The Born Supremacy: Quantum Advantage and Training of an Ising Born Machine

Brian Coyle[1], Daniel Mills[1], Vincent Danos[1, 2], and Elham Kashefi[1, 3]

[1]School of Informatics, 10 Crichton Street, University of Edinburgh, United Kingdom.
[2]École Normale Supérieure, Paris, CNRS, 75005 Paris, France.
[3]Laboratoire d'Informatique de Paris 6, CNRS, UPMC - Sorbonne Universités, 4 place Jussieu, 75005 Paris, France

April 5, 2019

Abstract

The search for an application of near-term quantum devices is widespread. Quantum Machine Learning is touted as a potential utilisation of such devices, particularly those which are out of the reach of the simulation capabilities of classical computers. In this work, we propose a generative Quantum Machine Learning Model, called the Ising Born Machine (IBM), which we show cannot, in the worst case, and up to suitable notions of error, be simulated efficiently by a classical device. We also show this holds for all the circuit families encountered during training. In particular, we explore quantum circuit learning using non-universal circuits derived from Ising Model Hamiltonians, which are implementable on near term quantum devices.

We propose two novel training methods for the IBM by utilising the *Stein Discrepancy* and the *Sinkhorn Divergence* cost functions. We show numerically, both using a simulator within Rigetti's Forest platform and on the Aspen-1 16Q chip, that the cost functions we suggest outperform the more commonly used Maximum Mean Discrepancy (MMD) in particular. We also propose an improvement to the MMD by performing a novel utilisation of a *quantum kernels* which we also demonstrate provides improvements over its classical counterpart. We discuss the potential of these methods to learn '*hard*' quantum distributions, a feat which would demonstrate the advantage of quantum over classical computers, and provide the first formal definitions for what we call '*Quantum Learning Supremacy*'. We also propose a novel view on the area of quantum circuit compilation by using the IBM to 'mimic' target quantum circuits using classical output data only.

Contents

1 Introduction

As fault intolerant quantum devices, with $\sim 80 - 100$ qubits, begin to be built, we near the dawn of the Noisy Intermediate Scale Quantum (NISQ) [1] technology era. These devices will not be able to perform many of the most famous algorithms thought to demonstrate exponential speedups over classical algorithms [2, 3]. However, they could provide an efficient solution to other problems which cannot be solved in polynomial time by purely classical means, given the limited resources of these near term devices. Showing this to be true is referred to as a demonstration of *Quantum Computational Supremacy*[1].

Many of the aforementioned proof of principle problems utilise the measurement process inherent in quantum computation by generating samples from a quantum distribution. Typically, the distribution is sampled by applying a sequence of quantum gates and measurements to some initial state. If this sequence consists of quantum gates drawn from a 'universal' set, we can sample from any quantum distribution. A more restricted scenario is one where we are not permitted to utilise the full suite of universal gates. The motivation here is to generate circuits that are simpler to implement experimentally on NISQ devices. Some of these simpler circuits are thought still to be usable in demonstrations of quantum supremacy, with popular examples of sub-universal models including the process underpinning BosonSampling, [4], and Instantaneous Quantum Polynomial Time (IQP) Computations [5].

We will make an attempt to connect these sampling hardness results to their use in Quantum Machine Learning (QML), in more detail than has been previously attempted, [6]. In this spirit we utilise a learning model called the *Born Machine* [7, 8] to perform Quantum Circuit Learning (QCL) [9, 10] with NISQ devices. The core principle of the Born Machine is its ability to produce statistics which rely on the fundamental randomness of quantum mechanics, according to Born's Measurement Rule for a state $|\psi\rangle$ and measurement outcome $\mathbf{x}$:

$$\mathbf{x} \sim P(\mathbf{x}) = |\langle \mathbf{x}|\psi\rangle|^2 \tag{1}$$

By utilising quantum mechanics in machine learning we hope to be able to develop algorithms to be applied in the classical domain in areas such as accelerating recommendation systems or support vector machines [3, 11–13]. One may also find *new* applications for QML algorithms, which exist purely in the quantum domain. One example, which we will propose in Section 7.1, is in the mimicking of target quantum circuits by 'learning a circuit description' using samples from its output distribution. This could be seen as an attempt to automatically compile the model circuit onto the target circuit in such a way that the outputs from the circuits are indistinguishable to a classical observer.

This contrasts with previous attempts at quantum compilation which involve directly adapting one circuit to another. We believe out method is a new way to approach the problem of compilation on quantum hardware.

It is also hoped that quantum models would achieve 'better' performance on certain datasets, than any purely classical one. This is motivated by the supremacy arguments mentioned above, and will provide a central theme to this work. A physical demonstration of such a task would provide a definitive separation between quantum and classical machine learning algorithms *in practice*. This is a more challenging task than simply demonstrating quantum supremacy by itself, or even the verification of quantum supremacy, [14–17] but

[1]This and similar problems are often also refereed to as *quantum advantage* or *quantum superiority*

addresses the usefulness of these near term devices. These complexity theoretic supremacy arguments are even more relevant given recent work in QML algorithm 'dequantisations' [18–22], in which quantum algorithms thought to have an exponential speedup over any classical algorithms inspired completely *classical* algorithms with polynomial run time. We will motivate our version of the Born Machine with complexity-theoretic arguments, to defeat obvious methods for such a dequantisation. Hence we have our first guiding principle in this work.

Supremacy Base learning supremacy results on solid complexity theoretic grounds.

In contrast to previous work [8], which shows that quantum circuits with more layers, and hence more parameters, are more expressive, we investigate NISQ devices with relatively few parameters. Specifically, we define a variant of the Born Machine, called an *Ising Born Machine* (IBM)[2] from which we can recover more well known circuit classes which can theoretically demonstrate quantum supremacy and which are defined by unitary gates which are derived from an Ising Hamiltonian. Our focus on NISQ devices contrasts with those 'coherent' QML algorithms, such as the HHL linear equation solver, [3], which require quantum technologies that may take many years to develop, such as Quantum Random Access Memory (QRAM) [23]. Our second guiding principle is the following.

NISQ Develop algorithms which consider the limitations of NISQ technology.

Our contribution in this paper can be summarised as follows:

1. *Classical Hardness:* We connect our model to quantum sampling hardness results in more detail than studied previously [6, 24], and also its training, which has not been previously studied.

2. *Training Procedure:* Our main contribution is to introduce new training procedures for differentiable training of quantum generative models, alternative to those proposed previously [8, 10, 25, 26] which we validate numerically to outperform the previous standard gradient based method, [8] both on a simulator and quantum hardware.

3. *Quantum Learning Supremacy:* We propose the first definitions for what it would mean for a generative quantum model to outperform all possible classical models, for learning certain distribution classes. This provides a formulation of the idea of [27] to learn hard distributions.

4. *Compilation:* We propose a new viewpoint to quantum circuit compilation using classical data with our methods, with a similar mindset to other approaches, [28, 29].

This paper is organised as follows.

Section 2: Required terminology and background in Machine Learning and 'Quantum Supremacy' are introduced along with the quantum circuit classes we will use. We also provide the first definitions, to the best of our knowledge, for what we call '*Quantum Learning Supremacy*', the ability of a Quantum algorithm to learn a distribution which is not possible efficiently classically.

[2]The model has no relation to the International Business Machines Corporation (IBM).

Section 3: The Ising Born Machine is defined and related to previous work. Our first contribution, to illustrate how the underlying circuit in the model is hard to classically simulate up to a suitable notion of error, is discussed.

Section 4: Kernel methods are introduced, along with both 'classical' and 'quantum' kernels. We describe methods to train Born Machine models, and introduce our contributions to this area; two new cost functions leading to differentiable training of the model. These are the Stein Discrepancy and the Sinkhorn Divergence, which we argue to be better than current approaches.

Section 5: Many ingredients in the training algorithm are shown to be hard for a purely classical system to perform, leveraging the hardness of the underlying quantum circuit, and relating back to Section 3.

Section 6: Numerical results are presented to illustrate these new training techniques.

Section 7: Two novel applications for the Ising Born Machine are discussed. The first is automatic compilation of quantum circuits, using purely classical data, and the second is in learning hard distributions, providing a more methodological approach to fulfilling the definitions introduced in Section 2. In depth analysis of these final applications is left to future work.

2 Preliminaries

We introduce some terminology and related work necessary for the reading of this paper.

2.1 Learning and Modelling

Machine learning broadly encapsulates the aspiration to be able to use algorithms to perform a task without explicit instruction, but deducing solutions using patterns or inference. Two common tasks for which ML techniques are useful are *discriminative tasks*, such as classification, and *generative modelling*. The former usually falls under the umbrella of '*Supervised Learning*', in which an algorithm is trained using labelled data. The latter is typically used in '*Unsupervised Learning*', in which no labels are provided, and the algorithm learns the relationships in the data by itself. The use case in this work will be parameterised *generative* modelling, which consists of three key components:

Target/Data: A phenomenon from which we can extract sample observations.

Model: A parameterised structure which represents a characterisation of the target.

Training: A process of updating the model parameters, based on observations of the target, by sampling from the model and the target. This is achieved by evaluating some cost or notion of '*closeness*' and calling some optimiser to compute updates.

The goal of the training is to ensure that samples from the model match the behaviour one would expect from the target.

The particular model we shall introduce will perform a relatively newly defined paradigm known as Quantum Circuit Learning [30]. The general methodology for training in QCL can be broken down as follows:

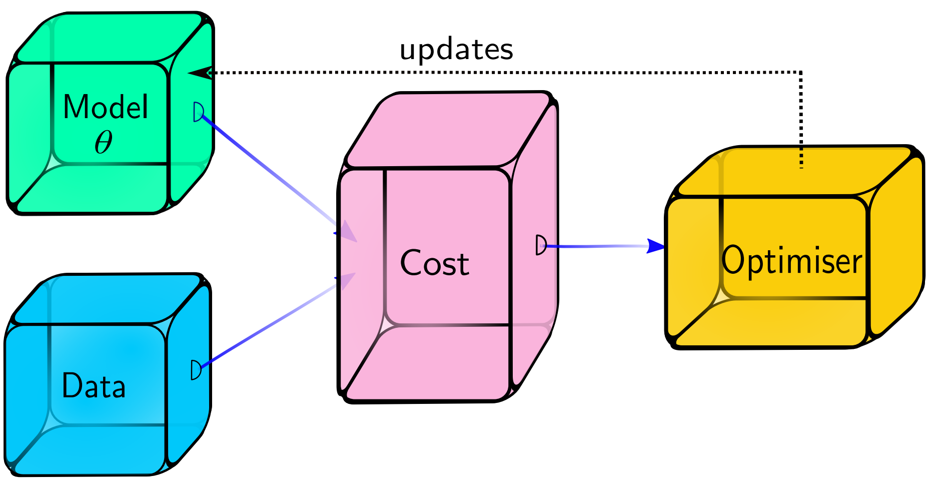


Figure 1: Illustration of generative modelling procedure.

Compute Phase: A parameterised quantum circuit is applied to an initial configuration of qubits, typically the $|0\rangle^{\otimes n}$ state, resulting in a parameterised final state.

Compare Phase: Some information about this state is extracted, be it a series of samples via measurements in the generative case, as we explore here, or some expectation value of an observable in the classification case [9].

Modify Phase: Based on this information, the parameters of the circuit are updated by a classical optimiser to minimise some cost function or 'error'. In gradient based methods, this update will depend on the gradient of the cost function.

This process is repeated multiple times, until there is some convergence or for a specified number of updates, in order to refine the model.

2.2 Classical Simulation of Quantum Computations

The central question behind Quantum Computational Supremacy is whether or not it is possible to design a classical algorithm which could produce a probability distribution $q(\mathbf{x})$, which is *close* to a given quantum output distribution $p(\mathbf{x})$. This notion of reproducing a quantum distribution can be formalised as classical simulation, of which there are two types. For our purposes, the more relevant notion is instead that of *weak simulation*, which better captures the process of sampling.

Definition 2.1 (Strong and Weak Classical Simulation)**.** *[16, 31] A uniformly generated quantum circuit, C, from a family of circuits, with input size n, is weakly simulatable 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. On the other hand, a strong simulator of the family would be able to compute the output probabilities,* $p(\mathbf{x})$*, and also all the marginal distributions over any arbitrary subset of the outputs. Both of these notions apply to some notion of error, ϵ.*

As mentioned in [16], strong simulation[3] is a harder task than weak simulation, and it is this weak simulatability which we want to rule out as being classically hard. The specific

[3]The suitable notion of error, ϵ, for strong simulation would be the precision to which the probabilities can be computed.

instances of problems which are classically hard is captured by worst case and *average case* hardness. Informally, worst case implies there is *at least* one instance of the problem which is hard to simulate. This worst case hardness holds for IQP / QAOA circuits, [16, 32], which we will illustrate in Section 2.3. A stronger notion is that of average case hardness, which has been proven for Random Circuit Sampling, [15], and BosonSampling, [4], but is only conjectured to hold for IQP circuits for example.

One could ask "What if we do not care about getting samples from the *exact* distribution, and instead an approximation is good enough?". Exact in this case refers to the outcome probabilities of the simulator being identical to those outputted by the quantum device; $q(\mathbf{z}) = p(\mathbf{z}) \forall \mathbf{z}$ or $\epsilon = 0$. This is a very important and relevant question to ask when discussing quantum supremacy since experimental noise means it could be that even quantum computers cannot produce the exact dynamics that they are supposed to, according to the theory. Worse still, noise typically results in decoherence and the destruction of entanglement and interference in quantum circuit, so in the presence of noise the resulting output distribution could become classically simulatable.

We wish to have strong theoretical guarantees that experiments which claim to demonstrate supremacy, even in the presence of reasonable noise, do in fact behave as expected. Since we are dealing fundamentally with probability distributions, there are many notions of error one could choose. This question is the backbone of this work, and is extremely relevant since it provides many variants of the problem that one could work with. One of the simplest examples of which is multiplicative error.

Definition 2.2 (Multiplicative Error). *A circuit family is weakly simulatable within multiplicative (relative) error, if there exists a classical probabilistic algorithm, Q, which produces samples, $\mathbf{z}$, according to the distribution, $q(\mathbf{z})$, in time which is polynomial in the input size, such that it differs from the ideal quantum distribution, $p(\mathbf{z})$, by a multiplicative constant, $c > 1$:*

$$\frac{1}{c} p(\mathbf{z}) \leq q(\mathbf{z}) \leq c p(\mathbf{z}) \qquad \forall \mathbf{z} \tag{2}$$

As noted in [33], it would be desirable to have a quantum sampler which could achieve the bound, (2), but this is not believed to be an experimentally reachable goal[4]. That is why much effort has been put in trying to find systems for which supremacy could be provably demonstrated according to the variational distance error condition, (3), which is easier to achieve on near term quantum devices.

Definition 2.3 (Total Variation (TV) Error). *A circuit family is weakly simulable within 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 polynomial time, such that it differs from the ideal quantum distribution, $p(\mathbf{x})$ in total variation distance, ϵ:*

$$\sum_{\mathbf{x}} |p(\mathbf{x}) - q(\mathbf{x})| \leq \epsilon \tag{3}$$

Intuitively, multiplicative error sampling is 'harder' since it must hold *for all* samples, i.e. the classical algorithm, Q, must capture all the fine features of the target distribution, p. In contrast, variation distance error indicates that the distributions only have to be similar 'overall'.

[4]In the sense that it is not believed a *physical* quantum device, could achieve such a multiplicative error bound on its probabilities, relative to its ideal functionality, i.e. replacing q in (2) by the output distribution of a noisy quantum device.

2.3 Quantum Circuit Classes

The circuit classes we introduce is strongly related to two classes, which have both had their relationship to Quantum Supremacy studied extensively [16, 32, 34–36]. Both are 'sub-universal', in the sense that they are not powerful enough to directly simulate arbitrary quantum computations, but are believed to achieve something outside of the classically tractable regime. They are derived from an Ising-type Hamiltonian, and differ only in the final 'measurement' gate applied, which is a rotation gate applied immediately preceding a measurement.

Initially, a Hadamard basis preparation is performed. This is followed by a unitary evolution by m operators, each acting on the qubits in the set S_j and described by:

$$U\left(\alpha_j, S_j\right) = \exp\left(i\alpha_j \bigotimes_{k\in S_j} Z_k\right) \tag{4}$$

This is followed by the *measurement* unitary U_f which is built from single qubit gates acting on each qubit.

$$U_f\left(\mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}\right) = \exp\left(i\sum_{k=1}^{n} \Gamma_k X_k + \Delta_k Y_k + \Sigma_k Z_k\right) \tag{5}$$

Where X_k, Y_k, Z_k are the canonical Pauli operators, [37], acting on qubit k. The final circuit is the following:

$$U_f\left(\mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}\right)\prod_{j=1}^{m} U(\alpha_j, S_j) H^{\otimes n} \left|0\right\rangle^{\otimes n} \tag{6}$$

Sampling from distributions produced by these circuits is performed by computational basis measurements of all qubits.

We now show how the specific choices of parameters in (6) retrieve the circuit classes mentioned above.

2.3.1 Instantaneous Quantum Polynomial Time Computations (IQP)

The first example of a sub-universal class is that of Instantaneous Quantum Polynomial Time Computations (IQP) circuits [5]. IQP circuits have exactly the form of (6), but with the parameters of the measurement unitary set in the following way:

$$U_f^{\mathsf{IQP}} = U_f\left(\forall k : \Gamma_k = \frac{\pi}{2\sqrt{2}}, \Delta_k = \frac{\pi}{2\sqrt{2}}, \Sigma_k = 0\right) = \bigotimes_{k=1}^{n} \exp\left(\frac{i\pi}{2\sqrt{2}}(X_k + Z_k)\right) \tag{7}$$

This results in a final Hadamard gate applied to every qubit, since:

$$H = \frac{1}{\sqrt{2}}(X + Z) \tag{8}$$

Using only gates diagonal in the Pauli-X basis, and thus which commute, make IQP *instantaneous* but mean it is not able to a achieve the full power of universal quantum computation. However, it is still believed to be hard to classically simulate [16]:

Theorem 2.1 (informal from [16]). *If the output probability distributions generated by uniform families of* IQP *circuits could be weakly classically simulated then the polynomial hierarchy (*PH*) would collapse to its third level.*

A collapse of PH is thought to be unlikely at *any* level, giving us confidence in the hardness of IQP. In some sense, such a collapse to a certain level would be a generalisation of P = NP, which would correspond to a full collapse to the zeroth level.

Theorem 2.1 and similar results in [38] are remarkable in their demonstration that quantum computers which are very much weaker than a universal BQP machine are still very difficult to classically simulate. In fact supremacy results of IQP also exist in the case of the more realistic variation distance error.

Theorem 2.2 (informal from [34]). *Assume either one of two conjectures, relating to the hardness of Ising partition function and the gap of degree 3 polynomials, and the stability of the* PH*, it is hard to classically sample from the output probability distribution of any* IQP *circuit in polynomial time, up to a total variation error of* $\epsilon = \frac{1}{384}$.

2.3.2 Quantum Approximate Optimisation Algorithm (QAOA)

The second well known class which can be recovered from (6) is the shallowest depth version of the *Quantum Approximate Optimisation Algorithm* (QAOA) [36]. The QAOA is an algorithm to approximately prepare a desired quantum state, which encodes the solution to some problem that can be extracted by measuring the final state. The canonical example is MaxCut [36], which is an example of a constraint satisfaction problem. The QAOA is defined in terms of a 'cost' Hamiltonian, $\mathcal{H}_z$, and a 'mixer' Hamiltonian, $\mathcal{H}_x$ (borrowing the terminology of [39]). The mixer Hamiltonian is assumed to be one which has an easily prepared ground state (typically a product state), for example:

$$\mathcal{H}_x = \sum_{i=1}^{n} X_i \tag{9}$$

The goal of the QAOA is to produce a ground (or thermal) state of the 'cost' Hamiltonian, $\mathcal{H}_z$, which encodes some problem solution. This cost Hamiltonian can be exactly the exponent of the unitary in (4), where for each j, $S_j \subseteq [n]$:

$$\mathcal{H}_z = \sum_j \alpha_j \bigotimes_{i \in S_j} Z_i \tag{10}$$

In the most general form, a QAOA circuit consists of applying the unitaries (9) and (10) in an alternating fashion. A depth $p -$ QAOA has p layers of these same gate sets acting in an alternating fashion, i.e. it produces a state:

$$|\psi_{\hat{\gamma},\hat{\beta}}\rangle = \prod_{i=1}^{p} e^{-i\beta_i \mathcal{H}_x} e^{-i\gamma_i \mathcal{H}_z} |+\rangle^{\otimes n} \tag{11}$$

The $2p$ parameters, $\{\hat{\gamma}, \hat{\beta}\} = \{\gamma_1, \dots, \gamma_p, \beta_1, \dots, \beta_p\}$ are optimised to produce the required state, which is assumed to be difficult to prepare directly.

We are interested in the shallowest depth version of the algorithm, which produces states of the form:

$$|\psi_{\gamma,\beta}\rangle = e^{-i\beta\mathcal{H}_x} e^{-i\gamma\mathcal{H}_z} |+\rangle^{\otimes n} \tag{12}$$

Since the mixer Hamiltonian in (9) is 1-local (each term acts on only a single qubit), the evolution by the unitary $U_f^{\mathsf{QAOA}} = e^{-i\beta\mathcal{H}_x}$ can be decomposed into a tensor product of single qubit unitaries corresponding to rotations around the Pauli-X axis.

The γ parameters in (12) can be absorbed into the Hamiltonian parameters α_j, and we allow β to be different for each qubit. Therefore, it can be seen that this corresponds to the following setting of the parameters in (5).

$$U_f^{\mathsf{QAOA}} = U_f(\forall k : \Gamma_k = -\Gamma_k, \Delta_k = 0, \Sigma_k = 0) = \exp\left(-i\sum_{k=1}^{n}\Gamma_k X_k\right) \tag{13}$$

QAOA is interesting for our purposes because of the following supremacy result.

Theorem 2.3 (informal from [32])**.** *Given an arbitrary QAOA circuit C of the form (6) with U_f^{QAOA} as in (13) the probability distribution over measurement outcomes is:*

$$p^{\mathsf{QAOA}}(\mathbf{x}) = \left| \langle \mathbf{x}| U_f^{\mathsf{QAOA}} \prod_{j=1}^{m} U_z(\alpha_j, S_j) H^{\otimes n} |0\rangle^{\otimes n} \right|^2 \tag{14}$$

If we have a poly-time randomised classical algorithm that takes as input a description of C and outputs a string $\mathbf{x}$ with probability $q(\mathbf{x})$ satisfying the multiplicative error bound[5]*:*

$$\left| q(\mathbf{x}) - p^{\mathsf{QAOA}}(\mathbf{x}) \right| \leq 0.1 p^{\mathsf{QAOA}}(\mathbf{x}) \tag{15}$$

then PH collapses.

2.4 Supremacy of Quantum Learning

Here we provide, to the best of our knowledge, the first formalisation of what we call '*Quantum Learning Supremacy*', specifically for distribution learning. We model our definitions around those provided in [40], which pertain to the theory of classical distribution learnability.

Intuitively, a generative quantum machine learning algorithm can be said to have demonstrated 'Quantum Learning Supremacy', if it is possible to efficiently learn a representation of a distribution which for which there does not exist a classical learning algorithm achieving the same end. More specifically, the quantum device has the ability to produce samples according to a distribution that is close in total variation to some distribution, using a polynomial number of samples from that distribution. However, there should be no classical algorithm which could achieve this.

We now formalise this intuition. First we must understand the inputs and outputs to learning algorithm. The inputs are samples from the distribution to be learnt, either classical bitstrings, or which could be quantum states encoding a superposition of such bitstring states, i.e. *qsamples* [41]. A generator can be interpreted as a routine that simulates sampling from the distribution. As in [40], we will assume only discrete distribution classes, $\mathcal{D}_n$, over binary vectors of length n.

Definition 2.4 (Generator [40])**.** *A class of distributions, $\mathcal{D}_n$ has efficient Generators, GEN_D, if for every distribution $D \in \mathcal{D}_n$, GEN_D produces samples in $\{0,1\}^n$ according to the exact distribution D, using polynomial resources. The generator may take a string of uniformly random bits, of size polynomial in n, $r(n)$, as input.*

[5]This form of multiplicative error is essentially the same as that in Definition (2.2).

The reader will notice that this definition allows, for example, for the Generator to be either a classical circuit, or a quantum circuit, with polynomially many gates. Further, in the definition of a classical Generator [40] a string of uniformly random bits is taken as input, and then transformed into the randomness of D. However, a quantum Generator would be able to produce its own randomness and so no such input is necessary. In this case the algorithm could ignore the input string $r(n)$.

While we are predominately interested in efficient learning with a Generator, one can also define a similar *Evaluator*:

Definition 2.5 (Evaluator [40])**.** *A class of distributions, $\mathcal{D}_n$ has efficient Evaluators, EVAL_D, if for every distribution $D \in \mathcal{D}_n$, EVAL_D produces the weight of an input $\mathbf{y}$ in $\{0,1\}^n$ under the exact distribution D, i.e. the probability of $\mathbf{y}$ according to D. The Evaluator is* efficient *if it uses polynomial resources.*

The distinction between EVAL and GEN is important and interesting in this case since the output probabilities of even IQP circuits are $\#\mathsf{P}$-Hard to compute, [16] and also hard to sample from by classical means, yet the distributions they produce can be sampled from efficiently by a quantum computer. This draws parallels to examples in [40] where certain classes of distributions are shown not to be learnable efficiently with an Evaluator, but they *are* learnable with a Generator. We also wish to highlight the connections to the definitions of strong and weak simulators of quantum circuits, Definition 2.1 to reinforce the similarity between Supremacy and Learning. An Evaluator for a quantum circuit would be a strong simulator of it, and a Generator would be a weak simulator. However, we keep these definitions separate in order to connect the hardness and learnability ideas explicitly.

For our purposes, the following definitions of learnable will be used. In contrast to [40], who was concerned with defining a 'good' generator to be one which achieves closeness relative to the Kullback-Leibler (KL) divergence, we wish to expand this to general cost functions, d. This is due to the range of cost functions we have access to and our wish to connect to the quantum circuit hardness results mentioned above, which typically strive for closeness in (TV).

Definition 2.6 ((d,ϵ)-Generator)**.** *For a cost function, d, let $D \in \mathcal{D}_n$. Let $\mathsf{GEN}_{D'}$ be a Generator for a distribution D'. We say GEN is a (d,ϵ)-Generator for D if $d(D,D') \leq \epsilon$.*

A similar notion of an ϵ-good Evaluator could be defined.

Definition 2.7 ((d,ϵ,C)-Learnable)**.** *For a metric d, $\epsilon > 0$, and complexity class C, a class of distributions $\mathcal{D}_n$ is called (d,ϵ,C)-learnable (with a Generator) if there exists an algorithm $\mathcal{A} \in \mathsf{C}$, called a learning algorithm for $\mathcal{D}_n$, which given $0 < \delta < 1$ as input, and given access to GEN_D for any distribution $D \in \mathcal{D}_n$, outputs $\mathsf{GEN}_{D'}$, a (d,ϵ)-Generator for D, with high probability:*

$$Pr\left[d\left(D,D'\right) \leq \epsilon\right] \geq 1-\delta \tag{16}$$

$\mathcal{A}$ should run in time $poly(1/\epsilon, 1/\delta, n)$.

In Definition 2.7, ϵ may, for example, be a function of the inputs to the learning algorithm. We may also wish to require a learnability definition which holds for all $\epsilon > 0$. This definition would, however, be too strong for our purposes. In order to claim Quantum Learning Supremacy of a Learning Algorithm, we only need to achieve closeness up to a *fixed* TV distance. This will be discussed in more detail in Section 7.2. An illustration of the procedure can be seen in Figure 2. Finally, we define what it would mean for a quantum algorithm to be superior to any classical algorithm for the problem of distribution learning:

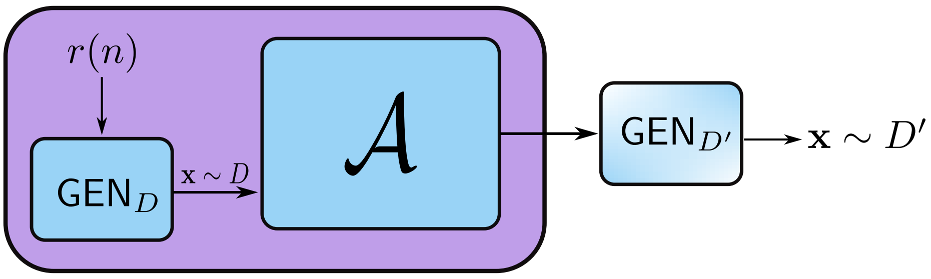


Figure 2: Illustration of a learning procedure using a Generator.

Definition 2.8 (Quantum Learning Supremacy)**.** *An algorithm $\mathcal{A} \in \mathsf{BQP}$ is said to have demonstrated the supremacy of quantum learning over classical learning if there exists a class of distributions $\mathcal{D}_n$ for which there exists d, ϵ such that $\mathcal{D}_n$ is $(d, \epsilon, \mathsf{BQP})$-Learnable, but $\mathcal{D}_n$ is not $(d, \epsilon, \mathsf{BPP})$-Learnable.*

As mentioned above, a typical choice of d would be $d = \mathsf{TV}$, but one could imagine weaker definitions by using weaker cost functions, which will be discussed further in Section 7.2. One may also be more restrictive and look for a demonstration of Learning superiority by a class which was efficiently IQP-Learnable, but not BPP-Learnable. This case may be more challenging to prove theoretically, but may be more amenable for the near term, precisely the original motivation for Quantum Supremacy, and, indeed, implies Definition 2.8.

3 Ising Born Machine

The Born Machine (BM) [7, 8] is the natural utilisation of the measurement postulate of quantum mechanics in generative modelling and applied to QCL. In particular, the Born rule gives this generative model both its name and its sampling process, as detailed in Section 3.1.

The Born Machine definition originated from tensor network approaches to define generative algorithms and the connection between physical systems and machine learning problems [42–48]. Since then, other works have given variants of the original definition [25], adversarial training approaches for the model [49], and adaptions to the continuous variable regime [50].

It is likely, since the statistics which the Born machine produces are generated by the fundamental randomness of quantum mechanics, that there should be no classical analogue to the model. In this regard, there is hope that a model could be defined which provably cannot be simulated by classical means, and by extension, could outperform *any* classical model for certain learning problems, as discussed in Section 7.2.

To accommodate our requirement for the model to be implementable on NISQ devices we will restrict the Born machine, which in general could be implemented using any quantum circuit, to an Ising version. In particular, the model will be a parameterised circuit of the form discussed in Section 2.3.

3.1 Definitions

The Model The Ising Born Machine Model is the state produced by the circuit discussed in Section 2.3, where in this case we have fixed S to be the set of all $S_j \subseteq [n]$ such that

$|S_j| \leq 2$, i.e the computation consists of gates acting on either one or two qubits. The state obtained by this circuit is the following.

$$|\psi^\theta\rangle = U_f\left(\mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}\right) \prod_{j=1}^{m} U(\alpha_j, S_j) H^{\otimes n} |0\rangle^{\otimes n} = \bigotimes_{k=1}^{n} U_f(\Gamma_k, \Delta_k, \Sigma_k) \prod_{j=1}^{m} U(\alpha_j, S_j) |+\rangle^{\otimes n} \tag{17}$$

This restriction to two qubit gates suffices for the hardness proofs we discuss in Section 3.2.

The goal of the training will be to alter the parameters α_j so that, upon measuring the state, it produces samples according to the target distribution. This will be done using QCL discussed in Section 2.1.

This approach can be compared to [9] where the authors use QCL in a classification algorithm. Typical approaches so far [8, 30] require making the circuit depth as large as possible and introducing extra parameters through single qubit gates. Clearly, this approach would lead to better approximations to the data since more parameters typically leads to more accurate fits.

However, the approach we will use is somewhat different. We are interested in choosing a circuit class which is as shallow as possible, but which is sufficiently complex to be hard to simulate classically. For this purpose, we choose a model which encapsulates the sub-universal circuit classes mentioned in Section 2.3. Notice, we also choose the final gate to be U_f, defined by (5), rather than the more standard $R_zR_xR_z$ decomposition found in other Born Machine works, [25, 45]. Both are effectively equivalent; they can both generate any arbitrary single qubit gate (up to a phase). However, we chose our construction to make the hardness connection more transparent.

With the restrictions set discussed above, the term in the exponent of the diagonal unitary (18) can be written as an Ising Model Hamiltonian, [31, 34]:

$$\sum_j \alpha_j \bigotimes_{i \in S_j} Z_i = \mathcal{H}_z = \sum_{i<j} J_{ij} Z_i Z_j + \sum_{k=1}^{n} b_k Z_k \tag{18}$$

$$\implies \prod_{j=1}^{m} U_j(\alpha_j, S_j) = e^{i\mathcal{H}_z} \equiv U_z(\boldsymbol{\alpha}) \tag{19}$$

The parameters $\boldsymbol{\alpha} = \{J_{ij}, b_k\}$ can be viewed as the coupling and local magnetic fields respectively. The evolved state is then:

$$|\psi^\theta\rangle = \bigotimes_{k=1}^{n} U_f\left(\Gamma_k, \Delta_k, \Sigma_k\right) U_z(\boldsymbol{\alpha}) |+\rangle^{\otimes n} \tag{20}$$

Where $\theta = \{\mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}, \boldsymbol{\alpha}\}$ refers to *all* parameters of the IBM circuit. To implement this circuit on NISQ hardware it is necessary to decompose the unitary, $U_z(\boldsymbol{\alpha})$ into single and two qubit gates. This is straightforward to do since all the terms in $U_z(\boldsymbol{\alpha})$ mutually commute. It is possible to find such a decomposition into two qubit $CZ(\alpha)$ gates, defined by the unitary matrix, (21), and single qubit rotations around the Pauli-Z axis.

$$CZ(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{pmatrix} \tag{21}$$

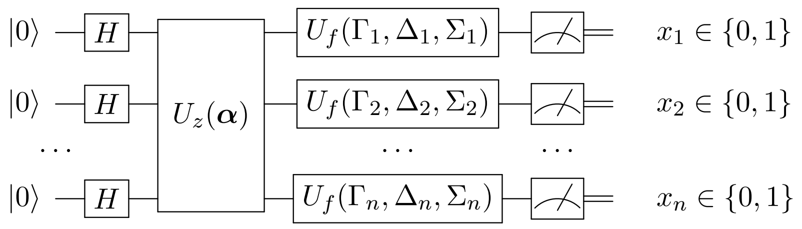
Using the decomposition from [51], the relationship between a $CZ_{ij}(\alpha)$ gate between qubit i and j, and an Ising interaction, $\exp(i\alpha Z_i \otimes Z_j)$ is as follows:

$$CZ_{ij}(\alpha) = e^{i\alpha |1\rangle\langle 1|_i \otimes |1\rangle\langle 1|_j} = e^{\frac{i\alpha}{4}(\mathbb{1}_i - Z_i)\otimes(\mathbb{1}_j - Z_j)} = e^{\frac{i\alpha}{4}\mathbb{1}_i\otimes\mathbb{1}_j} e^{-\frac{i\alpha}{4}\mathbb{1}_i\otimes Z_j} e^{-\frac{i\alpha}{4}Z_i\otimes\mathbb{1}_j} e^{\frac{i\alpha}{4}Z_i\otimes Z_j} \tag{22}$$

Therefore, $U_z(\boldsymbol{\alpha})$ can be expanded as follows:

$$\begin{aligned} U_z(\boldsymbol{\alpha}) &= \prod_{i<j} e^{iJ_{ij}Z_iZ_j} \prod_k e^{ib_kZ_k} \\ &= \prod_{i<j,k} e^{-iJ_{ij}} e^{iJ_{ij}\mathbb{1}_i\otimes Z_j} e^{iJ_{ij}Z_i\otimes\mathbb{1}_j} CZ_{i,j}(4J_{ij}) R_z^k(-2b_k) \\ &= \prod_{i<j,k} e^{-iJ_{ij}} CZ_{i,j}(4J_{ij}) R_z^i(-2J_{ij}) R_z^j(-2J_{ij}) R_z^k(-2b_k) \end{aligned} \tag{23}$$

Hence the specific circuit used will be given by (24):



$|0\rangle$
H
$|0\rangle$
H
$\cdots$
$|0\rangle$
H
$U_z(\boldsymbol{\alpha})$
$U_f(\Gamma_1, \Delta_1, \Sigma_1)$
$U_f(\Gamma_2, \Delta_2, \Sigma_2)$
$\cdots$
$U_f(\Gamma_n, \Delta_n, \Sigma_n)$
$x_1 \in \{0,1\}$
$x_2 \in \{0,1\}$
$x_n \in \{0,1\}$

(24)

Previous approaches, [8, 25] do not add parameters to the entangling gates, (CNOT gates in those cases), which has an advantage from an experimental point of view, but immediately increases the circuit depth. In contrast, by parameterising the entangling gates $U_z(J_{ij}) = \exp(iJ_{ij}Z \otimes Z)$ also, we can get up to n^2 more parameters 'for free'. Of course, this may reduce the effectiveness of the learning algorithm in a physical implementation, since in the latter approach there is only a need to create *some* entanglement, whereas we require creating *specific* entanglement, i.e. a precise implementation of the parameters, J_{ij}. However, we will leave a more thorough treatment of this trade-off to future work.

In the following we will use the notation to refer to an Ising Born Machine with parameters θ: $\mathsf{IBM}(\theta) = \mathsf{IBM}(\boldsymbol{\alpha}, \boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma}) = \mathsf{IBM}(\{J_{ij}, b_k\}, \boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma})$.

Sampling As mentioned above, the probability distribution of measuring a quantum state, $|\psi\rangle$ in the computational basis, with outcome string, $\mathbf{x}$, is given by Born's Rule, (1), which defines the core operating principle of a Born Machine. The resulting output probability distribution can be written as follows:

$$p_\theta(\mathbf{x}) = \left| \langle \mathbf{x}| U_f(\boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma}) \prod_{j=1}^{m} U_j(\alpha_j, S_j) |+\rangle^{\otimes n} \right|^2 = \mathsf{tr}\left(|\mathbf{x}\rangle \langle \mathbf{x}|\psi^\theta\rangle \langle \psi^\theta| \right) \tag{25}$$

Training We provide novel training methods for the model, which could outperform those given in the literature [8, 10, 25]. This out-performance will be measured by the closeness of the model to the data relative to total variation. As we shall see, this will involve a trade-off between the classical and quantum computing power required to train the model, which will be specific to the training algorithm. Previous methods advocate increasing the

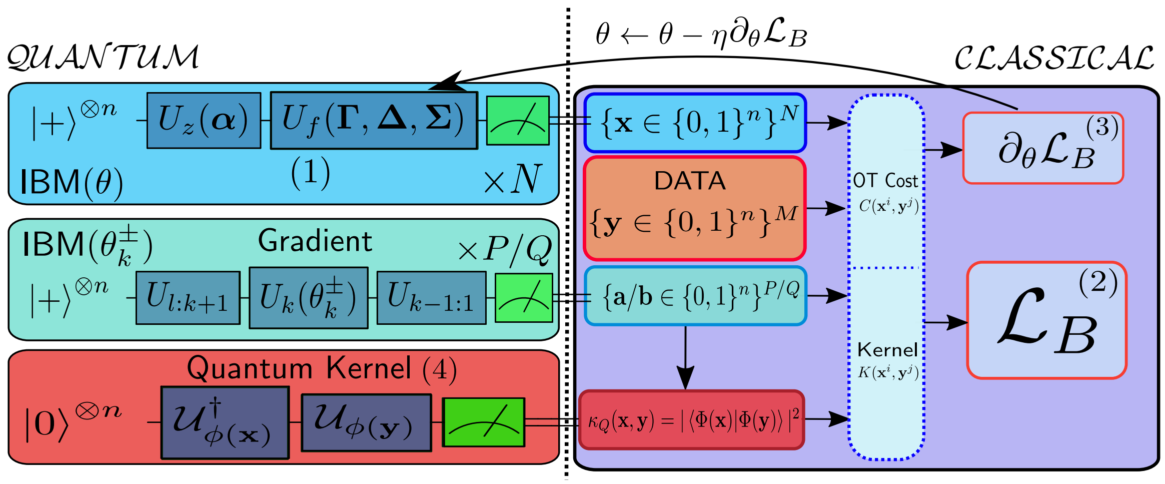

$\theta \leftarrow \theta - \eta\partial_\theta\mathcal{L}_B$
QUANTUM
CLASSICAL
$|+\rangle^{\otimes n}$
$U_z(\boldsymbol{\alpha})$
$U_f(\boldsymbol{\Gamma},\boldsymbol{\Delta},\boldsymbol{\Sigma})$
IBM(θ)
(1)
$\times N$
IBM($\theta_k^{\pm}$)
Gradient
$\times P/Q$
$|+\rangle^{\otimes n}$
$U_{l:k+1}$
$U_k(\theta_k^{\pm})$
$U_{k-1:1}$
Quantum Kernel (4)
$|0\rangle^{\otimes n}$
$\mathcal{U}^\dagger_{\phi(\mathbf{x})}$
$\mathcal{U}_{\phi(\mathbf{y})}$
$\{\mathbf{x} \in \{0,1\}^n\}^N$
DATA
$\{\mathbf{y} \in \{0,1\}^n\}^M$
$\{\mathbf{a}/\mathbf{b} \in \{0,1\}^n\}^{P/Q}$
$\kappa_Q(\mathbf{x},\mathbf{y}) = |\langle\Phi(\mathbf{x})|\Phi(\mathbf{y})\rangle|^2$
OT Cost
$C(\mathbf{x}^i,\mathbf{y}^j)$
Kernel
$K(\mathbf{x}^i,\mathbf{y}^j)$
$\partial_\theta\mathcal{L}_B$ (3)
$\mathcal{L}_B$ (2)

Figure 3: Hybrid training procedure of Ising Born Machine.

quantum part (by increasing the circuit depth), whereas we attempt to provide means to reduce the circuit depth, by leveraging more classical computation power to squeeze as much out of the given circuit as possible. This type of approach would make training quantum models more applicable to near term devices, and corresponds to leaning more heavily on the 'Classical' part of Figure 3. We adopt a 'plug-and-play' mentality in this work, depending on the compute resources available, whether quantum or classical, we expand the realm of choices one can choose from. Our contributions in this regard can be summarised as follows, corresponding to the various parts of the Figure 3.

(1) We define a new sub-universal circuit class, the IBM(θ) class.

(2) We introduce new cost functions, namely $\mathcal{L}_B$.

(3) We derive the corresponding gradients for these costs.

(4) We introduce the option to have quantum kernels in generative modelling.

The training process of the IBM is a hybrid approach, containing one quantum '*generation*' phase, and one classical '*update*' phase. It proceeds as in Figure 3 and as follows:

1. **Compute Phase:** Instantiate circuit parameters, $\theta = \{J_{ij}, b_k, \Gamma_k, \Delta_k, \Sigma_k\}$ at random, and either in full generality or adapted to the specific hardware, [52].

2. **Compare Phase:** Apply circuit (17) to initial state, and measure all qubits in computational basis to generate one sample, $\mathbf{z}$. Repeat N times to generate a set of $\hat{z} = \{\mathbf{z}^1, \dots, \mathbf{z}^N\}$ samples. This is illustrated by the left of Figure 3. Compute some loss function, $\mathcal{L}_B$ with respect to data samples. This is the 'classical' part of the algorithm in Figure 3.

3. **Modify Phase:** Compute gradient of $\mathcal{L}_B$ and update parameters θ using a classical optimiser according to update rule, $\theta \leftarrow \theta - \eta\partial_\theta\mathcal{L}_B$. Repeat, indicated by leftward arrow in Figure 3, until convergence of $\mathcal{L}_B$ to a minimum.

This training process, and cost functions, $\mathcal{L}_B$, will be detailed in Section 4.

3.2 Hardness of Ising Born Machine Circuits

3.2.1 Multiplicative Error Hardness of Ising Born Machine Circuits

In this section, we demonstrate how the core of the IBM, the underlying circuit, should be hard to sample from efficiently by purely classical means, up to multiplicative error, by pulling together the relevant hardness results from previous works in this area. We will prove the following.

Theorem 3.1. *If the output probability distributions generated by uniform families of* $\mathsf{IBM}(\theta)$ *circuits could be weakly classically simulated to within multiplicative error* $1 \leq c \leq \sqrt{2}$ *then* $\mathsf{PostBPP} = \mathsf{PP}$ *(and hence the polynomial hierarchy collapses to the third level), where* $\forall k, i, j,$*:*

$$J_{ij}, b_k = \begin{cases} \frac{(2l+1)\pi}{8d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{26}$$

and for each of the following instances:

$$\Gamma_k = 0, \Delta_k = \begin{cases} \frac{(2l+1)\pi}{8d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{27}$$

$$\Delta_k = 0, \Gamma_k = \begin{cases} \frac{(2l+1)\pi}{4d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{28}$$

$$\Delta_k = 0, \Gamma_k = \Sigma_k = \begin{cases} \frac{(2l+1)\pi}{2\sqrt{2}d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{29}$$

As discussed in Section 2 the following choices of $\mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}$ give the more well known circuit classes:

$$\mathsf{IBM}\left(\{J_{ij}, b_k\}, \forall k : \Gamma_k = \frac{\pi}{2\sqrt{2}}, \Delta_k = 0, \Sigma_k = \frac{\pi}{2\sqrt{2}}\right) = \mathsf{IQP}(\{J_{ij}, b_k\}) \tag{30}$$

$$\mathsf{IBM}(\{J_{ij}, b_k\}, \mathbf{\Gamma} = -\mathbf{\Gamma}, \mathbf{0}, \mathbf{0}) = \mathsf{QAOA}_{p=1}(\{J_{ij}, b_k\}, \mathbf{\Gamma}) \tag{31}$$

These are simply *worst case* hardness results and they tell us nothing about the *specific* circuit which would be hard, only that there exists one, generated by these intermediate gates, which is hard to simulate classically up to multiplicative error. See Appendix A for a proof of Theorem (3.1), and a concrete separation of our work from previous works.

3.2.2 Total Variation Distance Hardness of Ising Born Machine Circuits

We can improve the hardness of the model by incorporating a stronger result about the circuit class, IQP. The IBM model must be initialised to some setting of the circuit parameters to begin the training process. One such possible initialisation is to randomly assign J_{ij}, b_k to some subset with uniform probability. The random initialisation is typical in Machine Learning, but it is not the only way one could initialise the parameters. In this case however, we will do so in order to use the IQP result of [34].

Firstly, define the *Partition Function*, $\mathcal{Z}$, associated with the Ising Hamiltonian, (18):

$$\mathcal{Z} = \sum_{z \in \{\pm 1\}^n} e^{i \sum_{i<j} J_{ij} z_i z_j + i \sum_{k=1}^{n} b_k z_k} \tag{32}$$

Now, an output amplitude of an IQP circuit can be written as this partition function:

$$2^n \langle \mathbf{z}| U_z(\boldsymbol{\alpha}) |\mathbf{z}\rangle = \mathcal{Z} \tag{33}$$

Now the more formal result of Theorem 2.2 states the following:

Theorem 3.2 ([34])**.** *Assume it is* # P*-hard to approximate* $|\mathcal{Z}|^2$ *up to a relative error* $1/4+o(1)$ *for* $1/24$ *fraction of instances over the choice of the weights and biases,* J_{ij}, b_k*. If it is possible to classically sample from the output probability distribution of any* IQP *circuit in polynomial time, up to an additive error of* $1/384$ *in total variation distance, then there is a* $\mathsf{BPP}^{\mathsf{NP}}$ *algorithm to solve any problem in* $\mathsf{P}^{\#\,\mathsf{P}}$*, and hence the polynomial hierarchy collapses to the third level)*

This holds if the parameters are chosen uniformly at random from $\{J_{ij}, b_k\} \in \{0, \frac{\pi}{8}, \dots, \frac{7\pi}{8}\}$, which corresponds to randomly choosing a circuit from the IQP circuit class according to some measure over the unitary group. We refer to [34] for the proof of the above. This is done by introducing a conjecture (included in Theorem (3.2)), which claims that *on average* (i.e. over a $\frac{1}{24}$ fraction of instances) the Ising partition function is hard to compute up to a multiplicative error. Again, this holds only in the worst case, and translates between an average case conjecture in multiplicative error into a worst case result in total variation error. Using this, if we were to choose the setting of parameters such that they correspond to an IQP circuit, IBM $\left(\{J_{ij}, b_k\}, \{\frac{\pi}{2\sqrt{2}}\}, \mathbf{0}, \{\frac{\pi}{2\sqrt{2}}\}\right)$, and initialise J_{ij}, b_k to the values above, then we shall start with an IBM in a regime which is also hard to simulate classically up to a variation distance error in the worst case. Now, a random initialisation of parameterised quantum circuits has been shown to lead to 'barren plateaus', [53], in which the gradient with respect to some quantum circuits becomes exponentially small, and so one would need exponential resources to estimate it. This indeed could be an issue for training such IBM circuits as the number of qubits scales, but this question is currently under investigation, with some potential solutions found for circuit initialisation [54]. However, in the following sections, we will assume a random (but fixed) initialisation for simplicity as we are solely interested in the performance of various cost functions. Furthermore the barren plateau issue should not be a major issue for the small system sizes we consider here.

4 Training the Ising Born Machine

4.1 Kernel Methods

Before diving into the cost functions used to train the IBM, we will need to refer to *kernel methods*, which are a workhorse of machine learning. Kernel methods are a popular technique, for example in dimensionality reduction, and are beginning to be brought into the quantum realm [55, 56]. A kernel is a symmetric function, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is positive definite, meaning that the matrix it induces, called a *Gram* matrix, is positive semi-definite.

In models like support vector machines (SVM), the idea is to to *embed* the underlying sample space, $\mathcal{X}$, into a Hilbert Space using a non-linear map, $\phi : \mathcal{X} \to \mathcal{H}$ called a *feature map*. A kernel is simply defined by the inner product of this feature map at two different points in a Reproducing Kernel Hilbert Space (RKHS). Intuitively, the map should be designed to make the points more easily comparable in the RKHS. The choice of the kernel function (i.e. the choice of the RKHS) may allow different properties to be compared. For a comprehensive review of techniques in kernel embeddings, see [57].

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:*

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}} \tag{34}$$

The full proof that this feature map gives rise to a kernel, a positive semi-definite and symmetric function, can be found in [55] for quantum feature maps, $\phi(\mathbf{x})$.

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

We can also define the *mean embedding*, which will become relevant in Section 4.3:

$$\mu_P = \mathbb{E}_P(\kappa(\mathbf{x}, \cdot)) = \mathbb{E}_P[\phi(\mathbf{x})] \in \mathcal{H} \tag{35}$$

A canonical example of a kernel is the Gaussian Kernel, defined by:

$$\kappa_G(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||_2^2}{2\sigma^2}\right) \tag{36}$$

where $|| \cdot ||_2^2$ is the Euclidean distance, or squared ℓ_2 norm. The parameter, σ, is known as a *bandwidth*, and determines the scale at which the samples are compared. It is also possible to use a Gaussian mixture; a sum of Gaussians with different bandwidth parameters, as in [8], which will be our benchmark with which to compare.

As mentioned above, we wish to use *quantum* kernels, i.e. those arises as a result of an state overlap in a Quantum RKHS. This notion was first presented by [55]. To the best of our knowledge, we are the first to investigate the possibility of introducing quantum kernels to generative modelling, as we shall do. The form of the kernel assumed by [55] is the one induced by the inner product on a quantum (i.e. complex) Hilbert space:

$$\kappa_Q(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle \tag{37}$$

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 wave functions which encode two samples and is therefore a transition amplitude. As mentioned in [55], it would be desirable to find kernels which are hard to compute classically, in order to gain some quantum advantage. The first potential candidate for such a situation was derived by [56]. As such, to remain consistent with the notation of [56], the kernel is defined as the real transition *probability* instead. As mentioned in [55] it is possible to define a kernel this way by taking the modulus squared of (37), 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:*

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

The final ingredient is the discussion of the computability of this kernel function, since the evaluation of the cost function introduced in Section 4.2 and training process requires this to be done many times.

In [55], 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, a key ingredient in continuous variable quantum computing, [23], which are known to be classically simulable. However, to gain a quantum advantage it would be desirable to use a kernel which is (conjectured to be) *not* classically computable efficiently. For this reason, we use the kernel proposed in [56], which is defined by a feature map constructed by the following quantum circuit:

$$\Phi : \mathbf{x} \in \mathcal{X} \rightarrow |\Phi(\mathbf{x})\rangle \tag{38}$$

$$|\Phi(\mathbf{x})\rangle = \mathcal{U}_{\Phi(\mathbf{x})} |0\rangle^{\otimes n} = U_{\Phi(\mathbf{x})} H^{\otimes n} U_{\Phi(\mathbf{x})} H^{\otimes n} |0\rangle^{\otimes n} \tag{39}$$

so the resulting kernel is:

$$\kappa_Q(\mathbf{x}, \mathbf{y}) = |\langle \Phi(\mathbf{x}) | \Phi(\mathbf{y}) \rangle|^2 \tag{40}$$

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

$$U_{\Phi(\mathbf{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\mathbf{x}) \prod_{i \in S} Z_i \right) \tag{41}$$

This is exactly the same form as the diagonal unitary in the IBM, (18), 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[6] in (39).

It should be noted that this is experimentally favourable to work alongside the original IBM 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 Ising scenario, only single and two-qubit are operations are required:

$$U_{\phi_{\{l,m\}}(\mathbf{x})} = \exp\left(i\phi_{\{l,m\}}(\mathbf{x}) Z_l \otimes Z_m\right) \qquad U_{\phi_{\{k\}}(\mathbf{x})} = \exp\left(i\phi_{\{k\}}(\mathbf{x}) Z_k\right) \tag{42}$$

The arguments of the gates, (42), used to encode the samples is given by:

$$\phi_{\{l,m\}}(\mathbf{x}) = \left(\frac{\pi}{4} - x_l\right)\left(\frac{\pi}{4} - x_m\right) \qquad \phi_{\{k\}}(\mathbf{x}) = \frac{\pi}{4} x_k \tag{43}$$

The choice of this kernel is motivated by [56], which conjectures that this overlap, (40), 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. A rough bound for the number of measurements required to compute the full matrix is also given by [56]. The only difference is that in that case, the Gram matrix was computed using only samples

[6]It is noted in [56] 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 feature map, (38), as it will be #P-hard for some sample values. However, to rule out an *additive* error approximation, it is necessary to add the second encoding layer.

from the same source. In our example, this need not be the case. We shall see that it in general, it requires $|B|$ samples from one source (the IBM for example), and $|D|$ samples from another (the data). The resulting Gram matrix will then be of dimension $|B| \times |D|$. However, to simplify the expressions, we can assume that we have the same number of samples from each: $|B| = |D| = T$. Therefore, directly from [56], to compute a single kernel entry to precision $\tilde{\epsilon} = O(R^{-1/2})$ requires $R = O(\tilde{\epsilon}^{-2})$ measurement shots. Then an approximation, $\hat{K}$, of the Gram matrix for the kernel, $K_{\mathbf{x},\mathbf{y}} = \kappa(\mathbf{x}, \mathbf{y})$, which has $T(T-1)$ non-trivial entries, that is ϵ-close in operator norm $||K - \hat{K}|| \leq \epsilon$ can be determined using $R = O(\epsilon^{-2}T^4)$ measurement shots.

4.2 Cost Functions for Ising Born Machine

To train the Born Machine in general, it is necessary to have some '*cost function*' to compare instantaneously between our model distribution, given by $p_\theta(\mathbf{x})$, and the data distribution which we are trying to learn, represented by $\pi(\mathbf{y})$, using the notation of [8]. This should be distinguished from the kernel methods in the previous section, a kernel can only compare *between individual points*, $\mathbf{x}, \mathbf{y}$. However, we need to also be able to distinguish between the *distributions* which generate those points.

Furthermore, we will consider only gradient based methods to train the model, for example stochastic gradient descent. Of course, it is possible to train these models using gradient free approaches [10], but these tend to perform poorly in large parameter spaces. Gradient descent involves the following updates to the parameters of the model at each 'epoch', d, where one epoch refers to one iteration over all parameters:

$$\theta^{d+1} \leftarrow \theta^d - \eta \partial_{\theta^d} \mathcal{L}_B \tag{44}$$

where $\mathcal{L}_B$ is the cost function in question. η is the *learning rate*, and controls the speed of the descent.

The first distribution measure which could be used is the Kullback-Leibler (KL) Divergence. The cost function associated with the KL divergence is given by:

$$\mathcal{L}_{\mathsf{KL}} = -\sum_{\mathbf{z}} \pi(\mathbf{z}) \log\left(p_\theta(\mathbf{z})\right) = -\mathbb{E}_{\mathbf{z}\sim\pi}(\log(p_\theta(\mathbf{z})) \tag{45}$$

Unfortunately, it is not possible to use the KL divergence for training the Born Machine, as noted by [8]. This is because the gradient of the KL cost, $\mathcal{L}_{\mathsf{KL}}$, cannot be estimated efficiently for these types of QCL models:

$$\frac{\partial \mathcal{L}_{\mathsf{KL}}}{\partial \theta} \sim \sum_{\mathbf{z}} \frac{\pi(\mathbf{z})}{p_\theta(\mathbf{z})} \left(p_\theta^-(\mathbf{z}) - p_\theta^+(\mathbf{z})\right) \tag{46}$$

The notation $p_\theta^\pm$ will be explained in the next section. As noted in [8, 16], computing the required probabilities p_θ is $\#$P-hard, and so (46) cannot be computed efficiently. Furthermore, the KL Divergence is an example of a so-called f-divergence [58], and such measures are notoriously hard to estimate in practice. A potential solution to this is to use an alternative family of discrepancies, called *Integral Probability Metrics* (IPMs). This approach was first adopted by [8] in the form of the *Maximum Mean Discrepancy* (MMD) to train a Quantum Circuit Born Machine (QCBM), and we will expand on it in this work.

General IPM's are defined by the following equation:

$$\gamma_{\mathcal{F}}(P,Q) = \sup_{\phi\in\mathcal{F}} \left| \int_{\mathcal{M}} \phi dP - \int_{\mathcal{M}} \phi dQ \right| \tag{47}$$

$$= \sup_{\phi\in\mathcal{F}} \left(\mathbb{E}_P[\phi] - \mathbb{E}_Q[\phi]\right) \tag{48}$$

Where P, Q are defined over a measurable space, $\mathcal{M}$. The class of functions which are chosen, $\mathcal{F}$, defines the specific metric.

The MMD is one of the simplest such example of an IPM, since it is relatively easy to compute and more details on this point will be given in Section 7. The MMD was first used in [59] as a hypothesis test, to test if two distributions are identical. For a more thorough treatment of the properties of this metric, see [60, 61].

The MMD can be extracted from (47) by restricting the class of functions, $\mathcal{F}$, to be a unit ball in a Hilbert space:

$$\mathcal{F}_{\mathsf{MMD}} = \{\phi \in \mathcal{H} : ||\phi||_{\mathcal{H}} \leq 1\} \tag{49}$$

The second IPM which will be relevant here is the total variation (TV) was also shown in [58], to be the only cost function which is both an IPM, and an f-divergence. The TV can be obtained by taking the function space as follows:

$$\mathcal{F}_{\mathsf{TV}} = \{\phi : ||\phi||_{\infty} \leq 1\} \tag{50}$$

Where $||\phi||_{\infty} = \sup\{|\phi(\mathbf{x})|, \mathbf{x} \in \mathcal{M}\}$. Also, when working with a discrete[7] space, $\mathcal{M} = \mathcal{X}$, the total variation is given by:

$$\mathsf{TV} = \sum_{\mathbf{x}\in\mathcal{X}} |P(\mathbf{x}) - Q(\mathbf{x})| \tag{51}$$

This measure is particularly important, since it occurs in the definition of additive error simulation hardness, Definition (2.3). It will also be the benchmark we shall use for numerical experiments in Section 6.

Finally, the third useful IPM is the Kantorovich Metric, defined by:

$$\mathcal{F}_{\mathsf{W}} = \{\phi : ||\phi||_{L} \leq 1\} \tag{52}$$

where $||\cdot||_L$ is the Lipschitz semi-norm, defined by: $||\phi||_L := \sup\{|\phi(x) - \phi(y)|/d(\mathbf{x},\mathbf{y}) : \mathbf{x} \neq \mathbf{y} \in \mathcal{M}\}$, where d is a metric on $\mathcal{X}$. It turns out due to the Kantorovich-Rubinstein theorem [62], this is also equivalent to the *Wasserstein Metric* and related to the notion of Optimal Transport, which shall be discussed in more detail in Section 4.5.

4.3 MMD Training of Ising Born Machine

Due to the restriction of the function space for the MMD, it is possible to use the kernel trick discussed in Section 4.1 and equate the class of functions to a feature map in a RKHS, $\phi \in \mathcal{F}$, where $\mathcal{F} = \{\phi : ||\phi||_{\mathcal{H}} \leq 1\}$. In this case the feature maps embed samples into the unit ball in the Hilbert Space, where $||\cdot||_{\mathcal{H}}$ is the norm in the space, $\mathcal{H}$. It can be

[7]More generally if the space is countable.

shown that the MMD is exactly the difference in mean embeddings (35) between the two distributions, [63, 64]:

$$\gamma_{\mathsf{MMD}}(P,Q) = ||\mu_P - \mu_Q||_{\mathcal{H}} \tag{53}$$

Using this, it is possible to define a cost function associated to this metric, exactly the one used in [8, 25] for their versions of the Born Machine:

$$\mathcal{L}_{\mathsf{MMD}} = \gamma_{\mathsf{MMD}}(P,Q)^2 = ||\mathbb{E}_P[\phi(\mathbf{x})] - \mathbb{E}_Q[\phi(\mathbf{x})]||^2_{\mathcal{H}} \tag{54}$$

$$\mathcal{L}_{\mathsf{MMD}} = \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim P\\ \mathbf{y}\sim P}}(\kappa(\mathbf{x},\mathbf{y})) + \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim Q\\ \mathbf{y}\sim Q}}(\kappa(\mathbf{x},\mathbf{y})) - 2\mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim P\\ \mathbf{y}\sim Q}}(\kappa(\mathbf{x},\mathbf{y})) \tag{55}$$

where κ is the MMD kernel. A requirement for the MMD to be useful (as a hypothesis test) is that the kernel used to define it must be *characteristic* or universal, [65, 66]. This is one property which allows it to be a valid metric on the space of probability distributions, with $\gamma_{\mathsf{MMD}}(P,Q) = 0 \iff P \equiv Q$. This enables the MMD to be used in hypothesis testing to determine if the null hypotheses ($P = Q$) holds or not. The Gaussian kernel (36) is indeed one which is characteristic [65], and some effort has been made to find conditions under which a kernel is characteristic [60].

In the case where the support of the distributions is discrete, as is the case here, the condition on the kernel being characteristic is *strict positive definiteness* [58]. Strict positive definiteness of the kernel is defined as the resulting Gram matrix being positive definite[8]. Now, to set the scene, we revisit the work of [8], in which the MMD is used to train the Born Machine. In [58], it is shown that the MMD can be estimated, given i.i.d. samples from two distributions, $\mathbf{x} = (x_1, \dots, x_N) \sim P$, $\mathbf{y} = (y_1, \dots, y_M) \sim Q$. This is done by simply replacing the expectation values in (55) with their empirical values.

$$\begin{aligned}\tilde{\mathcal{L}}_{\mathsf{MMD}} &= \tilde{\gamma}_{\mathsf{MMD}}(P_N, Q_M)^2 \\ &= \frac{1}{M(M-1)}\sum_{i\neq j}^{N}\kappa(x_i,x_j) + \frac{1}{N(N-1)}\sum_{i\neq j}^{M}\kappa(y_i,y_j) - \frac{2}{MN}\sum_{i,j}^{M,N}\kappa(x_i,y_j)\end{aligned} \tag{56}$$

As shown in [58], the above (56) demonstrates both consistency and lack of bias, and furthermore it converges fast in probability to the true MMD:

$$|\tilde{\gamma}_{\mathsf{MMD}}(P_N, Q_M) - \gamma_{\mathsf{MMD}}(P,Q)| \leq O_{P,Q}(N^{-1/2} + M^{-1/2}) \tag{57}$$

This quadratic convergence rate is highly desirable, since it does not depend on the dimension of the space from which the samples are drawn. This will be discussed further in Section 4.5.1. There is one point that must be checked in order to carry on with this. The derivation of the MMD loss estimator requires that the kernel be a bounded and measurable function. This is the case for Gaussian kernels, but we must check it holds for the quantum kernel, (40). Happily, this is the case: the overlap between two states is bounded above by 1, and the sample space consists of binary strings.

$$\kappa(\mathbf{x},\mathbf{y}) = |\langle\Phi(\mathbf{x})|\Phi(\mathbf{y})\rangle|^2 \leq 1 \qquad \forall \mathbf{x},\mathbf{y} \in \mathcal{X} \tag{58}$$

Now, to compute the derivative of this cost function, we follow the method of [8]. This approach will apply to all cost functions we employ in this work. In that work, the quantum

[8]A matrix is positive definite $\iff \forall \mathbf{x} \in \mathbb{R}^d/\mathbf{0}, \mathbf{x}^T K\mathbf{x} > 0$

gates with trainable parameters, η, are of the form $U(\eta) = \exp(-i\frac{\eta}{2}\Sigma)$, where Σ can be a series of operators satisfying $\Sigma^2 = 1$.

It is known [8, 30] that the gradient of an observable of the circuit, B, with respect to a parameter, η, is given by:

$$\frac{\partial \langle B\rangle_\eta}{\partial \eta} = \frac{1}{2}\left(\langle B\rangle_{\eta^+} - \langle B\rangle_{\eta^-}\right) \tag{59}$$

where $\eta^\pm$ are parameter shifted versions of the original circuits. As noted in [8], taking the observable to be a measurement in the computational basis, $B = M_{\mathbf{z}} = |\mathbf{z}\rangle\langle\mathbf{z}|$, we arrive at the following form of the gradient of the probabilities:

$$\frac{\partial p_\theta(\mathbf{z})}{\partial \theta_k} = p^-_{\theta_k}(\mathbf{z}) - p^+_{\theta_k}(\mathbf{z}) \tag{60}$$

The factor of $-1/2$ difference with this gradient from that of [8] is due to the slightly different parameterisation we choose, our gates are instead of the form: $\exp(i\eta\Sigma) : \{D_1(b_k) = \exp ib_kZ_k, D_2(J_{ij}) = \exp iJ_{ij}Z_iZ_j\}$. As mentioned above, this gradient requires computing the parameter shifted versions of the original circuit, $p^\pm_{\theta_k}$. This notation indicates that the kth parameter in the original circuit has been shifted by a factor of $\pm\frac{\pi}{2}$; $p^\pm_{\theta_k} = p_{\theta_k\pm\pi/2}$. This formula is actually valid for the parameter in any unitary which has *at most* two distinct eigenvalues [67].

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

$$\frac{\partial \mathcal{L}_{\mathsf{MMD}}}{\partial \theta_k} = 2\underset{\substack{\mathbf{a}\sim p^-_{\theta_k}\\ \mathbf{x}\sim p_\theta}}{\mathbb{E}}(\kappa(\mathbf{a},\mathbf{x})) - 2\underset{\substack{\mathbf{b}\sim p^+_{\theta_k}\\ \mathbf{x}\sim p_\theta}}{\mathbb{E}}(\kappa(\mathbf{b},\mathbf{x})) - 2\underset{\substack{\mathbf{a}\sim p^-_{\theta_k}\\ \mathbf{y}\sim \pi}}{\mathbb{E}}(\kappa(\mathbf{a},\mathbf{y})) + 2\underset{\substack{\mathbf{b}\sim p^+_{\theta_k}\\ \mathbf{y}\sim \pi}}{\mathbb{E}}(\kappa(\mathbf{b},\mathbf{y})) \tag{61}$$

$$\frac{\partial \mathcal{L}_{\mathsf{MMD}}}{\partial \theta_k} \approx \frac{2}{PN}\sum_{p,i}^{P,N}\kappa(\mathbf{a}^p,\mathbf{x}^i) - \frac{2}{QN}\sum_{q,n}^{Q,N}\kappa(\mathbf{b}^q,\mathbf{x}^i) - \frac{2}{PM}\sum_{p,m}^{P,M}\kappa(\mathbf{a}^p,\mathbf{y}^m) + \frac{2}{QM}\sum_{q,m}^{Q,M}\kappa(\mathbf{b}^q,\mathbf{y}^m) \tag{62}$$

where we have P, Q samples, $\hat{\mathbf{a}} = \{\mathbf{a}^1,\dots,\mathbf{a}^P\}, \hat{\mathbf{b}} = \{\mathbf{b}^1,\dots,\mathbf{b}^Q\}$ drawn from the parameter shifted circuits, $p^-_{\theta_k}(\mathbf{a}), p^+_{\theta_k}(\mathbf{b})$ respectively and N, M samples, $\hat{\mathbf{x}} = \{\mathbf{x}^1,\dots,\mathbf{x}^N\}, \hat{\mathbf{y}} = \{\mathbf{y}^1,\dots,\mathbf{y}^M\}$ drawn from the the original Born circuit and the data distribution respectively, $p_\theta(\mathbf{x}), \pi(\mathbf{y})$. The parameter shifted circuits are of the form (63):

$$|0\rangle^{\otimes n} - \boxed{H^{\otimes n}} - \boxed{U_{l:k+1}} - \boxed{U_k(\theta_k^\pm)} - \boxed{U_{k-1:1}} - \boxed{\nearrow} = \mathbf{a}^p/\mathbf{b}^q \in \{0,1\}^n \tag{63}$$

where the notation indicates $U_{l:m} = U_lU_{l-1}\dots U_{m+1}U_m$, and each gate k carries one of the Ising parameters, $\boldsymbol{\alpha}_k$ or one of the set of measurement unitaries, $\{\boldsymbol{\Gamma}_k, \boldsymbol{\Delta}_k, \boldsymbol{\Sigma}_k\}$, since these also could be trained.

4.4 Stein Discrepancy Training of Ising Born Machine

While the MMD seems to work as an alternative cost function to the KL Divergence to train a Born Machine, [8, 25] and it has many advantages such as being in a simple form, and efficiently and accurately computable using finite samples, it is known to be a relatively weak measure of discrepancy between distributions. This 'weakness' will be discussed further in Section 7. As such, we propose the first alternative cost function for training the IBM, called the *Stein Discrepancy* (SD). This discrepancy has become popular for goodness-of-fit tests, i.e. testing whether a given set of samples come from a particular distribution or not, as opposed to the MMD, which is typically used for kernel two sample tests. The SD was originally proposed in [68], which also provided as a method to compute it using linear programs, since in general the computation involves high dimensional integrals, similarly to the MMD. A simpler form was derived by [69], which introduces a kernelised version, allowing it to be computed in closed form.

The Discrepancy is derived from Stein's Identity which is given by (in the case where the sample space is one-dimensional, $x \in \mathcal{X} \subseteq \mathbb{R}$):

$$\mathbb{E}_q\left[\mathcal{A}_q\phi(x)\right] = \mathbb{E}_q\left[s_q(x)\phi(x) + \nabla_x\phi(x)\right] = 0 \tag{64}$$

where $s_q(x) = \nabla_x \log(q(x))$ is the *Stein Score* function of the distribution q, and $\mathcal{A}_q$ is a so-called *Stein operator* of q. The functions, ϕ, which obey the above Identity, (64), are said to be in the *Stein class* of the distribution q. From Stein's Identity, (64), one can define a discrepancy between two distributions, p, q, by the following optimisation problem, [70]:

$$D_S(p||q) = \sup_{\phi\in\mathcal{F}} \left(\mathbb{E}_p[\mathcal{A}_q\phi] - \mathbb{E}_p[\mathcal{A}_p\phi]\right)^2 \tag{65}$$

Note the second term in (65) is zero by (64) if and only if $p \equiv q$, in which case $D_s(p||q) = 0$ by (64). Exactly as with the MMD, the power of the above discrepancy, (65) will depend on the choice of the function space, $\mathcal{F}$, and by choosing it to be a RKHS, a kernelised form which is computable in closed form can be obtained. Also, this form of (65) is very reminiscent of that of the Integral Probability Metrics, (47).

However, to make the SD applicable to this case, we must make a key alteration. The above, (64) is only defined for smooth probability densities, p, q, which are supported on *continuous* domains, e.g. $\mathbb{R}$, due to the gradient term, ∇_x, in (64). In the case of the Born Machine, the support of the density is discrete (it is composed of binary strings), so the standard gradient is not defined on the sample space. Fortunately, this has been addressed by [70], which adapted the kernelised SD to the discrete domain, by introducing a discrete gradient 'shift' operator. Without loss of generality, from here on we will assume we are dealing with d-dimensional sample vectors, $\mathbf{x} \in \mathcal{X}^d \subseteq \mathbb{R}^d$. First of all, we shall need some definitions introduced by [70]:

Definition 4.1 (Cyclic Permutation). *For a set $\mathcal{X}$ of finite cardinality, a cyclic permutation $\neg : \mathcal{X} \to \mathcal{X}$ is a bijective function such that for some ordering $x^{[1]}, x^{[2]}, \dots, x^{[|\mathcal{X}|]}$ of the elements in $\mathcal{X}$, $\neg x^{[i]} \mapsto x^{[(i+1) \mod |\mathcal{X}|]}, \forall i = 1, 2, \dots, |\mathcal{X}|$*

Definition 4.2 (Partial Difference Operator and Difference Score Function). *Given a cyclic permutation $\neg$ on $\mathcal{X}$, for any vector, $\mathbf{x} = (x_1, \dots, x_d)^T \in \mathcal{X}^d$. For any function $f : \mathcal{X}^d \to \mathbb{R}$, denote the (partial) difference operator as:*

$$\Delta_{x_i} f(\mathbf{x}) := f(\mathbf{x}) - f(\neg_i\mathbf{x}) \forall i = 1, \dots, d \tag{66}$$

with $\Delta f(\mathbf{x}) = (\Delta_{x_1} f(\mathbf{x}), \dots \Delta_{x_d} f(\mathbf{x}))^T$. Define the (difference) score function for a positive *probability mass function, $p(\mathbf{x}) > 0 \forall \mathbf{x}$ as:*

$$\mathbf{s}_p(\mathbf{x}) = \frac{\Delta p(\mathbf{x})}{p(\mathbf{x})} \tag{67}$$

$$(\mathbf{s}_p(\mathbf{x}))_i = \frac{\Delta_{x_i} p(\mathbf{x})}{p(\mathbf{x})} = 1 - \frac{p(\neg_i \mathbf{x})}{p(\mathbf{x})} \tag{68}$$

Furthermore, [70] also defines the inverse permutation by $\lrcorner$: $x^{[i]} \mapsto x^{[(i-1) \mod |\mathcal{X}|]}$, and the *inverse* shift operator by:

$$\Delta^*_{x_i} f(\mathbf{x}) := f(\mathbf{x}) - f(\lrcorner_i \, \mathbf{x}) \forall i = 1, \dots, d \tag{69}$$

However, this is slightly too general for our purposes, as the sample space for a single qubit is binary: $\mathcal{X} = \{0, 1\}$. In this case, the forward and reverse permutations are identical, so $\Delta \equiv \Delta^*$.

The discrete versions of the Stein Identity, and the kernelised Stein discrepancy are also derived in [70], and are in an identical form to the continuous case. There is however, one interesting difference. In the continuous case, the class of functions, $\phi \in \mathcal{F}$, which obey Stein's Identity (i.e. those which are in the Stein class of the operator $\mathcal{A}_p$) are only those for which $\phi(\mathbf{x})p(\mathbf{x})$ vanishes at the boundary of the set, $\partial \mathcal{X}$ [69]. Interestingly, this restriction is not necessary in the discrete case. Due to the definition of the discrete shift operator, it turns out that *all* functions $\phi : \mathcal{X}^d \to \mathbb{R}$ are in the Stein class of the discrete Stein operator. This allows us the freedom to use the quantum kernel, κ_Q, (40), in the SD. This is possible by making a small adaption to the proof of Theorem 2 in [70] so that the discrete Stein Identity also holds for all *complex* valued functions also, due to the complex nature of the feature map in (38).

Theorem 4.3 (Difference Stein's Identity for complex valued functions, adapted from Theorem 2 in [70])**.** *:*
For any function $\phi : \mathcal{X}^d \to \mathbb{C}$, and a probability mass function p on $\mathcal{X}^d$:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim p} [\mathcal{A}_p \phi(\mathbf{x})] = \mathbb{E}_{\mathbf{x}\sim p} [\mathbf{s}_p(\mathbf{x})\phi(\mathbf{x}) - \Delta\phi(\mathbf{x})] = 0 \tag{70}$$

The proof is given in Appendix C. Furthermore, as in [70], the functions can also be extended into complex valued *vector* functions, with an analogue of the above result. In this case, for $\Phi : \mathcal{X}^d \to \mathbb{C}^m$, the discrete Stein Identity is:

$$\mathbb{E}_{\mathbf{x}} [\mathcal{A}_p \Phi(\mathbf{x})] = \mathbb{E}_{\mathbf{x}} \left[\mathbf{s}_p(\mathbf{x})\Phi(\mathbf{x})^T - \Delta\Phi(\mathbf{x})\right] = \mathbf{0} \tag{71}$$

where $\Delta\Phi(\mathbf{x})$ is an $d \times m$ matrix: $(\Delta\Phi)_{ij} = \Delta_{x_i}\phi_j(\mathbf{x})$, i.e. shifting the i^{th} element of the j^{th} function value. Now we can reproduce Theorem (7) from [70]:

Theorem 4.4 (Theorem 7 in [70])**.** *The Discrete Kernelised SD is given by:*

$$\mathcal{L}_{\mathsf{SD}}(P, Q) = D_S(P||Q)^2 = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p} [\kappa_q(\mathbf{x}, \mathbf{y})] \tag{72}$$

where κ_q is the Stein kernel:

$$\kappa_q(\mathbf{x}, \mathbf{y}) = s_q(\mathbf{x})^T \kappa(\mathbf{x}, \mathbf{y}) s_q(\mathbf{y}) - s_q(\mathbf{x})^T \Delta^*_{\mathbf{y}} \kappa(\mathbf{x}, \mathbf{y}) - \Delta^*_{\mathbf{x}} \kappa(\mathbf{x}, \mathbf{y})^T s_q(\mathbf{y}) + tr(\Delta^*_{\mathbf{x},\mathbf{y}} \kappa(\mathbf{x}, \mathbf{y})) \tag{73}$$

As long as the Gram matrix for the kernel, $K_{ij} = \kappa(\mathbf{x}^i, \mathbf{y}^j)$ is positive definite, the kernel, κ, will be *strictly* positive definite. Hence, if a given kernel on a discrete space gives rise to a positive definite Gram matrix, this kernel induces a valid discrepancy measure [70]. Note that as mentioned above in Section 4.3, this criterion is exactly the same as that which makes the MMD a valid discrepancy measure in the discrete case, [58].

While the SD is in a similar form to the MMD, there is one key difference. Namely, the score function, $s_q(\mathbf{y})$ in the worst case, requires computing the probabilities of the distribution we are trying to learn, $q(\mathbf{y})$. If we let $p_\theta(\mathbf{x})$ be the output from the IBM, and again, $\pi(\mathbf{x})$ is the data distribution over binary strings, we have the Stein Cost function for the IBM given by:

$$\mathcal{L}_{\mathsf{SD}}(p_\theta, \pi) = D_S(p_\theta||\pi)^2 = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_\theta}\left[\kappa_\pi(\mathbf{x},\mathbf{y})\right] \tag{74}$$

$$\kappa_\pi(\mathbf{x},\mathbf{y}) = s_\pi(\mathbf{x})^T\kappa(\mathbf{x},\mathbf{y})s_\pi(\mathbf{y}) - s_\pi(\mathbf{x})^T\Delta^*_{\mathbf{y}}\kappa(\mathbf{x},\mathbf{y}) - \Delta^*_{\mathbf{x}}\kappa(\mathbf{x},\mathbf{y})^T s_\pi(\mathbf{y}) + \mathsf{tr}(\Delta^*_{\mathbf{x},\mathbf{y}}\kappa(\mathbf{x},\mathbf{y})) \tag{75}$$

Now, to employ the SD loss function in gradient descent, we will need its gradient with respect to a given parameter, θ_k, which is given by:

$$\frac{\partial\mathcal{L}_{\mathsf{SD}}}{\partial\theta_k} = \underset{\substack{\mathbf{x}\sim p_\theta^-\\ \mathbf{y}\sim p_\theta}}{\mathbb{E}}[\kappa_\pi(\mathbf{x},\mathbf{y})] - \underset{\substack{\mathbf{x}\sim p_\theta^+\\ \mathbf{y}\sim p_\theta}}{\mathbb{E}}[\kappa_\pi(\mathbf{x},\mathbf{y})] + \underset{\substack{\mathbf{x}\sim p_\theta\\ \mathbf{y}\sim p_\theta^-}}{\mathbb{E}}[\kappa_\pi(\mathbf{x},\mathbf{y})] - \underset{\substack{\mathbf{x}\sim p_\theta\\ \mathbf{y}\sim p_\theta^+}}{\mathbb{E}}[\kappa_\pi(\mathbf{x},\mathbf{y})] \tag{76}$$

The gradient derivation is identical to that of the MMD and given in Appendix D. The major difference between (76), and the gradient of the MMD (61) is the different kernel which is used, κ_π vs. simply κ. This clearly makes the Stein Discrepancy more challenging to compute, but it also potentially gives it extra power as a distribution comparison mechanism. This fact is numerically reinforced in Section 6. Also, in contrast to the MMD, the SD in (74) is asymmetric since the π weighted kernel only depends on the distribution π.

The SD is computed between the instantaneous model distribution (p_θ) and the distribution to be learned, (π) by drawing $N = |B|$ samples from the IBM, $\mathbf{x} \sim p_\theta(\mathbf{x})$, and for each pair of samples, $(\mathbf{x}, \mathbf{y})$ the π-weighted kernel, $\kappa_\pi(\mathbf{x},\mathbf{y})$ is computed.

In doing so for a single pair, we must compute the original kernel, κ as a subroutine. If the quantum kernel of (40) is chosen, this requires $O\left(\epsilon^{-2}N^4\right)$[9] [56], measurements of the quantum circuit, as discussed in Section 4.1.

However, adding to the computational burden, we also must compute the following 'shifted' versions of the kernel for each pair of samples, $(\mathbf{x},\mathbf{y})$, in both parameters:

$$\Delta_{\mathbf{x}}\kappa(\mathbf{x},\mathbf{y}) \qquad \Delta_{\mathbf{y}}\kappa(\mathbf{x},\mathbf{y}) \qquad \mathsf{tr}\Delta_{\mathbf{xy}}\kappa(\mathbf{x},\mathbf{y}) \tag{77}$$

which are required in (75). Let K be a polynomial, where it takes $O(K)$ time to compute the entire original kernel matrix:

$$K = O(\epsilon^{-2}N^4) \times T(n) \tag{78}$$

where $T(n)$ is a polynomial in the size of the input n (the number of qubits) describing the running time of the quantum circuit required to compute the kernel for *one* pair of samples, $(\mathbf{x},\mathbf{y})$. We now examine the efficiency of computing the shifted terms in (74) using the quantum kernel of (40).

[9]Notice here that we do not need to compute the kernel with respect to the *data samples*, $\mathcal{X}_{\text{Data}}$, since the dependency on π in (75) is altered with respect to the MMD.

For example:

$$\Delta_{\mathbf{x}}\kappa(\mathbf{x},\mathbf{y}) = (\kappa(\mathbf{x},\mathbf{y}),\dots,\kappa(\mathbf{x},\mathbf{y}))^T - (\kappa(\neg_1\mathbf{x},\mathbf{y}),\dots,\kappa(\neg_n\mathbf{x},\mathbf{y}))^T \tag{79}$$

We assume $\kappa(\mathbf{x},\mathbf{y})$ has been computed for a single pair of samples in time $O(\epsilon^{-2}N^2 \times T(n))$, and we need to compute $\kappa(\neg_i\mathbf{x},\mathbf{y})$ for $i = \{1,\dots,n\}$. Therefore, computing the shifted kernel operator in a single parameter takes $O(\epsilon^{-2}N^2 \times T(n) \times (n+1))$. The same holds for the kernel gradient with respect to the second argument, $\Delta_{\mathbf{y}}\kappa(\mathbf{x},\mathbf{y})$. For $\Delta_{\mathbf{x},\mathbf{y}}\kappa(\mathbf{x},\mathbf{y})$, the process is slightly more involved because:

$$\begin{aligned}\mathsf{tr}\Delta_{\mathbf{x},\mathbf{y}}\kappa(\mathbf{x},\mathbf{y}) &= \mathsf{tr}\Delta_{\mathbf{x}}[\Delta_{\mathbf{y}}\kappa(\mathbf{x},\mathbf{y})] = \mathsf{tr}\Delta_{\mathbf{x}}[\kappa(\mathbf{x},\mathbf{y}) - \kappa(\mathbf{x},\neg\mathbf{y})]\\ &= n\kappa(\mathbf{x},\mathbf{y}) - \sum_{i=1}^{n}\kappa(\mathbf{x},\neg_i\mathbf{y}) - \sum_{i=1}^{n}\kappa(\neg_i\mathbf{x},\mathbf{y}) + \sum_{i=1}^{n}\kappa(\neg_i\mathbf{x},\neg_i\mathbf{y})\end{aligned}$$

Each individual term in the respective sums requires the same complexity, i.e. $O(\epsilon^{-2}N^2 \times T(n))$ so the term $\mathsf{tr}\Delta_{\mathbf{x},\mathbf{y}}\kappa(\mathbf{x},\mathbf{y})$ overall requires $O(\epsilon^{-2}N^2 \times T(n) \times (3n+1))$.

Therefore, each term in κ_π can be computed efficiently using the quantum kernel, (40). That is, with the exception of the score function s_π for the Data distribution. If we are given oracle access to the probabilities, $\pi(\mathbf{y})$, then there is no issue and SD will be computable. Unfortunately, in any practical application this will not be the case. To deal with such a case, in Section B, we give two approaches to approximate the Score function via samples from π. We call these methods the 'Identity', and 'Spectral' methods for convenience. We will only use the Spectral method in training the IBM in the numerical results in Section 6, since the former method does not give an immediate out-of-sample method to compute the score, as discussed further in Section B. Notice that even with the hurdle (difficulty in compute the score), the SD is still more suitable than the KL divergence to train these models, since the latter requires computing the *circuit* probabilities, $p_\theta(\mathbf{x})$, as mentioned in Section 4, which is in general intractable, and so could not be done for *any* dataset. In contrast, the Stein Discrepancy does not require the circuit probabilities, only the data probabilities, which may make it ameanable for QCL using some datasets.

4.5 Sinkhorn Training of Ising Born Machine

The final cost function we shall consider is the so-called *Sinkhorn Divergence* (SH). This is a relatively new way to compare probability distributions, [71–73]. As discussed above, the Stein Discrepancy still is not totally suitable for our purposes. While it seems to be a stronger cost function than the MMD, it may have exponential complexity required to compute the probabilities required in the score function. It seems we have overshot the mark. Now, can we find some middle ground? In other words, does there exist a cost function we can choose, which is still 'stronger' than the MMD, but easier to compute than the SD.

As motivated by our introduction of the SD, a stronger distance would presumably provide better results than the MMD for generative modelling. Firstly, we may consider the Wasserstein or Kantorovich Metric, (52). The Wasserstein Metric happens to be a special case of so-called *optimal transport*. The solution of the 'optimal transport' problem gives the optimal way to move, or transport, probability mass from one distribution to another, and hence gives a means of determining distribution similarity. This problem of optimal transport has been widely studied, and has a rich history, so we refer to [74] for a thorough treatment.

The Optimal Transport Distance is given by:

$$\mathsf{OT}^c(p, q) = \min_{U\in\mathcal{U}(P,Q)} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}\times\mathcal{Y}} c(\mathbf{x}, \mathbf{y})U(\mathbf{x}, \mathbf{y}) \tag{80}$$

where P, Q are the marginal distributions of U, i.e. $\mathcal{U}(P, Q)$ is the space of joint distributions over $\mathcal{X} \times \mathcal{Y}$ such that $\sum_{\mathbf{x}} U(\mathbf{x}, \mathbf{y}) = Q(\mathbf{y}), \sum_{\mathbf{y}} U(\mathbf{x}, \mathbf{y}) = P(\mathbf{x})$, in the discrete case. $c(\mathbf{x}, \mathbf{y})$ is the '*cost*' of transporting an individual 'point', $\mathbf{x}$, to another point $\mathbf{y}$. It somewhat plays a similar role to the kernel function in the MMD. If we take the optimal transport 'cost', to be a metric on the sample space, $\mathcal{X}\times\mathcal{Y}$, i.e. $c(\mathbf{x}, \mathbf{y}) = d(\mathbf{x}, \mathbf{y})$ we get the Wasserstein metric, which turns out to be equivalent to the Kantorovich Metric, revealing the connection to the IPMs discussed in Section 4.2:

$$W^d(P, Q) = \min_{U\in\mathcal{U}(P,Q)} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}^2} d(\mathbf{x}, \mathbf{y})U(\mathbf{x}, \mathbf{y}) \tag{81}$$

The Wasserstein Metric has become very popular in classical ML literature, for example in the definition of an Generative Adversarial Network (WGAN) [75][10].

However, it does suffer from a severe drawback; In d dimensions, its sample complexity scales as $O(1/N^{1/d})$, and furthermore it is difficult to compute. As a remedy to this, *regularisation* was introduced in order to smooth the problem, and ease computation, which is a standard technique in ML to prevent overfitting.

Now, as noted in [78], (one) regularised version of optimal transport introduces an entropy term, in the form of the KL Divergence as follows:

$$\mathsf{OT}^c_\epsilon(P, Q) = \min_{U\in\mathcal{U}(P,Q)} \left(\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}\times\mathcal{Y}} c(\mathbf{x}, \mathbf{y})U(\mathbf{x}, \mathbf{y}) + \epsilon\,\mathsf{KL}(U|P\otimes Q) \right) \tag{82}$$

$$\mathsf{KL}(U|P \otimes Q) \equiv \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}\times\mathcal{Y}} U(\mathbf{x}, \mathbf{y}) \log \left(\frac{U(\mathbf{x}, \mathbf{y})}{(P \otimes Q)(\mathbf{x}, \mathbf{y})} \right) \tag{83}$$

It turns out that the regularised version allows a cost function to be defined, which interpolates between the Wasserstein, and the MMD. As such, it allows one to take advantage of the small sample complexity, and therefore ease of computability of the MMD, but the desirable properties of the Wasserstein Distance.

The relative entropy term serves to determine how distant the coupling distribution, π, is from a product distribution $P \otimes Q$, and effectively smooths the problem, such that it becomes more efficiently solvable. Based on this, [72, 73] define the *Sinkhorn Divergence* which we can appropriate for the IBM:

$$\mathcal{L}^\epsilon_{\mathsf{SH}}(p_\theta, \pi) = \mathsf{OT}^c_\epsilon(p_\theta, \pi) - \frac{1}{2}\mathsf{OT}^c_\epsilon(p_\theta, p_\theta) - \frac{1}{2}\mathsf{OT}^c_\epsilon(\pi, \pi) \tag{84}$$

As shown by [73], as $\epsilon \to 0$, (84) becomes the unregularised Optimal Transport distance: $\mathcal{L}^0_{\mathsf{SH}}(p_\theta, \pi) \underset{\epsilon\to 0}{\to} \mathsf{OT}^c(p_\theta, \pi)$, and as $\epsilon \to \infty$, (84) becomes the MMD with a kernel given by the negative of the Sinkhorn cost, $\kappa(\mathbf{x}, \mathbf{y}) = -c(\mathbf{x}, \mathbf{y})$: $\mathcal{L}^\epsilon_{\mathsf{SH}}(p_\theta, \pi) \underset{\epsilon\to\infty}{\to} \mathsf{MMD}_{-c}(p, \pi)$.

[10]GANs are alternative generative models, which train *adversarially* via a competition between two neural networks, one *discriminator*, and one *generator*. Quantum models of GANs have also been defined, for example, [26, 76] and an experimental implementation [77].

The extra terms in (84) relative to (80) ensure the Sinkhorn Divergence is unbiased, since $\mathsf{OT}_\epsilon^c(P,P) \neq 0$ in general.

Now, similarly to the previous two cost functions, we can derive gradients of the Sinkhorn Divergence, with respect to the given parameter, θ_k. According to [73], each term in (84) can be written as follows:

$$\begin{aligned}\mathsf{OT}_\epsilon^c(p_\theta,\pi) &= \langle p_\theta, f\rangle + \langle \pi, g\rangle \\ &= \sum_{\mathbf{x}} p_\theta(\mathbf{x})f(\mathbf{x}) + \pi(\mathbf{x})g(\mathbf{x})\end{aligned} \tag{85}$$

f and g are the so-called optimal Sinkhorn potentials, arising from a primal-dual formulation of optimal transport. These are computed using the Sinkhorn Algorithm, which gives the divergence its name, [79]. These vectors will be initialised at $f^0(\mathbf{x}) = 0 = g^0(\mathbf{x})$, and iterated in tandem according to (88) for a fixed number of 'Sinkhorn iterations' until convergence. The number of iterations required will depend on the value of ϵ, and the specifics of the problem. Typically, smaller values of epsilon will require more iterations, since this is bringing the problem closer to unregularised optimal transport, which is more challenging to compute. For further discussions on regularised optimal transport and its dual formulation, see [73, 80]. Now, following [73], the SH can be written as:

$$\mathcal{L}_{\mathsf{SH}}^\epsilon(p_\theta,\pi) = \sum_{\mathbf{x}} \left[p_\theta(\mathbf{x})\left(f(\mathbf{x}) - s(\mathbf{x})\right) + \pi(\mathbf{x})\left(g(\mathbf{x}) - t(\mathbf{x})\right)\right] \tag{86}$$

s, t are the '*autocorrelation*' dual potentials, arising from the terms $\mathsf{OT}_\epsilon^c(p_\theta,p_\theta)$, $\mathsf{OT}_\epsilon^c(\pi,\pi)$ in (84).

The reader should note that in (86), the dependence on the data distribution, π, is hidden in the dual vector, f. Following [73], we can see how by discretising the situation based on N, M samples from p_θ, π respectively; $\hat{\mathbf{x}} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\} \sim p_\theta(\mathbf{x}), \hat{\mathbf{y}} = \{\mathbf{y}^1, \dots, \mathbf{y}^M\} \sim \pi(\mathbf{y})$. With this, the optimal dual vectors, f, g are given by:

$$f^{l+1}(\mathbf{x}^i) = -\epsilon \mathrm{LSE}_{k=1}^M\left(\log\left(\pi(\mathbf{y}^k) + \frac{1}{\epsilon}g^l(\mathbf{y}^k) - \frac{1}{\epsilon}C_{ik}(\mathbf{x}^i,\mathbf{y}^k)\right)\right) \tag{87}$$

$$g^{l+1}(\mathbf{y}^j) = -\epsilon \mathrm{LSE}_{k=1}^N\left(\log\left(p_\theta(\mathbf{x}^k) + \frac{1}{\epsilon}f^l(\mathbf{x}^k) - \frac{1}{\epsilon}C_{kj}(\mathbf{x}^k,\mathbf{y}^j)\right)\right) \tag{88}$$

$C(\mathbf{x},\mathbf{y})$ is the so-called optimal transport *cost matrix* derived from the cost function applied to all samples, $C_{ij}(\mathbf{x}^i,\mathbf{y}^j) = c(\mathbf{x}^i,\mathbf{y}^j)$ and $\mathrm{LSE}_{k=1}^N(\mathbf{V}_k) = \log\sum_{k=1}^N \exp(\mathbf{V}_k)$ is a log-sum-exp reduction for a vector $\mathbf{V}$, used to give a smooth approximation to the true dual potentials.

The autocorrelation potential, s, is given by:

$$s(\mathbf{x}^i) = -\epsilon \mathrm{LSE}_{k=1}^N\left(\log\left(p_\theta(\mathbf{x}^k) + \frac{1}{\epsilon}s(\mathbf{x}^k) - \frac{1}{\epsilon}C(\mathbf{x}^i,\mathbf{x}^k)\right)\right) \tag{89}$$

$t(\mathbf{y}^i)$ can be derived similarly by replacing $p_\theta \to \pi$ in (89) above. However, the autocorrelation dual can be found using a well-conditioned fixed point update [73], and convergence to the optimal potentials can be observed with much fewer Sinkhorn iterations, l:

$$s(\mathbf{x}^i) \leftarrow \frac{1}{2}\left[s(\mathbf{x}^i) - \epsilon \mathrm{LSE}_{k=1}^N\left(\log\left(p_\theta(\mathbf{x}^\mathbf{k}) + \frac{1}{\epsilon}s(\mathbf{x}^k) - \frac{1}{\epsilon}C(\mathbf{x}^i,\mathbf{x}^k)\right)\right)\right] \tag{90}$$

Of course, now that we have discussed how to compute the Sinkhorn Divergence, it is time to examine its gradient with respect to the circuit parameters, as usual. The gradient of $\mathcal{L}^{\epsilon}_{\mathsf{SH}}$ with respect to a single probability of the *observed* samples, $p_\theta(\mathbf{x}^i)$ is given by [73]:

$$\frac{\partial \mathcal{L}^{\epsilon}_{\mathsf{SH}}(p_\theta, \pi)}{\partial p_\theta(\mathbf{x}^i)} = f(\mathbf{x}^i) - s(\mathbf{x}^i) \tag{91}$$

However, this only applies to the samples which have been *used* to compute f, s in the first place. If one encounters a sample from $p_{\theta_k^{\pm}}$ (which we shall in the gradient), $\mathbf{x}^s$, which one has not seen in the original samples from p_θ, one has no value for the corresponding vectors at this point $f(\mathbf{x}^s), s(\mathbf{x}^s)$. Fortunately, as shown in [73], the gradient does extend smoothly to this point (and all points in the sample space) and in general is given by:

$$\frac{\partial \mathcal{L}^{\epsilon}_{\mathsf{SH}}(p_\theta, \pi)}{\partial p_\theta(\mathbf{x})} = \varphi(\mathbf{x}) \tag{92}$$

$$\begin{aligned} \varphi(\mathbf{x}) = -\epsilon \mathrm{LSE}_{k=1}^{M}\left(\log\left(\pi(\mathbf{y}^k) + \frac{1}{\epsilon}g^0(\mathbf{y}^k) - \frac{1}{\epsilon}C(\mathbf{x}, \mathbf{y}^k)\right)\right) \\ + \epsilon \mathrm{LSE}_{k=1}^{N}\left(\log\left(p_\theta(\mathbf{y}^{\mathbf{k}}) + \frac{1}{\epsilon}s^0(\mathbf{x}^k) - \frac{1}{\epsilon}C(\mathbf{x}, \mathbf{x}^k)\right)\right) \end{aligned} \tag{93}$$

Where $g^{(0)}, s^{(0)}$, are the optimal vectors which solve the original optimal transport problem, (88, 89) at convergence, given the samples, $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ from p_θ, π respectively. Given this, the gradient of the Sinkhorn Divergence with respect to the parameters, θ_k is given by:

$$\begin{aligned} \frac{\partial \mathcal{L}^{\epsilon}_{\mathsf{SH}}(p_\theta, \pi)}{\partial \theta_k} &= \sum_{\mathbf{x}} \frac{\partial \mathcal{L}^{\epsilon}_{\mathsf{SH}}(p_\theta, \pi)}{\partial p_\theta(\mathbf{x})} \frac{\partial p_\theta(\mathbf{x})}{\partial \theta_k} \\ &= \sum_{\mathbf{x}} \varphi(\mathbf{x}) \left(p_{\theta_k^-}(\mathbf{x}) - p_{\theta_k^+}(\mathbf{x})\right) \\ &= \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\theta_k^-}}[\varphi(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\theta_k^+}}[\varphi(\mathbf{x})] \end{aligned} \tag{94}$$

Therefore, one can compute the gradient by drawing samples from the distributions, $\hat{\mathbf{x}} \sim p_{\theta_k^{\pm}}$, and computing the vector $\varphi(\mathbf{x})$, for each sample, $\mathbf{x} \in \hat{\mathbf{x}}$, using the vectors, $g^{(0)}, s^{(0)}$ already computed during the evaluation of SH at each epoch.

4.5.1 Sample Complexity of Sinkhorn Divergence

It is of critical interest to our purpose to investigate the sample complexity of the Sinkhorn Divergence, and in doing so mention the sample complexity of Wasserstein, and the MMD. This will be necessary in our attempts to use them for a quantum model. We shall see that due to the results of [81], we will be able to define a somewhat 'optimal' cost function for the Born Machine, in order to squeeze as many favourable metric properties from the Wasserstein Metric, but still leave the resulting model efficiently computable. This will be determined by the parameter ϵ in (84).

As mentioned above, the sample complexity of the MMD scales as $O(1/\sqrt{N})$, and it is known that the Wasserstein Distance scales as $O(1/N^d)$ [82] for a distribution supported on a subset of $\mathbb{R}^d$. The former indicates that the MMD can be computed to an accuracy ϵ with $O(\epsilon^{-2})$ samples regardless of the underlying space, and the latter means the Wasserstein requires $O(\epsilon^{-d})$ which is exponential in the size of the space. Since we are dealing with binary vectors of length n, the support of our distributions (the IBM and the data distribution) will be $\mathcal{X} = \{0,1\}^n \subseteq \mathbb{R}^n$[11]. These two cases correspond to the extreme regularisation values of $\epsilon \to \infty$ and $\epsilon \to 0$ respectively. However, the motivation for using the Sinkhorn Divergence is to leverage the favourable sample complexity of the MMD, with the stronger Wasserstein Distance. This ϵ-regularisation hyperparameter allows us to choose a cost function which optimally suits our needs. By the results of [73], we can be guaranteed that no matter which value is chosen, the SH will be suitable as a cost function to train the IBM, and in fact any generative model. This is because it is zero for any distributions which are identical, and strictly positive otherwise:

$$\mathcal{L}^{\epsilon}_{\mathsf{SH}}(P,Q) = 0 \iff P = Q \tag{95}$$

$$\mathcal{L}^{\epsilon}_{\mathsf{SH}}(P,Q) \geq \mathcal{L}^{\epsilon}_{\mathsf{SH}}(P,P) = 0 \tag{96}$$

It also metrizes convergence in law, effectively meaning it can be estimated using N samples, and will converge to its true value in the limit of large samples:

$$\mathcal{L}^{\epsilon}_{\mathsf{SH}}(\hat{P}_N,P) \to 0 \iff \hat{P}_N \rightharpoonup P \tag{97}$$

Now, from [81], we have the following two results. The first is the mean difference between the true Sinkhorn, $\mathcal{L}^{\epsilon}_{\mathsf{SH}}(P,Q)$ and its estimator derived from the empirical distributions, $\mathcal{L}^{\epsilon}_{\mathsf{SH}}(\hat{P}_N,\hat{Q}_N)$ is given by:

Theorem 4.5 (Theorem 3 from [81]). *Consider the Sinkhorn Divergence between two distributions, P, and Q on two bounded subsets, $\mathcal{X},\mathcal{Y}$ of $\mathbb{R}^d$, with a C^∞, $L-$Lipshitz cost c. One has:*

$$\mathbb{E}|\mathcal{L}^{\epsilon}_{\mathsf{SH}}(P,Q) - \mathcal{L}^{\epsilon}_{\mathsf{SH}}(\hat{P}_N,\hat{Q}_N)| = O\left(\frac{e^{\frac{\kappa}{\epsilon}}}{\sqrt{N}}\left(1+\frac{1}{\epsilon^{\lfloor k/2\rfloor}}\right)\right) \tag{98}$$

where $\kappa = 2L|\mathcal{X}| + ||c||_\infty$ and constants only depend on $|\mathcal{X}|, |\mathcal{Y}|, c$ and $||c^l||_\infty$ for $l = 0, \dots, \lfloor d/2 \rfloor$.

The second is the following concentration result:

Corollary 4.5.1 (Corollary 1 from [81]). *With probability at least $1-\delta$,*

$$|\mathcal{L}^{\epsilon}_{\mathsf{SH}}(P,Q) - \mathcal{L}^{\epsilon}_{\mathsf{SH}}(\hat{P}_N,\hat{Q}_N)| \leq 6B\frac{\lambda K}{\sqrt{N}} + C\sqrt{\frac{2\log\frac{1}{\delta}}{N}} \tag{99}$$

where $\kappa = 2L|\mathcal{X}| + ||c||_\infty$, $C = \kappa + \epsilon e^{\frac{\kappa}{\epsilon}}$, $B \leq 1 + e^{\left(2\frac{L|\mathcal{X}|+||c||_\infty}{\epsilon}\right)}$, $\lambda = O(1+\frac{1}{\epsilon^{\lfloor d/2\rfloor}}))$ and $K = \max_{\mathbf{x}\in\mathcal{X}} \kappa_S(\mathbf{x},\mathbf{x})$.

κ_S is the Matern or the Sobolev kernel, associated to the Sobolev Space, $\mathbf{H}^s(\mathbb{R}^d)$, which is a RKHS for $s > d/2$, but we will not go into further detail here.

The more exact expression for (4.5) is given by:

$$\mathbb{E}|\mathcal{L}^{\epsilon}_{\mathsf{SH}} - \hat{\mathcal{L}}^{\epsilon}_{\mathsf{SH}}| \leq 3\frac{B\lambda\sqrt{K}}{\sqrt{N}} \tag{100}$$

$$\leq \frac{3K}{\sqrt{N}}\left(1 + e^{\left(2\frac{L|\mathcal{X}|+||c||_\infty}{\epsilon}\right)}\right) O\left(1+\frac{1}{\epsilon^{\lfloor d/2\rfloor}}\right) \tag{101}$$

[11]A general quantum distribution can be supported on this entire space.

Now, for our particular case, we wish to choose the Hamming distance as a metric on the Hamming hypercube. However, due to the smoothness requirement of the above theorems, $c \in C^\infty$, this would not hold in the discrete case we are dealing with. However, we can take a broader view to simply use the ℓ^2, or Euclidean distance, and embed the Hamming hypercube in a larger space. This is possible because the Hamming distance is exactly the squared ℓ^2 cost[12], but restricted to binary vectors:

$$||\mathbf{x} - \mathbf{y}||_2^2 = \sum_{i=1}^{d} (x_i - y_i)^2 \tag{102}$$

$$= \sum_{i=1}^{d} (x_i - y_i) = d_H(\mathbf{x}, \mathbf{y}) \forall x_i, y_i \in \{0, 1\} \tag{103}$$

In this scenario, formally, we are dealing with the general hypercube in $\mathbb{R}^d$, but where the probability masses are strictly concentrated on the vertices of the hypercube. Now, we can compute directly some of the constants in the above, Theorem (4.5) and Corollary (4.5.1). Taking $\mathcal{X}$ to be the unit hypercube in $\mathbb{R}^d$, and taking the Sinkhorn cost to be the ℓ^2 cost, which is Lipschitz continuous, we can compute the following:

$$|\mathcal{X}| = \sup_{\mathbf{x},\mathbf{y} \in \mathcal{X}} ||\mathbf{x} - \mathbf{y}||_2 = \sqrt{d}, \qquad ||c(\mathbf{x}, \mathbf{y})||_\infty = \sup\{|c(\mathbf{x}, \mathbf{y})| : (\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}\} = d \tag{104}$$

A rough upper bound for the Lipschitz constant, L, can be obtained as follows. For a function, $f : \mathcal{A} \to \mathcal{B}$, the Lipschitz constant is the smallest value L such that:

$$d_\mathcal{B}(f(\mathbf{x}), f(\mathbf{y})) \leq L d_\mathcal{A}(\mathbf{x}, \mathbf{y}) \tag{105}$$

If we take $d_\mathcal{A}$ to be the sum metric on the product space, $\mathcal{X} \times \mathcal{Y}$, and f to be the ℓ_2 norm squared, we get:

$$|(c(\mathbf{x}_1, \mathbf{y}_1) - c(\mathbf{x}_2, \mathbf{y}_2))| \leq L \left[d_\mathcal{X}(\mathbf{x}_1, \mathbf{x}_2) + d_\mathcal{Y}(\mathbf{y}_1, \mathbf{y}_2)\right] \tag{106}$$

For two points, $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2) \in \mathcal{X} \times \mathcal{Y}$, and the cost $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}, c = || \cdot ||_2^2$.

Now,

$$\frac{\left|\sum_i (\mathbf{x}_1^i - \mathbf{y}_1^i)^2 - \sum_i (\mathbf{x}_2^i - \mathbf{y}_2^i)^2\right|}{\left[\sqrt{\sum_i (\mathbf{x}_1^i - \mathbf{x}_2^i)^2} + \sqrt{\sum_i (\mathbf{y}_1^i - \mathbf{y}_2^i)^2}\right]} \leq L \tag{107}$$

We want to find an upper bound for the left hand side of (107), assuming that $\mathbf{x}_1 \neq \mathbf{x}_2$ and $\mathbf{y}_1 \neq \mathbf{y}_2$[13]. Now, applying the trick that we have embedded the Hamming hypercube in the general hypercube, we can assume $\mathbf{x}_{1,2}^i, \mathbf{y}_{1,2}^i \in \{0, 1\} \forall i$. To derive such a bound, we can bound both the numerator and the denominator of the LHS of (107) independently. We find the denominator is as small as possible, when only one element of $\mathbf{x}_{1,2}$ or $\mathbf{y}_{1,2}$ is equal to one, and all the rest zero. The numerator is as large as possible when one of $\mathbf{x}_{1,2}$ or $\mathbf{y}_{1,2}$ is the all-one vector. In this case, the LHS is upper bounded by d, so we can choose $L = d$, which is a constant for a fixed d.

[12]The squared ℓ^2 distance is in fact *not* a metric, however it is suitable to use as a more general cost function in the Sinkhorn Divergence, since it is Lipshitz and smooth.

[13]In this case, both numerator and denominator are zero, so any non-zero L will satisfy (106).

So, (103) becomes:

$$\mathbb{E}|\mathcal{L}^{\epsilon}_{\mathsf{SH}} - \hat{\mathcal{L}}^{\epsilon}_{\mathsf{SH}}| \leq \frac{3K}{\sqrt{M}}\left(1 + e^{\left(2\frac{d\sqrt{d}+d}{\epsilon}\right)}\right) O\left(1 + \frac{1}{\epsilon^{\lfloor d/2 \rfloor}}\right) \tag{108}$$

The constants in $O\left(1 + \frac{1}{\epsilon^{\lfloor d/2 \rfloor}}\right)$, depend on $|\mathcal{X}|, |\mathcal{Y}|, d,$ and $||c^{(k)}||_\infty$ which are at most linear in d. Similarly the concentration bound is:

$$|\mathcal{L}^{\epsilon}_{\mathsf{SH}} - \hat{\mathcal{L}}^{\epsilon}_{\mathsf{SH}}| \leq 6B\frac{\lambda K}{\sqrt{N}} + C\sqrt{\frac{2\log\frac{1}{\delta}}{N}} \tag{109}$$

$$\leq \frac{6K}{\sqrt{N}}\left(1 + e^{\left(2\frac{L|\mathcal{X}|+||c||_\infty}{\epsilon}\right)}\right) O\left(1 + \frac{1}{\epsilon^{\lfloor d/2 \rfloor}}\right) + \kappa\sqrt{\frac{2\log\frac{1}{\delta}}{N}} + \epsilon e^{\frac{\kappa}{\epsilon}}\sqrt{\frac{2\log\frac{1}{\delta}}{N}} \tag{110}$$

$$= \frac{1}{\sqrt{N}}\left[6\left(1 + e^{\left(\frac{O(d^{3/2})}{\epsilon}\right)}\right) O\left(1 + \frac{1}{\epsilon^{\lfloor d/2 \rfloor}}\right) + O(d^{3/2})\sqrt{2\log\frac{1}{\delta}} + \epsilon e^{\frac{O(d^{3/2})}{\epsilon}}\sqrt{2\log\frac{1}{\delta}}\right] \tag{111}$$

$$= O\left(\frac{1}{\sqrt{N}}\left[\left(1 + e^{\left(\frac{d^{3/2}}{\epsilon}\right)}\right)\left(1 + \frac{1}{\epsilon^{\lfloor d/2 \rfloor}}\right) + d^{3/2}\sqrt{2\log\frac{1}{\delta}} + \epsilon e^{\frac{d^{3/2}}{\epsilon}}\sqrt{2\log\frac{1}{\delta}}\right]\right) \tag{112}$$

Since, $\kappa = 2d\sqrt{d} + d = O(d^{3/2})$. Now, clearly, due to the asymptotic behaviour of the Sinkhorn Divergence, we would like to choose ϵ sufficiently large in order to remove as much dependence on the dimension, d, as possible. This is because, in our case, the dimension of the space is equivalent to the number of qubits, $d = n$, and hence to derive a favourable sample complexity, we would hope for the dependence on d to be polynomial in the number of qubits. By examining, (112), we could see that a good choice might be $\epsilon = O(d^{3/2}) = O(n^{3/2})$. In this case, we get:

$$|\mathcal{L}^{\epsilon}_{\mathsf{SH}} - \hat{\mathcal{L}}^{\epsilon}_{\mathsf{SH}}| \leq O\left(\frac{1}{\sqrt{M}}\left[\left(1 + \frac{1}{d^{\lfloor d/2 \rfloor}}\right) + d\sqrt{2\log\frac{1}{\delta}}\right]\right) \tag{113}$$

with probability $1 - \delta$. It is likely in practice however, that a much smaller value of ϵ could be chosen, without blowing up the sample complexity. This is evidenced by numerical results in [72, 73, 81].

5 Hardness of Classically Simulating the Ising Born Machine

In this section, we revisit the hardness results from Section 3.2, which dealt only with the underlying circuit class of the Born Machine itself for a variety of parameters. We are now in a position to claim that all of the individual ingredients in the training algorithm should be equally as hard as the original circuit. Of course, by its very nature, the algorithm is a hybrid one, hence classical optimisation is clearly crucial to facilitate training. Since the computation of the cost functions rely on computing expectation values of the outputs of quantum circuits, we might not expect a provable advantage, since a classical algorithm could likely compute the probabilities to a comparable accuracy as a quantum circuit to do so, and then compute an estimator for the cost functions, given the # P-Hardness of the probabilities. However, we can leverage the sampling hardness results to claim the individual ingredients should not be possible by purely classical means efficiently, including the updated circuits found by gradient descent, and the parameter shifted gradient circuits themselves, (63).

Algorithm 1 $\mathsf{IBM}(\theta)$ Training Algorithm

```
 1: Initialise all parameters of Ising Born Machine (IBM), $\theta^{(0)} = \{\{J_{ij}, b_k\}, \mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}\}$ at random, within the regime proven to be hard or a subset of the parameters as constrained by hardware. Choose cost function, $B \in \{\mathsf{MMD}, \mathsf{SD}, \mathsf{SH}\}$.
 2: Inputs: Data samples, $\{\mathbf{y}^j\}_{j=1}^M \sim \pi(\mathbf{y})$
 3: for all d in epochs do
 4:     for i in N repetitions do
 5:         Prepare $n$ qubits in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.
 6:         for all qubits do
 7:             Apply $CZ(4J_{ij}),\ Z(-2J_{ij}), Z(-2b_k)$
 8:         end for
 9:         Apply $U_f(\mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma})$
10:         for $l$ in qubits do
11:             Measure qubit $l$ in the comp. basis, $M_{\mathbf{x}_l^i} = |\mathbf{x}_l^i\rangle \langle \mathbf{x}_l^i|$, with outcome $\mathbf{x}_l^i$.
12:         end for
13:     end for
14:     Compute cost, $\mathcal{L}_B\left(\{\mathbf{x}^i\}, \{\mathbf{y}^j\}\right)$, using $K_{ij}$ for $B = \mathsf{MMD}\,/\,\mathsf{SD}$ or $C_{ij}$ for $B = \mathsf{SH}$.
15:     for $\theta_k^{(d)}$ in $\{\theta^{(d)}\}$ do
16:         for $p = \{1, \dots, P\}, q \in \{1, \dots, Q\}$ do
17:             Generate samples from the shifted distributions, $\mathbf{a}^p \sim p_{\theta_k}^-, \mathbf{b}^q \sim p_{\theta_k}^+$.
18:         end for
19:         Compute gradient of $\mathcal{L}_B$ using all samples, $\frac{\partial \mathcal{L}_B}{\partial \theta_k^{(d)}}\left(\{\mathbf{x}^i, \mathbf{y}^j, \mathbf{a}^p, \mathbf{b}^q\}\right)$
20:         $\theta_k^{(d+1)} \leftarrow \theta_k^{(d)} - \eta \frac{\partial \mathcal{L}_B}{\partial \theta_k^{(d)}}$
21:     end for
22: end for
```

Collecting the results required in Theorem (3.1) is necessary, since we want to make the strongest arguments for the intractability of classically simulating the Ising Born Machine. If we were not careful, the parameter updates could lead us into a regime which was classically simulable. Now, the initialisation of the parameters, $\theta^{(0)}$, will lead to a circuit class which is hard to sample from classically in the worst case, up to variation distance error, if U_f is chosen such that the IBM is exactly an IQP circuit, otherwise it will only be (provably) hard up to multiplicative error. These samples can then be used to compute the cost function, $\mathcal{L}_B$. Now computing the gradients, $\frac{\partial \mathcal{L}}{\partial \theta_k}$, for all choices of cost function, $B = \{\mathsf{MMD}, \mathsf{SD}, \mathsf{SH}\}$ requires running parameter shifted circuits, $p_{\theta_k}^-, p_{\theta_k}^+$. If the original circuit is hard to sample from, these shifted circuits will *also* be hard to sample from. This is due to the fact that they only differ by a single parameter shift, which can be reabsorbed into the original parameters such that the resulting circuit also resides in the hard circuit classes. As an example, using an IQP circuit, if the parameter to be updated is J_{ij}, then that parameter should have been initialised according to Theorem (A.2):

$$J_{ij} = \begin{cases} \frac{(2l+1)\pi}{8d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{114}$$

Therefore:

$$J_{ij} \pm \frac{\pi}{2} = \begin{cases} \frac{(2(l\pm 4d)+1)\pi}{8d} & \text{for integers, } d, l \\ 2\left(\frac{2\nu\pm 1}{4}\right)\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{115}$$

A relabelling of $l \pm 4d \to k, \frac{1}{4}(2\nu \pm 1) \to \mu$, where μ is irrational if ν is, and k is still an integer, gives that the parameter shifted circuits should have exactly the same structure as (114), and hence should be just as hard.

The above argument illustrates how the circuits required to compute the gradient for a given parameter, J_{ij}, should be as hard as the original circuit. However, the training algorithm requires update the parameters of the IBM circuit, over a number of epochs, in order to provide better fits to the data. As detailed above, this is done using the following update rule:

$$\theta_l^{(d+1)} \leftarrow \theta_l^{(d)} - \eta \frac{\partial \mathcal{L}_B}{\partial \theta_l^{(d)}} \tag{116}$$

We can ensure that the system remains in a class which has worst case hardness 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, $\theta^{(0)} = 2\pi\nu$, where ν is irrational, then an update will be of the form $\theta^{(d+1)} = \theta^{(d)} + \mu$. If we choose $\mu = 2\pi\alpha$, then the new parameters at epoch, $d+1$, will be $\theta^{(d+1)} = 2\pi\beta, \beta = \nu + \alpha$[14] Since ν was irrational originally, β will also be irrational, and therefore the new configuration should be hard to simulate classically. Although, clearly we cannot allow updates like $\alpha = -\nu(+\delta)$, where δ is rational since this will result in the parameter going to $0(\delta)$ and could make the model classically simulatable. As an example of this, choose the following circuit parameters for the IBM$(\{J_{ij}, b_k\}, -\boldsymbol{\Gamma}_k, \Delta_k = 0, \Sigma_k = 0)$, which is a $p = 1$ QAOA circuit, and then set $\Gamma_k = 0$. 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, 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. Another example is if we were to allow the entangling parameters, $J_{ij} \to 0$. This would lead to a circuit composed only of single qubit gates, which again is not classically hard.

There is, of course, a caveat to this argument. The hardness results in Theorem (3.1) are only valid in the *worst case*. This means that there exists *some* instance of the problem generated by the set of parameters which cannot be classically simulated efficiently. More specifically, while with each instance of 'hard' parameter values, we may end up in a parameter landscape which admits is a worst case hardness result, there is obviously no

[14] α will be the learning rate multiplied by the gradient, $\eta\frac{\partial \mathcal{L}}{\partial \theta_l^{(d)}}$.

guarantee that the particular instance *implemented* in the IBM is in fact *the* hard one, and we do not claim to have found such a thing. Furthermore, there is also no guarantee that we go from the 'hard' circuit generated from one set of parameters, to the hard circuit using the next set of parameters (given by the gradient update). Nevertheless, we believe this is a step forward in the methodology to design Quantum Machine Learning algorithms which demonstrate some quantum supremacy over their classical counterparts.

Further adding to this hardness argument is the required computation of the intermediate quantum-hard kernel (40), should that be the one which is chosen. This computation must be done *for all* pairs of samples required in the loss, $\mathcal{L}_B$ and its gradient, and as such, increases the conjectured hardness of classically simulating the training algorithm. This is due to the argument of [56], which conjectures that this kernel should be hard to compute for any classical algorithm, up to additive error.

6 Numerical Results

In this section, we present numerical results demonstrating the ability of Algorithm 1 to learn a particular data distribution. We implement the algorithm with the three cost functions of Section 4, namely the MMD, Stein Discrepancy and Sinkhorn Divergence. The results are produced using Rigetti's Forest platform [83]. Both the simulator, or Quantum Virtual Machine (QVM), and sub-lattices on the Quantum Processing Unit (QPU), the Rigetti `Aspen` 16Q chip, are used. This gives us access to perfect and realistic implementations respectively. Our implementation can be found at [84], which uses some code from [85] to compute the Sinkhorn Divergence and its gradient.

The data used is a collection of samples from the distribution of (117). This is the same simple toy example which was used to train a Quantum Boltzmann Machine, [86], and the Quantum Approximate Boltzmann Machine (QABoM) [39].

$$\pi(\mathbf{y}) = \frac{1}{T}\sum_{k=1}^{T} p^{N-d_H^{(k,\mathbf{y})}}(1-p)^{d_H^{(k,\mathbf{y})}} \tag{117}$$

The use of this distribution is in contrast to other recent works on the Born Machine [6, 8, 10, 25] which use the more canonical Bars and Stripes dataset [87] to train.

To generate this data, T 'Bernoulli' hidden modes are randomly chosen. Each mode (k) is randomly chosen to be a binary string, $\mathbf{s}^k = [s_1^k, \dots, s_n^k]$, and a given sample, $\mathbf{y}$, is produced with a probability which depends on the Hamming distance, $d_H^{(k,\mathbf{y})}$, between each mode string, k, and that sample. In all of the following, the Adam [88] optimiser was applied, using the hyperparameters suggested in that paper, i.e. $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1\times 10^{-8}$, and initial learning rate, η_{init}. This was chosen since it was found to be more robust to sampling noise [8]. Also, in all of the simulated results below, we assume a fully connected QVM for simplicity, as illustrated in Figure 6a and Figure 7a.

We use the total variation distance, TV, to compare training methods. We chose TV as an objective measure, as opposed to, say the KL divergence as chosen in [52], for a couple of reasons. Firstly, the classical hardness results mentioned in this work all aim to prove Quantum Supremacy with respect to this measure. Secondly, TV plays a key role in our discussion of learning hard distributions, covered in Section 7.

We aim for two properties that our cost functions should exhibit in order to claim they have outperformed the MMD training method:

- *Speed of Converge:* Both cost functions, SD and SH should achieve equal or lower TV than the MMD in a *shorter* time period (even accounting for various learning rates).
- *Accuracy:* Since the cost functions we employ are in some sense 'stronger' than MMD, we want to see them achieve a *smaller* TV than is possible with the MMD in an equal or quicker time.

It should be noted that the TV appears to behave strangely during training, often initially increasing. The reason for this is that it is not be minimised *directly*, only through auxiliary cost functions. As such, we would not expect to see a monotonically decreasing curve. It is also clear that it would not be possible to compute TV in general as the number of qubits scales, but it is possible to do so to benchmark the small examples we use here.

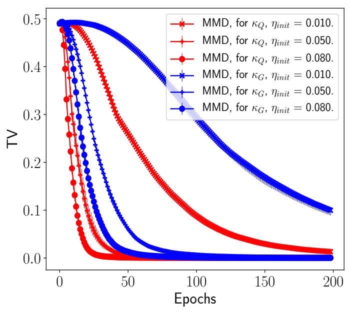
For all of examples here, we use a QAOA structured IBM, with the final measurement angle fixed for all qubits; $\Gamma_k = \pi/4 \forall k$ for simplicity. As such, only the Ising weights and biases, $\{J_{ij}, b_k\}$ were trained. We also did not initialise or restrict the parameter updates to be in the 'hard' regime discussed here, but allowed them to vary arbitrarily according to gradient descent. This is because, as far as the numerical results are concerned, we were only interested in testing the performance of our novel cost functions.

Firstly, in Figure 4 and Figure 5, we compare the use of the quantum kernel of (40) against the classical Gaussian one of (36) which was used in [8]. These results indicate that the Quantum kernel (40) is both suitable for training the IBM, and seems to converge faster when used in the MMD with all other parameters being identical. This is indicated in Figure 4a and Figure 5a. However, the computation of this kernel is likely to not be feasible for near term quantum devices, but does illustrate the potential of quantum kernels first explored in [55, 56].

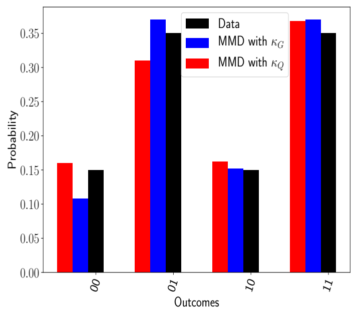
Figure 4 and Figure 5 illustrate the difference in training between the Quantum and Gaussian kernels, for two and four qubits respectively. The data was taken during 5 independent runs in each case, with averages and errors computed over the runs. This behaviour is apparent even for a range of learning rates, which we show. In the two qubit case, there is not a major difference between the two, but the discrepancy becomes more apparent as the number of qubits increases. This could be an indicator of the quantum kernel's extra expressive power and ability to capture finer details of the distribution, as in the four qubit case it can actually achieve a *lower* TV than the Gaussian kernel, used in [8]. Of course, this quantum kernel may not outperform all other kernels, or even different choices of the bandwith parameters in (36), but it is encouraging that such results were observed on a randomly selected dataset. Whether or not this performance is due to some fundamental 'quantum' reason, or simply that the kernel is more complicated, requires further study. The latter seems unlikely since the dataset is completely classical.

Next, Figure 6 and Figure 7 illustrate the differences between using the MMD cost function and either the Sinkhorn Divergence or the Stein Discrepancy to learn the same dataset. We found that with the exception of the two qubit case, both the Stein Discrepancy and the Sinkhorn Divergence are able to learn with a higher *speed of convergence*, and *accuracy* as shown in Figure 6b Figure 7b. It should be noted that the results for the Sinkhorn training were highly dependent on the value of the regularisation, as expected. Also, simply because the Sinkhorn achieved better results on one particular data set, does not imply that it would do so over most. However, the fact that the distribution in question was randomly generated, does look encouraging and supports this claim.

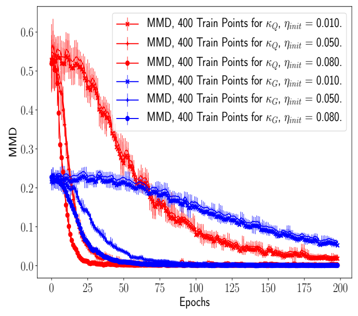
In the case of the Stein Discrepancy, we run the training algorithm for 125 epochs. The reason for this is twofold. Firstly, when training using the 'Exact' Stein Score (i.e. with



(a) TV Difference between κ_G [🟦] and κ_Q [🟥].

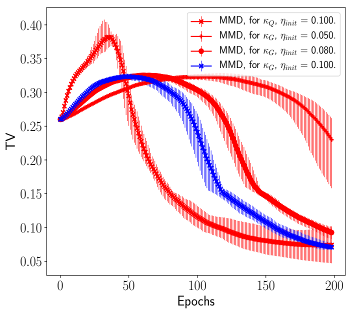


(b) Final learned probabilities using κ_G [🟦] and κ_Q [🟥] with $\eta_{\text{init}} = 0.08$.

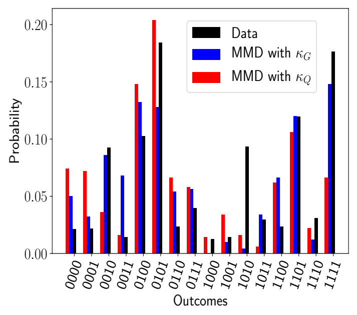


(c) MMD using κ_G [🟥] vs. κ_Q [🟦] on 400 training data points, 100 test data points.

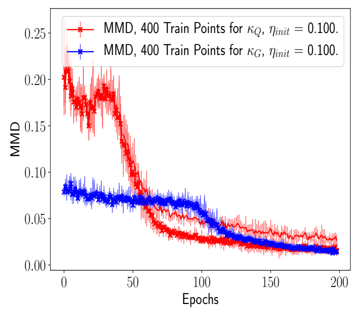
Figure 4: Quantum vs. Gaussian kernel for 2 qubits. 500 samples from IBM and Data, Batch Size = 250.



(a) TV Difference between κ_G [🟦] and κ_Q [🟥].



(b) Final learned probabilities using κ_G [🟦] and κ_Q [🟥] with $\eta_{\text{init}} = 0.1$.



(c) MMD using κ_G [🟦] vs. κ_Q [🟥] on 400 training data points, 100 test data points.

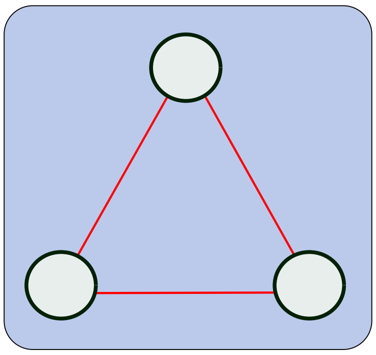
Figure 5: Quantum vs. Gaussian kernel for 4 qubits. 500 samples from IBM and Data, Batch Size = 250.

oracle access to the data probabilities π) the convergence is so fast that the model would tend to jump out of its convergence point minimum after a certain period of time. Hence we employed a form of early stopping, as indicated in Figure 6b. Secondly, for the case of the 'Spectral' Score method, (Figure 6b and Figure 7b) we used much less samples than with the previous methods as can be observed. This is due to the (classical) computational cost of computing the Score using this method in our naive implementation. We also took the best run of the Stein training algorithm, rather than an average in the other cases. This was due to the low sample numbers, which lead to a high deviation in the training path taken, so we removed it in order to not obscure the other results. For computing the Spectral Score, we used 3 and 6 Nyström eigenvectors for 3 and 4 qubits respectively, see Appendix B.

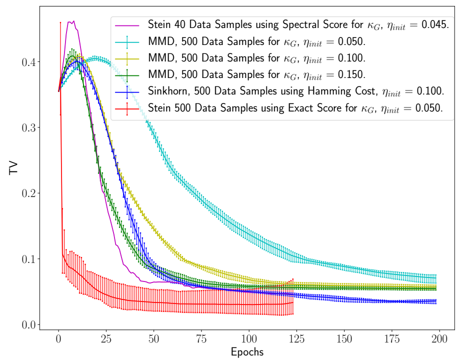
Finally, one may comment on the fact that we allowed the Stein Discrepancy to use the exact probabilities of the data, π, and this constitutes and unfair advantage against the MMD. In fact, the high number of samples we used ensured that the approximate data distributions that the MMD received was in fact very close to the exact data, and we found no major improvement for training with the MMD by allowing oracle data access, i.e. the exact probabilities, π, reinforcing the fundamental weakness of the MMD that is discussed above.

Finally, we perform experiments on the 16 Qubit QPU of Rigetti, `Aspen`, as seen in Figure 8 and Figure 9, to determine the performance of the training in practice. We used two sublattices for 3 and 4 qubits respectively, the `Aspen-4-3Q-A` and `Aspen-4-4Q-A`, and their respective `qvm` versions. As before we ran 5 independent runs of the training procedure from the same initial condition, and took averages over the run. We restricted to the native connectivity of the chip, as seen in Figure 8e, and used the available qubits in the `Aspen-4-3Q-A` chip, $(10, 11, 17)$, with no direct connection between qubits 11 and 17, as illustrated. Taking this into account, we did not enforce a fully connected topology since this would have resulted in the compiler implementing SWAP gates. A similar restriction was enforced for the 4 qubit version in Figure 7a. This is one reason why the models trained reasonably well on the hardware, with the fact that the two qubit gates in our IBM are native to the Rigetti chip (they implement CZ gates as the entangling links) also providing an advantage.

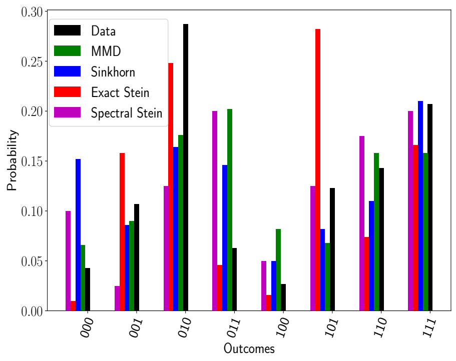
Figure 8a and Figure 9a compare training using the MMD with a Gaussian kernel and the Sinkhorn Divergence, benchmarked relative to TV as with the previous numerical results. As expected, the QPU noise leads to a higher variance between training runs, but the large number of samples taken (500) still permits the model to learn quickly. In both examples, it can been observed that the Sinkhorn cost function enabled faster and more accurate training, which becomes more evident as the number of qubits increases. It was also necessary to use quite an aggressive learning rate for four qubits while training with the MMD over SH, $0.15 vs. 0.1$, in order to get the model to train *at all* within the given timeframe. Finally, it was also observed that the absolute time for the training to complete (over 100 epochs) was actually less when training was performed (with 4 qubits) on the actual QPU itself, rather than when performed using the QVM exclusively. Such a phenomenon is obviously expected in general, but it was surprising to see it for only 4 qubits.



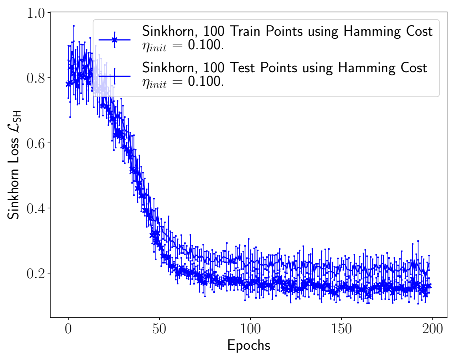
(a) Qubit topology in fully connected graph for three qubits, i.e. Rigetti `3q-qvm`.



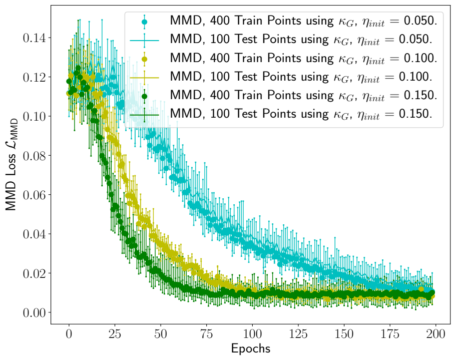
(b) TV Difference between MMD, Sinkhorn and Stein Training, with regularisation parameter $\epsilon = 0.08$ for SH, and 3 eigenvectors for Spectral Stein method.



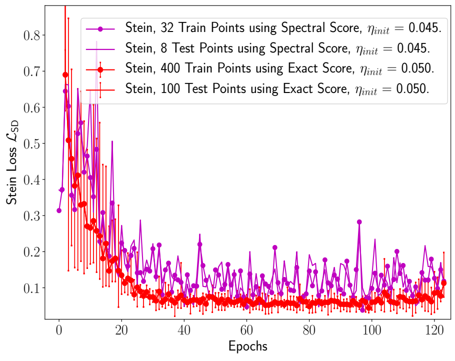
(c) Final learned probabilities of target data [⬛] using MMD [🟩] with $\eta_{init} = 0.15$, Sinkhorn [🟦], Exact Stein,[🟥] and Spectral Stein, [🟪].



(d) $\mathcal{L}^{0.08}_{\mathsf{SH}}$ for 3 qubits trained on the data (117) using 500 samples and a batch size of 250.

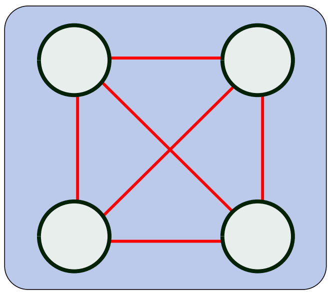


(e) $\mathcal{L}_{\mathsf{MMD}}$ for 3 qubits trained on the data (117) using 500 samples and a batch size of 250, with three different initial learning rates.

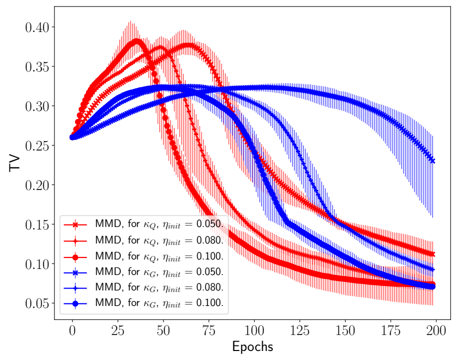


(f) $\mathcal{L}_{\mathsf{SD}}$ for 3 qubits trained on the data (117) using 500 samples and a batch size of 250 for Exact Score, and 40 Samples and batch size of 20 for the Spectral Score.

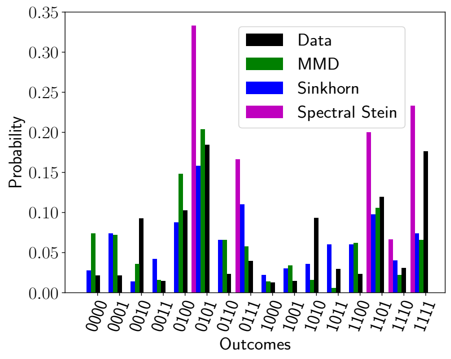
Figure 6: MMD [🟦, 🟨, 🟩] vs. Sinkhorn [🟦] and Stein training with Exact Score function [🟥] and Spectral Score method [🟪] for 3 qubits.



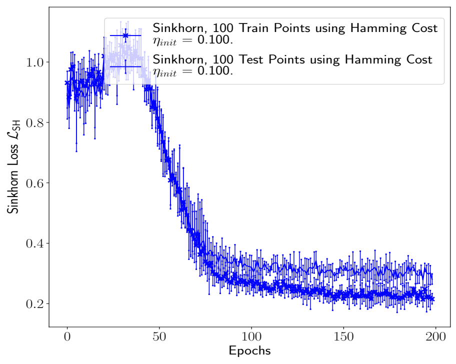
(a) Qubit topology in fully connected graph for four qubits, i.e. Rigetti `4q-qvm`.



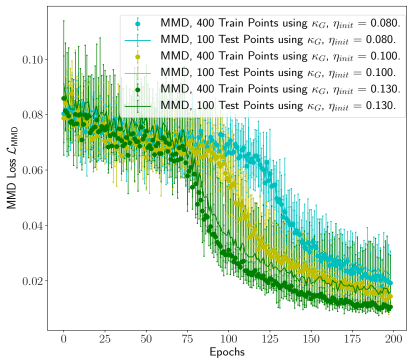
(b) TV Difference between MMD, Sinkhorn and Stein Training, with regularisation parameter $\epsilon = 0.1$ for SH, and 6 eigenvectors for the Spectral Stein method.



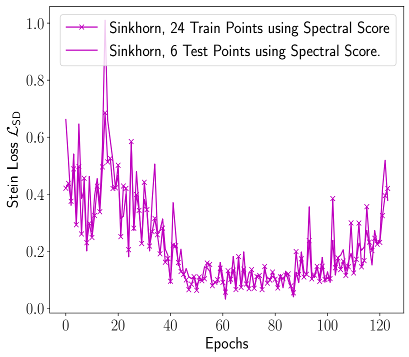
(c) Final learned probabilities of target data [⬛] using MMD [🟩] with $\eta_{init} = 0.15$, Sinkhorn [🟦], and Spectral Stein, [🟪].



(d) $\mathcal{L}_{\text{SH}}^{0.1}$ for 4 qubits trained on the data (117) using 500 samples and a batch size of 250.

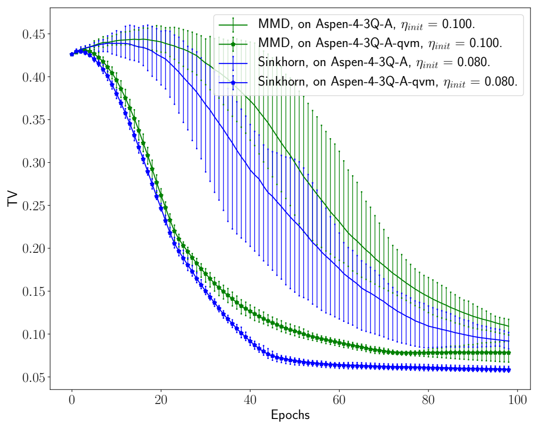


(e) $\mathcal{L}_{\text{MMD}}$ for 4 qubits trained on the data (117) using 500 samples and a batch size of 250, with three different initial learning rates.

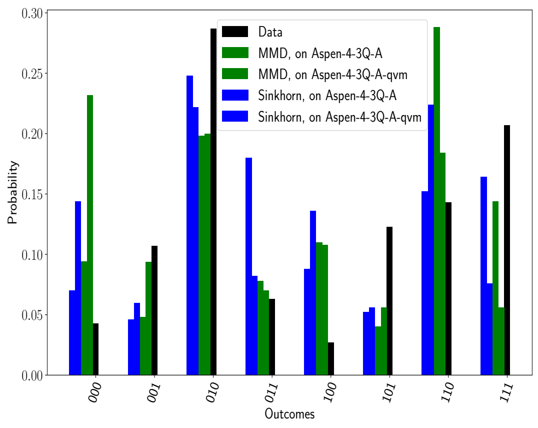


(f) $\mathcal{L}_{\text{SD}}$ for 4 qubits trained on the data (117) using 30 Samples and batch size of 20 for the Spectral Score.

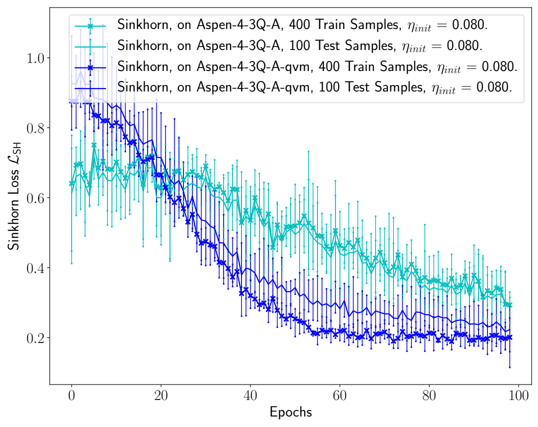
Figure 7: MMD [🟦, 🟨, 🟩] vs. Sinkhorn [🟦] and Stein training with Spectral Score method [🟪] for 4 qubits.



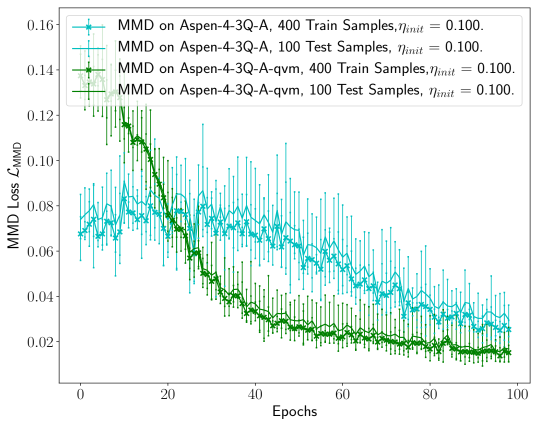
(a) TV Difference between MMD [🟩], and Sinkhorn [🟦] with regularisation parameter $\epsilon = 0.1$ on QVM vs QPU.



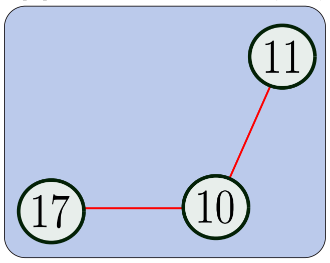
(b) Final learned probabilities of target data [⬛] using MMD [🟩] LR $\eta_{\text{init}} = 0.1$ and Sinkhorn [🟦] with $\epsilon = 0.1, \eta_{\text{init}} = 0.08$.



(c) $\mathcal{L}^{0.1}_{\text{SH}}$ for 3 qubits trained on the data (117)on QVM [🟦] vs. QPU [🟦].

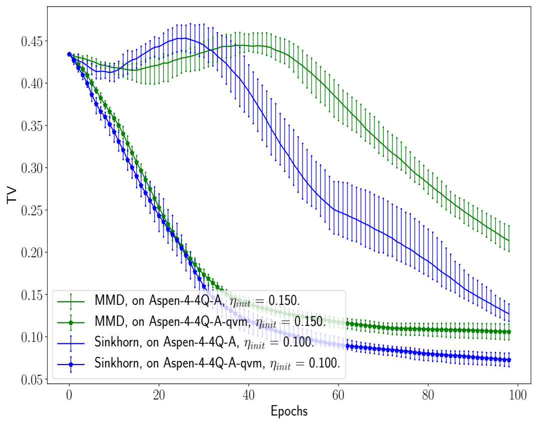


(d) $\mathcal{L}_{\text{MMD}}$ for 3 qubits trained on the data (117) on QVM [🟩] vs. QPU [🟦].

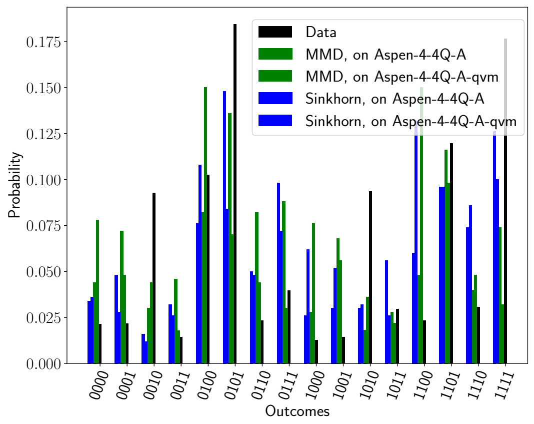


(e) Qubit 'line' topology in Rigetti `Aspen-4-3Q-A` chip, using qubits, $(10, 11, 17)$.

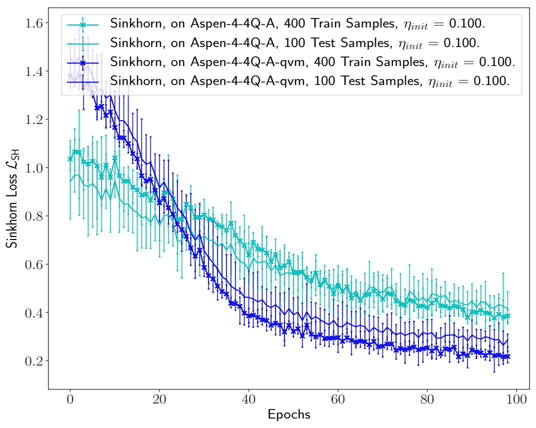
Figure 8: MMD vs. Sinkhorn for 3 qubits comparing performance on the real QPU (`Aspen-4-3Q-A`) vs. simulated behaviour on QVM (`Aspen-4-3Q-A-qvm`) using 500 samples and a batch size of 250.



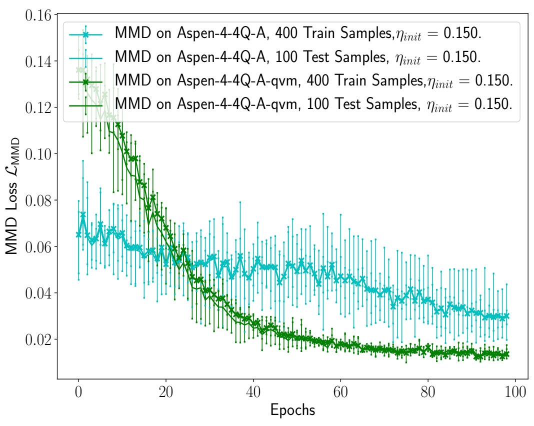
(a) TV Difference between MMD [🟩], and Sinkhorn [🟦] with regularisation parameter $\epsilon = 0.1$ on QVM vs QPU.



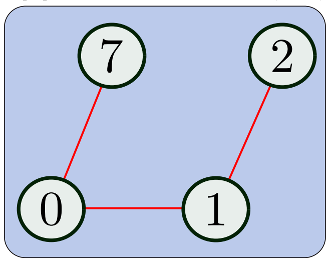
(b) Final learned probabilities of target data [⬛] using MMD [🟩] LR $\eta_{\text{init}} = 0.1$ and Sinkhorn [🟦] with $\epsilon = 0.1, \eta_{\text{init}} = 0.1$.



(c) $\mathcal{L}_{\text{SH}}^{0.1}$ for 4 qubits trained on the data (117)on QVM [🟦] vs. QPU [🟦].



(d) $\mathcal{L}_{\text{MMD}}$ for 4 qubits trained on the data (117) on QVM [🟩] vs. QPU [🟦].



(e) Qubit 'line' topology in Rigetti `Aspen-4-4Q-A` chip, using qubits, $(0, 1, 2, 7)$.

Figure 9: MMD vs. Sinkhorn for 4 qubits comparing performance on the real QPU (`Aspen-4-4Q-A`) vs. simulated behaviour on QVM (`Aspen-4-4Q-A-qvm`) using 500 samples and a batch size of 250.

7 Applications of the IBM

We propose two applications of the IBM. The first has more practical applicability, while the second is of more theoretical interest.

7.1 Automatic Compilation of Quantum Circuits with Classical Data

The first application is the use of the IBM to compile quantum circuits using classical data from a target circuit. This is likely a process which could not be simulated efficiently classically, particularly given the arguments presented in this paper. By its very nature, a Born machine provides a means of *automatic compilation* of quantum circuits, it finds an optimal configuration of the circuit parameters, θ, which reproduces a given probability distribution that could arise from a quantum circuit. This is similar in flavour to the recent work, [29] and [28], which provides a closely related idea using a parameterised quantum circuit, $V(\theta)$, as an ansatz, and updating the parameters to reproduce the behaviour of a given target unitary, U. The associated cost function minimises some cost between the two unitaries, but this cost function involves embedding the two unitaries in a larger circuit, which would be expensive in terms of quantum resources.

The approach we suggest is different to the standard approaches in quantum circuit compilation[15], in which the problem is to compile one (known) unitary directly into another. In the near term it is unlikely we would be able to embed unitaries into larger circuits in order to compute these cost functions, as in [28]. In particular, this straightforward approach becomes unfeasible as the size and complexity of circuits grows, so we should increase the tools we have available to tackle the problem. A more (at least experimentally) realistic task would be to try and compile based on classical measurement results from the target circuit. Of course, in this approach, there is no guarantee that neither the target unitaries, nor its parameters, would be identical in terms of their operation, only that the output distributions would be somewhat close, given the ansatz training circuit we have chosen. Nevertheless, the ability to compile quantum circuits using classical data may be a useful toolkit to have, particularly for benchmarking quantum device relative to one another. For example, this could be useful in a case where one does not have access to the direct evolution being implemented, but only measurement statistics from it, perhaps in the simulation of molecules.

Furthermore, this mindset could potentially be incorporated into quantum chip design, to extract the qubit layout/connectivity which gives a maximal amount of *quantumness*, given experimental constraints.

To illustrate this in a simple example, we apply the same techniques as above to train a circuit with a random initialisation of parameters. We begin with a $p = 1$ QAOA circuit, with a fixed final parameter in the measurement unitary, $\Gamma_k = \pi/4 \forall k$, and train the parameters, $\{J_{ij}^{\mathsf{QAOA}}, b_k^{\mathsf{QAOA}}\}$ exactly as before to represent the output distribution of an IQP circuit (where the measurement unitary is simply a Hadamard) with parameters, $\{J_{ij}^{\mathsf{IQP}}, b_k^{\mathsf{IQP}}\}$. Notice this is effectively the same as one IBM attempting to learn to represent another:

$$\mathsf{IBM}\left(\left\{J_{ij}^{\mathsf{QAOA}}, b_k^{\mathsf{QAOA}}\right\}, \left\{\Gamma_k = \frac{\pi}{4}\right\}, 0, 0\right) \tag{118}$$

$$\to \mathsf{IBM}\left(\left\{J_{ij}^{\mathsf{IQP}}, b_k^{\mathsf{IQP}}\right\}, \left\{\Gamma_k = \frac{\pi}{2\sqrt{2}}\right\}, 0, \left\{\Sigma_k = \frac{\pi}{2\sqrt{2}}\right\}\right) \tag{119}$$

[15]See [28] and references therein.

While this may *seem* trivial, since the two parameterised circuits seem the same initially as the layout of the parameterised sections will be the same, they will actually give rise to different distributions due to the interference provided by the final measurement unitary, which is different in both cases. We illustrate this in Figures (10 11) for two and three qubits. We assumed a fully connected graph, which is possible to access using the Rigetti QVM simulator but it would not be possible to directly access such connectivity on the QPU. We leave a more thorough analysis of this, similar to the work of [52], to future work.

Figure 10a and Figure 11a illustrate the circuit parameters which are achieved by Sinkhorn training. For ease of viewing, the cirucuit parameters are multiplied by a factor of 10. It is interesting to notice that, as discussed above, the final parameters of the QAOA circuit are not the same as the IQP target, however the resulting distributions only deviate by 1.5×10^2 in total variation, in the three qubit example in Figure 11c. It is likely that increasing the depth of the ansatz circuit would achieve better fits.

7.2 Learning Hard Distributions

Now, we hope to motivate why it could be possible to combine ideas in the previous Section 7.1 with the definitions in Section 2.4 so that the IBM could learn '*hard*' distributions. This concept is tightly related to the theory of PAC (Probably Approximately Correct) Learning which has been introduced to the quantum realm, [89], and in fact the theory of distribution learning was inspired by the PAC framework [40]. A major question in this field is whether or not *qsamples* (i.e. classical data samples presented in superposition to a learning algorithm) improve the learnability of certain concept classes. In some cases this is the case, but it is not generally true, [90]. In any case, it is a very interesting avenue to explore.

As intuition, and a motivating example, if a particular quantum chip is developed in the near future, which has provably demonstrated a quantum advantage in the sampling problems discussed above. Given a series of samples from such a quantum device, we could test to see if that distribution could be *learnt* by a new quantum device, which could not be represented by any classical device[16]. If the new device could, for example learn an approximation which was ϵ-close to the 'hard' distribution in TV, then the new device must also have the capability to do something non-classical. This line of thinking could potentially lead to a method to determine new classically-hard circuit classes. A major proof for an advantage using a quantum machine learning algorithm on a near term device, would be the ability of the IBM to learn a hard distribution efficiently.

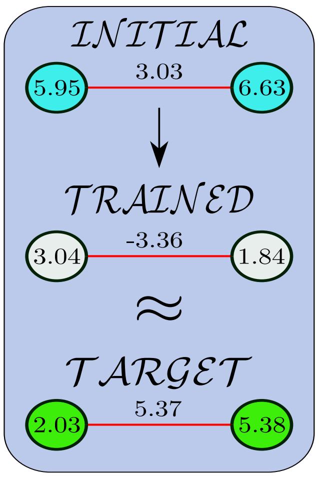
This is related to the Definition 2.8 in Section 2.4 in the following way. Let's suppose the metric we choose is the total variation distance, $d(p_\theta, p_H) = \mathsf{TV}(p_\theta, p_H)$ between our learning model (the IBM, p_θ) and p_H which is a distribution which is 'hard' for a classical device to sample from, for example an IQP distribution.

From the definition, 'Quantum Learning Supremacy' would be achieved if a quantum device (BQP) was able to output a generator, GEN_{p_θ} (a parameterised quantum circuit), which was ϵ-close to the true distribution, p_H, with probability $1 - \delta$:

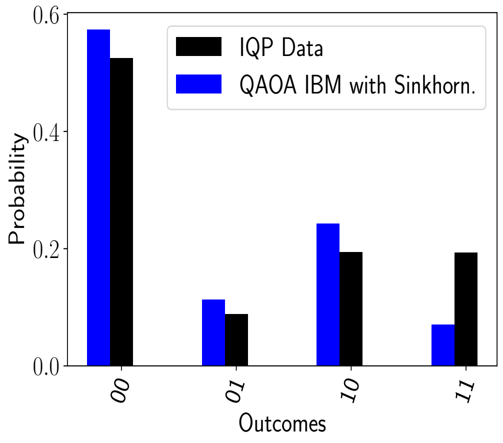
$$Pr\left[\mathsf{TV}\left(p_\theta, p_H\right) \leq \epsilon\right] \geq 1 - \delta \tag{120}$$

Now we know that the Born Machine has the *capacity* to represent that hard data distribution, [6] but can it actually infer enough about it in order to represent it itself?

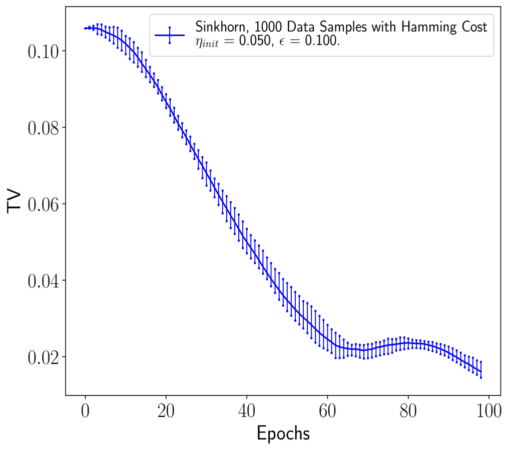
[16] According to the relevant complexity theory conjectures behind quantum sampling hardness arguments.



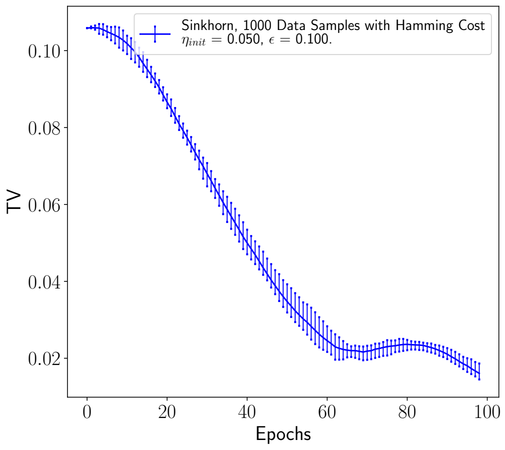
(a) Initial [🟦] and trained [⬜] QAOA circuit parameters for two qubits. Target IQP circuit parameters [🟩].



(b) Final learned probabilities of IBM (QAOA) [🟦] circuit versus 'data' probabilities (IQP) [⬛].

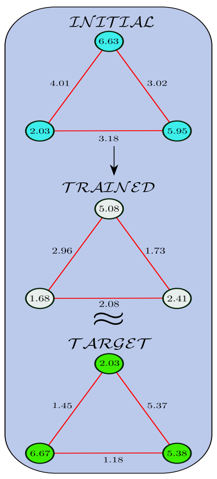


(c) Total Variation Distance

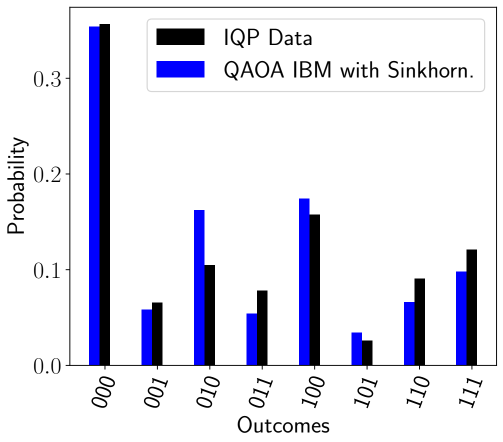


(d) Sinkhorn Divergence for 800 Train Samples, 200 Test Samples.

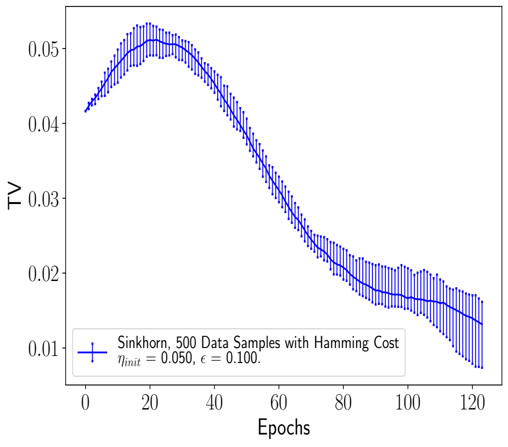
Figure 10: Automatic Compilation of IQP circuit to a $p = 1$ QAOA circuit with two qubits with $\epsilon = 0.05$.



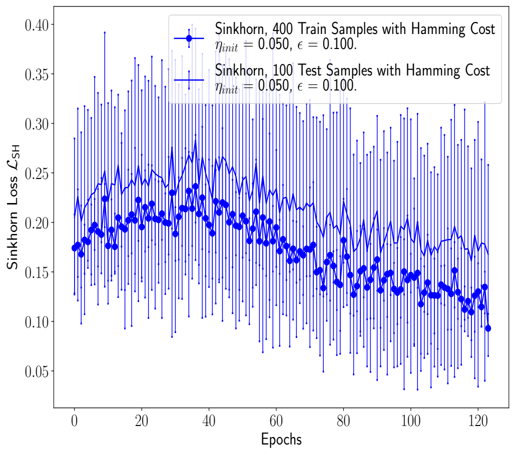
(a) Initial [🟦] and trained [⬜] QAOA circuit parameters for three qubits. Target IQP circuit parameters [🟩].



(b) Final learned probabilities of IBM (QAOA) [🟦] circuit versus 'data' probabilities (IQP) [⬛].



(c) Total Variation Distance during training.



(d) Sinkhorn Divergence for 400 Train samples, 100 Test Samples.

Figure 11: Automatic Compilation of IQP circuit to a $p = 1$ QAOA circuit with two qubits with $\epsilon = 0.05$.

An obvious approach to this problem would be to use the (TV) itself as a cost function, and minimise it with respect to the hard distribution. Now, if p_H is produced by a quantum supremacy circuit class, then by the hardness arguments, there can exist no randomised poly-time classical algorithm, Q, that can produce samples according to a distribution, q such that the following holds:

$$\mathsf{TV}(p_H, q) \leq \delta \tag{121}$$

Where δ is ideally a constant but will depend on the hard distribution p_H, for example, with IQP, $\delta = 1/384$ [34].

Now, if it is possible to *learn* a representation of p_H using the IBM, then we can claim that no classical algorithm could exist to simulate the IBM either to the required precision in TV. More concretely, if we are able to train the IBM to achieve closeness in a variation distance to a constant, ϵ, of the hard distribution, p_H then:

$$\mathsf{TV}(p_\theta, p_H) \leq \epsilon \tag{122}$$

so:

$$\begin{aligned}\mathsf{TV}(p_H, q) &= \frac{1}{2}\sum_{\mathbf{x}} |p_H(\mathbf{x}) - q(\mathbf{x})| = \frac{1}{2}\sum_{\mathbf{x}} |p_H(\mathbf{x}) - p_\theta(\mathbf{x}) + p_\theta(\mathbf{x}) - q(\mathbf{x})| \\ &\leq \frac{1}{2}\sum_{\mathbf{x}} |p_H(\mathbf{x}) - p_\theta(\mathbf{x})| + \frac{1}{2}\sum_{\mathbf{x}} |p_\theta(\mathbf{x}) - q(\mathbf{x})| \\ &\leq \epsilon + \frac{1}{2}\sum_{\mathbf{x}} |p_\theta(\mathbf{x}) - q(\mathbf{x})| \leq 2\epsilon = \delta \end{aligned} \tag{123}$$

Using the triangle inequality, and the last line follows if it possible to get within ϵ of p_θ using the algorithm Q, and so this is a contradiction to the implausibility of (121).

This idea is very closely connected to the *verification* of these supremacy circuits and resulting distributions. The verification problem for a supremacy experiment is to determine if the physical quantum device is actually producing samples from the *correct* or *ideal* distribution, be it a BosonSampling experiment, or a Random/IQP circuit for example. As discussed above, the desired supremacy criterion, would be such that no classical algorithm could reach the distribution in question up to a constant total variation distance. Therefore, we must check whether or not the physical quantum device is actually achieving this! This is of crucial importance with the approach of the quantum supremacy era, since benchmarking a quantum device is almost as critical as building it in the first place. Potential solutions currently include including some bias in the circuit which could be checked, [5, 17], making assumptions about the device using cross-entropy benchmarking, [15, 91] or by computing certain probabilities of the circuit [14].

However, actually training the Born Machine using the TV is not a practical thing to do; we can see from (122) that to do so would require computing the output probabilities of the IBM, and the hard data distribution, which is #P-Hard in general.

Therefore, we could suggest using *another* metric as a cost function, ideally one which upper bounds the TV. If such a metric could be found, which is sample-efficient to compute, one could minimise it in a learning algorithm, which would directly minimise the TV through the upper bound.

Unfortunately, the MMD is not ideal for this purpose, as it actually *lower bounds* the total variation distance, [58]:

$$TV(p_\theta, p_s) \geq \frac{\gamma_{\mathsf{MMD}}(P,Q)}{\sqrt{C}} \tag{124}$$

if $C := \sup_{\mathbf{x}\in\mathcal{X}^d} \kappa(\mathbf{x},\mathbf{x}) < \infty$. In the cases of the two kernels we use in this work, $C = 1$:

$$\kappa_G(\mathbf{x},\mathbf{x}) = \frac{1}{c}\sum_{i=1}^{c} e^{-\frac{1}{2\sigma_i}|\mathbf{x}-\mathbf{x}|^2} = \frac{1}{c}(c) = 1 \tag{125}$$

$$\kappa_Q(\mathbf{x},\mathbf{x}) = |\langle\phi(\mathbf{x})|\phi(\mathbf{x})\rangle|^2 = |\langle 0|0\rangle^{\otimes n}|^2 = 1\forall\mathbf{x} \tag{126}$$

So the supremum is equal to one, for all $\mathbf{x}$. Unfortunately, a lower bound on the TV is not sufficient, and as a result minimising the MMD between the distributions says nothing necessarily about the value of the TV between them. This is also the reason why we opted for stronger cost functions than the MMD in this work. An alternative choice would be the Optimal Transport Metric or Wasserstein Distance (81). In fact, TV is in fact a form of Optimal Transport with a trivial cost, [92].

From [92], we get the following inequalities for a discrete space, $\mathcal{X}$[17]:

$$d_{min}\,\mathsf{TV}(p_\theta, p_H) \leq \mathsf{OT}_0^d(p_\theta, p_H) \leq \text{diam}(\mathcal{X})\,\mathsf{TV}(p_\theta, p_H) \tag{127}$$

where $\text{diam}(\mathcal{X}) = \max\{d(\mathbf{x},\mathbf{y}), \mathbf{x},\mathbf{y}\in\mathcal{X}\}$, $d_{min} = \min_{x\neq y} d(\mathbf{x},\mathbf{y})$, and $d(\mathbf{x},\mathbf{y})$ is the ground metric used in the Wasserstein Distance, (81). If, for instance, we were to choose $d(\mathbf{x},\mathbf{y})$ to be the ℓ_2 metric on the hypercube as discussed in Section 4.5.1 between the binary vectors, $\mathbf{x},\mathbf{y}$, i.e. the Hamming distance, then we get that $d_{min} = 1, \text{diam}(\mathcal{X}) = \sqrt{n}$, and so:

$$\mathsf{TV}(p_\theta, p_H) \leq \mathsf{OT}_0^{\ell_2}(p_\theta, p_H) \leq \sqrt{n}\,\mathsf{TV}(p_\theta, p_H) \tag{128}$$

We might hope therefore that by using the $\mathsf{OT}_0^{\ell_2}$ distance as a cost function to train the IBM, (which can be estimated using samples), we would be able to minimise it sufficiently to fall within the Total Variation threshold for Quantum Supremacy. Unfortunately, as noted in Section 4.5, the sample complexity of the Wasserstein distance is still exponential in the number of qubits, and hence we would not expect efficient training using this metric either.

Worse still, is the recent work of [93], in which it is proven the quantum supremacy 'candidates' that exist in the literature, for example, IQP, Random Circuit Sampling, BosonSampling, all require exponentially many samples to certify that the distribution arising from the quantum device is in fact close to the required one. More specifically, they show that no efficient 'testing' algorithm exists for these circuit classes which can pass a so-called Total Variation Identity Test. This problem involves being able to determine if the output distribution is correct (the sampled distribution is identical to the ideal one) or at least ϵ away in TV without making any assumptions about the device. This result potentially implies that learning these distributions efficiently is also not possible[18], due to the relationship between learning, and property testing of distributions, [94, 95].

[17]Note that the left hand inequality does not exist in the case that the distributions have continuous support.

[18]This result only holds for a 'device independent' scenario, but this is also likely the situation a corresponding learning algorithm would be in also.

However, hope is not lost. As conjectured in [93] we could hope for the existence of *some* distributions which can be efficiently tested (and also learned) but which still admit a sampling hardness result. It is with these hypothetical distributions for which we could hope to prove some classical-quantum separation in learnability to satisfy Definition 2.8.

Secondly, the result only applies to the *classical* certification of these distributions. In our case, we have a quantum model, and one could hope for the possibility that a quantum learning model would be able to infer more information from the samples than is possible using a classical model alone, given that these quantum models do have the capacity to *represent* hard distributions.

Finally on this note, we recall a last result from [81]:

Theorem 7.1 (Theorem 1. from [81])**.** *Let P and Q be probability distributions on $\mathcal{X}, \mathcal{Y}$ subsets of $\mathbb{R}^d$, such that $|\mathcal{X}| = |\mathcal{Y}| \leq D$ and assume that c is L-Lipschitz w.r.t. $\mathbf{x}, \mathbf{y}$. Then:*

$$0 \leq \mathcal{L}^{\epsilon}_{\mathsf{SH}}(P, Q) - \mathsf{OT}^{c}_{0}(P, Q) \leq 2\epsilon \log \left(\frac{e^2 LD}{\sqrt{d}\epsilon} \right) \tag{129}$$

$$\sim_{\epsilon \to 0} 2\epsilon \log(1/\epsilon) \tag{130}$$

Again, for our specific case, we have $D = \sqrt{d}$, $L = \sqrt{d}$[19]:

$$0 \leq \mathcal{L}^{\epsilon}_{\mathsf{SH}}(P, Q) - \mathsf{OT}^{\ell_2}_{0}(P, Q) \leq 2\epsilon \log \left(\frac{e^2 \sqrt{d}}{\epsilon} \right) \tag{131}$$

The log term will be positive as long as $\epsilon \leq \sqrt{d}e^2$, in which case the Sinkhorn Divergence will give an upper bound for the Wasserstein distance, and hence the TV through (127). Of course, for this low value of ϵ, it would take exponentially many samples to approximate the Sinkhorn Divergence to make this bound, as evidenced by [81], and Section 4.5.1 above, since there is a gap of a factor of d between the two values for ϵ. Note if we took the cost, c, to be the ℓ_2 metric directly in Section 4.5.1, the gap would be a factor of $\sqrt{d}$. It would be of interest to study this kind of relationship, hopefully this line of thinking would be able to define more optimal algorithms, for both verification of quantum devices using these cost functions, and the efficient learning of hard quantum distributions.

8 Conclusion

In conclusion, we have studied a specific generative quantum machine learning model, which we call the Ising Born Machine. We claim that most parts of the learning algorithm could not be simulated by purely classical means in the worst case, up to multiplicative error. We introduced two new cost functions, the Stein Discrepancy and the Sinkhorn Divergence, and adapted them to train the Ising Born Machine. We study numerically the potential advantage of using quantum kernels over classical kernels, and the advantage of both the Stein Discrepancy and the Sinkhorn Divergence over the MMD to achieve lower error in total variation. This was validated through numerical experiments run on Rigetti's QVM simulator and their Aspen chip. We also test an application in quantum circuit compilation

[19]The square root in this Lipschitz constant comes from taking the cost to be the ℓ_2 metric, and not its squared version, which is not a metric. This equates the Optimal Transport metric to the Wasserstein metric, in which case the upper bound on TV holds.

using classical data, and show numerically how this can be achieved. We discuss the possibility of using this model to learn certain hard distributions, for example those exhibiting Quantum Computational Supremacy, and provided a definition for a generative QML algorithm to demonstrate 'Quantum Learning Supremacy'. In this regard, we discuss how one could potentially achieve this by leveraging certain upper bounds on total variation relating to the integral probability metrics which we use as training cost functions.

Future work directions would be, for example, determining how limited connectivity and noise affect the performance of the training algorithm. We also will run experiments on different datasets, for example the Bars-and-Stripes. A second interesting direction would be to find example of distributions which could admit a demonstration of Quantum Learning Supremacy, and provide a definitive separation (up to relevant complexity theoretic conjectures) between quantum and classical machine learning algorithms in practice. While this is quite a strong claim, it is likely that weaker claims could be made which perhaps would be more ameanable to the near term.

9 Acknowledgements

We thank Andru Gheorghiu for useful discussions and title suggestion. We also thank Jean Feydy, Patric Fulop, Vojtech Havlicek and Jiasen Yang for clarifying pointers. This work was supported by the Engineering and Physical Sciences Research Council (grant EP/L01503X/1), EPSRC Centre for Doctoral Training in Pervasive Parallelism at the University of Edinburgh, School of Informatics and Entrapping Machines, (grant FA9550-17-1-0055). We also thank Rigetti Computing for the use of their Quantum compute resources, and the views expressed in this paper are solely those of the authors and have no relation to Rigetti Computing.

References

[1] John Preskill. Quantum Computing in the NISQ era and beyond. *arXiv:1801.00862 [cond-mat, physics:quant-ph]*, January 2018. URL http://arxiv.org/abs/1801.00862. arXiv: 1801.00862.

[2] P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi: 10.1137/S0097539795293172. URL https://doi.org/10.1137/S0097539795293172.

[3] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.*, 103(15):150502, October 2009. doi: 10.1103/PhysRevLett.103.150502. URL https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[4] Scott Aaronson and Alex Arkhipov. The Computational Complexity of Linear Optics. *Theory of Computing*, 9(1):143–252, February 2013. ISSN 1557-2862. doi: 10.4086/toc.2013.v009a004. URL http://www.theoryofcomputing.org/articles/v009a004.

[5] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. *Proceedings of the Royal Society A: Mathematical, Phys-*

ical and Engineering Science, January 2009. doi: 10.1098/rspa.2008.0443. URL http://rspa.royalsocietypublishing.org/content/early/2009/02/18/rspa.2008.0443.abstract.

[6] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. The Expressive Power of Parameterized Quantum Circuits. *arXiv:1810.11922 [quant-ph]*, October 2018. URL http://arxiv.org/abs/1810.11922. arXiv: 1810.11922.

[7] Song Cheng, Jing Chen, and Lei Wang. Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines. *arXiv:1712.04144 [cond-mat, physics:physics, physics:quant-ph, stat]*, December 2017. URL http://arxiv.org/abs/1712.04144. arXiv: 1712.04144.

[8] Jin-Guo Liu and Lei Wang. Differentiable Learning of Quantum Circuit Born Machine. *arXiv:1804.04168 [quant-ph, stat]*, April 2018. URL http://arxiv.org/abs/1804.04168. arXiv: 1804.04168.

[9] Edward Farhi and Hartmut Neven. Classification with Quantum Neural Networks on Near Term Processors. *arXiv:1802.06002*, February 2018. URL https://arxiv.org/abs/1802.06002.

[10] Marcello Benedetti, Delfina Garcia-Pintos, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *arXiv:1801.07686 [quant-ph]*, January 2018. URL http://arxiv.org/abs/1801.07686. arXiv: 1801.07686.

[11] Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. *arXiv:1603.08675 [quant-ph]*, March 2016. URL http://arxiv.org/abs/1603.08675. arXiv: 1603.08675.

[12] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum Support Vector Machine for Big Data Classification. *Phys. Rev. Lett.*, 113(13):130503, September 2014. doi: 10.1103/PhysRevLett.113.130503. URL https://link.aps.org/doi/10.1103/PhysRevLett.113.130503.

[13] Iordanis Kerenidis and Alessandro Luongo. Quantum classification of the MNIST dataset via Slow Feature Analysis. *arXiv:1805.08837 [quant-ph]*, May 2018. URL http://arxiv.org/abs/1805.08837. arXiv: 1805.08837.

[14] Scott Aaronson and Lijie Chen. Complexity-Theoretic Foundations of Quantum Supremacy Experiments. *arXiv:1612.05903 [quant-ph]*, December 2016. URL http://arxiv.org/abs/1612.05903. arXiv: 1612.05903.

[15] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. Quantum Supremacy and the Complexity of Random Circuit Sampling. *arXiv:1803.04402 [quant-ph]*, March 2018. URL http://arxiv.org/abs/1803.04402. arXiv: 1803.04402.

[16] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. 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):459–472, 2011. ISSN 1364-5021. doi: 10.1098/rspa.2010.0301. URL http://rspa.royalsocietypublishing.org/content/467/2126/459.

[17] Daniel Mills, Anna Pappa, Theodoros Kapourniotis, and Elham Kashefi. Information Theoretically Secure Hypothesis Test for Temporally Unstructured Quantum Computation (Extended Abstract). *Electronic Proceedings in Theoretical Computer Science*, 266:209–221, February 2018. ISSN 2075-2180. doi: 10.4204/EPTCS.266.14. URL http://arxiv.org/abs/1803.00706. arXiv: 1803.00706.

[18] Ewin Tang. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv:1811.00414 [quant-ph]*, October 2018. URL http://arxiv.org/abs/1811.00414. arXiv: 1811.00414.

[19] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *arXiv:1807.04271 [quant-ph]*, July 2018. URL http://arxiv.org/abs/1807.04271. arXiv: 1807.04271.

[20] Alexandr Andoni, Robert Krauthgamer, and Yosef Pogrow. On Solving Linear Systems in Sublinear Time. *arXiv:1809.02995 [cs]*, September 2018. URL http://arxiv.org/abs/1809.02995. arXiv: 1809.02995.

[21] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *arXiv:1811.04852 [quant-ph]*, November 2018. URL http://arxiv.org/abs/1811.04852. arXiv: 1811.04852.

[22] Andrs Gilyn, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv:1811.04909 [quant-ph]*, November 2018. URL http://arxiv.org/abs/1811.04909. arXiv: 1811.04909.

[23] Seth Lloyd and Samuel L. Braunstein. Quantum Computation over Continuous Variables. *Physical Review Letters*, 82(8):1784–1787, February 1999. doi: 10.1103/PhysRevLett.82.1784. URL https://link.aps.org/doi/10.1103/PhysRevLett.82.1784.

[24] Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, Maria Schuld, Nicols Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *arXiv:1806.06871 [quant-ph]*, June 2018. URL http://arxiv.org/abs/1806.06871. arXiv: 1806.06871.

[25] Yuxuan Du, Tongliang Liu, and Dacheng Tao. Bayesian Quantum Circuit. *arXiv:1805.11089 [quant-ph]*, May 2018. URL http://arxiv.org/abs/1805.11089. arXiv: 1805.11089.

[26] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *arXiv:1804.08641 [quant-ph]*, April 2018. URL http://arxiv.org/abs/1804.08641. arXiv: 1804.08641.

[27] Andrea Rocchetto, Edward Grant, Sergii Strelchuk, Giuseppe Carleo, and Simone Severini. Learning hard quantum distributions with variational autoencoders. *npj Quantum Information*, 4(1):28, June 2018. ISSN 2056-6387. doi: 10.1038/s41534-018-0077-z. URL https://doi.org/10.1038/s41534-018-0077-z.

[28] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. Quantum-assisted quantum compiling. *arXiv:1807.00800 [quant-ph]*, July 2018. URL http://arxiv.org/abs/1807.00800. arXiv: 1807.00800.

[29] Tyson Jones and Simon C. Benjamin. Quantum compilation and circuit optimisation via energy dissipation. *arXiv:1811.03147 [quant-ph]*, November 2018. URL http://arxiv.org/abs/1811.03147. arXiv: 1811.03147.

[30] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum Circuit Learning. *arXiv:1803.00745 [quant-ph]*, March 2018. URL http://arxiv.org/abs/1803.00745. arXiv: 1803.00745.

[31] Keisuke Fujii and Tomoyuki Morimae. Commuting quantum circuits and complexity of Ising partition functions. *New Journal of Physics*, 19(3): 033003, March 2017. ISSN 1367-2630. doi: 10.1088/1367-2630/aa5fdb. URL http://stacks.iop.org/1367-2630/19/i=3/a=033003?key=crossref.cefbe34cf11242886552ceea447a4526.

[32] Edward Farhi and Aram W. Harrow. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *arXiv:1602.07674 [quant-ph]*, February 2016. URL http://arxiv.org/abs/1602.07674. arXiv: 1602.07674.

[33] Keisuke Fujii, Hirotada Kobayashi, Tomoyuki Morimae, Harumichi Nishimura, Shuhei Tamate, and Seiichiro Tani. Impossibility of Classically Simulating One-Clean-Qubit Model with Multiplicative Error. *Phys. Rev. Lett.*, 120(20):200502, May 2018. doi: 10.1103/PhysRevLett.120.200502. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.200502.

[34] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Average-Case Complexity Versus Approximate Simulation of Commuting Quantum Computations. *Phys. Rev. Lett.*, 117(8):080501, August 2016. doi: 10.1103/PhysRevLett.117.080501. URL https://link.aps.org/doi/10.1103/PhysRevLett.117.080501.

[35] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum*, 1: 8, April 2017. ISSN 2521-327X. doi: 10.22331/q-2017-04-25-8. URL https://doi.org/10.22331/q-2017-04-25-8.

[36] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014. URL http://arxiv.org/abs/1411.4028. arXiv: 1411.4028.

[37] Michael A. Nielsen. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, tenth anniversary edition.. edition, 2010. ISBN 978-1-107-00217-3.

[38] Matty J. Hoban, Joel J. Wallman, Hussain Anwar, Nari Usher, Robert Raussendorf, and Dan E. Browne. Measurement-Based Classical Computation. *Phys. Rev. Lett.*, 112(14):140505, April 2014. doi: 10.1103/PhysRevLett.112.140505. URL https://link.aps.org/doi/10.1103/PhysRevLett.112.140505.

[39] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. A quantum algorithm to train neural networks using low-depth circuits. *arXiv:1712.05304 [cond-mat, physics:quant-ph]*, December 2017. URL http://arxiv.org/abs/1712.05304. arXiv: 1712.05304.

[40] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the Learnability of Discrete Distributions. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 273–282, New York, NY, USA, 1994. ACM. ISBN 0-89791-663-8. doi: 10.1145/195058.195155. URL http://doi.acm.org/10.1145/195058.195155. event-place: Montreal, Quebec, Canada.

[41] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, 2018. ISBN 978-3-319-96423-2. URL https://www.springer.com/us/book/9783319964232.

[42] Ya-Hui Zhang. Entanglement Entropy of Target Functions for Image Classification and Convolutional Neural Network. *arXiv:1710.05520 [cond-mat]*, October 2017. URL http://arxiv.org/abs/1710.05520. arXiv: 1710.05520.

[43] Yoav Levine, David Yakira, Nadav Cohen, and Amnon Shashua. Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design. *arXiv:1704.01552 [quant-ph]*, April 2017. URL http://arxiv.org/abs/1704.01552. arXiv: 1704.01552.

[44] Xun Gao, Zhengyu Zhang, and Luming Duan. An efficient quantum algorithm for generative machine learning. *arXiv:1711.02038 [quant-ph, stat]*, November 2017. URL http://arxiv.org/abs/1711.02038. arXiv: 1711.02038.

[45] Ding Liu, Shi-Ju Ran, Peter Wittek, Cheng Peng, Raul Blzquez Garca, Gang Su, and Maciej Lewenstein. 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]*, October 2017. URL http://arxiv.org/abs/1710.04833. arXiv: 1710.04833.

[46] Vasily Pestun and Yiannis Vlassopoulos. Tensor network language model. *arXiv:1710.10248 [cond-mat, stat]*, October 2017. URL http://arxiv.org/abs/1710.10248. arXiv: 1710.10248.

[47] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised Generative Modeling Using Matrix Product States. *arXiv:1709.01662 [cond-mat, physics:quant-ph, stat]*, September 2017. URL http://arxiv.org/abs/1709.01662. arXiv: 1709.01662.

[48] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. Equivalence of restricted Boltzmann machines and tensor network states. *Phys. Rev. B*, 97(8):085104, February 2018. doi: 10.1103/PhysRevB.97.085104. URL https://link.aps.org/doi/10.1103/PhysRevB.97.085104.

[49] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. Learning and Inference on Generative Adversarial Quantum Circuits. *arXiv:1808.03425 [quant-ph, stat]*, August 2018. URL http://arxiv.org/abs/1808.03425. arXiv: 1808.03425.

[50] Jonathan Romero and Alan Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *arXiv:1901.00848 [quant-ph]*, January 2019. URL http://arxiv.org/abs/1901.00848. arXiv: 1901.00848.

[51] Xun Gao, Sheng-Tao Wang, and L.-M. Duan. Quantum Supremacy for Simulating a Translation-Invariant Ising Spin Model. *Phys. Rev. Lett.*, 118(4):040502, January 2017. doi: 10.1103/PhysRevLett.118.040502. URL https://link.aps.org/doi/10.1103/PhysRevLett.118.040502.

[52] Vicente Leyton-Ortega, Alejandro Perdomo-Ortiz, and Oscar Perdomo. Robust Implementation of Generative Modeling with Parametrized Quantum Circuits. *arXiv:1901.08047 [quant-ph]*, January 2019. URL http://arxiv.org/abs/1901.08047. arXiv: 1901.08047.

[53] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *arXiv:1803.11173 [physics, physics:quant-ph]*, March 2018. URL http://arxiv.org/abs/1803.11173. arXiv: 1803.11173.

[54] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *arXiv:1903.05076 [quant-ph]*, March 2019. URL http://arxiv.org/abs/1903.05076. arXiv: 1903.05076.

[55] Maria Schuld and Nathan Killoran. Quantum machine learning in feature Hilbert spaces. *arXiv:1803.07128 [quant-ph]*, March 2018. URL http://arxiv.org/abs/1803.07128. arXiv: 1803.07128.

[56] Vojtch Havlek, Antonio D. Crcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, March 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-0980-2. URL https://doi.org/10.1038/s41586-019-0980-2.

[57] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schlkopf. Kernel Mean Embedding of Distributions: A Review and Beyond. *Foundations and Trends in Machine Learning*, 10(1-2):1–141, 2017. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000060. URL http://arxiv.org/abs/1605.09522. arXiv: 1605.09522.

[58] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schlkopf, and Gert R. G. Lanckriet. On integral probability metrics, \phi-divergences and binary classification. *arXiv:0901.2698 [cs, math]*, January 2009. URL http://arxiv.org/abs/0901.2698. arXiv: 0901.2698.

[59] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schlkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13:723773, March 2012. ISSN 1533-7928. URL http://jmlr.csail.mit.edu/papers/v13/gretton12a.html.

[60] Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. Universality, Characteristic Kernels and RKHS Embedding of Measures. *Journal of Machine Learning Research*, 12(Jul):2389–2410, 2011. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v12/sriperumbudur11a.html.

[61] Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schlkopf, and Gert R. G. Lanckriet. Hilbert Space Embeddings and Metrics on Probability Measures. *Journal of Machine Learning Research*, 11(Apr):1517–1561, 2010. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v11/sriperumbudur10a.html.

[62] R. M. Dudley. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2002. doi: 10.1017/CBO9780511755347.

[63] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schlkopf, and Alex J. Smola. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics (Oxford, England)*, 22(14):e49–57, July 2006. ISSN 1367-4811. doi: 10.1093/bioinformatics/btl242.

[64] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Schlkopf, and Alex J. Smola. A Kernel Method for the Two-Sample-Problem. In B. Schlkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, 2007. URL http://papers.nips.cc/paper/3110-a-kernel-method-for-the-two-sample-problem.pdf.

[65] Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schlkopf. Kernel Measures of Conditional Dependence. In *NIPS*, 2007.

[66] Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Gert R. G. Lanckriet, and Bernhard Schlkopf. Injective Hilbert Space Embeddings of Probability Measures. In *COLT*, 2008.

[67] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *arXiv:1811.11184 [quant-ph]*, November 2018. URL http://arxiv.org/abs/1811.11184. arXiv: 1811.11184.

[68] Jackson Gorham and Lester Mackey. Measuring Sample Quality with Stein\textquotesingle s Method. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 226–234. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5768-measuring-sample-quality-with-steins-method.pdf.

[69] Qiang Liu, Jason D. Lee, and Michael Jordan. A Kernelized Stein Discrepancy for Goodness-of-fit Tests. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 276–284, New York, NY, USA, 2016. JMLR.org. URL http://dl.acm.org/citation.cfm?id=3045390.3045421.

[70] Jiasen Yang, Qiang Liu, Vinayak Rao, and Jennifer Neville. Goodness-of-Fit Testing for Discrete Distributions via Stein Discrepancy. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5561–5570, Stockholmsmssan, Stockholm Sweden, July 2018. PMLR. URL http://proceedings.mlr.press/v80/yang18c.html.

[71] Aaditya Ramdas, Nicolas Garcia, and Marco Cuturi. On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests. *arXiv:1509.02237 [math, stat]*, September 2015. URL http://arxiv.org/abs/1509.02237. arXiv: 1509.02237.

[72] Aude Genevay, Gabriel Peyr, and Marco Cuturi. Learning Generative Models with Sinkhorn Divergences. *arXiv:1706.00292 [stat]*, June 2017. URL http://arxiv.org/abs/1706.00292. arXiv: 1706.00292.

[73] Jean Feydy, Thibault Sjourn, Franois-Xavier Vialard, Shun-ichi Amari, Alain Trouv, and Gabriel Peyr. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. *arXiv: 1810.08278*, October 2018. URL https://arxiv.org/abs/1810.08278.

[74] Cdric Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, Berlin Heidelberg, 2009. ISBN 978-3-540-71049-3. URL //www.springer.com/gb/book/9783540710493.

[75] Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*, January 2017. URL http://arxiv.org/abs/1701.07875. arXiv: 1701.07875.

[76] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *arXiv:1804.09139 [quant-ph]*, April 2018. URL http://arxiv.org/abs/1804.09139. arXiv: 1804.09139.

[77] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, and Luyan Sun. Quantum generative adversarial learning in a superconducting quantum circuit. *Science Advances*, 5(1):eaav2761, January 2019. doi: 10.1126/sciadv.aav2761. URL http://advances.sciencemag.org/content/5/1/eaav2761.abstract.

[78] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances. *arXiv:1306.0895 [stat]*, June 2013. URL http://arxiv.org/abs/1306.0895. arXiv: 1306.0895.

[79] Richard Sinkhorn. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, June 1964. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177703591. URL https://projecteuclid.org/euclid.aoms/1177703591.

[80] Gabriel Peyr and Marco Cuturi. Computational Optimal Transport. *arXiv:1803.00567 [stat]*, March 2018. URL http://arxiv.org/abs/1803.00567. arXiv: 1803.00567.

[81] Aude Genevay, Lnaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyr. Sample Complexity of Sinkhorn divergences. *arXiv:1810.02733 [math, stat]*, October 2018. URL http://arxiv.org/abs/1810.02733. arXiv: 1810.02733.

[82] Jonathan Weed and Francis Bach. Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance. *arXiv:1707.00087 [math, stat]*, June 2017. URL http://arxiv.org/abs/1707.00087. arXiv: 1707.00087.

[83] pyquil: A Python library for quantum programming using Quil, July 2018. URL https://github.com/rigetticomputing/pyquil. original-date: 2017-01-09T21:30:22Z.

[84] BrianCoyle. Implementation of Quantum Ising Born Machine., February 2019. URL https://github.com/BrianCoyle/IsingBornMachine. original-date: 2018-07-10T19:39:22Z.

[85] jeanfeydy. MMD, Hausdorff and Sinkhorn divergences scaled up to 1,000,000 samples.: jeanfeydy/global-divergences, November 2018. URL https://github.com/jeanfeydy/global-divergences. original-date: 2018-08-15T09:29:01Z.

[86] Mohammad H. Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum Boltzmann Machine. *Physical Review X*, 8(2):021050, May 2018. doi: 10.1103/PhysRevX.8.021050. URL https://link.aps.org/doi/10.1103/PhysRevX.8.021050.

[87] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. ISBN 0-521-64298-1.

[88] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

[89] Srinivasan Arunachalam and Ronald de Wolf. A Survey of Quantum Learning Theory. *arXiv:1701.06806 [quant-ph]*, January 2017. URL http://arxiv.org/abs/1701.06806. arXiv: 1701.06806.

[90] Srinivasan Arunachalam, Alex B. Grilo, and Aarthi Sundaram. Quantum hardness of learning shallow classical circuits. *arXiv:1903.02840 [quant-ph]*, March 2019. URL http://arxiv.org/abs/1903.02840. arXiv: 1903.02840.

[91] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, June 2018. ISSN 1745-2481. doi: 10.1038/s41567-018-0124-x. URL https://doi.org/10.1038/s41567-018-0124-x.

[92] Alison L. Gibbs and Francis Edward Su. On Choosing and Bounding Probability Metrics. *International Statistical Review*, 70(3):419–435, 2002. doi: 10.1111/j.1751-5823.2002.tb00178.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2002.tb00178.x.

[93] Dominik Hangleiter, Martin Kliesch, Jens Eisert, and Christian Gogolin. Sample complexity of device-independently certified ”quantum supremacy”. *arXiv:1812.01023 [quant-ph]*, December 2018. URL http://arxiv.org/abs/1812.01023. arXiv: 1812.01023.

[94] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property Testing and Its Connection to Learning and Approximation. *J. ACM*, 45(4):653–750, July 1998. ISSN 0004-5411. doi: 10.1145/285055.285060. URL http://doi.acm.org/10.1145/285055.285060.

[95] Dana Ron. Property Testing: A Learning Theory Perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008. ISSN 1935-8237. doi: 10.1561/2200000004. URL http://dx.doi.org/10.1561/2200000004.

[96] P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. On Universal and Fault-Tolerant Quantum Computing. *arXiv:quant-ph/9906054*, June 1999. URL http://arxiv.org/abs/quant-ph/9906054. arXiv: quant-ph/9906054.

[97] Shakir Mohamed and Balaji Lakshminarayanan. Learning in Implicit Generative Models. *arXiv:1610.03483 [cs, stat]*, October 2016. URL http://arxiv.org/abs/1610.03483. arXiv: 1610.03483.

[98] Peter J. Diggle and Richard J. Gratton. Monte Carlo Methods of Inference for Implicit Statistical Models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227, 1984. ISSN 00359246. URL http://www.jstor.org/stable/2345504.

[99] Yingzhen Li and Richard E. Turner. Gradient Estimators for Implicit Models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJi9WOeRb.

[100] Jiaxin Shi, Shengyang Sun, and Jun Zhu. A Spectral Approach to Gradient Estimation for Implicit Distributions. *arXiv:1806.02925 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1806.02925. arXiv: 1806.02925.

[101] E. J. Nystrm. ber Die Praktische Auflsung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. *Acta Mathematica*, 54:185–204, 1930. doi: 10.1007/BF02547521. URL https://doi.org/10.1007/BF02547521.

A Proof of Theorem 3.1

The proof of Theorem 3.1 involves stitching together several results, some well known, some less so, and, to the best of out knowledge, some unknown. The proof will proceed systematically through the instances of the parameters, $\{J_{ij}, b_k\}, \mathbf{\Gamma}, \mathbf{\Delta}, \mathbf{\Sigma}$, and will show that in each of the cases mentioned in Theorem 3.1 the class of circuits generated cannot, in the worst case, be simulated to within multiplicative error efficiently by any classical means. For the remainder of this proof, we use the phrase 'simulate' to mean exactly this.

A.1 IQP

As stated above, IQP circuits correspond to the setting of the parameters, $\{J_{ij}, b_k\}, \forall k : \Gamma_k = \pi/2\sqrt{2}, \Delta_k = 0, \Sigma_k = \pi/2\sqrt{2}$. This means the unitary $U_z(\theta)$ is applied to the initial superposition state, $|+\rangle^{\otimes n}$, followed finally by a Hadamard applied to all qubits, $H^{\otimes n}$. It is known that IQP circuits, with the homogeneous parameters $\alpha = J_{ij} = b_k = \pi/8\ \forall i, j, k$ in (18), are hard to simulate in the worst case [16]. We will denote IQP circuits with the two sets of parameters used to define them, $\{J_{ij}, b_k\}$ by $\mathsf{IQP}(J_{ij}, b_k)$.

Theorem A.1 (Theorem 2 from [16]). *If the output probability distributions generated by uniform families of* $\mathsf{IQP}(\{\pi/8\},\{\pi/8\})$ *circuits could be weakly classically simulated to within multiplicative error* $1 \leq c \leq \sqrt{2}$ *then* $\mathsf{PostBPP} = \mathsf{PP}$ *(and hence the polynomial hierarchy collapses to the third level).*

A collapse of PH to any level is considered unlikely, and in some sense is a generalisation of $\mathsf{P} \neq \mathsf{NP}$.

Next, we consider the question of for *which* homogeneous parameters, $\boldsymbol{\alpha}_l = \alpha = J_{ij} = b_k$ result in IQP circuit families which are hard to simulate. The answer is almost all of them.

Theorem A.2 (Adapted from Theorem 5 of [31]). *If the output probability distributions generated by uniform families of* $\mathsf{IQP}(\theta,\theta)$ *circuits could be weakly classically simulated to within multiplicative error* $1 \leq c \leq \sqrt{2}$ *then* $\mathsf{PostBPP} = \mathsf{PP}$ *(and hence* PH *collapses to the third level), where*

$$\theta = \begin{cases} \frac{(2l+1)\pi}{8d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{132}$$

Finally we note that there is no need for the circuit parameters to be homogeneous. We may also allow each single and two qubit gate to have an independent parameter, as long as they are *all* of the form (132).

In Theorem A.1 it was shown that a general computation can be simulated by IQP circuits generated by a homogeneous parameter value, $\theta = \pi/8$. We will now use the arguments of [31] to show why $\mathsf{IQP}(J_{ij}, b_k)$ circuits, with almost all parameter angles, can simulate $\mathsf{IQP}(\pi/8, \pi/8)$ efficiently. The result of [31] will be a subcase of this, which accounts for the case $J_{ij} = b_k \neq \pi/8$. Specifically, if the original IQP circuit is generated by gates in the form $D_1(\pi/8) = e^{i\frac{\pi}{8}Z}, D_2(\pi/8) = e^{i\frac{\pi}{8}Z\otimes Z}$, general IQP circuits with gates acting on at most two qubits ($|S_j| \leq 2$) will be generated by gates $D_1(b_k) = e^{ib_kZ}, D_2(J_{ij}) = e^{iJ_{ij}Z\otimes Z}$. Therefore, it is necessary to show how each of these gates can simulate each of the former gates with a homogeneous rotation angle, $\pi/8$. To do so we can use the error measure defined as follows, [37], as the difference between the operations of two arbitrary gates on a quantum state, when maximised over all possible states:

$$E(U,V) = \max_{|\psi\rangle} ||(U-V)\,|\psi\rangle\,|| \tag{133}$$

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 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:

$$D_2(\theta_2)^m = D_2(m\theta_2) = D_2\left(\frac{\pi}{8}+\epsilon\right) \tag{134}$$

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 [31, 37]. The same thing holds for the single qubit gates with angle θ_1 approximating Z or P for example.

The irrationality of the parameter values allows the above, (134), to be true since $2\pi\nu m$ mod 2π is distributed uniformly. See [96] 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. In this construction, each individual gate, j, will have an error, ϵ_j, in contrast to [31] in which all of the errors would be the same (or a constant multiple ($\epsilon = \epsilon_j \forall k$). However, as long as this parameter, ϵ_j, for each gate is lower than the threshold for fault-tolerant quantum computation, as noted in [31], then we can reliably simulate universal quantum computation, and hence $\mathsf{PostIQP}(b_k, J_{ij}) = \mathsf{PostIQP}(\pi/8, \pi/8)$.

A.2 $p = 1$ QAOA

Now, we can prove the analogous statement as with IQP with this setting of the parameters, in an identical way.

1. The case $\forall k : \Gamma_k = -\pi/4$ is covered by [32] where it is shown how $\mathsf{QAOA}(\pi/8, \pi/8, \pi/4)$ equipped with postselection is equivalent to BQP with postselection. The proof involves the design of a gadget similar to the IQP gadget in the IQP proof of hardness, and results in a similar collapse of the PH as in Theorem (A.1).

2. Extending the parameters, $J_{ij} = b_k = \pi/8$ in the diagonal unitary $U_z(\theta)$ to general parameters, J_{ij}, b_k follows identically to the argument for IQP in the previous section. More specifically, any diagonal gate with parameter, $\pi/8$, can be simulated by a gate with any parameter of the form (26), applied a constant number of times to get within a fixed error of the desired $\pi/8$ gate.

3. Finally, if we wish to simulate the behaviour of any gate $\tilde{H} = e^{-i\pi/4X_k}$, by any general gate $U_f(\Gamma_k) = e^{-i\Gamma_k X_k}$ on qubit k, we just need to apply the latter gate a constant, $m = O(1/\epsilon)$, number of times, to get within ϵ, of the gate $\tilde{H}$, exactly as in (134). Now, we will have k such gates, each requiring the application of a constant number of repetitions, $m_k = O(1/\epsilon_k)$, so the total number of gates that would have to be applied is $\prod_{k=1}^{n} m_k$. Overall, we will acquire a polynomial overhead in the simulation of the circuit generated by the fixed parameters, $\forall i, j, k : J_{ij} = \pi/8, b_k = \pi/8, \Gamma_k = -\pi/4$. Hence we can achieve universal quantum computation with postselection in exactly the same way, and hence $\mathsf{PostQAOA}(\{J_{ij}, b_k\}, \mathbf{\Gamma}) = \mathsf{PostBQP}$.

A.3 Remaining Cases

The above two sections, Section A.1, Section A.2, cover many of the angles for the final measurement unitary to be hard. We can immediately extend the argument to cover almost all of the angles. The following case is a generalisation of IQP:

$$\Delta_k = 0, \Gamma_k = \Sigma_k = \begin{cases} \frac{(2l+1)\pi}{2\sqrt{2}d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0, 1) \text{ irrational.} \end{cases} \tag{135}$$

Where the final angle is a rotation in the 'Hadamard' axis, i.e. a rotation around the axis $1/\sqrt{2}(X+Z)$, but by a more general angle than π.

Secondly, when the final measurement unitary is a rotation around the Pauli-Y, i.e. the IBM has the following parameters:

$$\Sigma_k = 0, \Gamma_k = 0, \Delta_k = \begin{cases} \frac{(2l+1)\pi}{4d} & \text{for integers, } d, l \\ 2\nu\pi & \nu \in [0,1) \text{ irrational.} \end{cases} \tag{136}$$

where the final measurement unitary is:

$$U_f(\Delta_k) = R_y(-2\Delta_k) = e^{i\Delta_k Y_k} \tag{137}$$

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

$$p_{\mathsf{IQP}}(\mathbf{z}) = |\langle\mathbf{z}| H^{\otimes n} U_z(\boldsymbol{\alpha}) H^{\otimes n} |0\rangle^{\otimes n}|^2 \tag{138}$$

Let p^* be the output distribution produced by these types of circuits with a final Pauli-Y rotation:

$$p^*(\mathbf{z}') = |\langle\mathbf{z}'| \bigotimes_{i=1}^{n} e^{i\frac{\Delta_i}{2}Y_i} U_z(\boldsymbol{\alpha}) H^{\otimes n} |0\rangle^{\otimes n}|^2 \tag{139}$$

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 $\Delta_k = \pi/4$, we get that the two distributions, (138, 139) are related as follows:

$$p_{\mathsf{IQP}}(\mathbf{z}) = |i^{-n} \langle\mathbf{z}| X^{\otimes n} \sqrt{Y}^{\otimes n} U_z(\boldsymbol{\alpha}) H^{\otimes n} |0\rangle^{\otimes n}|^2 \tag{140}$$

$$= |\langle\mathbf{z}'| \bigotimes_{i=k}^{n} e^{i\frac{\pi}{4}Y_k} U_z(\boldsymbol{\alpha}) H^{\otimes n} |0\rangle^{\otimes n}|^2 = p^*(\mathbf{z}') \tag{141}$$

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 the choice of parameters $\mathsf{IBM}(\boldsymbol{\alpha}, \mathbf{0}, \{\Delta_k = \pi/4\}, \mathbf{0})$ is also classically hard to sample from, where $\boldsymbol{\alpha} = \{b_k, J_{ij}\}$ have the same constraints as in IQP or QAOA. The extension from $\Delta_k = \pi/4$ to almost any general Δ_k, according to (136), follows in an identical fashion from Section A.1, Section A.2, so these circuit classes with postselection are also equal to PostBPP and if they could be simulated, once again the PH collapses.

B Computing the Stein Score Function

In Section 4.4 it was shown that most parts of the Stein Discrepancy can be computed efficiently, even when using the quantum kernel of (40). However, we have not addressed the computability of the score function (67) itself. For every sample, $\mathbf{x} \sim p_\theta$, that we receive from the Born machine we require the score function of that outcome being outputted from the data distribution, $\mathbf{x} \sim \pi$. This involves computing $\pi(\mathbf{x})$, and also $\Delta_{\mathbf{x}}\pi(\mathbf{x})$, i.e. $\pi(\neg_i\mathbf{x}), \forall i \in \{1, \dots, n\}$. In fact, since the score function involves the ratio of these two quantities, it is actually not necessary to compute the probabilities *themselves*. This makes the Stein Discrepancy useful for distributions which are normalised by some intractable normalisation factor, for example in a Boltzmann Machine.

Of course, the Stein Discrepancy can be immediately used for any classical problem (classical datasets) for which the probability density is already known, or can be computed efficiently. However, if one is interested in *implicit* models[20], models which admit a means of sample generation, but not explicit access to the probability density, then this not immediate. In particular, for those distributions which admit some complexity theory result indicating that they cannot be simulated on a classical device efficiently, it will not be possible to efficiently compute the probabilities required in order to compute the score.

Here we present two approaches, [99, 100], to compute approximations to the score function, and their application in this specific circumstance. In all the following, we assume it is the score of the data, π, which we want to compute. Of course, the most obvious approach to computing the score, using (D) samples alone, would be to simply accumulate the empirical distribution which is observed by the samples, $\hat{\pi}(\mathbf{x}^m) = \frac{1}{M}\sum_{m=1}^M \mathbb{I}(\mathbf{x} = \mathbf{x}^m)$ and compute the score from this distribution. However, this immediately has a severe drawback. Since the score for a given outcome, $s_\pi(\mathbf{x}^m)$, requires *also* computing the probabilities of all shifted samples, $\pi(\neg_i \mathbf{x}^m)\forall i$, if we have not seen any of the outcomes $\neg_i \mathbf{x}^m$ in the observed data, we will not have values for these outcomes in the empirical distribution, and hence we cannot compute the score. This would be a major issue as the number of qubits in our system grows, since we will have exponentially many outcomes, many of which we will not see with $poly(n)$ samples.

B.1 Identity Approximation of Stein Score

As a first attempt, we shall try the method of [99]. This involves noticing that the score function appears in Stein Identity, and inverting Stein's Identity gives a procedure to approximate the score.

Of course, we shall need to use the discrete version of Stein's Identity in our case, and rederive the result of [99] but there are no major differences. If we have generated M samples from the data distribution, we denote the score matrix, G, at each of those sample points as follows:

$$G^\pi = \begin{pmatrix} \mathbf{s}_\pi^1(\mathbf{x}^1) & \mathbf{s}_\pi^1(\mathbf{x}^2) & \dots & \mathbf{s}_\pi^1(\mathbf{x}^M) \\ \mathbf{s}_\pi^2(\mathbf{x}^1) & \mathbf{s}_\pi^2(\mathbf{x}^2) & \dots & \mathbf{s}_\pi^2(\mathbf{x}^M) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{s}_\pi^n(\mathbf{x}^1) & \mathbf{s}_\pi^n(\mathbf{x}^2) & \dots & \mathbf{s}_\pi^n(\mathbf{x}^M) \end{pmatrix} \qquad G^\pi_{i,j} = \mathbf{s}_\pi^i(\mathbf{x}^j) = \frac{\Delta_{x_i^j}\pi(\mathbf{x}^j)}{\pi(\mathbf{x}^j)} \tag{142}$$

Each column is the term which corresponds to the score function for the distribution, π, and that given sample.

Now, to compute, $\hat{G}^\pi \approx G^\pi$ we can invert the discrete version of Stein's Identity, (143), similar to [99] which does this in the continuous case:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\pi}[\mathbf{s}_\pi(\mathbf{x})\mathbf{f}(\mathbf{x})^T - \Delta\mathbf{f}(\mathbf{x})] = \mathbf{0} \tag{143}$$

[20]Implicit models are also present in the classical domain, for example in Generative Adversarial Networks, [97, 98], and it is of great interest to find methods of dealing with them.

Rearranging (143) in terms of the score function, and following [99]:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim\pi}[\mathbf{s}_\pi \mathbf{f}(\mathbf{x})^T] = \mathop{\mathbb{E}}_{\mathbf{x}\sim\pi}[\Delta \mathbf{f}(\mathbf{x})] \tag{144}$$

$$\implies \sum_{\mathbf{x}} \pi(\mathbf{x})\mathbf{s}_\pi(\mathbf{x})\mathbf{f}(\mathbf{x})^T = \sum_{\mathbf{x}} \pi(\mathbf{x})\Delta\mathbf{f}(\mathbf{x}) \tag{145}$$

Approximated with M samples, from the data distribution, $\pi(\mathbf{x})$:

$$\implies \frac{1}{M}\sum_{i=1}^{M} \mathbf{s}_\pi(\mathbf{x}^i)\mathbf{f}(\mathbf{x}^i)^T \approx \frac{1}{M}\sum_{i=1}^{M} \Delta_{\mathbf{x}^i}\mathbf{f}(\mathbf{x}^i) \tag{146}$$

Defining:

$$F = [\mathbf{f}(\mathbf{x}^1), \mathbf{f}(\mathbf{x}^2), \dots, \mathbf{f}(\mathbf{x}^M)]^T, \qquad \hat{G}^\pi = [\mathbf{s}^\pi(\mathbf{x}^1), \mathbf{s}^\pi(\mathbf{x}^2), \dots, \mathbf{s}^\pi(\mathbf{x}^M))]^T, \tag{147}$$

$$\overline{\Delta_{\mathbf{x}}\mathbf{f}} = \frac{1}{D}\sum_{i=1}^{D} \Delta_{\mathbf{x}^i}\mathbf{f}(\mathbf{x}^i), \qquad \Delta_{\mathbf{x}^i}\mathbf{f}(\mathbf{x}^i) = [\Delta_{\mathbf{x}^i} f_1(\mathbf{x}^i), \Delta_{\mathbf{x}^i} f_l(\mathbf{x}^i)]^T \tag{148}$$

Now the optimal value for the approximate Stein Matrix, $\hat{G}^\pi$ will be the solution to the following ridge regression problem, and adding a regularisation term, with parameter, η, to avoid the matrix being non-singular:

$$\hat{G}^\pi = \mathop{\arg\min}_{\hat{G}^\pi \in \mathbb{R}^{D\times n}} ||\overline{\Delta_{\mathbf{x}}\mathbf{f}} - \frac{1}{M}F\hat{G}^\pi||_F^2 + \frac{\eta}{M^2}||\hat{G}^\pi||_F^2 \tag{149}$$

Where $||\cdot||_F$ is the Frobenius norm: $||A||_F = \sqrt{\mathsf{tr}\,(A^T A)}$. The analytic solution of this ridge regression problem is well known and can be found by differentiating the above (149) with respect to $\hat{G}^\pi$ and setting to zero:

$$\hat{G}^\pi = M(K + \eta\mathbf{1})^{-1}F^T\overline{\Delta_{\mathbf{x}}\mathbf{f}} \tag{150}$$

$$\hat{G}^\pi = M(K + \eta\mathbf{1})^{-1}\langle\Delta, K\rangle \tag{151}$$

Now the method of [99] involves implicitly setting the test function to be a feature map in a RKHS, $\mathbf{f} = \Phi$. If this is the case we get $K = F^T F$, and also $\langle\Delta, K\rangle_{ab} = \frac{1}{M}\sum_{i=1}^{M} \Delta_{x_b^i}\kappa(\mathbf{x}^a, \mathbf{x}^i)$. Unfortunately, there is no motivation given in [99] for which choice of feature map should be used to compute (151). For example, we could use the quantum feature map of (40), the mixture of Gaussians kernel, (36), or the exponentiated Hamming Kernel, which is suggested as a sensible kernel to use in (binary) discrete spaces by [70]:

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-H(\mathbf{x}, \mathbf{y})\right) \tag{152}$$

$$H(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i| \tag{153}$$

Any of these kernels could be used, since the only requirement on the above method is that the feature map obeys the discrete Stein Identity, which we have seen is *any* complex vector valued function.

B.2 Spectral Approximation of Stein Score

While the method used to approximate the score function method which we showed in Section B.1 is straightforward, it does not give a method of computing the score accurately at sample points which have *not* been seen in the data distribution, π. This is a problem if, for instance, we come across a sample from the Born Machine, which has not been seen in the data set. Again, this becomes exponentially more likely as the number of qubits grows. If this were to occur during training, a possible solution mentioned in [99], is simply to add that sample to the sample set, and recompute the score function by the method of Section B.1. However, this is expensive, so more streamlined approaches would be desirable. Worse still, this tactic would potentially introduce bias to the data, since there is no guarantee that the given sample from the Born machine, does not have zero probability in the true data, and hence would *never* occur.

The approach we take is that of [100], which uses the Nyström method as a subroutine to approximate the score. The Nyström method is a technique to approximately solve integral equations [101]. It works by finding eigenfunctions of a given kernel with respect to the target probability mass function, π. As in the case of [99], the method was defined when π is a continuous probability measure, and as such we must make suitable alterations to adapt it to the discrete setting.

We summarise the parts of [100] which are necessary in the discretisation. For the most part the derivation follows cleanly from [100], and from Section B.1. Firstly, the eigenfunctions in question are given by the following summation equation:

$$\sum_{\mathbf{y}} \kappa(\mathbf{x}, \mathbf{y})\psi_j(\mathbf{y})\pi(\mathbf{y}) = \mu\psi_j(\mathbf{x}) \tag{154}$$

where $\{\psi_j\}_{j=1}^N \in \ell^2(\mathcal{X}, \pi)$, and $\ell^2(\mathcal{X}, \pi)$ is the space of all square-summable sequences with respect to π, over the discrete sample space, $\mathcal{X}$. If the kernel is a quantum one, as in (40), the feature space has a basis, $\{\psi_j = \langle s_j|\psi\rangle\}_{j=1}^N \in \ell^2(\mathcal{X}, \pi)$, where $|s_j\rangle$ are for example computational basis states. We also have the constraint that these functions are orthonormal under the discrete π:

$$\sum_{\mathbf{x}} \psi_i(\mathbf{x})\psi_j(\mathbf{x})\pi(\mathbf{x}) = \delta_{ij} \tag{155}$$

Approximating (154) by a Monte-Carlo estimate drawn with M samples, and finding the eigenvalues and eigenvectors of the covariance kernel matrix, $K_{ij} = \kappa(\mathbf{x}^i, \mathbf{y}^j)$, in terms of the approximate ones given by the Monte-Carlo estimate exactly, as in [100], we get:

$$\psi_j(\mathbf{x}) \approx \hat{\psi}_j(\mathbf{x}) = \frac{\sqrt{M}}{\lambda_i} \sum_{m=1}^{M} u_j(\mathbf{x}^m)\kappa(\mathbf{x}, \mathbf{x}^m) \tag{156}$$

$\{u_j\}_{j=1,\dots J}$ are the J^{th} largest eigenvalues of the kernel matrix, K, with eigenvalues, λ_j. The true eigenfunctions are related to these 'sampled' versions by: $\psi_j(\mathbf{x}^m) \approx \sqrt{M}u_{jm}\forall m \in \{1, \dots, M\}, \mu_j \approx \lambda_j/M$.

Assuming [21] that the discrete score functions are square summable with respect to π, i.e. $s^i(\mathbf{x}) \in \ell^2(\mathcal{X}, \pi)$, we can be expand the score in terms of the eigenfunctions of the $\ell^2(\mathcal{X}, \pi)$:

$$s^i(\mathbf{x}) = \sum_{j=1}^{N} \beta_{ij}\psi_j(\mathbf{x}) \tag{157}$$

[21]

Because the eigenfunctions, ψ_j are complex valued, they automatically obey the Discrete Stein's Identity (70) and we get the same result as [100]:

$$\beta_{ij} = -\mathbb{E}_\pi \Delta_{x_i} \psi_j(\mathbf{x}) \tag{158}$$

Proceeding as in [100], we apply the discrete shift operator, Δ_{x_i}, to both sides of (154) to give an approximation for the term, $\hat{\Delta}_{x_i}\psi(\mathbf{x}) \approx \Delta_{x_i}\psi(\mathbf{x})$:

$$\hat{\Delta}_{x_i}\psi(\mathbf{x}) = \frac{1}{\mu_j M}\sum_{m=1}^{D} \Delta_{x_i}\kappa(\mathbf{x}, \mathbf{x}^m) \tag{159}$$

It can also be shown in this case that $\hat{\Delta}_{x_i}\psi(\mathbf{x}) \approx \Delta_{x_i}\hat{\psi}(\mathbf{x})$, by comparing (159) with (156), and hence we arrive at the estimator for the score function:

$$\hat{s}^i(\mathbf{x}) = \sum_{j=1}^{J} \hat{\beta}_{ij}\hat{\psi}_j(\mathbf{x})\hat{\beta}_{ij} = -\frac{1}{M}\Delta_{x_i}\hat{\Psi}_j(\mathbf{x}^m) \tag{160}$$

If the sample space is the space of binary strings of length n, the number of eigenfunctions, N will be exponentially large, $N = 2^n$, and so the sum in (157) is truncated to only include the J^{th} largest eigenvalues and corresponding eigenvectors.

C Proof of Theorem 4.3

The proof of Theorem 4.3, and in particular (70), is a simple extension of Theorem 2 in [70]

Proof.

$$\begin{aligned}
\phi(\mathbf{x}) &= a(\mathbf{x}) + ib(\mathbf{x}) && (161)\\
\mathbb{E}[\mathcal{A}_p\phi(\mathbf{x})] &= \sum_{\mathbf{x}\in\mathcal{X}^d} [\phi(\mathbf{x})\Delta p(\mathbf{x}) - \Delta^*\phi(\mathbf{x})] && (162)\\
&= \sum_{\mathbf{x}\in\mathcal{X}^k} [a(\mathbf{x})\Delta p(\mathbf{x}) - p(\mathbf{x})\Delta^* a(\mathbf{x})] + i\sum_{\mathbf{x}\in\mathcal{X}^d} [b(\mathbf{x})\Delta p(\mathbf{x}) - p(\mathbf{x})\Delta^* b(\mathbf{x})] && (163)
\end{aligned}$$

For each term, j, the real part is given by:

$$\sum_{\mathbf{x}\in\mathcal{X}^d} a(\mathbf{x})\Delta_{x_j}p(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{X}^d} a(\mathbf{x})p(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}^d} a(\mathbf{x})p(\neg_j\mathbf{x}) \tag{164}$$

$$\sum_{\mathbf{x}\in\mathcal{X}^d} p(\mathbf{x})\Delta^*_{x_j}a(\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{X}^d} p(\mathbf{x})a(\mathbf{x}) - \sum_{\mathbf{x}\in\mathcal{X}^d} p(\mathbf{x})a(\lrcorner_j\,\mathbf{x}) \tag{165}$$

Since the sum is taken over all the sample space, $\mathcal{X}^d$, and $\neg_i(\lrcorner_i\,\mathbf{x}) = \mathbf{x}$, i.e. $\neg$ and $\lrcorner$ are inverse operations (in fact they are equal in our case), (164) is equal to (165), and so the real parts of (163) cancel out. The same result hold for the imaginary parts, completing the proof.

□

D Proof of (76)

Here we compute the gradient of the Stein Cost function of (76), between the Born Machine, p_θ and the data distribution, π, given by (74).

Proof. The derivation is very similar to the derivation of the MMD cost function gradient. Using the fact that the Stein kernel, (75), does not depend on the parameter, θ, the gradient is given by:

$$\begin{aligned}
\frac{\partial \mathcal{L}^\theta_{\mathsf{SD}}}{\partial \theta_k} &= \sum_{\mathbf{x},\mathbf{y}} \frac{\partial p_\theta(\mathbf{x})}{\partial \theta_k} \kappa_\pi(\mathbf{x},\mathbf{y}) p_\theta(\mathbf{y}) + \sum_{\mathbf{x},\mathbf{y}} p_\theta(\mathbf{x}) \kappa_\pi(\mathbf{x},\mathbf{y}) \frac{\partial p_\theta(\mathbf{y})}{\partial \theta_k} &(166)\\
&= \sum_{\mathbf{x},\mathbf{y}} p_\theta^-(\mathbf{x}) \kappa_\pi(\mathbf{x},\mathbf{y}) p_\theta(\mathbf{y}) - \sum_{\mathbf{x},\mathbf{y}} p_\theta^+(\mathbf{x}) \kappa_\pi(\mathbf{x},\mathbf{y}) p_\theta(\mathbf{y}) &(167)\\
&+ \sum_{\mathbf{x},\mathbf{y}} p_\theta(\mathbf{x}) \kappa_\pi(\mathbf{x},\mathbf{y}) p_\theta^-(\mathbf{y}) - \sum_{\mathbf{x},\mathbf{y}} p_\theta(\mathbf{x}) \kappa_\pi(\mathbf{x},\mathbf{y}) p_\theta^+(\mathbf{y}) &(168)\\
&= \underset{\substack{\mathbf{x}\sim p_\theta^- \\ \mathbf{y}\sim p_\theta}}{\mathbb{E}} [\kappa_\pi(\mathbf{x},\mathbf{y})] - \underset{\substack{\mathbf{x}\sim p_\theta^+ \\ \mathbf{y}\sim p_\theta}}{\mathbb{E}} [\kappa_\pi(\mathbf{x},\mathbf{y})] + \underset{\substack{\mathbf{x}\sim p_\theta \\ \mathbf{y}\sim p_\theta^-}}{\mathbb{E}} [\kappa_\pi(\mathbf{x},\mathbf{y})] - \underset{\substack{\mathbf{x}\sim p_\theta \\ \mathbf{y}\sim p_\theta^+}}{\mathbb{E}} [\kappa_\pi(\mathbf{x},\mathbf{y})] &(169)
\end{aligned}$$

We have used (60) for the Ising Born Machine Gradient, [8, 30, 67] and that the Stein kernel, κ_π, of (75) does not depend on the parameter, θ_k.

□