \pi) \tag{4.2}$$

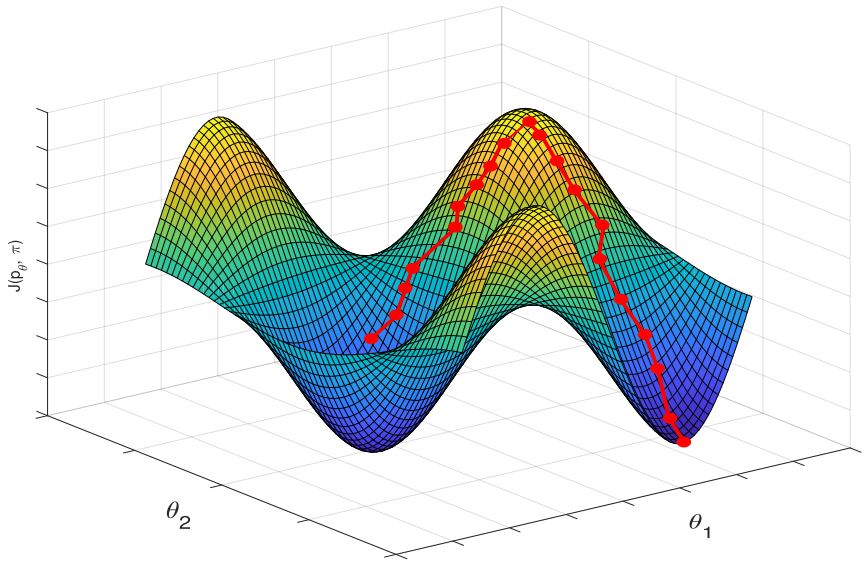


Figure 4.2: Plot of cost function in parameter space with two parameters, $\theta = \{\theta_1, \theta_2\}$.

Figure (4.2) illustrates the training process for a very simple cost function, which is a function of two parameters, $\theta = \{\theta_1, \theta_2\}$. The training algorithm will most likely start at a

[3]Ideally this will be a global minimum, but often local minima are good enough in practice.
[4]With the caveat that going too fast could result in getting stuck in local minima.
[5]One epoch is a complete iteration over all the parameters.

peak (the model and data are initially as far as possible), and gradient descent traverses along the direction of steepest descent to reach a minimum. In the case of the above figure, there are two equally good minima since the function happens to be periodic.

In general, the cost function, J, could be a highly non-convex function with many optima, which makes training difficult. Furthermore, there are many such candidate cost functions one could choose for a given problem. Indeed, this is a key question we will address later on in this work. A characteristic which is favoured by one function may not be at all by a different one.

The quantity, η, is called the *learning rate* and it is one of the so-called 'hyperparameters' we can choose beforehand to aid training. In essence, the learning rate controls *how far* in parameter space the algorithm will step, at each epoch. If the learning rate is too large, the algorithm will likely overshoot the desired minimum. If it is too small, the algorithm will take too long to converge. Therefore, choosing the learning rate is itself an artform. A possible approach is to define a learning rate which is time-dependent; for example it is large initially, but decreases as training progresses so as to get a balance between speed and convergence.

4.1.2 Quantum Boltzmann Machine

In this section, we will recall one of the first attempts to define a *quantum* generative model. This was done by [8], where the authors quantized the Boltzmann Machine (BM) model. Since the Boltzmann Machine is defined by the Ising Model of statistical physics, it is a very natural choice to run on a quantum computer.

The classical Boltzmann Machine is a form of a stochastic recurrent neural network, first defined in 1985, [6]. As with any neural network, it is represented in terms of *nodes* or *units*. These nodes can be either deactivated or activated, corresponding to a binary variable, $z \in \{0, 1\}$ respectively. Some of these nodes are labelled as *visible* which produce the samples in the generative model, and some are *hidden*, used for extra representational power.

The BM is defined on an energy function, E_z, which is an Ising model in its standard form:

$$E_{\mathbf{z}} = -\sum_{i<j} J_{ij} z_i z_j - \sum_i b_i z_i \tag{4.3}$$

If the Boltzmann Machine is viewed in terms of a graph, J_{ij} represents the weights of the graph (or the coupling between node i and node j) and b_i is the *bias* of an

individual node (or local magnetic field of the Ising Model). The probability of a given configuration of the nodes, $\mathbf{z} = (z_1, \ldots, z_n) = (\mathbf{v}, \mathbf{h}) = \{v_1, \ldots, v_{n_v}, h_1, \ldots, h_{n_h}\}$, where n_v, n_h are the number of nodes which are visible, or hidden respectively, is given by a Boltzmann distribution[6],

$$p(\mathbf{z}) = \mathcal{Z}^{-1} e^{-E_{\mathbf{z}}} \tag{4.4}$$

where $\mathcal{Z}$ is the *partition function*, an object of fundamental importance in statistical mechanics, and ensures normalization:

$$\mathcal{Z} = \sum_{\mathbf{z}} e^{-E_{\mathbf{z}}} \implies \sum_{\mathbf{z}} p(\mathbf{z}) = 1 \tag{4.5}$$

For the specific case of the Boltzmann Machine, the 'randomness' required to generate samples probabilistically is derived from the statistical ensemble over possible configurations of the system. The probability is an inverse exponential in the energy of a particular configuration, indicating that the lowest energy states are the more likely to occur, with the ground state being the minimum.

The probability distribution of interest is the marginal distribution over the hidden nodes, $p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{z})$.

Now, as is done in [8], the model can be quantized into a *Quantum* Boltzmann Machine (QBM). Firstly, each binary node is replaced by a qubit in the computational basis $z_i \to \sigma_z^i$.[7] For example, if qubit, i, is prepared in the computational basis, $\sigma_z^i \in \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ where $Z_i = \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{i-1} \otimes Z_i \otimes \mathbb{1}_{i+1} \otimes \cdots \otimes \mathbb{1}_n$ denotes the Pauli-Z operator acting on qubit i, with identity on all other qubits.

Now, the energy function is replaced by a Hamiltonian: $E_{\mathbf{z}} \to \mathcal{H}$:

$$\mathcal{H} = -\sum_{i<j} J_{ij} Z_i Z_j - \sum_i b_i Z_i \tag{4.6}$$

Now, the quantum state of the system described by the Hamiltonian, (4.6), is a so-called *Gibbs state* (or thermal state) which is a mixed state of all possible energy eigenvectors of the Hamiltonian, weighted by the corresponding Boltzmann probability.

$$\rho = \mathcal{Z}^{-1} e^{-\mathcal{H}}, \qquad \mathcal{Z} = \mathrm{Tr}\left(e^{-\mathcal{H}}\right) \tag{4.7}$$

The definition of the Hamiltonian,(4.6), has not done anything interesting yet; it is still completely diagonal in the computational basis, and hence is still '*classical*'. The

[6] Note that the inverse temperature has been normalized to one.

[7] The notation, σ_a^i shall denote the qubit in basis a, and the A shall be used for the operator in basis a.

diagonal terms of the above state, (4.7) correspond to the energies of the configurations of the original BM, but trivially encoded in a quantum state. To exploit interesting quantum behaviours, it is necessary to augment the Hamiltonian by introducing non-commuting terms, in this case, simple rotations around the Pauli-X axis.[8] The new QBM Hamiltonian is then:

$$\mathcal{H} = -\sum_{i<j} J_{ij} Z_i Z_j - \sum_i b_i Z_i - \sum_i \Gamma_i X_i \tag{4.8}$$

This Hamiltonian has n extra degrees of freedom in the parameters $\mathbf{\Gamma} = \{\Gamma_1, \ldots, \Gamma_n\}$. The resulting state is still given by, (4.7), however the new state has now off-diagonal terms in its density matrix.

4.1.2.1 QBM State Preparation

Of course to actually use the QBM, one has to be able to produce the above state, (4.7) on a quantum computer. The first method to do so is implemented in [8] using a D-Wave Quantum Annealer. Quantum Annealing is the quantum version of simulated annealing, a classical simulation technique. It is an optimization technique which has similar feel to that used in Adiabatic Quantum Algorithm (AQA) [27], and the QAOA. In some sense it is a superset of the AQA, since if the annealing procedure is done slowly enough, such that the Adiabatic Theorem holds, it becomes Adiabatic Quantum Computation. It is typically used in association with the transverse Ising model and defined by a time-*dependent* Hamiltonian:

$$\mathcal{H}(s) = -A(s)\left[\sum_{i<j} J_{ij} Z_i Z_j + \sum_i b_i Z_i\right] - B(s)\sum_i X_i \tag{4.9}$$

$$= A(s)\mathcal{H}_z + B(s)\mathcal{H}_x \tag{4.10}$$

The terms, $A(s), B(s)$ determine the evolution of the Hamiltonian where, $s = t/t_a$ is called the *annealing schedule*, t_a is the annealing time. A, B are chosen such that $A(0) \ll B(0), A(1) \gg B(1)$, i.e. the system starts in an eigenstate of $\mathcal{H}_x$ (which is a product state, and hence relatively easy to prepare), and then the system is time evolved such that it ends up in the eigenstate of the final Hamiltonian, $\mathcal{H}_z$ which is the desired one, and presumably difficult to prepare. If the annealing is done slowly enough and the system will remain in the same instantaneous eigenstate, as per the Adiabatic Theorem.

[8] This would correspond to introducing a *transverse* field to the Ising model.

The second approach to implement the QBM does not actually implement the full Hamiltonian, (4.8), instead it only defines an *approximate* version, the Quantum Approximate Boltzmann Machine (QABoM), [68]. It does this by implementing the QAOA to find the approximate thermal state after a finite sequence of steps.

It should be noted the distinction between the two sets of parameters, $\{J_{ij}, b_i\}$ and γ, β defined in the quantum state produced by the QAOA, and hence the QABoM, in (3.12). The first set are the actual parameters of the *machine learning model* and the target Hamiltonian, $\mathcal{H}_z$, in (3.12) which are used to produce a particular state to sample from, whereas the latter are optimized in the QAOA to produce the approximate ground or thermal state of the cost Hamiltonian, $\mathcal{H}_z$. In short, the QABoM of [68] uses the QAOA to prepare the thermal state by optimizing the γ, β parameters, which, once prepared, is then *trained* by updating the parameters, $\{J_{ij}, b_i\}$ in the actual Boltzmann Machine.

4.1.2.2 Training the Quantum Boltzmann Machine

The upgrade of the classical Boltzmann Machine to its quantum counterpart lead to interesting added difficulties in training, which we will briefly mention in this section. The cost function which is chosen in [8] is the Kullback-Leibler (KL) Divergence which is the standard cost function used in ML applications to compare two probability distributions, [44]. This is an information theoretic measure of difference because it is also the relative entropy between two probability distributions. The KL divergence is defined as follows for two distributions which are discrete, [50]:

$$D_{KL}(P||Q) = \sum_{\mathbf{x}} P(\mathbf{x}) \log \left(\frac{P(\mathbf{x})}{Q(\mathbf{x})} \right) \tag{4.11}$$

The QBM was trained via gradient descent with the negative log likelihood, $\mathcal{L}_{KL}$ (equivalent to the KL Divergence as a cost function) and the parameters updated between epochs (n) as:

$$J_{ij}^{(n+1)} = J_{ij}^{(n)} - \eta \partial_{J_{ij}} \mathcal{L}_{KL} \qquad b_k^{(n+1)} = b_k^{(n)} - \eta \partial_{b_k} \mathcal{L}_{KL} \tag{4.12}$$

$$\mathcal{L}_{KL} = -\sum_{\mathbf{v}} p(\mathbf{v}) \log p(\mathbf{v}) \tag{4.13}$$

As found in [8], the computation of the gradient above requires the evaluation of a difficult integral for training which is not an efficient and scalable solution[9]. It was noticed

[9]This could be done explicitly for the small problem size they investigate with eight visible and two hidden qubits.

that the QBM outperformed the classical Boltzmann Machine in learning a probability distribution, in terms of accuracy with respect to the KL Divergence.

Since this original work, attempts have been made to improve it, for example the Approximate QBM (QABoM) mentioned above in Section (3.2) [68], which is seen again to outperform the classical model even in the presence of moderate depolarising noise[10] and given that it was only an approximate solution. Finally, an attempt was made by [42] to improve the efficiency of the training process of the QBM and further they defined an alternative QBM with a fermionic Hamiltonian (defined by fermionic creation and annihilation operators) which they also conjectured to be hard to classically simulate as it is not *stoquastic*[11]. It is somewhat a mixture of these two mentalities that we want to emulate in this work; to define a quantum machine learning algorithm which is relatively simple to implement, but still complex enough to be hard to classically simulate, and therefore provide a quantum advantage.

The original motivation of the work undertaken in this dissertation was to study the complexity of the Quantum Boltzmann Machine, in order to explain its advantage in accuracy over its classical counterpart. Unfortunately, due to the nature of the state produced by the QBM, it was not possible to connect the hardness results of Section (3.3), to the training of the QBM, in order to make a conjecture about the origin of the quantum advantage. This is illustrated by [68], in the description of the state which is prepared in the QABoM. Using the QAOA, the QABoM starts in a thermal state of the mixer Hamiltonian, $\mathcal{H}_x$ and 'digitally anneals' to the thermal state of the target Hamiltonian, $\mathcal{H}_z$:

$$\rho_x = \mathcal{Z}_M^{-1} e^{-\mathcal{H}_x} \rightarrow \rho_z = \mathcal{Z}_z^{-1} e^{-\mathcal{H}_z} \tag{4.14}$$

This process, (4.14), is implemented via annealing in [8], and digitally using quantum gates in [68]. However, the thermal state is a mixed state of the form:

$$\rho_x = \mathcal{Z}_x^{-1} e^{-\mathcal{H}_x} = \mathcal{Z}_x^{-1} \sum_{\mathbf{s}\in\{0,1\}^n} e^{-|\mathbf{s}|} \left|\pm_{\mathbf{s}}\right\rangle \left\langle\pm_{\mathbf{s}}\right| \tag{4.15}$$

where $|\mathbf{s}|$ is the Hamming weight[12] of the string $\mathbf{s}$. The normalization factor is not

[10] Depolarising noise is a simple model used to state that instead of the correct state being produce, ρ, in a task, instead a mixed state which has an equal probability of having one of the three Pauli errors applied to it instead: $\rho_p = (1-p)\rho + p/3Z\rho Z + p/3X\rho X + p/3Y\rho Y$

[11] A stoquastic Hamiltonian [15] is one which has all off-diagonal terms real and non-positive in the computational basis. It is believed that these types of Hamiltonians can be simulated using quantum Monte Carlo methods, whereas non-stoquastic Hamiltonians cannot.

[12] The Hamming weight is a measure of distance between binary strings, it counts the number of operations required to transform one string into another, i.e. the number of bits in which the two distributions differ.

important for this discussion. Now, the difficulty becomes apparent. The state of (4.15) is a classical mixture of all possible configurations of the qubits in the eigenstates of $\mathcal{H}_x$, each weighted by a Boltzmann factor given by the distance from the ground state (the state which consists of every qubit in the state $|+\rangle$). For a single qubit, this is:

$$\rho_x \sim |+\rangle\langle+| + e^{-1}|-\rangle\langle-| \tag{4.16}$$

If we were to now apply the cost Hamiltonian unitary, $\mathcal{U}_z = e^{-i\mathcal{H}_z}$, to this state, which decomposes into the diagonal gates in our sub-universal models in Section (3.3)[13], and try to claim a classical-hardness result, we would be faced with a discontinuity. The state (4.15) is a mixed state, whereas the initial states used in the sub-universal models are always *pure states*. This means the distribution we would get from the former is something like (up to normalization):

$$P(\mathbf{z}) \sim \sum_{\mathbf{s}\in\{0,1\}^n} e^{-|\mathbf{s}|}\,\mathrm{Tr}\left(|\pm_{\mathbf{z}}\rangle\langle\pm_{\mathbf{z}}|\,\mathcal{U}_z\,\underbrace{|\pm_{\mathbf{s}}\rangle\langle\pm_{\mathbf{s}}|}_{\substack{\text{Contribution to}\\ \text{Mixed State}}}\,\mathcal{U}_z^\dagger\right) \tag{4.17}$$

Which is *not* the same as the one produced by an IQP circuit:

$$P(\mathbf{z}) \sim \mathrm{Tr}\left(|\pm_{\mathbf{z}}\rangle\langle\pm_{\mathbf{z}}|\,\mathcal{U}_z\,\underbrace{|+\rangle\langle+|}_{\text{Pure State}}\,\mathcal{U}_z^\dagger\right) \tag{4.18}$$

The latter, (4.18), is classically hard to produce samples from, but we cannot say the same about the former, even though it is the sum of two IQP-like distributions. This is because we cannot say that just because two distributions are hard to sample from individually, this does not mean their *sum* is also hard to sample from. Indeed there are simple examples where this is not the case.

4.1.3 The Ising Born Machine

The Quantum Boltzmann Machine relies crucially on the preparation of thermal states. However, motivated by the previous section we can ask the question; "*What if we do not prepare a thermal state to sample from, but instead a single pure state?*". This idea arises from examining the nature of how a Boltzmann Machine generates statistics. In the classical case (and quantum case) the distribution which we try to train is produced from a statistical ensemble of possible configurations of the BM (or graph equivalently).

[13] We will show this explicitly in Section (4.1.3).

However, this randomness is purely classical, and represents only our lack of information about the instantaneous state of the system. The introduction of quantum techniques opens the path to generate a distribution, not from statistical uncertainty, but instead from the *fundamental* randomness inherent in Quantum Mechanics *itself*. This possibility is hinted at by introducing the transverse field into the QBM Hamiltonian, (4.8). This QBM also partially exploits the 'quantum' nature of the system, since the resulting state is no longer diagonal in the computational basis.

This line of thought has led to the definition and development of the *Born Machine*, by [22], [48] during the course of this dissertation, and the techniques of which have been incorporated into it. The Born Machine definition sprung from tensor network approaches to define generative algorithms and the connection between physical systems and machine learning problems, [21, 33, 38, 45, 47, 56, 71].

The 'Born Machines' defined by [22] implicitly assume that *any* quantum state can define a Born Machine, mixed or pure. In this respect, the QBM could be viewed as *both* a Boltzmann Machine and a Born Machine. However, since we are interested in a very specific case of these Born Machines (i.e. those which could demonstrate quantum supremacy) we will only consider pure states, $|\psi\rangle$. For clarity, when we say 'Born Machine' in the rest of this work, we really mean *Pure State* Born Machines. This distinction allows a nice contrast between the two types of randomness inherent in quantum mechanics, the intrinsic *quantum* randomness derived from measurements (in the Pure State Born Machine), and the *classical* randomness arising from statistical uncertainty (in the Boltzmann Machine). The Quantum Boltzmann Machine possesses some randomness of each type.

The probability distribution of measuring a quantum state, $|\psi\rangle$ in the computational basis, with outcome string, $\mathbf{z}$ is given by Born's Rule:

$$p(\mathbf{z}) = |\langle \mathbf{z}|\psi\rangle|^2 \tag{4.19}$$

Since the state is pure, we have no normalisation factor because $p(\mathbf{z}) \leq 1$ by design, in contrast to [22]. This simple equation, (4.19), is the core of our Born Machine. Now, how can we train this model to reproduce some target probability distribution? To do so, we first must introduce some parameters in the quantum state to train: $|\psi\rangle \rightarrow |\psi_\theta\rangle$. This can be done using the new technique broadly referred to as *quantum circuit learning* (QCL), which has been used both for discriminative and generative models, [25, 29, 40, 48, 59]. This technique involves applying a parametrized quantum circuit to some initially prepared state, using a sequence of unitary operators with parameters, θ,

$U(\theta)$. As an example, [29], uses QCL for a classification algorithm.

We will follow the method and notations of [48] as we will be using an approach similar to theirs in order to derive a generative learning algorithm.

- We are given some data distribution which we wish to learn, π.
- We have a parametrized circuit, $\mathcal{U}_\theta$, with parameters θ operating on an initial state (simply n qubits in the computational basis), $|0\rangle^{\otimes n}$.
- The outcome statistics are governed by measurements in the computational basis, $M_{\mathbf{z}} = |\mathbf{z}\rangle\langle\mathbf{z}|$. These measurements are all orthogonal and the full set $\{M_{\mathbf{z}}\}_{\mathbf{z}\in\{0,1\}^n}$ gives all possible 2^n binary strings of length n. The outcomes are bitstrings $\mathbf{z} = \{z_1, \ldots, z_n\} \in \{0,1\}^n$, corresponding to the computational basis states $|0\rangle, |1\rangle$ respectively.[14]
- The probability distribution is then:

$$p_\theta(\mathbf{z}) = \left|\langle\mathbf{z}|\,\mathcal{U}_\theta\,|0\rangle^{\otimes n}\right|^2 = \mathrm{Tr}\left(|\mathbf{z}\rangle\langle\mathbf{z}|\,\mathcal{U}_\theta\,|0\rangle^{\otimes n}\langle 0|^{\otimes n}\,\mathcal{U}_\theta^\dagger\right) \tag{4.20}$$

The second question to ask is: "How to we choose the parametrized circuit?". For our purposes it is sufficient to choose one which is complex enough to demonstrate quantum supremacy, but still as shallow as possible to be easily implementable on NISQ devices. This is in contrast to the previous works in the area, [25, 48] which aim to use as deep a circuit as possible in order to have enough parameters to capture the complexity of a target probability distribution. The question we are attempting to answer is somewhat the reverse:

> *Given* a quantum circuit layout which has the potential to display quantum computational supremacy, *how well* can we learn a given dataset (probability distribution)?

An answer to this question would give a potential application for near term quantum devices, ideally those which can display supremacy; they could be used for machine learning. It is the hope of the community that quantum devices would provide an advantage to classical machine learning, with the HHL algorithm being the main example held up for this purpose. We give an alternative method utilising the ability of quantum computers to solve a sampling problem exponentially faster than is possible classically. Now, it is

[14] The measurement outcomes are actually ± 1, corresponding to the *eigenvalues* of the measurement operators, but for consistency with classical computer science we refer to the eigenvector labels, $0, 1$ as the outcomes instead.

true that these quantum circuit examples (IQP, QAOA) have been *engineered* to be hard to classically sample from, and it is still unknown whether the output distribution from these circuits would be able to learn distributions which we care about in practice. However, [51] demonstrates the ability of quantum circuit learning methods to learn several classes of functions so the answer to this question is optimistic. We illustrate in Section (4.2) how to train the model, in such a way which exploits the ease of sampling from a quantum device. It should be noted that the plan for doing this is to find some cost function (and its gradient) which can be computed more efficiently on a quantum device than is possible on any classical machine. The evaluation of this cost function (and its gradient) is of vital importance to the training algorithm, as we shall see.

The circuit involved in $\mathcal{U}_\theta$, will be one of the three possibilities mentioned in Chapter (3), all of which essentially differ only in the final measurement, but all 3 conjectured to be hard to simulate classically. We will jointly refer to all of these circuit types as an '*Ising Born Machine*' (IBM), since the core ingredient of all three is the Ising interaction:

- Initial Hadamard operation on all n qubits.
- Apply unitary which is diagonal in the computational basis, given by:

$$U_z(\theta) = \exp\left(i \sum_{i<j} J_{ij} Z_i Z_j + i \sum_k b_k Z_k\right) \tag{4.21}$$

  where $\{\theta\} = \{\mathbf{J}, \mathbf{b}\} = \{J_{ij}, b_k\}_{i<j,\forall k}$ are the parameters to be trained.
- Apply a final '*measurement*' unitary, U_f^A, depending on our choice of circuit:
  - $\underline{\mathsf{IQP}(J_{ij}, b_k)}$

$$U_f^H = H^{\otimes n} \tag{4.22}$$

  - $\underline{\text{p} = 1\ \mathsf{QAOA}(J_{ij}, b_k, \Gamma_l)}$

$$U_f^X = \prod_{k=1}^{n} \exp\left(-i\Gamma_l X_l\right) = \exp\left(-i \sum_l \Gamma_l X_l\right) \tag{4.23}$$

    which is a rotation around the Pauli-X axis.
  - $\underline{\mathsf{IQP}_{\text{y}}(J_{ij}, b_k, \Delta_l)}$

$$U_f^Y = \prod_{k=1}^{n} \exp\left(-i\Delta_l Y_l\right) = \exp\left(-i \sum_l \Delta_l Y_l\right) \tag{4.24}$$

    which is a rotation around the Pauli-Y axis.

- Measure in the computational basis, $M_{\mathbf{z}} = |\mathbf{z}\rangle\langle\mathbf{z}|$, with outcome string $\mathbf{z}$.

In the end, the resulting probability distribution is:

$$p_\theta(\mathbf{z}) = \left|\langle\mathbf{z}| U_f^A U_z(\theta) |0\rangle^{\otimes n}\right|^2 = \mathrm{Tr}\left(|\mathbf{z}\rangle\langle\mathbf{z}| U_f^A U_z(\theta) |0\rangle\langle 0|^{\otimes n} U_z^\dagger(\theta) U_f^{A\dagger}\right) \tag{4.25}$$

The superscript, A, indicates what the final measurement basis should be, i.e. $X, Y,$ or H. For now, we also assume the parameters, $\mathbf{\Gamma}, \mathbf{\Delta}$ are untrainable, and just set to a constant. The advantage of including them is to have the *option* to train them, and also to draw parallels to the QBM. A circuit describing the model is given by (4.26):

$|0\rangle^{\otimes n}$ — $H^{\otimes n}$ — $U_z(\theta)$ — U_f^A — ⟋ ═ $\mathbf{z} \in \{0,1\}^n$ (4.26)

In all three of the circuit models mentioned above, the initial state is created after the initial Hadamard operation on all qubits. This state can be clearly seen as the uniform superposition over all possible 2^n binary strings of length n:

$$|s\rangle = H|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{k\in\{0,1\}^n} |k\rangle \tag{4.27}$$

The specific implementation of the diagonal unitaries, $\mathcal{U}_z(\theta)$ are as follows:

$$\mathcal{U}_z(\theta) = \exp\left(i\sum_{i<j} J_{ij} Z_i Z_j + i\sum_k b_k Z_k\right) = \prod_{i<j} e^{iJ_{ij}Z_iZ_j} \prod_k e^{ib_kZ_k} \tag{4.28}$$

Now, we use the following fact from [34], relating a $CZ(\theta)$ gate to the '*Ising*' interaction, $\exp(i\theta Z\otimes Z)$:

$$CZ(\theta) = e^{i\theta|1\rangle\langle 1|_i \otimes |1\rangle\langle 1|_j} = e^{\frac{i\theta}{4}(\mathbb{1}_i - Z_i)\otimes(\mathbb{1}_j - Z_j)} \tag{4.29}$$

$$= e^{\frac{i\theta}{4}\mathbb{1}_i\otimes\mathbb{1}_j} e^{-\frac{i\theta}{4}\mathbb{1}_i\otimes Z_j} e^{-\frac{i\theta}{4}Z_i\otimes\mathbb{1}_j} e^{\frac{i\theta}{4}Z_i\otimes Z_j} \tag{4.30}$$

Where the subscripts, i, j indicate the operators act on qubits i or j respectively. The term $|1\rangle\langle 1|_i \otimes |1\rangle\langle 1|_j$ is written as follows:

$$|1\rangle\langle 1| \otimes |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.31}$$

Taking the matrix exponential of (4.31), with a parameter $i\theta$ gives the $CZ(\theta)$. If $\theta = \pi$, this recovers the maximally entangling CZ gate. Now, we can write the Ising interaction in terms of $CZ(\alpha)$ gates, and local $Z(\beta)$ rotations:

$$e^{i\theta Z_i\otimes Z_j} = e^{-i\theta} e^{\theta\mathbb{1}_i\otimes Z_j} e^{\theta Z_i\otimes\mathbb{1}_j} CZ_{i,j}(4\theta) \tag{4.32}$$

The first term in (4.30) is simply a global phase, which can effectively be ignored because it has no effect on the physical properties of the system. The terms, $e^{\theta \mathbb{1}_i \otimes Z_j}, e^{\theta Z_i \otimes \mathbb{1}_j}$ simply become local Z rotations on qubits, i, j respectively:

$$e^{-i\theta \mathbb{1}_i \otimes Z_j} = \mathbb{1}_i \otimes R_z^j(2\theta) = R_z^j(2\theta) \qquad e^{-i\theta Z_i \otimes \mathbb{1}_j} = R_z^i(2\theta) \otimes \mathbb{1}_j = R_z^i(2\theta)$$

$$\implies \mathcal{U}_\theta = \prod_{i<j} e^{iJ_{ij} Z_i Z_j} \prod_k e^{ib_k Z_k} = \prod_{i<j} e^{-iJ_{ij}} e^{iJ_{ij} \mathbb{1}_i \otimes Z_j} e^{iJ_{ij} Z_i \otimes \mathbb{1}_j} CZ_{i,j}(4J_{ij}) \prod_k R_z^k(-2b_k) \tag{4.33}$$

$$= \prod_{i<j} e^{-iJ_{ij}} CZ_{i,j}(4J_{ij}) R_z^i(-2J_{ij}) R_z^j(-2J_{ij}) \prod_k R_z^k(-2b_k) \tag{4.34}$$

The above, (4.34) implies that the Ising gate can be decomposed into a $CZ(\alpha)$ and two corresponding local Z rotations which are corrections applied to the two qubits on which the controlled gate acts. These extra local rotations can be absorbed into the b_k parameters, and the number of these corrections will depend on the graph structure. For example, Figure (4.3) illustrates a potential qubit layout in a nearest neighbour lattice. Qubit 1 picks up contributions from each entangling gate that operates on it with strength $J_{1i}, \forall i$. The green boxes represent the local magnetic fields, b_k, that operate on each qubit. It should be noted that for a graph link labelled by J_{ij} in the figure, the actual operation which would be applied is $CZ(4J_{ij})$.

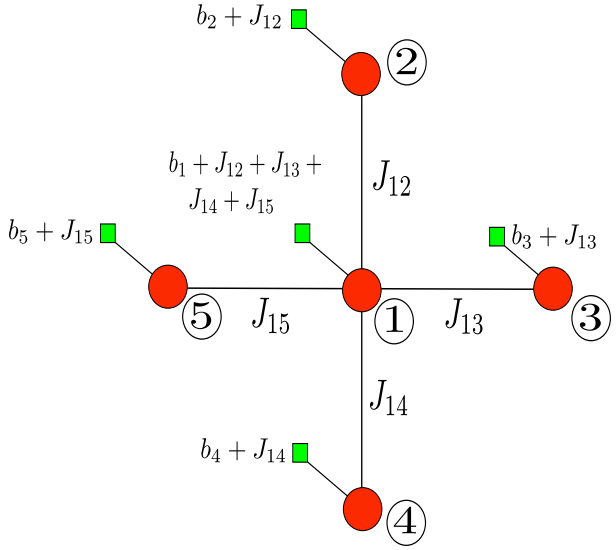


Figure 4.3: Example 5 qubit graph layout with corresponding interaction strengths and local rotations.

4.2 Training the Born Machine

Of crucial importance to this problem is the method used to train these Born Machine models to represent probability distributions. In fact, this is still an open and widely debated topic even in the classical machine learning community, with several questions including:

- Which cost function to use?
- How many parameters to consider?
- How many hyperparameters to introduce and how to control them?
- Gradient based or gradient free?

Firstly, we only consider gradient based methods in this work, in contrast to [10], which uses a gradient free method to train a Born Machine, with a classical method called particle swarm optimization. It is desirable however to have a method which does provide a gradient since it has been observed in classical literature that gradient free methods tend to behave poorly in large parameter spaces, [54].

Secondly, we choose parameters in a similar fashion to the Boltzmann Machine; we imagine a graph structure connecting n qubits, the qubit connections are given by the graph weights on the edges, J_{ij}, and the biases of individual qubits, b_i. In this way, we have at most $n(n-1)/2$ weights, and n biases to train. This also assumes a fully connected graph structure.

4.2.1 Kullback-Leibler Divergence

The initial approach we employ is to use the KL Divergence (specifically the log-likelihood) as our cost function. As described in the previous section, after applying the Ising circuit, the evolved state is:

$$|\psi_f\rangle = U_f^A U_z |s\rangle \implies \rho_f^\theta = U_f^A U_z(\theta) |s\rangle \langle s| U_z^\dagger(\theta) U_f^{A\dagger} \tag{4.35}$$

where the outcome distribution measured over all qubits is:

$$p_\theta(\mathbf{z}) = \left|\langle \mathbf{z}| U_f^A U_z(\theta) |0\rangle^{\otimes n}\right|^2 = \mathrm{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| U_f^A U_z(\theta |0\rangle \langle 0|^{\otimes n} U_z^\dagger(\theta) U_f^{A\dagger}\right) \tag{4.36}$$

Furthermore, it is also possible to emulate the Boltzmann Machine construction, where some qubits are designated as *hidden*, $\mathbf{h}$, and *visible*, $\mathbf{v}$, so the probability distribution

we want to train is actually given by the marginal over the hidden qubits:

$$\begin{aligned} p_\theta(\mathbf{v}) &= \sum_{\mathbf{h}} p_\theta(\mathbf{v},\mathbf{h}) = \sum_{\mathbf{h}} \mathrm{Tr}\left\langle \mathbf{v},\mathbf{h}\right| \rho_f^\theta \left|\mathbf{v},\mathbf{h}\right\rangle \\ &= \sum_{\mathbf{h}} \mathrm{Tr}\left\langle \mathbf{v},\mathbf{h}\right| U_f^A U_z(\theta) \left|s\right\rangle \left\langle s\right| U_z(\theta)^\dagger U_f^{A\dagger} \left|\mathbf{v},\mathbf{h}\right\rangle \\ p_\theta(\mathbf{v}) &= \sum_{\mathbf{h}} \left|\left\langle \mathbf{v},\mathbf{h}\right| U_f^A U_z(\theta) \left|s\right\rangle\right|^2 \end{aligned} \tag{4.37}$$

To begin the training, we must compute the KL divergence, and its gradient:

$$D_{KL}(\pi(\mathbf{v})||p_\theta(\mathbf{v})) = \sum_{\mathbf{v}} \pi(\mathbf{v}) \log\left(\frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})}\right) \tag{4.38}$$

It turns out that minimizing the KL divergence is equivalent to minimizing the negative log-likelihood of the distribution, a technique referred to a Maximum Likelihood Estimation (MLE) defined by:

$$\mathcal{L}_{KL}^\theta = -\sum_{\mathbf{v}} \pi(\mathbf{v}) \log\left(p_\theta(\mathbf{v})\right) \tag{4.39}$$

$$\implies \arg\min_\theta D_{KL}(p_\theta(\mathbf{v})||\pi(\mathbf{v})) = \arg\min_\theta \mathcal{L}_{KL}^\theta \tag{4.40}$$

The equivalence can be easily derived by expanding out the logarithm in (4.38) and using the fact that the entropy of the data distribution is constant through the training. Now, minimizing $\mathcal{L}_{KL}^\theta$ requires taking the derivative w.r.t θ (See Appendices (A.1, A.3) for full derivation):

$$\frac{\partial \mathcal{L}_{KL}^\theta}{\partial \theta} = -\sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} 2\,\mathrm{Im}\, \underbrace{\left\langle \mathbf{v},\mathbf{h}\right| U_f^A U_z \left|s\right\rangle}_{\text{Amp 1}} \underbrace{\left\langle s\right| U_z^\dagger \partial_\theta H_z U_f^{A\dagger} \left|\mathbf{v},\mathbf{h}\right\rangle}_{\text{Amp 2}} \tag{4.41}$$

$$= \sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} \underbrace{\left|\left\langle \mathbf{v},\mathbf{h}\right| U_f^A U_z^{k-\pi/2(\theta)} \left|\mathbf{s}\right\rangle\right|^2}_{p_\theta^-} - \underbrace{\left|\left\langle \mathbf{v},\mathbf{h}\right| U_f^A U_z^{k+\pi/2}(\theta) \left|\mathbf{s}\right\rangle\right|^2}_{p_\theta^+} \tag{4.42}$$

From these expressions, (4.41, 4.42), we can see a difficulty which arises. The computation of the two amplitudes in (4.41) can be computed using the Hadamard test, as described in Appendix (A.2), but this cannot be done in general to multiplicative error. The reason is that these amplitudes (of IQP/QAOA circuits)[15] can be related to instances of problems which are #P-hard to compute in the worst case. It is not believed that quantum computers would be able to solve #P-hard problems, or even NP-hard

[15]Note that the amplitudes, Amp 1,2 (and the probabilities p_θ^+, p_θ^-) are just the parameter shifted versions of the original IQP/QAOA circuits in the Born Machine. If the original version is hard, so is the parameter shifted version as discussed in Section (3.3.4).

ones. Further, it is also not possible to compute the outcome probabilities efficiently, $p_\theta(\mathbf{v})$, which correspond to the modulus squared of the amplitudes. Recall the ability to *strongly* simulate these circuit families would imply one *could* compute these amplitudes. For the example of IQP circuits, ending in U_f^H, it turns out that the amplitudes can be related to a Constraint Satisfaction Problem, the efficient solutions of which would result in a collapse of the PH to the zeroth level, P = NP, [28]. The unlikeliness of this result implies the hardness of computing these quantities efficiently.

This can be seen by examining the Hadamard test as one possible method to compute them. Fundamentally, this is a probabilistic process. It relies on repeatedly running a circuit and computing the probability of an ancilla being in the state $|0\rangle$. However, for those amplitudes which correspond to outcomes that are exponentially unlikely (they have probabilities whose values are exponentially decreasing in the size of the circuit input size), the Hadamard test would have to be run an exponential number of times to *even see* these outcomes *once*. This immediately kills any exponential speedup we want to achieve from the quantum device. Since the gradient (4.41) requires a sum over all sample configurations ($\mathbf{v}$), we would need to compute the probabilities, $p_\theta(\mathbf{v})$, for each configuration of the visible variables $\mathbf{v}$. If a non-negligible fraction of these outcomes is exponentially unlikely, the resulting gradient which we compute (in practice) using only polynomially many runs could differ wildly from the true gradient. In terms of the learning algorithm, this inaccurate gradient could send us in the completely wrong direction in parameter space. Unfortunately, the very nature of choosing a circuit family which is hard to simulate weakly (sample from), is exactly why the KL divergence does not admit a quantum advantage for the task.

However, hope is not lost. The key realisation here is that, to gain a quantum advantage, it is necessary to exploit a feature of these quantum circuits which we already know (or at least strongly believe) gives a quantum speedup. This feature is the ability of quantum circuits to solve *sampling problems* efficiently, i.e. we must use a cost function to compare these two distributions (the Born Machine and the target data distribution) using *only* samples from the distributions. After all, that is exactly what the KL Divergence is doing: comparing the distributions. Admittedly, using a finite number of samples from a distribution will not give an exactly faithful representation, but it could be good enough for our purposes.

Fortunately, a candidate solution was discovered during the course of this dissertation by [48] for exactly the purpose of training a Quantum Circuit Born Machine (QCBM). The authors also note this shortcoming of the KL Divergence in that it can not be written

as a sampling problem on the Born Machine. This is due to the factor of $p_\theta(\mathbf{v})$ in the denominator of (4.41, 4.42). The solution they propose is to use the *Maximum Mean Discrepancy* (MMD), and we will adopt this approach also. Admittedly, they did not approach the problem from a 'supremacy' point of view, they were just interested in devising an efficient method for training the QCBM. Furthermore, they noticed that the MMD served as a good alternative for the KL Divergence to train the Born Machine based on empirical simulations.

4.2.2 Maximum Mean Discrepancy (MMD)

The Maximum Mean Discrepancy is an alternative method to compare two probability distributions, P, Q. It is an example of an *Integral Probability Metric* (IPM) [63], in contrast, the KL Divergence is a form of a so-called ϕ-divergence. In the classical Machine Learning community, alternative cost functions to compare probability distributions are currently being investigated in the quest for better or more efficient ML algorithms. It is conjectured that a cost function used to compare distributions, in for example a generative algorithm, should be perhaps a metric[16]. The Wasserstein Metric is another example of an IPM [63] and which was used to define a Wasserstein Generative Adversarial Network (WGAN). Generative Adversarial Networks are similar to the generative models we examine here, with the difference that they are trained *adversarially*, i.e. there exists a discriminator, working in tandem with the generator, which seeks to determine if the samples produced by the generating algorithm are correct, with respect to the target distribution. However, it has been shown [9] that using the Wasserstein Metric admits gradients which are *biased*[17], and this is clearly an undesirable property to have. In the same work [9], the authors propose the Energy distance or MMD as a potential alternative to either the Wasserstein Metric or the KL Divergence due to its favourable properties, in particular it does not have biased gradients.

The reason the MMD was chosen (as opposed to the Wasserstein for instance) as an alternative to the KL Divergence is twofold. Firstly, and most obviously, it had already been used specifically to train a Born Machine in [48]. The second reason is that it can be estimated efficiently and in an unbiased way using a sampling technique for the quantum circuit learning models, in contrast to the KL Divergence, which cannot.

[16]The KL divergence is in fact a *divergence* and not a metric. It does not take into account differences *between points* in the two distributions

[17]If an estimator, $\hat{\theta}$, of a random variable, θ, is biased, this means there is a discrepancy between the expectation of the estimator and the true variable, i.e. $\mathbf{E}(\hat{\theta}) - \theta \neq 0$.

IPM's are defined by the following equation:

$$\gamma_{\mathcal{F}}(P,Q) = \sup_{\phi\in\mathcal{F}} \left| \int_{\mathcal{M}} \phi dP - \int_{\mathcal{M}} \phi dQ \right| = \sup_{\phi\in\mathcal{F}} \left(\mathbb{E}_P[\phi(x)] - \mathbb{E}_Q[\phi(y)] \right) \tag{4.43}$$

Where P, Q are defined over a measurable space, $\mathcal{M}$. The class of functions which are chosen, $\mathcal{F}$, defines the metric that is used. In our case, the class of functions will be so-called *feature maps*, $\phi \in \mathcal{F}$, where $\mathcal{F} = \{\phi : ||\mathcal{F}||_{\mathcal{H}} \leq 1\}$, $\mathcal{H}$ is the *Reproducing Kernel Hilbert Space* (RKHS) for a reason which will become apparent shortly. In other words, these feature maps take samples from the respective distributions, P, Q (which we want to compare), and map them into the unit ball in the Hilbert Space, where $||\cdot||_{\mathcal{H}}$ is the norm in the space, $\mathcal{H}$. These feature maps are also referred to as '*embeddings*', since they *embed* the samples in the RKHS. If a different class of functions is chosen, one acquires the Wasserstein Metric or the Dudley Metric for instance.

4.2.2.1 MMD Kernel

A key component of the MMD is the kernel function which is used to describe it. This kernel function now opens the door to a vast store of classical knowledge of so-called '*kernel methods*'. The kernel is simply defined by the inner product of the feature map at two different points in the RKHS. In this way, any inner product space can be used as the range of the feature map, however we will always assume these inner product spaces are complete, and hence they form Hilbert Spaces. Intuitively, the introduction of these feature maps and kernels allow the samples (from the probability distributions, P and Q, which could be the output from the Born Machine and the training data for example) to be mapped to a high dimensional space, in which they can be compared more easily. The choice of the kernel function (i.e. the choice of the RKHS) may allow different properties of the distributions to be compared. For clarity, we now reproduce some definitions of various quantities as given in [52, 60], since [60] was the first work to explore the idea of using 'quantum kernels', and this fact leads to the key innovation or novelty of this work. For a comprehensive review of techniques in kernel embeddings, see [52].

Definition 4.1: Kernel

Let $\mathcal{X}$ be a non-empty set. A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ is called a kernel if the Gram matrix K with entries, $K_{m,m'} = \kappa(x^m, x^{m'})$ is positive semidefinite, in other words, if for any finite subset, $\{x^1, \ldots, x^M\} \subseteq \mathcal{X}$ with $M \geq 2$ and $c_1, \ldots, c_M \in \mathbb{C}$:

$$\sum_{m,m'=1}^{M} c_m c_{m'} \kappa(x^m, x^{m'}) \geq 0 \tag{4.44}$$

Every feature map has a corresponding kernel, given by the inner product on the Hilbert Space:

Theorem 4.1: Feature Map Kernel

Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature map. The inner product of two inputs mapped to a feature space defines a kernel via:

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}} \tag{4.45}$$

The full proof that this feature map gives rise to a kernel can be found in [60]. The above shows that if we can define a feature map to a Hilbert Space, a kernel is given by the inner product of two mappings in that Hilbert Space.

Finally, we give a formal definition of a Hilbert space and Reproducing Kernel Hilbert Spaces for completeness [62].

Definition 4.2: Hilbert spaces

Let $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ be an inner product on a real or complex vector space $\mathcal{V}$ and let $|| \cdot || : \mathcal{V} \to \mathbb{R}$ be the associated norm defined as:

$$||\mathbf{v}|| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}, \forall \mathbf{v} \in \mathcal{V} \tag{4.46}$$

The vector space $\mathcal{V}$ is a Hilbert space if it is complete with respect to this norm, i.e., every Cauchy sequence in $\mathcal{V}$ has a limit which is an element of $\mathcal{V}$.

Definition 4.3: Reproducing Kernel Hilbert Space (RKHS)

Let $\mathcal{X}$ be a non-empty input set, and let $\mathcal{H}$ be a Hilbert Space of functions. Let $\langle \cdot, \cdot \rangle$ be an inner product defined on $\mathcal{H}$ (which gives rise to a norm defined as above). $\mathcal{H}$ is a RKHS if every point evaluation is a continuous functional $F : f \to f(x), \forall x \in \mathcal{X}$. This means there exists a function, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ such that:

$$\langle f, \kappa(\mathbf{x}, \cdot) \rangle = f(\mathbf{x}) \forall f \in \mathcal{H}, \mathbf{x} \in \mathcal{X} \tag{4.47}$$

with $\kappa(\mathbf{x}, \cdot) \in \mathcal{H}$

The Hilbert Space is '*reproducing*' because the inner product of the kernel at a point in the sample space, with a function on the Hilbert space, in some sense, *evaluates* or reproduces the function at that point. The form of the kernel assumed by [60] is the one induced by the inner product on a quantum Hilbert space:

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle \tag{4.48}$$

Given that we are trying to exploit some advantage by using quantum computers, this definition is natural. However, it should be noted that the above definition of a kernel defines it to be a mapping from the sample space to $\mathbb{C}$, i.e. the kernel is the inner product of two wavefunctions which encode two samples and is therefore a transition amplitude. For our purposes, remaining consistent with [40], the kernel is defined as the real transition *probability* instead. As mentioned in [60] it is possible to define a kernel this way by taking the modulus squared of (4.48), and defining the RKHS as follows:

Theorem 4.2: Quantum RKHS

let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature map over an input set $\mathcal{X}$, giving rise to a *real* kernel: $\kappa(\mathbf{x}, \mathbf{y}) = |\langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle|^2$. The corresponding RKHS is therefore:

$$\mathcal{H}_\kappa = \{ f : \mathcal{X} \to \mathbb{R} | f(\mathbf{x}) = |\langle w | \phi(\mathbf{x}) \rangle|^2, \forall \mathbf{x} \in \mathcal{X}, w \in \mathcal{H} \} \tag{4.49}$$

This is so that we can directly use the kernel described in [40]. The final ingredient is the discussion of the computability of this kernel function, since the evaluation of the MMD, and training process requires this to be done many times. If we can find a kernel which can be computed more efficiently via quantum methods, we will have another quantum advantage.[18]

[18] On top of the classical sampling hardness.

In [60], the authors investigate kernels which are classically efficiently computable in order to facilitate testing of their classification algorithm, for example they only study so-called Gaussian states, which are known to be classically simulable. Gaussian states are a key ingredient in CV quantum computing. However, in our case, we require the kernel specifically to *not* be classically computable. For this reason, we can use the kernel defined in [40]:

$$\Phi : \mathbf{x} \in \mathcal{X} \to |\Phi(\mathbf{x})\rangle \tag{4.50}$$

where it is constructed by the following quantum circuit:

$$|\Phi(\mathbf{x})\rangle = \mathcal{U}_{\Phi(\mathbf{x})} |0\rangle^{\otimes n} = U_{\Phi(\mathbf{x})} H^{\otimes n} U_{\Phi(\mathbf{x})} H^{\otimes n} |0\rangle^{\otimes n} \tag{4.51}$$

so the resulting kernel is the squared overlap between two of these states evaluated at different points:

$$\kappa(\mathbf{x}, \mathbf{y}) = |\langle \Phi(\mathbf{x}) | \Phi(\mathbf{y}) \rangle|^2 \tag{4.52}$$

Of particular interest, and to add further motivation to why this particular type of circuit is chosen, is its relationship to the IQP model. This can be seen through the choice of the unitary encoding operators, $U_{\Phi(\mathbf{x})}$:

$$U_{\Phi(\mathbf{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\mathbf{x}) \prod_{i \in S} Z_i \right) \tag{4.53}$$

This is exactly the same form as in (3.4), with the parameters, θ replaced by the feature map with sample $\mathbf{x}$, $\phi_S(\mathbf{x})$. However, this is *not* an IQP circuit due to the extra final layer of these diagonal gates, which are absent in an IQP circuit.

It should be noted that this is experimentally favourable to work alongside the original Born Machine circuit since the same setup (i.e. layout of entanglement gates, and single qubits rotations) is required for both circuit types, albeit with different parameters. Just like in the IQP scenario, only single and two-qubit are operations are required:

$$U_{\phi_{\{l,m\}}(\mathbf{x})} = \exp\left(i\phi_{\{l,m\}}(\mathbf{x}) Z_l Z_m\right) \qquad U_{\phi_{\{k\}}(\mathbf{x})} = \exp\left(i\phi_{\{k\}}(\mathbf{x}) Z_k\right) \tag{4.54}$$

Exactly as in the actual Born Machine, this kernel circuit can be decomposed into $CZ(\alpha), Z(\beta)$ gates as in (4.34):

$$\implies U_{\phi(\mathbf{x})} = \prod_{i<j} e^{i\phi_{\{ij\}}(\mathbf{x}) Z_i Z_j} \prod_k e^{i\phi_{\{k\}}(\mathbf{x}) Z_k} \tag{4.55}$$

$$= \prod_{i<j} e^{-i\phi_{\{ij\}}} CZ_{i,j}(4\phi_{\{ij\}}(\mathbf{x})) R_z^i(-2\phi_{\{ij\}}(\mathbf{x})) R_z^j(-2\phi_{\{ij\}}(\mathbf{x})) \prod_k R_z^k(-2\phi_{\{k\}}(\mathbf{x})) \tag{4.56}$$

To encode the data, the feature map is chosen such that $\phi_{\{lm\}}(\mathbf{x}) = x_l x_m, \phi_{\{k\}}(\mathbf{x}) = x_k$, or in general $\phi_S(\vec{x}) = \prod_{i \in S} x_i$. In [40], the samples, $\mathbf{x} = (x_1, \ldots, x_n)$, are elements of $\mathbb{R}^d$, so individual elements, x_j, are real numbers. In our case however, the samples are simplified to be only binary vectors, such that the feature map is given by:

$$x_l x_m = \begin{cases} \frac{\pi}{4} \iff x_l = x_m = 1 \\ 0 \text{ otherwise.} \end{cases} \qquad x_k = \begin{cases} \frac{\pi}{4} \iff x_k = 1 \\ 0 \text{ otherwise} \end{cases} \tag{4.57}$$

The choice of this kernel is motivated by the conjecture proposed in [40], which says this overlap will be classically hard to estimate up to polynomially small error, whereas the kernel can be computed using a quantum device, up to an additive sampling error of $\tilde{\epsilon}$ using $O(\tilde{\epsilon}^{-2})$ sampling shots. An approximation, $\hat{K}$, of the Gram matrix for the kernel, $K_{\mathbf{x},\mathbf{y}} = \kappa(\mathbf{x}, \mathbf{y})$, which has $O(|B| \times |D|)$ non-trivial entries, (where we have $|B|$ samples from the Ising Born Machine, and $|D|$ from the data) that is $\epsilon-$close in operator norm[19] $||K - \hat{K}|| \leq \epsilon$ can be determined using $R = O(\epsilon^{-2}|B|^2 \times |D|^2)$ measurement shots.

It is noted in [40] that if we only care about computing the overlap between the states to a multiplicative error (or exactly) it is sufficient to ignore the second layer of diagonal gates, $U_{\phi(\mathbf{x})}$ in the kernel, as will be #P-hard, as in the above IQP circuit. However, in this case, to be able to rule out an *additive* error approximation[20], it is necessary to add the second encoding layer.

The specific complexity theory results indicating why this type of kernel is conjectured to be hard to compute classically is given in detail in [40] and so we will not repeat it here.

4.2.2.2 MMD Estimator

Now that we have a candidate kernel, we can return to the computation of the MMD. In this part, we draw heavily on [63], which shows that the MMD can be estimated as follows given i.i.d. samples from two distributions, $\mathbf{x} = (x_1, \ldots, x_m)$, $\mathbf{y} = (y_1, \ldots, y_n)$ drawn from P, Q respectively:

$$\text{MMD}[P, Q] \approx \sup_{\phi \in \mathcal{H}} \left(\frac{1}{m} \sum_{i=1}^{m} \phi(x_i) - \frac{1}{n} \sum_{j=1}^{n} \phi(y_j) \right) \tag{4.58}$$

where $\mathcal{H} = \{\phi : ||\phi||_{\mathcal{H}} \leq 1\}$, indicates we are dealing with the unit ball in a Hilbert Space, and hence it is the MMD (as the specific example of an IPM). This estimator is

[19] The operator norm is defined by: $||O|| = \inf\{k \geq 0 : ||Ov|| \leq k||v|| \forall v \in \mathcal{V}\}$ where $\mathcal{V}$ is the vector space on which O acts.

[20] This is more experimentally relevant.

simply derived by replacing the expectation values in (4.43) with their empirical values, computed using the finite samples.

If we define the *mean embedding*:

$$||\mu_P||_{\mathcal{H}} = \mathbb{E}_P(\kappa(\mathbf{x}, \cdot)) = \mathbb{E}_P[\phi(\mathbf{x})] \tag{4.59}$$

which is simply the expectation value of the distribution, P, when mapped to the Hilbert space by the feature map, [12, 36], we can show the MMD is exactly the difference in mean embeddings between the two distributions:

$$\begin{aligned}
\mathsf{MMD}[P, Q] &= \sup_{||\phi||\leq 1} \left| \int \phi(\mathbf{x}) dP(\mathbf{x}) - \int \phi(\mathbf{y}) dQ(\mathbf{y}) \right| && (4.60)\\
&= \sup_{||\phi||\leq 1} \left| \langle \phi, \int \kappa(\mathbf{x}, \cdot) dP(\mathbf{x}) \rangle - \langle \phi \int \kappa(\mathbf{y}, \cdot) dQ(\mathbf{y}) \rangle \right| && (4.61)\\
&= \sup_{||\phi||\leq 1} |\langle \phi, \mu_P - \mu_Q \rangle| = ||\mu_P - \mu_Q||_{\mathcal{H}} && (4.62)
\end{aligned}$$

Given this, a consistent unbiased estimator for the squared MMD can be derived, [63], which is exactly the form used in [48] (in the discrete case) to train their QCBM:

$$\begin{aligned}
\mathcal{L}_{\mathsf{MMD}} &= \mathsf{MMD}^2 = ||\mathbb{E}_P[\phi(\mathbf{x})] - \mathbb{E}_Q[\phi(\mathbf{x})]||^2_{\mathcal{H}} && (4.63)\\
\mathcal{L}_{\mathsf{MMD}} &= \underset{\mathbf{x}\sim P, \mathbf{y}\sim P}{\mathbb{E}}(\kappa(\mathbf{x}, \mathbf{y})) + \underset{\mathbf{x}\sim Q, \mathbf{y}\sim Q}{\mathbb{E}}(\kappa(\mathbf{x}, \mathbf{y})) - \underset{\mathbf{x}\sim P, \mathbf{y}\sim Q}{2\mathbb{E}}(\kappa(\mathbf{x}, \mathbf{y})) && (4.64)
\end{aligned}$$

Where the estimator is defined as follows, given $\mathbf{x} = (x_1, \ldots, x_m)$, $\mathbf{y} = (y_1, \ldots, y_n)$ drawn from P, Q:

$$\widehat{\mathcal{L}_{\mathsf{MMD}}} = \widehat{\mathsf{MMD}}^2[P, Q] = \frac{1}{m(m-1)} \sum_{i\neq j}^{m} \kappa(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i\neq j}^{m} \kappa(y_i, y_j) - \frac{2}{mn} \sum_{i,j}^{m} \kappa(x_i, y_j) \tag{4.65}$$

The estimator should be consistent and unbiased, where consistency is defined as convergence to the true value (of the expectation value) in the limit of large samples. For the estimator to be unbiased, the expectation value of the estimator should be equal to the true expectation value of the quantity being estimated.

As shown in [63], the above (4.65) demonstrates both of these properties, and furthermore it converges fast in probability to the true MMD:

$$|\widehat{\mathsf{MMD}}[P_n, Q_m] - \mathsf{MMD}[P, Q]| \leq O_{P,Q}(n^{-1/2} + m^{-1/2}) \tag{4.66}$$

where P_n, Q_m are the empirical distributions acquired using n, m samples from P, Q respectively.

In contrast, estimators for the KL divergence can be derived which are also consistent, but which are slow to converge, requiring many more samples and which can depend on the data dimension. For example, in d-dimensional spaces, estimators of the KL-divergence can be arbitrily slow to converge, [70]. The derivation of the MMD loss estimator requires only that the kernel be a bounded and measurable function, which it will be in our case: the overlap between two states is bounded above by 1, and the sample space consists of binary strings.

Now, to compute the derivative of this cost function, we follow the method of [48]. In that work, the gates with trainable parameters, η, are of the form $\exp\left(-i\frac{\eta}{2}\Sigma\right)$, where Σ can be series of operators satisfying $\Sigma^2 = 1$. The derivation of the gradient is given in Appendix (A.3). However, it is necessary to point out some differences between our circuit, and that of [48], as some care is required with the numerical factors.

The gradient of an observable of the circuit, B, with respect to a parameter, η, is given by:

$$\frac{\partial \langle B \rangle_\eta}{\partial \eta} = \frac{1}{2}\left(\langle B \rangle_{\eta^+} - \langle B \rangle_{\eta^-}\right) \tag{4.67}$$

where $\eta^\pm$ are parameter shifted versions of the original circuits. In our case, we have gates, $\{D_1(b_k) = \exp\{ib_k Z_k\}, D_2(J_{ij}) = \exp\{iJ_{ij}Z_iZ_j\}\}$. As noted in [48], taking the observable, $B = |\mathbf{z}\rangle\langle\mathbf{z}|$, we arrive at the following form of the gradient of the probabilities:

$$\frac{\partial p_\theta(\mathbf{z})}{\partial \theta_k} = \frac{1}{2}\left(p_{\theta_k^+}(\mathbf{z}) - p_{\theta_k^-}(\mathbf{z})\right) \tag{4.68}$$

As mentioned above, this gradient requires computing the parameter shifted versions of the original circuit, $p_{\theta_k^\pm}$. This notation indicates that the k^{th} parameter in the original circuit has been shifted by a factor of $\pm\frac{\pi}{2}$; $\theta_k^\pm = \theta_k \pm \pi/2$.

However since the unitaries we apply are not quite parametrized the same as in [48], the form of our gradient is actually given by:

$$\frac{\partial p_\theta(\mathbf{z})}{\partial \theta_k} = \left(p_{\theta_k^-}(\mathbf{z}) - p_{\theta_k^+}(\mathbf{z})\right) \tag{4.69}$$

As such, the derivative of this loss function with respect to a given parameter, θ_k, is given by:

$$\frac{\partial \mathcal{L}_{\mathsf{MMD}}}{\partial \theta_k} = \underset{\mathbf{x}\sim p_{\theta_k^-},\mathbf{y}\sim p_\theta}{2\mathbb{E}}\left(\kappa(\mathbf{x},\mathbf{y})\right) - \underset{x\sim p_{\theta_k^+},y\sim p_\theta}{2\mathbb{E}}\left(\kappa(\mathbf{x},\mathbf{y})\right) - \underset{\mathbf{x}\sim p_{\theta_k^-},\mathbf{y}\sim \pi}{2\mathbb{E}}\left(\kappa(\mathbf{x},\mathbf{y})\right) + \underset{\mathbf{x}\sim p_{\theta_k^+},\mathbf{y}\sim \pi}{2\mathbb{E}}\left(\kappa(\mathbf{x},\mathbf{y})\right) \tag{4.70}$$

$$\frac{\partial \mathcal{L}_{\mathsf{MMD}}}{\partial \theta_k} \approx \frac{2}{pm}\sum_{i,j}^{p,m}\kappa(a_i,x_j) - \frac{2}{qm}\sum_{i,j}^{q,m}\kappa(b_i,x_j) - \frac{2}{pn}\sum_{i,j}^{p,n}\kappa(a_i,y_j) + \frac{2}{qn}\sum_{i,j}^{q,n}\kappa(b_i,y_j) \tag{4.71}$$

where we have p, q samples, $\mathbf{a} = \{a_1, \ldots, a_p\}, \mathbf{b} = \{b_1, \ldots, b_q\}$ drawn from the parameter shifted circuits, $p^-_{\theta_k}(\mathbf{a}), p^+_{\theta_k}(\mathbf{b})$ respectively and m, n samples, $\mathbf{x} = \{x_1, \ldots, x_m\}, \mathbf{y} = \{y_1, \ldots, y_n\}$ drawn from the the original Born circuit and the data distribution respectively, $p_\theta(\mathbf{x}), \pi(\mathbf{y})$.

The full algorithm is given by Algorithm (1) in Appendix (A.4), and for illustration, (4.4), gives a heuristic picture of the operation of the algorithm.

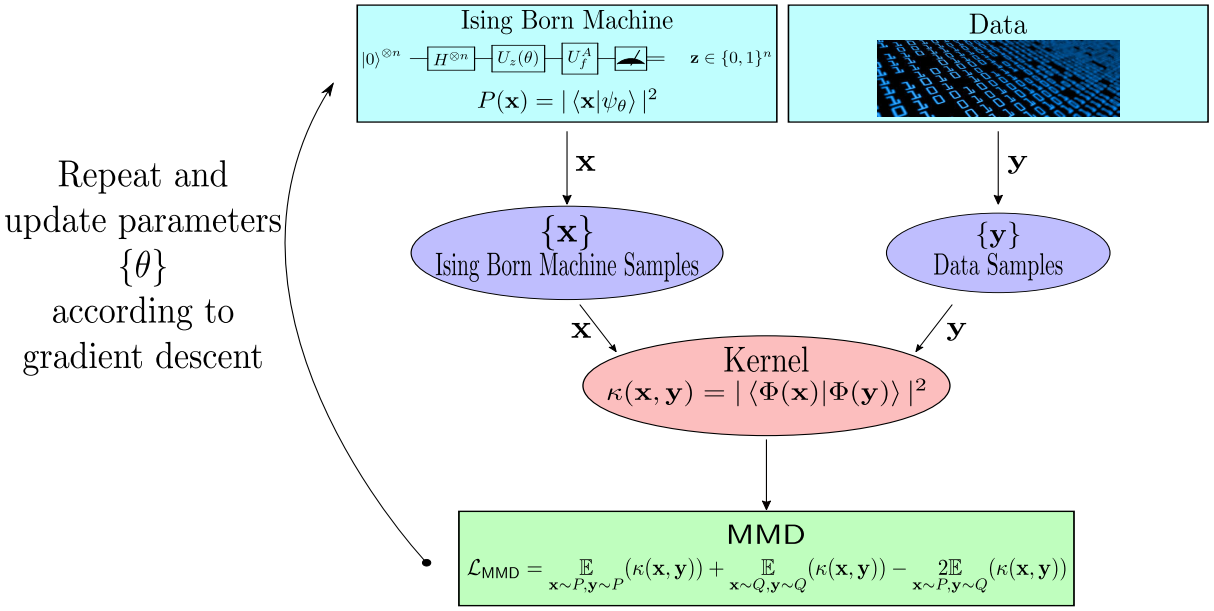


Figure 4.4: Illustrated operation of the Ising Born Machine, with MMD and quantum-hard kernel.

4.3 Numerical Results

In this section, we present numerical results to test the functionality of the algorithm in its ability to 'learn' a given data distribution. We implemented the algorithm with the two above cost functions, namely the KL Divergence and the MMD. The results were produced using Rigetti's quantum simulator and the Forest platform, which operates as a Python library called PyQuil, [2].

The 'data' was taken to be a simple toy example, which was the same used to train the QBM, [8], and the QABoM [68], namely a distribution over binary strings of length n. The samples produced by the Ising Born Machine are simply binary strings also, since

they result from measurements.

$$\pi(\mathbf{v}) = \frac{1}{M}\sum_{k=1}^{M} p^{N-d_{\mathbf{v}}^k}(1-p)^{d_{\mathbf{v}}^k} \tag{4.72}$$

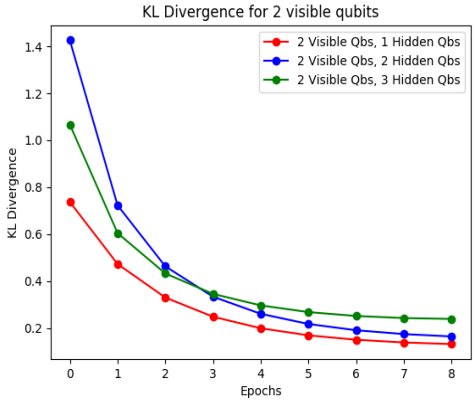
To generate this artificial data, M 'Bernoulli' hidden modes are randomly chosen. Each mode (k) is randomly chosen to be a binary string, $\mathbf{s}^k = [s_1^k, \ldots, s_{n_v}^k]$, and a given configuration on the visible variables (i.e. one data sample), $\mathbf{v}$, is produced with a probability which depends on the Hamming weight, $d_{\mathbf{v}}^k$, between each mode string, k, and the given configuration, $\mathbf{v}$. It should also be noted that these hidden modes are not necessarily related to the hidden qubits used in the QBM or our Born Machine trained with the KL Divergence. As an illustration, if a single centre mode ($M = 1$) over two bits is chosen to be 01, then all possible binary strings of length 2 are generated with probabilities:

$$\mathbf{v} = \begin{cases} 00, \implies d_{00}^1 = 1 & p^{2-1}(1-p)^1 \\ 01, \implies d_{01}^1 = 0 & p^{2-0}(1-p)^0 \end{cases} \qquad \mathbf{v} = \begin{cases} 10, \implies d_{10}^1 = 2 & p^{2-2}(1-p)^2 \\ 11, \implies d_{11}^1 = 1 & p^{2-1}(1-p)^1 \end{cases} \tag{4.73}$$

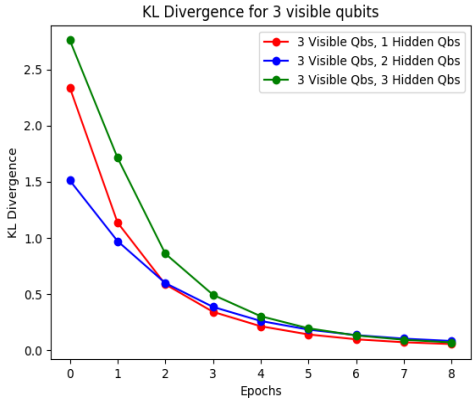
In the case of the KL Divergence[21], the gradient can be computed exactly using the Hadamard test as described in Section (4.2.1), but clearly this method doesn't scale to large numbers of input qubits, n, due to the hardness of computing outcome amplitudes or probabilities from quantum circuits. Furthermore, since a simulator was used, it is possible to directly access the output amplitudes of the Born Machine circuit, but obviously this would not be possible on a real quantum device. It should be noted that due to the random initialization of the parameters, $\{J_{ij}, b_k\}$[22], on a fully connected graph, the initial cost function values would be different in separate runs of the algorithm. As such, one should repeat the algorithm with the same conditions multiple times and take averages, but this was not possible here due to the time cost of quantum simulation. The $\mathsf{QAOA}(J_{ij}, b_k, \pi/4)$ circuit class was chosen for all the following. Also important to note is that *vanilla* or *batch* gradient descent was consistently used, which means to update a single parameter, *all* the samples from the Born Machine, and the training data were used to compute the gradient. In the case of the exact gradients, this effectively means the exact probabilities were used. This is typically the least efficient, but one of the most robust method for gradient descent in classical ML.

[21] We will use the terms KL Divergence and log-likelihood interchangeably since minimising one is equivalent to minimising the other, as discussed in Section (4.1.2.2)

[22] The parameters are chosen initially to be uniform on the interval $(0, \frac{\pi}{4})$



(a) KL Divergence for two qubits with one, two and three hidden qubits.

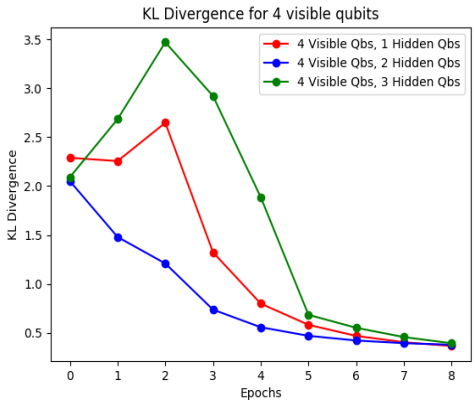


(b) KL Divergence for three qubits with one, two and three hidden qubits.

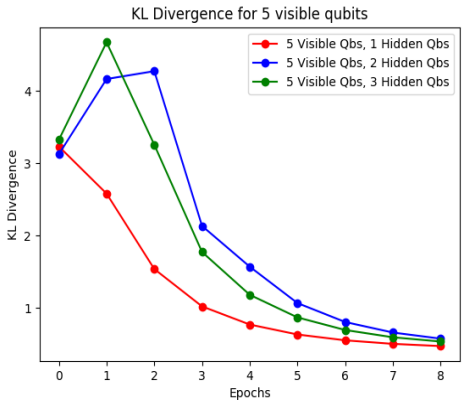
Figure 4.5: KL Divergence for two and three qubits, learning binary strings of length two and three respectively.

Figures (4.5, 4.6) show the results for learning the distribution (4.72) using the KL Divergence when both the gradient of the log-likelihood and the toy data distribution were computed exactly. In the case of the former, this can be done via the simulation of the Hadamard test, which produces the exact gradient since we have access to the amplitudes of the test circuit. In the latter case, the probabilities of the distribution can be written analytically. In all Figures (4.5-4.6), a constant learning rate of $\eta = 0.1$ was chosen, with the number of hidden modes in the data being $M = 8$, consistent with [8]. As the number of qubits increases, the distributions (produced by the Born Machine and the toy data) are more likely to initialize further away (when distance is measured according to the KL Divergence), hence the higher initial values of the KL divergence. Due to the fact the exact gradient is being computed, fast convergence is observed, in all reasonable cases, the KL divergence reaches minimums of less than one. In standard ML models, the number of epochs would typically be on the order of hundreds or thousands. These runtimes were unreachable in this case for two reasons. The first is the exponential overhead accumulated by simulating a quantum computation on a classical device with increasing number of qubits. The second is the intractability of computing the gradient of the KL loss function, which must be done exactly.

Despite the numerous shortcomings of the model, which we were aware of beforehand, these results are presented for one reason only. This is to indicate the ability of the model *to learn at all*, which it clearly does to some extent. Also, the hidden qubits



(a) KL Divergence for four qubits with one, two and three hidden qubits.



(b) KL Divergence for five qubits with one, two and three hidden qubits.

Figure 4.6: KL Divergence for four and five qubits, learning binary strings of length four and five respectively.

were introduced to draw parallels to the Boltzmann Machine, they will be dropped when investigating the MMD. This is because it is not clear if the sampling hardness results would hold if some of the qubits were marginalized over to produce the final distribution. This requires further investigation and is mentioned in (5.1).

Secondly, the MMD cost function, (4.65), and its gradient, (4.71), was used to train the model. Since the MMD requires the computation of a kernel, which encodes samples from the data and the Ising Born Machine, two different kernel choices were investigated. The first was the quantum kernel of [40], (4.52). The second was the Gaussian kernel, (4.74), used in [48], since this is a common choice in classical ML literature also, to act as a benchmark to previous work.

$$\kappa(\mathbf{x}, \mathbf{y}) = \frac{1}{c} \sum_{i=1}^{c} e^{-\frac{1}{2\sigma_i}|x-y|^2} \tag{4.74}$$

where c is the number of Gaussians we use to explore the difference between the data and the model distributions at different scales, which are controlled by the bandwidth parameters, σ_i. $|\cdot|$ denotes the l_2 norm in this case since the samples, $\mathbf{x}, \mathbf{y}$ are bit strings. In all the following, $\sigma = [0.25, 10, 1000]$ was chosen to be consistent with [48]. It should be noted, that the approach of using the MMD with this Gaussian kernel for the Born Machine has been studied in that work, we are adding the novelty of the quantum-hard kernel. Also, the dataset they used as a target distribution was not the one we chose.

When training the model using the MMD, two approaches were taken. The initial

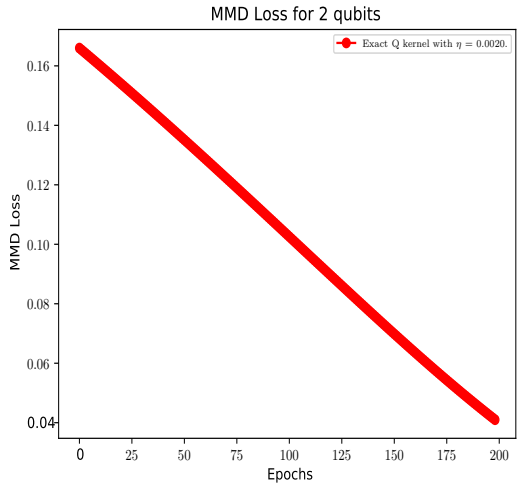
approach was to compute the MMD cost function and its gradient *exactly*. This was possible due to the nature of quantum simulators, which allow direct access to the circuit amplitudes (and hence probabilities). The method was essentially the same as computing the exact gradient for the KL Divergence above[23], and done to validate the potential of using the MMD as a cost function. The initial hopes for this were optimistic, due to the results of [48] which indicated the MMD was a good alternative to the KL Divergence, in the sense that the observed behaviour of both cost functions was similar for circuits of the same size. Also, to speed up the training in the exact case, the quantum and classical kernels were precomputed. It should be noted though that this precomputation would not be possible in supremacy-sized quantum computers, where a sampling approach is used, since the (quantum) kernel would need to be computed on-the-fly and it would require an exponentially large (classical) memory to store the kernel values.

Figures (4.7a-4.9a) illustrate that the algorithm can also learn when using the exact MMD for certain learning rates with two, three and four qubits respectively. Figures (4.7b-4.9b) illustrate the difference that the learning rate can have on the algorithm, in cases where it is too large (~ 0.1), the algorithm can actually perform gradient *ascent*, and settle in a local maximum. This could be resolved by allowing the algorithm to run for more Epochs in some cases. It was noticed that the algorithm behaved differently on separate runs of the algorithm, mainly due to the random initialisation of the parameters. Again, this illustrates the need to repeat the algorithm many times to get a more thorough picture of what is happening, and to fully study its ability to learn. Also, we mentioned the desire to study and compare the difference when using the quantum-hard kernel, (4.52), versus a kernel which is classically computable, the Gaussian kernel (4.74). Firstly, it should be noted that we did not expect to see an obvious quantum advantage in these small test cases (the data and model over 2-4 qubits is classically simulable), it would only be when scaled up to supremacy-sized quantum where we could hope to see an improvement in practice. Furthermore, it clearly is unfair to compare these kernels directly, due to the fact that they each cause the MMD to be evaluated differently, and as such, a more thorough comparison needs to be undertaken. These results can be seen in Figures (4.7c-4.9c). Given the above scepticism, it is encouraging to see a faster convergence rate apparent when using the quantum-hard kernel (the red line) with all three cases. Granted, this behaviour was dependent on the learning rate being small, $\eta = 0.001$, but it is cause for

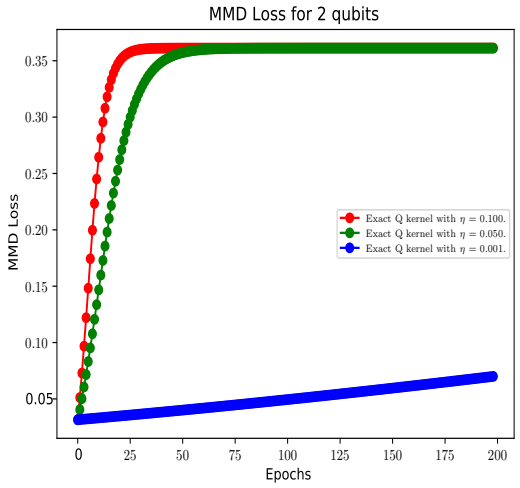
[23]Minus the need for the Hadamard test, since the gradient of the MMD requires simply expectation values, which can be derived directly by computing the kernel, and outputting the required probabilities from the quantum simulator for the various circuits, $p_\theta, p_\theta^\pm$.

hope nonetheless.

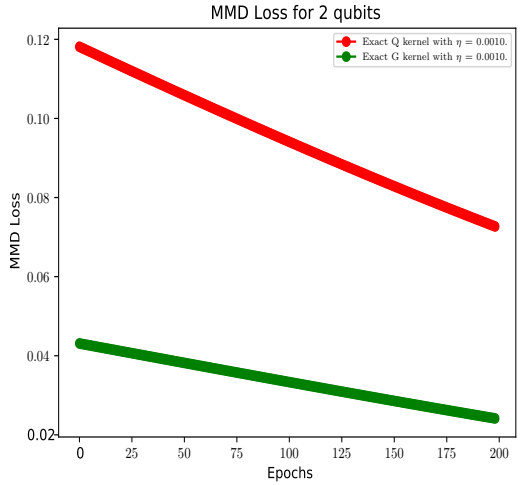
The second was a sampling approach which is the ultimate goal of a realistic implementation of the algorithm, i.e. drawing a certain number of samples from the Born Machine and the same toy data distribution. In particular, a finite samples number of samples was used to compute the expectation values used in (4.71), where the samples were drawn i.i.d from whichever distribution was required, i.e. $p_\theta, p_\theta^\pm, \pi$ for the respective contribution in the sum of (4.71). The kernel itself was also computed using a finite number of runs, (4.2.2.1). Figures (4.7d-4.9d) illustrate the algorithms operation for a sampling approach with two to four qubits respectively. Unfortunately, these results are not as encouraging as in the exact case, since the model seems to fail to learn the data at all, both with the Gaussian and quantum-hard kernel. Surprisingly, the performance did not seem to improve as the number of samples increased as in (4.8d), i.e. as the approximation improved, nor when the number of Epochs increased, as in Figure (4.9d). This is in contrast to the work of [48], in which the model does seem to train properly, albeit not as well as in the exact case. This indicates there is some error with the code, or the nature of the approximation which was used. In any case, this is something to be addressed in future work, as the correct functionality of the algorithm using a sampling approach is critical to gaining a quantum advantage.



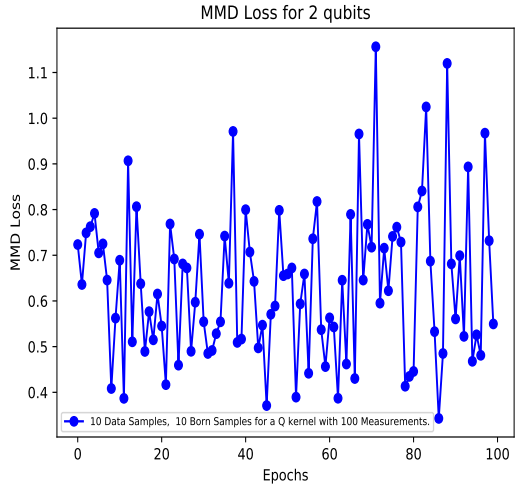
(a) Exact MMD for two qubits with $\eta = 0.002$.



(b) Exact MMD for two qubits with $\eta = 0.1, 0.05, 0.001$.

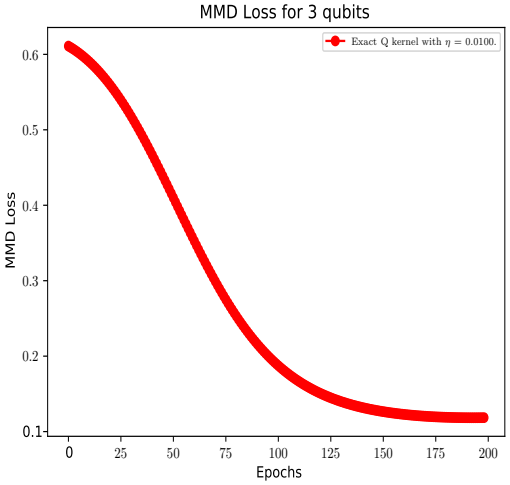


(c) Exact MMD for two qubits with both quantum-hard, and Gaussian kernel.

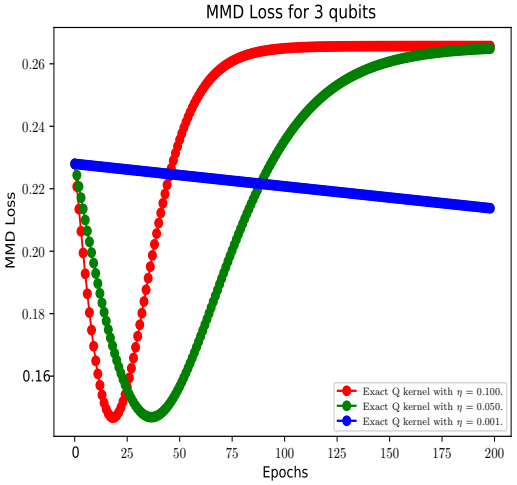


(d) Sampled MMD for two qubits with quantum-hard kernel computed with 10 samples, 10 samples from the Born Machine and Data, for 100 Epochs, with $\eta = 0.1$.

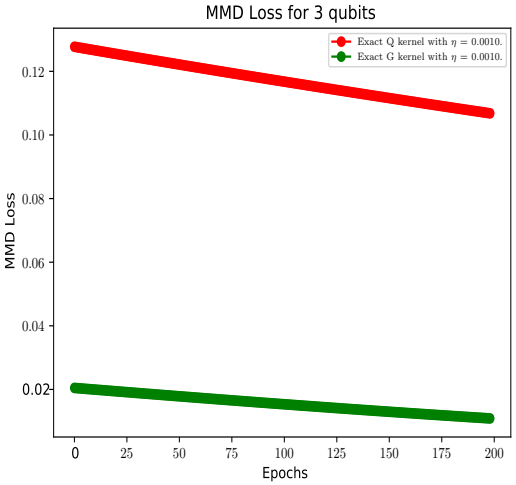
Figure 4.7: MMD for two qubits, learning binary strings of length two.



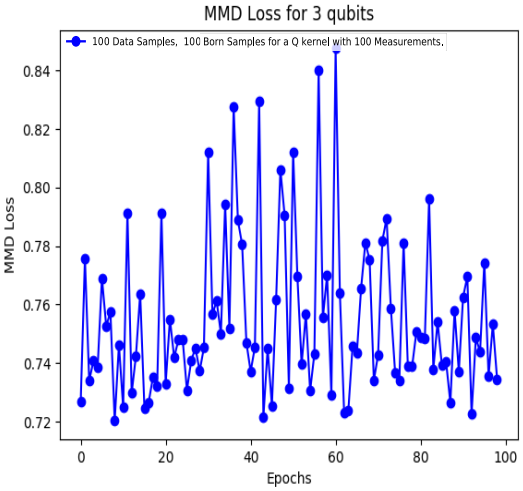
(a) MMD for three qubits with $\eta = 0.01$.



(b) Exact MMD for three qubits with $\eta = 0.1, 0.05, 0.001$.

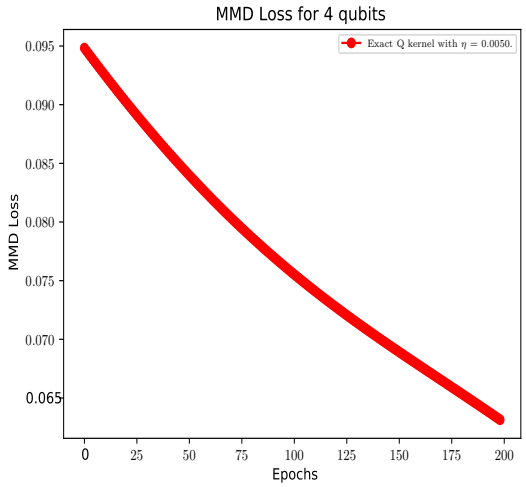


(c) Exact MMD for three qubits with both quantum-hard, and Gaussian kernel.

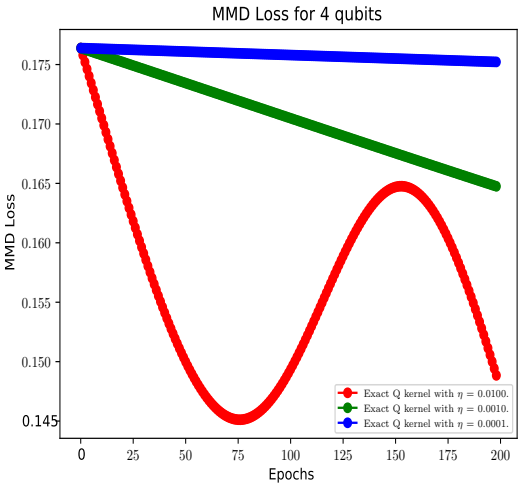


(d) Sampled MMD for three qubits with quantum-hard kernel computed with 100 samples, 100 samples from the Born Machine and Data, for 100 Epochs, with $\eta = 0.1$.

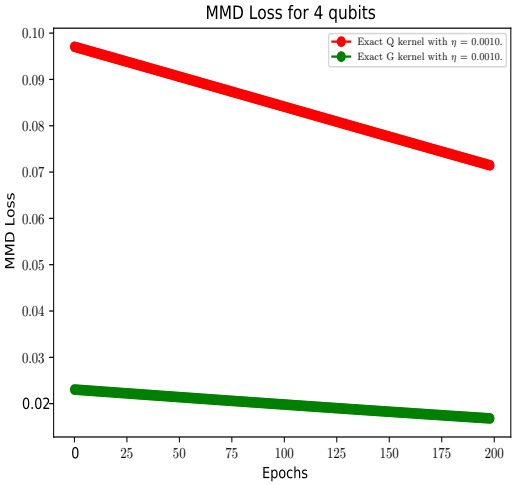
Figure 4.8: MMD for three qubits, learning binary strings of length three.



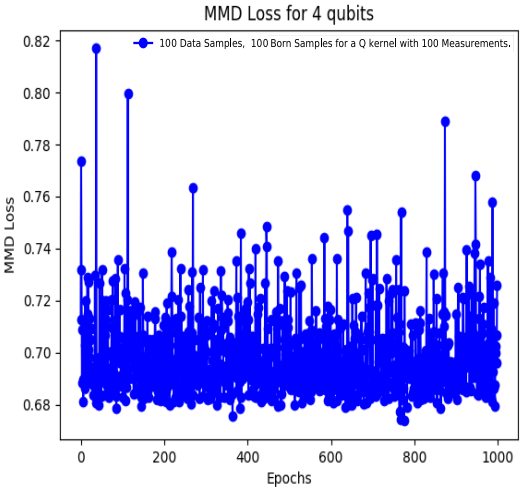
(a) MMD for four qubits with $\eta = 0.005$.



(b) MMD for four qubits with $\eta = 0.01, 0.001, 0.0001$.



(c) MMD for four qubits with both quantum-hard, and Gaussian kernel.



(d) Sampled MMD for four qubits with quantum-hard kernel computed with 100 samples, 100 samples from the Born Machine and Data, for 1000 Epochs, with $\eta = 0.1$.

Figure 4.9: Exact MMD for four qubits, learning binary strings of length four.

Chapter 5

Conclusion

In conclusion, we have developed a quantum generative machine learning algorithm called an Ising Born Machine, derived from quantum circuits in the classes $\{\mathsf{IQP}, p = 1\,\mathsf{QAOA}, \mathsf{IQP}_{\mathrm{y}}\}$. We conjecture that this algorithm cannot be simulated in polynomial time on a classical computer due to certain complexity theory results, due to the following two ingredients:

1. A classical algorithm cannot produce the required samples for training in polynomial time, up to multiplicative error.

2. Even if the samples could be produced efficiently, the kernel can not be computed up to additive error in polynomial time.

The sampling part holds for all parameter values which are irrationally related to 2π, and so we can update the parameters accordingly in the Born Machine to train it.

Of course, we are only conjecturing that a classical algorithm would not be able to *simulate* this learning algorithm, and this does not necessarily imply that this quantum algorithm will give an advantage in *learning* over any classical algorithm (in terms of accuracy in reproducing a distribution). However, given the use of distributions and kernel which are not classically computable efficiently, we conjecture that this Ising Born Machine could be used to learn probability distributions that could not be learned classically. By this statement, we mean that there could exist probability distributions for which the QML algorithms demonstrating supremacy, such as the one defined here, could learn better than *any* ML algorithm by purely classical means. Likely candidates for these hard-to-learn distributions could be any which themselves demonstrate quantum supremacy, or those arising from physical systems. As noted in [49], it is unlikely this would be the case for any data distribution which does not demonstrate supremacy itself,

but it could very well be true for any that does, with a particular idea of long term application in quantum simulation. For example, the output distribution of some highly entangled many body system, the dynamics of which are thought to be intractable for classical devices, could be a candidate for this algorithm to consider. This idea has precedence in recent work involving (classically) learning many body systems using artificial neural networks, [20]. This would be a natural place to insert these types of '*quantum supreme*' learning algorithms, and use simple circuits (or quantum neural networks) to learn complicated many body systems.

Furthermore, it should be noted that these types of algorithms are favourable (or at least more favourable) to run on NISQ devices, since they do not *require* quantum data, in contrast to the algorithms for recommendation, or linear equation solving (HHL) mentioned in Chapter (4), which require creating and *storing* large amounts of qubits in superposition in a qRAM structure.

To test the model, we have implemented classical numerical simulations on Rigetti's Forest Platform, PyQuil, to learn the probability distribution of a toy model. This could be done for the KL Divergence, and the model exhibited a fast convergence to a minimum of this cost function. It could also be done for the MMD as a cost function, when all involved terms were computed exactly, and potentially also showed encouraging results for the use of a quantum-hard kernel, as opposed to one which could be classically computed efficiently. As mentioned above, however, this behaviour depended heavily on the choice of hyperparameters (i.e. the learning rate), along with the initialisation of the parameters, $\{\mathbf{J}, \mathbf{b}\}$ and as such, it makes it difficult to make many concrete statements. Finally, the MMD was also computed and trained approximately using a finite number of samples from the target data to be learned, and the Born Machine itself. Unfortunately, less encouraging results were observed in this case and this is worrying for quantum supremacy conjectures, as discussed in Section (4.3). Fortunately, we also believe that this can be remedied and will be done in future work.

5.1 Future Work

- First and foremost, the most obvious next step is to test the model more thoroughly to determine the cause of the poor results achieved when using a sampling approach for the MMD cost function. As mentioned, this is critical to correct functionality of the learning algorithm. We also would investigate alternative methods to improve the model, for example by adding momentum, or by such a classical parameter

optimizer like Adam, as in [48].

- In all of the above simulations, it has been assumed that the graph structure underlying the Ising Born Machine is fully connected. It would be interesting to investigate performance of the algorithm under the types of connectivity that are currently available in near term devices and further investigate the effects of noise induced by these systems.
- It is desirable to find a candidate probability distribution which could demonstrate supremacy, and investigate whether the quantum model outperforms the existing classical techniques when attempting to learn it.
- As mentioned above, one could add hidden qubits to increase the representational power of the model, in line with the Boltzmann Machine, and study how the hardness results change in this case.
- To make a more realistic claim for the quantum supremacy of this model, it would be necessary to strengthen the complexity theory assumptions to the more general and realistic case of additive error sampling approximations, and therefore investigate how changing the parameter values of the model affects these.
- The "quantum-hard" kernel of [40] is only the first such example of a kernel which cannot be computed classically. As such, more examples should be found, perhaps which are more experimentally realizable.
- Investigate the alternative implementation of this model in continuous variable quantum computing platforms and simulators, such as Strawberry Fields, [43], which is becoming a prominent platform for Quantum Machine Learning.
- Test how training the $\mathbf{\Gamma}, \mathbf{\Delta}$ parameters in $\mathsf{QAOA}(J_{ij}, b_k, \Gamma_l)\,\mathsf{IQP}_{\mathrm{y}}(J_{ij}, b_k, \Delta_l)$ affects the representational power of the model.

Bibliography

[1] CTAN: Package q-circuit.

[2] pyquil: A Python library for quantum programming using Quil, July 2018. original-date: 2017-01-09T21:30:22Z.

[3] Aaronson, S. Quantum computing, postselection, and probabilistic polynomial-time. PROC R SOC A 461, 2063 (Nov. 2005), 3473.

[4] Aaronson, S. Read the fine print. Nature Physics 11 (Apr. 2015), 291.

[5] Aaronson, S., and Arkhipov, A. The Computational Complexity of Linear Optics. Theory of Computing 9, 1 (Feb. 2013), 143–252.

[6] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. A learning algorithm for boltzmann machines. Cognitive Science 9, 1 (Jan. 1985), 147–169.

[7] Adachi, S. H., and Henderson, M. P. Application of Quantum Annealing to Training of Deep Neural Networks. arXiv:1510.06356 [quant-ph, stat] (Oct. 2015). arXiv: 1510.06356.

[8] Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B., and Melko, R. Quantum Boltzmann Machine. Phys. Rev. X 8, 2 (May 2018), 021050.

[9] Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. The Cramer Distance as a Solution to Biased Wasserstein Gradients. arXiv:1705.10743 [cs, stat] (May 2017). arXiv: 1705.10743.

[10] Benedetti, M., Garcia-Pintos, D., Nam, Y., and Perdomo-Ortiz, A. A generative modeling approach for benchmarking and training shallow quantum circuits. arXiv:1801.07686 [quant-ph] (Jan. 2018). arXiv: 1801.07686.

[11] Benedetti, M., Realpe-Gómez, J., Biswas, R., and Perdomo-Ortiz, A. Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. Phys. Rev. A 94, 2 (Aug. 2016), 022308.

[12] Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. Integrating structured biological data by Kernel Maximum Mean Discrepancy. Bioinformatics 22, 14 (July 2006), e49–57.

[13] Born, M., and Fock, V. Beweis des Adiabatensatzes. Z. Physik 51, 3-4 (Mar. 1928), 165–180.

[14] Boykin, P. O., Mor, T., Pulver, M., Roychowdhury, V., and Vatan, F. On Universal and Fault-Tolerant Quantum Computing. arXiv:quant-ph/9906054 (June 1999). arXiv: quant-ph/9906054.

[15] Bravyi, S., DiVincenzo, D. P., Oliveira, R. I., and Terhal, B. M. The Complexity of Stoquastic Local Hamiltonian Problems. arXiv:quant-ph/0606140 (June 2006). arXiv: quant-ph/0606140.

[16] Bremner, M. J., Jozsa, R., and Shepherd, D. J. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 467, 2126 (2011), 459–472.

[17] Bremner, M. J., Montanaro, A., and Shepherd, D. J. Average-Case Complexity Versus Approximate Simulation of Commuting Quantum Computations. Phys. Rev. Lett. 117, 8 (Aug. 2016), 080501.

[18] Bremner, M. J., Montanaro, A., and Shepherd, D. J. Achieving quantum supremacy with sparse and noisy commuting quantum computations. Quantum 1 (Apr. 2017), 8.

[19] Broadbent, A., Fitzsimons, J., and Kashefi, E. Universal Blind Quantum Computation. In 2009 50th Annual IEEE Symposium on Foundations of Computer Science (Oct. 2009), pp. 517–526.

[20] Carleo, G., and Troyer, M. Solving the quantum many-body problem with artificial neural networks. Science 355, 6325 (2017), 602–606.

[21] Chen, J., Cheng, S., Xie, H., Wang, L., and Xiang, T. Equivalence of restricted Boltzmann machines and tensor network states. Phys. Rev. B 97, 8 (Feb. 2018), 085104.

[22] Cheng, S., Chen, J., and Wang, L. Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines. arXiv:1712.04144 [cond-mat, physics:physics, physics:quant-ph, stat] (Dec. 2017). arXiv: 1712.04144.

[23] Deutsch, D. Quantum theory, the Church–Turing principle and the universal quantum computer. Proc R Soc Lond A Math Phys Sci 400, 1818 (July 1985), 97.

[24] Dickson, N. G., Johnson, M. W., Amin, M. H., Harris, R., Altomare, F., Berkley, A. J., Bunyk, P., Cai, J., Chapple, E. M., Chavez, P., Cioata, F., Cirip, T., deBuen, P., Drew-Brook, M., Enderud, C., Gildert, S., Hamze, F., Hilton, J. P., Hoskinson, E., Karimi, K., Ladizinsky, E., Ladizinsky, N., Lanting, T., Mahon, T., Neufeld, R., Oh, T., Perminov, I., Petroff, C., Przybysz, A., Rich, C., Spear, P., Tcaciuc, A., Thom, M. C., Tolkacheva, E., Uchaikin, S., Wang, J., Wilson, A. B., Merali, Z., and Rose, G. Thermally assisted quantum annealing of a 16-qubit problem. Nature Communications 4 (May 2013), 1903.

[25] Du, Y., Liu, T., and Tao, D. Bayesian Quantum Circuit. arXiv:1805.11089 [quant-ph] (May 2018). arXiv: 1805.11089.

[26] Farhi, E., Goldstone, J., and Gutmann, S. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph] (Nov. 2014). arXiv: 1411.4028.

[27] Farhi, E., Goldstone, J., Gutmann, S., and Sipser, M. Quantum Computation by Adiabatic Evolution. arXiv:quant-ph/0001106 (Jan. 2000). arXiv: quant-ph/0001106.

[28] Farhi, E., and Harrow, A. W. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. arXiv:1602.07674 [quant-ph] (Feb. 2016). arXiv: 1602.07674.

[29] Farhi, E., and Neven, H. Classification with Quantum Neural Networks on Near Term Processors.

[30] Fitzsimons, J. F., and Kashefi, E. Unconditionally verifiable blind quantum computation. Phys. Rev. A 96, 1 (July 2017), 012303.

[31] Fujii, K., Kobayashi, H., Morimae, T., Nishimura, H., Tamate, S., and Tani, S. Impossibility of Classically Simulating One-Clean-Qubit Model with Multiplicative Error. Phys. Rev. Lett. 120, 20 (May 2018), 200502.

[32] Fujii, K., and Morimae, T. Commuting quantum circuits and complexity of Ising partition functions. New Journal of Physics 19, 3 (Mar. 2017), 033003.

[33] Gao, X., and Duan, L.-M. Efficient representation of quantum many-body states with deep neural networks. Nature Communications 8, 1 (Sept. 2017), 662.

[34] Gao, X., Wang, S.-T., and Duan, L.-M. Quantum Supremacy for Simulating a Translation-Invariant Ising Spin Model. Phys. Rev. Lett. 118, 4 (Jan. 2017), 040502.

[35] Giovannetti, V., Lloyd, S., and Maccone, L. Quantum Random Access Memory. Phys. Rev. Lett. 100, 16 (Apr. 2008), 160501.

[36] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A Kernel Two-Sample Test. Journal of Machine Learning Research 13 (Mar. 2012), 723-773.

[37] Han, Y., Hemaspaandra, L., and Thierauf, T. Threshold Computation and Cryptographic Security. SIAM Journal on Computing 26, 1 (1997), 59–78.

[38] Han, Z.-Y., Wang, J., Fan, H., Wang, L., and Zhang, P. Unsupervised Generative Modeling Using Matrix Product States. arXiv:1709.01662 [cond-mat, physics:quant-ph, stat] (Sept. 2017). arXiv: 1709.01662.

[39] Harrow, A. W., Hassidim, A., and Lloyd, S. Quantum Algorithm for Linear Systems of Equations. Phys. Rev. Lett. 103, 15 (Oct. 2009), 150502.

[40] Havlicek, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., and Gambetta, J. M. Supervised learning with quantum enhanced feature spaces. arXiv:1804.11326 [quant-ph, stat] (Apr. 2018). arXiv: 1804.11326.

[41] Kerenidis, I., and Prakash, A. Quantum Recommendation Systems. arXiv:1603.08675 [quant-ph] (Mar. 2016). arXiv: 1603.08675.

[42] Kieferová, M., and Wiebe, N. Tomography and generative training with quantum Boltzmann machines. Phys. Rev. A 96, 6 (Dec. 2017), 062327.

[43] Killoran, N., Izaac, J., Quesada, N., Bergholm, V., Amy, M., and Weedbrook, C. Strawberry Fields: A Software Platform for Photonic Quantum Computing. arXiv:1804.03159 [physics, physics:quant-ph] (Apr. 2018). arXiv: 1804.03159.

[44] Kullback, S., and Leibler, R. A. On Information and Sufficiency. Ann. Math. Statist. 22, 1 (1951), 79–86.

[45] Levine, Y., Yakira, D., Cohen, N., and Shashua, A. Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design. arXiv:1704.01552 [quant-ph] (Apr. 2017). arXiv: 1704.01552.

[46] Li, J., Yang, X., Peng, X., and Sun, C.-P. Hybrid Quantum-Classical Approach to Quantum Optimal Control. Phys. Rev. Lett. 118, 15 (Apr. 2017), 150503.

[47] Liu, D., Ran, S.-J., Wittek, P., Peng, C., García, R. B., Su, G., and Lewenstein, M. Machine Learning by Two-Dimensional Hierarchical Tensor Networks: A Quantum Information Theoretic Perspective on Deep Architectures. arXiv:1710.04833 [cond-mat, physics:physics, physics:quant-ph, stat] (Oct. 2017). arXiv: 1710.04833.

[48] Liu, J.-G., and Wang, L. Differentiable Learning of Quantum Circuit Born Machine. arXiv:1804.04168 [quant-ph, stat] (Apr. 2018). arXiv: 1804.04168.

[49] Lloyd, S., and Weedbrook, C. Quantum generative adversarial learning. arXiv:1804.09139 [quant-ph] (Apr. 2018). arXiv: 1804.09139.

[50] MacKay, D. J. C. Information Theory, Inference & Learning Algorithms. Cambridge University Press, New York, NY, USA, 2002.

[51] Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. Quantum Circuit Learning. arXiv:1803.00745 [quant-ph] (Mar. 2018). arXiv: 1803.00745.

[52] Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. Kernel Mean Embedding of Distributions: A Review and Beyond. Foundations and Trends® in Machine Learning 10, 1-2 (2017), 1–141. arXiv: 1605.09522.

[53] Neill, C., Roushan, P., Kechedzhi, K., Boixo, S., Isakov, S. V., Smelyanskiy, V., Megrant, A., Chiaro, B., Dunsworth, A., Arya, K., Barends, R., Burkett, B., Chen, Y., Chen, Z., Fowler, A., Foxen, B., Giustina, M., Graff, R., Jeffrey, E., Huang, T., Kelly, J., Klimov, P., Lucero, E., Mutus, J., Neeley, M., Quintana, C., Sank, D., Vainsencher, A., Wenner, J., White, T. C., Neven, H., and Martinis, J. M. A blueprint for demonstrating quantum supremacy with superconducting qubits. Science 360, 6385 (Apr. 2018), 195.

[54] Newell, A. Perceptrons. An Introduction to Computational Geometry. Marvin Minsky and Seymour Papert. M.I.T. Press, Cambridge, Mass., 1969. vi + 258 pp., illus. Cloth, $12; paper, $4.95. Science 165, 3895 (1969), 780–782.

[55] Nielsen, M. A. Quantum computation and quantum information, tenth anniversary edition.. ed. Cambridge University Press, Cambridge, 2010.

[56] Pestun, V., and Vlassopoulos, Y. Tensor network language model. arXiv:1710.10248 [cond-mat, stat] (Oct. 2017). arXiv: 1710.10248.

[57] Preskill, J. Quantum Computing in the NISQ era and beyond. arXiv:1801.00862 [cond-mat, physics:quant-ph] (Jan. 2018). arXiv: 1801.00862.

[58] Raussendorf, R., and Briegel, H. J. A One-Way Quantum Computer. Phys. Rev. Lett. 86, 22 (May 2001), 5188–5191.

[59] Schuld, M., Bocharov, A., Svore, K., and Wiebe, N. Circuit-centric quantum classifiers. arXiv:1804.00633 [quant-ph] (Apr. 2018). arXiv: 1804.00633.

[60] Schuld, M., and Killoran, N. Quantum machine learning in feature Hilbert spaces. arXiv:1803.07128 [quant-ph] (Mar. 2018). arXiv: 1803.07128.

[61] Shepherd, D., and Bremner, M. J. Temporally unstructured quantum computation. PROC R SOC A (Jan. 2009).

[62] Sopka, J. Introductory Functional Analysis with Applications (Erwin Kreyszig). SIAM Review 21, 3 (1979), 412–413.

[63] Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. G. On integral probability metrics, \phi-divergences and binary classification. arXiv:0901.2698 [cs, math] (Jan. 2009). arXiv: 0901.2698.

[64] Stockmeyer, L. J. The polynomial-time hierarchy. Theoretical Computer Science 3, 1 (1976), 1 – 22.

[65] Tang, E. A quantum-inspired classical algorithm for recommendation systems. arXiv:1807.04271 [quant-ph] (July 2018). arXiv: 1807.04271.

[66] Terhal, B. M., and DiVincenzo, D. P. Adaptive Quantum Computation, Constant Depth Quantum Circuits and Arthur-Merlin Games. arXiv:quant-ph/0205133 (May 2002). arXiv: quant-ph/0205133.

[67] Toda, S. PP is as Hard as the Polynomial-Time Hierarchy. SIAM Journal on Computing 20, 5 (1991), 865–877.

[68] Verdon, G., Broughton, M., and Biamonte, J. A quantum algorithm to train neural networks using low-depth circuits. arXiv:1712.05304 [cond-mat, physics:quant-ph] (Dec. 2017). arXiv: 1712.05304.

[69] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In Proceedings 2001 IEEE International Conference on Cluster Computing (Oct. 2001), pp. 279–287.

[70] Wang, Q., Kulkarni, S. R., and Verdu, S. Divergence estimation of continuous distributions based on data-dependent partitions. IEEE Transactions on Information Theory 51, 9 (Sept. 2005), 3064–3074.

[71] Zhang, Y.-H. Entanglement Entropy of Target Functions for Image Classification and Convolutional Neural Network. arXiv:1710.05520 [cond-mat] (Oct. 2017). arXiv: 1710.05520.

Appendix A

Calculations

A.1 Computation of the KL Divergence for the Born Machine

The negative log-likelihood (equivalent to the KL Divergence) between the data distribution, $\pi(\mathbf{v})$, and the Born Machine distribution over the visible qubits, $p_\theta(\mathbf{v})$, is given by:

$$\begin{aligned}
\mathcal{L}^\theta_{KL} &= -\sum_{\mathbf{v}} \pi(\mathbf{v}) \log p_\theta(\mathbf{v}) \\
\frac{\partial \mathcal{L}^\theta_{KL}}{\partial \theta} &= -\sum_{\mathbf{v}} \pi(\mathbf{v}) \partial_\theta \log p_\theta(\mathbf{v}) = -\sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \partial_\theta p_\theta(\mathbf{v}) \\
&= -\sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} \langle \mathbf{v}, \mathbf{h}| \, \partial_\theta \left[U_f^A U_z(\theta) \, |s\rangle \, \langle s| \, U_z(\theta)^\dagger U_f^{A^\dagger} \right] |\mathbf{v}, \mathbf{h}\rangle \\
&= -\sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} \Big[\langle \mathbf{v}, \mathbf{h}| \, U_f^A (\partial_\theta U_z(\theta)) \, |s\rangle \, \langle s| \, U_z(\theta)^\dagger U_f^{A^\dagger} \, |\mathbf{v}, \mathbf{h}\rangle \\
&\qquad + \langle \mathbf{v}, \mathbf{h}| \, U_f^A U_z(\theta) \, |s\rangle \, \langle s| \, (\partial_\theta U_z(\theta)^\dagger) U_f^{A^\dagger} \, |\mathbf{v}, \mathbf{h}\rangle \Big]
\end{aligned}$$

Now, let $U_z = U_z(\theta)$

$$\begin{aligned}
H_z &= \sum_{ij} J_{ij} Z_i Z_j + \sum_{ij} b_i Z_i \\
U_z = e^{iH_z} &\implies \partial_\theta U_z^\dagger = i U_z \partial_\theta H_z \\
U_z^\dagger = e^{-iH_z} &\implies \partial_\theta U_z^\dagger = -i U_z^\dagger \partial_\theta H_z
\end{aligned}$$

$$\begin{aligned}
\partial_\theta \mathcal{L} &= i \sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} \Big[\langle \mathbf{v}, \mathbf{h}| \, U_f^A U_z \partial_\theta H_z \, |s\rangle \, \langle s| \, U_z^\dagger U_f^{A^\dagger} \, |\mathbf{v}, \mathbf{h}\rangle \\
&\qquad - \langle \mathbf{v}, \mathbf{h}| \, U_f^A U_z \, |s\rangle \, \langle s| \, U_z^\dagger \partial_\theta H_z U_f^{A^\dagger} \, |\mathbf{v}, \mathbf{h}\rangle \Big] \\
&= \sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} 2 \operatorname{Im} \langle \mathbf{v}, \mathbf{h}| \, U_f^A U_z \, |s\rangle \, \langle s| \, U_z^\dagger \partial_\theta H_z U_f^{A^\dagger} \, |\mathbf{v}, \mathbf{h}\rangle
\end{aligned}$$

Now to compute this gradient using the amplitude formalism, we need to compute two amplitudes for each string $|\mathbf{z}\rangle = |\mathbf{v}, \mathbf{h}\rangle$:

$$\langle \mathbf{z}| U_f^A U_z |s\rangle = \langle 0|^{\otimes n} \bigotimes_{j=1}^{n} X^{z_j} U_f^A U_z H^{\otimes n} |0\rangle^{\otimes n} = \langle 0|^{\otimes n} \mathcal{U} |0\rangle^{\otimes n}$$

$$\langle s| U_z^\dagger \partial_\theta H_z U_f^{A^\dagger} |\mathbf{z}\rangle = \langle 0|^{\otimes n} H^{\otimes n} U_z^\dagger \partial_\theta H_z U_f^{A^\dagger} \bigotimes_{j=1}^{n} X^{z_j} |0\rangle^{\otimes n} = \langle 0|^{\otimes n} \mathcal{U}' |0\rangle^{\otimes n}$$

The derivative of H_z with respect to the parameters, θ, will result in terms like $\partial_{J_{ij}} H_z = Z_i Z_j = ie^{\pi/2 Z_i Z_j}$, $\partial_{b_i} H_z = Z_i = ie^{\pi/2 Z_i}$, which can be absorbed into $U_z^\dagger$, and as discussed in Section (3.3.4) the resulting expression should still demonstrate the same classical hardness result.

One aspect of the above should be noted. It can be seen that to compute the amplitude for outcome $\mathbf{z}$, we *need to know* $\mathbf{z}$ in order to apply the correct bit flip gates, X^{z_j}. However, we do in fact know this bitstring. This is because we essentially have to sum over *all* possible outcomes.

A.2 Amplitude computation using the Hadamard Test

To compute these amplitudes, the so-called Hadamard Test will be used, which is also partially employed in [29] to compute the derivative of the cost function in their classifier. The method presented here is slightly different than the standard Hadamard Test, but it computes the same quantity.

The idea behind the Hadamard Test is to add two ancilla qubits for each amplitude, one to compute the real part of the amplitude, and one to compute the complex and run the circuit induced by $\mathcal{U}$ or $\mathcal{U}'$. The circuit for $\mathcal{U}$ will be described, the one for $\mathcal{U}'$ is analogous.

Hadamard Test:

- Firstly, prepare the state with an ancilla in the $|+\rangle_a$ state:
- Apply either $i\mathcal{U}$ or $\mathcal{U}$, on $|\psi_0\rangle = |0\rangle^{\otimes n} = |\hat{0}\rangle$ conditioned on the ancilla being in the state $|1\rangle_a$:

$$C(i^k \mathcal{U}) |\hat{0}\rangle |+\rangle_a = \frac{1}{\sqrt{2}} \left(|\hat{0}\rangle |0\rangle_a + i^k \mathcal{U} |\hat{0}\rangle |1\rangle_a \right) = |\psi_1\rangle \qquad k \in \{0, 1\} \tag{A.1}$$

  Now, if $k = 0$, $\mathcal{U}$ is applied and as a result, the real part of the amplitude is computed. On the other hand, whereas if $k = 1$, $i\mathcal{U}$ is the evolution which is used and we compute the imaginary part.

- Applying Hadamard again to the ancilla results in the state:

$$\mathbb{1}^{\otimes n} \otimes H |\psi_1\rangle = \frac{1}{\sqrt{2}} \left(|\hat{0}\rangle |+\rangle_a + i^k \mathcal{U} |\hat{0}\rangle |-\rangle_a \right) \tag{A.2}$$

$$= \frac{1}{2} \left(|\hat{0}\rangle |0\rangle_a + |\hat{0}\rangle |1\rangle_a i^k \mathcal{U} |\hat{0}\rangle |0\rangle_a - i^n \mathcal{U} |\hat{0}\rangle |1\rangle_a \right) \tag{A.3}$$

$$= \frac{1}{2} \left(\left[|\hat{0}\rangle + i^k \mathcal{U} |\hat{0}\rangle \right] |0\rangle_a + \left[|\hat{0}\rangle - i^k \mathcal{U} |\hat{0}\rangle \right] |1\rangle_a \right) \tag{A.4}$$

- The final step is to measure the ancilla qubit in the computational basis. The probability of measuring the ancilla and finding it in the $|0\rangle$ state is given by:

$$P(|\psi\rangle_a = |0\rangle_a) = \frac{1}{2} + [i^k + (-i)^k] \operatorname{Re}\left\langle\hat{0}\right|\mathcal{U}\left|\hat{0}\right\rangle + [i^{k+1} - (-i)^{k+1}] \operatorname{Im}\left\langle\hat{0}\right|\mathcal{U}\left|\hat{0}\right\rangle \tag{A.5}$$

$$= \begin{cases} \frac{1}{2} + \frac{1}{2}\operatorname{Re}\left\langle\hat{0}\right|\mathcal{U}\left|\hat{0}\right\rangle & \text{if } k = 0, \\ \frac{1}{2} - \frac{1}{2}\operatorname{Im}\left\langle\hat{0}\right|\mathcal{U}\left|\hat{0}\right\rangle & \text{if } k = 1 \end{cases} \tag{A.6}$$

  and by running the test multiple times for different values of k, the real and imaginary parts of the amplitudes can be computed.

Of course, since this procedure is probabilistic, it may require exponentially many repetitions to get an non-zero value for an amplitude corresponding to an outcome, $\mathbf{z}$, which is exponentially unlikely to occur. This will kill any hope of a quantum speedup and it is the reason the KL divergence is not a suitable measure of closeness for our purposes.

A.3 Computation of MMD gradient

The gradient computation for the Born Machine is almost identical to that of [48], following the original method of [46, 51]. However, we have to make a slight recalculation, since in these circuits, only the single qubit gates were parametrized, but in this case we have two-qubit gates parametrized also, and secondly the parametrization of the unitaries we use is slightly different than the standard formalism, as discussed in Section (4.2.2.2):

$$\mathcal{L}_{\mathsf{MMD}} = \mathop{\mathbb{E}}_{x\sim P, y\sim P}(\kappa(x,y)) + \mathop{\mathbb{E}}_{x\sim Q, y\sim Q}(\kappa(x,y)) - 2\mathbb{E}_{x\sim P, y\sim Q}\mathbb{E}(\kappa(x,y)) \tag{A.7}$$

$$= \sum_{x,y} p_\theta(x)p_\theta(y)\kappa(x,y) + \sum_{x,y}\pi(x)\pi(y)\kappa(x,y) - 2\sum_{x,y} p_\theta(x)\pi(y)\kappa(x,y) \tag{A.8}$$

Therefore the derivative with respect to a parameter, θ_k, where $\theta = \{\mathbf{J}, \mathbf{b}\}$ is given by:

$$\frac{\partial \mathcal{L}_{\mathsf{MMD}}}{\partial \theta_k} = \sum_{x,y}\kappa(x,y)\left(p_\theta(y)\frac{\partial p_\theta(x)}{\partial\theta_k} + p_\theta(x)\frac{\partial p_\theta(y)}{\partial\theta_k}\right) - 2\sum_{x,y}\kappa(x,y)\frac{\partial p_\theta(x)}{\partial\theta_k}\pi(y) \tag{A.9}$$

For completeness, the gradient calculation will be repeated here. Assume we have l unitaries, $U_{1:l} = U_1 \dots U_l = U_f^A U_z(\{\mathbf{J}, \mathbf{b}\})$ acting on the initial uniform superposition state: $|s\rangle = H^{\otimes n}|0\rangle = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}|x\rangle$. Some of the unitaries are entangling $CZ(4J_{ij})$ gates, some are local Z rotations, $Z(-2b_i), Z(-2J_{ij})$ and the final gates are either another layer of H on each qubit, for IQP, rotations around the Pauli-X axis for QAOA or rotations around the Pauli-Y axis for IQP_y. To compute the above gradient, we need to compute the derivative of the probability:

$$p_\theta(\mathbf{z}) = \operatorname{Tr}\left(|\mathbf{z}\rangle\langle\mathbf{z}| U_{l:1}|s\rangle\langle s| U_{l:1}^\dagger\right) = |\langle\mathbf{z}| U_{l:1}|s\rangle|^2 \tag{A.10}$$

Hence, the derivative w.r.t a parameter, θ_k is:

$$\frac{\partial p_\theta(\mathbf{z})}{\partial \theta_k} = \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[\frac{\partial U_{l:1}}{\partial \theta_k} \rho U_{l:1}^\dagger + U_{l:1} \rho \frac{\partial U_{l:1}^\dagger}{\partial \theta_k} \right] \right) \tag{A.11}$$

Now,

$$\frac{\partial U_{l:1}}{\partial \theta_k} = U_{l:k+1} \frac{\partial U_k}{\partial \theta_k} U_{k-1:1} \qquad \frac{\partial U_{l:1}^\dagger}{\partial \theta_k} = U_{k-1:1}^\dagger \frac{\partial U_k^\dagger}{\partial \theta_k} U_{l:k+1}^\dagger \tag{A.12}$$

$$U_k = e^{i\theta_k \Sigma_k} \qquad U_k^\dagger = e^{-i\theta_k \Sigma_k} \tag{A.13}$$

$$\implies \frac{\partial U_k}{\partial \theta_k} = i\Sigma_k e^{i\theta_k \Sigma_k} = i\Sigma_k U_k \qquad \frac{\partial U_k^\dagger}{\partial \theta_k} = -i\Sigma_k e^{-i\theta_k \Sigma_k} = -i\Sigma_k U_k^\dagger \tag{A.14}$$

where the index, k, refers to the parameter/unitary number and *not* the qubit(s) on which the unitary, U_k, acts.

Proof.

$$\begin{aligned}
\frac{\partial p_\theta(\mathbf{z})}{\partial \theta_k} &= \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[U_{l:k+1} \frac{\partial U_k}{\partial \theta_k} U_{k-1:1} \rho U_{l:1}^\dagger + U_{l:1} \rho U_{k-1:1}^\dagger \frac{\partial U_k^\dagger}{\partial \theta_k} U_{l:k+1}^\dagger \right] \right) \\
&= i \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[U_{l:k+1} \Sigma_k U_k U_{k-1:1} \rho U_{l:1}^\dagger - U_{l:1} \rho U_{k-1:1}^\dagger \Sigma_k U_k^\dagger U_{l:k+1}^\dagger \right] \right) \\
&= i \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[U_{l:k} \Sigma_k U_{k-1:1} \rho U_{k-1:1}^\dagger U_{l:k}^\dagger - U_{l:k} U_{k-1:1} \rho U_{k-1:1}^\dagger \Sigma_k U_{l:k}^\dagger \right] \right) \\
&= i \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[U_{l:k} [\Sigma_k U_{k-1:1} \rho U_{k-1:1}^\dagger - U_{k-1:1} \rho U_{k-1:1}^\dagger \Sigma_k] U_{l:k}^\dagger \right] \right) \\
&= i \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[U_{l:k} [\Sigma_k, U_{k-1:1} \rho U_{k-1:1}^\dagger] U_{l:k}^\dagger \right] \right) \\
&\overset{(a)}{=} i^2 \operatorname{Tr}\left(|\mathbf{z}\rangle \langle \mathbf{z}| \left[U_{l:k} \left\{ U_k^{\pi/2} U_{k-1:1} \rho U_{k-1:1}^\dagger (U_k^{\pi/2})^\dagger - U_k^{-\pi/2} U_{k-1:1} \rho U_{k-1:1}^\dagger (U_k^{-\pi/2})^\dagger \right\} U_{l:k}^\dagger \right] \right) \\
&= -\operatorname{Tr}\left[|\mathbf{z}\rangle \langle \mathbf{z}| U_{l:k} U_k^{\pi/2} U_{k-1:1} \rho U_{k-1:1}^\dagger (U_k^{\pi/2})^\dagger U_{l:k}^\dagger - U_{l:k} U_k^{-\pi/2} U_{k-1:1} \rho U_{k-1:1}^\dagger (U_k^{-\pi/2})^\dagger U_{l:k}^\dagger \right] \\
&= -\operatorname{Tr}\left[|\mathbf{z}\rangle \langle \mathbf{z}| U_{l:1}^{k+\pi/2} \rho (U_{l:1}^{k+\pi/2})^\dagger - U_{l:1}^{k-\pi/2} \rho (U_{l:1}^{k-\pi/2})^\dagger \right] \\
&= -\operatorname{Tr}\left[|\mathbf{z}\rangle \langle \mathbf{z}| U_{l:1}^{k+\pi/2} \rho (U_{l:1}^{k+\pi/2})^\dagger \right] + \operatorname{Tr}\left[|\mathbf{z}\rangle \langle \mathbf{z}| U_{l:1}^{k-\pi/2} \rho (U_{l:1}^{k-\pi/2})^\dagger \right] \\
&= p_{\theta_k}^-(\mathbf{z}) - p_{\theta_k}^+(\mathbf{z})
\end{aligned}$$

□

where, $U_{l:1}^{k\pm\pi/2}$ indicates the parameter of the k^{th} unitary, $U_k(\theta_k)$ is shifted by $\pm\pi/2$, i.e. $U_{l:1}^{k\pm\pi/2} = U_l \dots U_{k+1} U_k(\theta_k \pm \pi/2) U_{k-1} \dots U_1$.

The commutator evaluation, (a), comes from the following property:

$$[P_j, \rho] = i \left[U_j\left(\frac{\pi}{2}\right) \rho U_j^\dagger\left(\frac{\pi}{2}\right) - U_j\left(-\frac{\pi}{2}\right) \rho U_j^\dagger\left(-\frac{\pi}{2}\right) \right] \tag{A.15}$$

Equation (A.15) holds as long as U_j has the above form, and P_j is such that $P_j^2 = \mathbb{1}$. ρ is an arbitrary operator, in our case, $\rho = |\mathbf{s}\rangle \langle \mathbf{s}|$.

Proof.

$$
\begin{aligned}
&i\left[U_j\left(\frac{\pi}{2}\right)\rho U_j^\dagger\left(\frac{\pi}{2}\right) - U_j\left(-\frac{\pi}{2}\right)\rho U_j^\dagger\left(-\frac{\pi}{2}\right)\right] = i\left[e^{-i\frac{\pi}{4}P_j}\rho e^{i\frac{\pi}{4}P_j} - e^{i\frac{\pi}{4}P_j}\rho e^{-i\frac{\pi}{4}P_j}\right] \\
&= i\left[\left(\cos\left(\frac{\pi}{4}\right)\mathbb{1} - i\sin\left(\frac{\pi}{4}\right)P_j\right)\rho\left(\cos\left(\frac{\pi}{4}\right)\mathbb{1} + i\sin\left(\frac{\pi}{4}\right)P_j\right)\right. \\
&\left. - \left(\cos\left(\frac{\pi}{4}\right)\mathbb{1} + i\sin\left(\frac{\pi}{4}\right)P_j\right)\rho\left(\cos\left(\frac{\pi}{4}\right)\mathbb{1} - i\sin\left(\frac{\pi}{4}\right)P_j\right)\right] \\
&= \frac{i}{2}\left[(\mathbb{1} - iP_j)\,\rho\,(\mathbb{1} + iP_j) - (\mathbb{1} + iP_j)\,\rho\,(\mathbb{1} - iP_j)\right] \\
&= \frac{i}{2}\left[(\rho - iP_j\rho)\,(\mathbb{1} + iP_j) - (\rho + iP_j\rho)\,(\mathbb{1} - iP_j)\right] \\
&= \frac{i}{2}\left[\left(\rho - iP_j\rho + i\rho P_j - i^2P_j\rho P_j\right) - \left(\rho - iP_j\rho + i\rho P_j - i^2P_j\rho P_j\right)\right] \\
&= \frac{i}{2}\left[\left(\not{\rho} - \underline{iP_j\rho} + \underline{\underline{i\rho P_j}} + \cancel{P_j\rho P_j}\right) - \left(\not{\rho} - \underline{iP_j\rho} + \underline{\underline{i\rho P_j}} + \cancel{P_j\rho P_j}\right)\right] = [P_j, \rho]
\end{aligned}
$$

□

The gradient of the KL Divergence can also be written in the form of probabilities using this method, as an alternative to computing the amplitudes, one can compute the probabilities:

$$
\frac{\partial \mathcal{L}_{KL}^{\theta}}{\partial \theta} = \sum_{\mathbf{v}} \frac{\pi(\mathbf{v})}{p_\theta(\mathbf{v})} \sum_{\mathbf{h}} |\left\langle \mathbf{v}, \mathbf{h} \right| U_f^A U_z^{k-\pi/2} \left| \mathbf{s} \right\rangle|^2 - |\left\langle \mathbf{v}, \mathbf{h} \right| U_f^A U_z^{k+\pi/2} \left| s \right\rangle|^2 \tag{A.16}
$$

However, we shall not use this formalism in this work. The gradient of the MMD loss with respect to the parameter, θ_k, is then:

Proof.

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{\mathsf{MMD}}}{\partial \theta_k} &= \sum_{x,y} \kappa(x,y) p^-_{\theta_k}(x) p_\theta(y) - \sum_{x,y} \kappa(x,y) p^+_{\theta_k}(x) p_\theta(y) + \sum_{x,y} \kappa(x,y) p_\theta(x) p^-_{\theta_k}(y) \\
&\quad - \sum_{x,y} \kappa(x,y) p_\theta(x) p^+_{\theta_k}(y) - 2\sum_{x,y} \kappa(x,y) p^-_{\theta_k}(x)\pi(y) + 2\sum_{x,y} \kappa(x,y) p^+_{\theta_k}(x)\pi(y) \\
&= \underset{x\sim p^-_{\theta_k}, y\sim p_\theta}{2\mathbb{E}}(\kappa(x,y)) - \underset{x\sim p^+_{\theta_k}, y\sim p_\theta}{2\mathbb{E}}(\kappa(x,y)) - \underset{x\sim p^-_{\theta_k}, y\sim \pi}{2\mathbb{E}}(\kappa(x,y)) + \underset{x\sim p^+_{\theta_k}, y\sim \pi}{2\mathbb{E}}(\kappa(x,y))
\end{aligned} \tag{A.17}
$$

□

Since the kernel is symmetric: $\kappa(x,y) = \kappa(y,x)$.

A.4 Generative Ising Born Machine Algorithm

The full operation of the algorithm is as follows:

Algorithm 1 Generative Ising Born Machine

(1) Initialize all parameters of Ising Born Machine, $\theta^{(0)} = \{J_{ij}, b_k\}$ at random or a subset of the parameters as constrained by hardware.

(2) Prepare n qubits in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

(3) Apply single and two-qubit operations corresponding to $CZ(4J_{ij})$, $Z(-2J_{ij}), Z(-2b_k)$ to all qubits, as determined by the graph connectivity (given by experimental constraints).

(4) Apply final unitary, $U_f^A = \left\{ H^{\otimes n}, e^{i\sum_{i=1}^{n}\Gamma_i X_i}, e^{i\sum_{i=1}^{n}\Delta_i Y_i} \right\}$, and measure all qubits in the computational basis, $M_{\mathbf{z}} = |\mathbf{z}\rangle\langle\mathbf{z}|$, with outcome string $\mathbf{z}$.

(5) Repeat steps (2) - (4), n times to generate n samples from the Ising Born Machine.

(6) Run quantum circuit to compute the kernel, $\kappa(\mathbf{x}, \mathbf{y})$, for each pair of samples from Born Machine, $p_\theta(\mathbf{x})$, and given data distribution, $\pi(\mathbf{y})$. Output MMD to compare between instantaneous Born Machine distribution and data distribution.

(7) Run quantum circuits corresponding to *shifted* Born Machine circuit, two for each parameter, θ_k, to be updated. Each circuit differs only from the original circuit by a shift of parameter, θ_k, by $\pm\frac{\pi}{2}$.

(8) Repeat step (4), p, q times to generate p, q samples from the shifted distributions, $p^-_{\theta_k}, p^+_{\theta_k}$ respectively.

(9)
- **if:** A classical kernel, $\kappa(\mathbf{x}, \mathbf{y})$ is to be used, i.e. $\kappa(\mathbf{x}, \mathbf{y})$ is a Gaussian kernel for example, compute $\kappa(\mathbf{a}, \mathbf{x}), \kappa(\mathbf{a}, \mathbf{y}), \kappa(\mathbf{b}, \mathbf{x}), \kappa(\mathbf{b}, \mathbf{y})$ for each pair of samples from the Born Machine and the data distribution with each pair from the shifted circuits directly.
- **else:** Run circuit to compute quantum-hard kernels required for the gradient of the MMD. This is done for each pair of samples from the Born Machine and the data distribution with each pair from the shifted circuits to compute $\kappa(\mathbf{a}, \mathbf{x}), \kappa(\mathbf{a}, \mathbf{y}), \kappa(\mathbf{b}, \mathbf{x}), \kappa(\mathbf{b}, \mathbf{y})$.
- **return:** $\kappa(\mathbf{a}, \mathbf{x}), \kappa(\mathbf{a}, \mathbf{y}), \kappa(\mathbf{b}, \mathbf{x}), \kappa(\mathbf{b}, \mathbf{y})$

(10) Update each parameter in the Ising Born Machine according to $\theta_k^{(d+1)} = \theta_k^{(d)} - \eta\frac{\partial\mathcal{L}}{\partial\theta_k^{(d)}}$

(11) Repeat steps (1) - (10) for each epoch, d, until the MMD cost, $\mathcal{L}_{\text{MMD}}$ has converged to a minimum, indicating the Ising Born Machine can output samples approximately from the data distribution, $\pi(\mathbf{y})$