Quantum methods for machine learning and classical dynamics

Barthe, A.M.

CHAPTER 5

Quantum Advantage in Learning Quantum Dynamics

5.1. Introduction

Identifying when quantum computers provide an advantage in learning tasks is a central challenge of quantum machine learning. Problems involving quantum systems are natural candidates, but many of their associated learning tasks can still be solved efficiently by classical computers. For instance, [153] shows that predicting ground state properties within a phase is classically tractable with limited training data. Nonetheless, the prevailing intuition is that access to a device that can efficiently simulate the target quantum system should be advantageous. While earlier works identified contrived problems exhibiting such separation [21, 22], identifying quantum-classical learning separations in natural settings remains a compelling open question.

We address this challenge by considering the supervised learning problem of *learning unknown Hamiltonian dynamics* from classical data. Specifically, we define a family of functions $f_\alpha(x) = \langle\psi(x,\alpha)|O|\psi(x,\alpha)\rangle$, where $|\psi(x,\alpha)\rangle = e^{iH(x,\alpha)t}|\psi_0\rangle$ is a time-evolved quantum state under a pa-

The contents of this chapter have been published in [37].

rameterized Hamiltonian $H(x, \alpha)$ for some fixed time t. The task is to learn f_α (in the PAC sense, which we explain shortly) from a dataset $\mathcal{D} = \{(x_j, f_\alpha(x_j))\}_j$ for some fixed, unknown α. Our settings assume a more restricted (and maybe more realistic) access to the dynamics than found in related works in the literature [154, 155], in particular excluding the use of Hamiltonian learning methods.

Our contributions can be summarized as follows. We introduce a quantum subroutine to extract the Fourier decomposition of PQC-based functions, which we call *Fourier coefficient extraction* subroutine. We then use this subroutine to define a quantum learning algorithm to solve the unknown Hamiltonian dynamics problem, which we formalize as the Hamiltonian dynamics concept class (defined shortly) in the probably approximately correct (PAC) framework. In our approach, we use the fact that the quantum dynamics class can be approximated via Hamiltonian simulation by a class of functions built around parameterized quantum circuits, which we call the PQC-based functions concept class. The latter class will serve as the effective hypothesis family of our learning algorithm. Our learning algorithm is then proven to be correct for both concept classes. The algorithm is efficient when the number of unknown parameters (of the PQC or Hamiltonian) scales logarithmically with the system size. We also prove that no classical algorithm can solve this learning task under complexity-theoretic assumptions, yielding a separation for this natural problem. We also consider the much more general setting of polynomially many unknown parameters. For this case, we analyze the potential and limitations of generalization of our method for this broader class of quantum dynamics problems. We identify conditional no-gos for provably efficient learners, but also propose a heuristic method for the problem, which may work in cases beyond what can be proven analytically.

This chapter is structured as follows. Section 5.2 introduces the *Fourier coefficient extraction* algorithm. Section 5.3 applies this to learn PQCs, and Section 5.4 extends this to Hamiltonian evolution. Section 5.6 discusses limitations and heuristic extensions. We finish this chapter with a brief conclusion section in Section 5.6.2.

5.1.1. Parameterized Quantum circuits

Parameterized quantum circuits, and in particular variational methods [4], have been at the centre of recent approaches to quantum machine learning and were extensively studied [156]. Here, we focus on parameterized quantum circuits where the inputs and other tunable parameters appear repeatedly as Pauli rotations.

Definition 5.1 (Pauli encoding)
A Pauli-encoded circuit is a parametrized quantum circuit on n qubits $U : \alpha \in [0,1]^d \to \mathcal{U}(2^n)$. It is composed of $N_f \in \text{poly}(n)$ fixed unitary gates. There are d parameters, each reuploaded L times, such that there are $dL \in \text{poly}(n)$ parametrized gates:

$$\{V_{s,j}(\alpha) \coloneqq e^{i\pi P_{s,j}\alpha_j}\}_{(s,j)\in\{1,\cdots,L\}\times\{1,\cdots,d\}}$$

where $P_{s,j}$ are Pauli strings.

It has been shown that such Pauli Quantum Circuits admit a finite Fourier representation [26].

Lemma 5.1
Any circuit U as defined in Definition 5.1 admits a finite Fourier representation with frequencies included in $\mathcal{L}' = l \in \{-L,\cdots,+L\}^d$ as follow:

$$|\phi(\alpha)\rangle = U(\alpha)\,|0\rangle \tag{5.1}$$

$$= \sum_{k\in\{0,1\}^n} \sum_{l\in\mathcal{L}'} a_{l,k} e^{i\pi\alpha\cdot l}\,|k\rangle\ . \tag{5.2}$$

Measuring the expectation value of some observable O for such a state results in what we call a *PQC function*, an input-output mapping as follows,

$$f(\alpha) = \langle 0|\,U^\dagger(\alpha) O U(\alpha)\,|0\rangle\ . \tag{5.3}$$

In the case of quantum reuploading models with Pauli encodings as in Definition 5.1, we have that

$$f(\alpha) = \sum_{l\in\{-2L,\cdots,2L\}^d} b_l e^{i\pi\alpha\cdot l}\,. \tag{5.4}$$

We call the coefficients b_l the *Fourier coefficients* of the *PQC function* f.

In the next section, we provide a subroutine for sampling from them. Building on this subroutine, we provide a sampling-based algorithm for their estimation.

5.2. Fourier coefficient extraction algorithm

5.2.1. Fourier representation of parameterized circuits

For our quantum learning algorithm, we introduce a new subroutine that allows us to prepare a state that amplitude-encodes the coefficients b_l

of a PQC function f, and describe a sampling-based method for their extraction. First we observe that the amplitudes of the output state of a PQC function admit a finite Fourier representation, as in Lemma 5.2.

Lemma 5.2
Any circuit U as defined in Definition 2.4 admits a finite Fourier representation as follows:

$$|\phi(x)\rangle = U(x)\,|0\rangle = \sum_{k\in\{0,1\}^n} \sum_{l\in[-L,+L]^D} a_{l,k} e^{i\pi x\cdot l}\,|k\rangle\ . \tag{5.5}$$

As we explain in Section 5.7.5, it is possible to estimate the coefficients given just appropriate black-box access to the classical function f. However, here we assume access to the gate-decomposition of the circuit, which yields a more elegant and significantly more gate-frugal method. Based on this gate decomposition, we propose an algorithm that transforms the description of this circuit to the description of a circuit with an additional register for frequencies.

Theorem 5.1
There exists an algorithm $\mathcal{A}$ that given the description of any parameterized circuit U as defined in Definition 5.1, returns the description of a non-parameterized quantum circuit $U' = \mathcal{A}(U)$ on n' qubits with L' gates, such that it prepares the Fourier representation state of U as follows:

$$|\phi'\rangle = \mathcal{A}(U)\,|0\rangle = \sum_{k\in\{0,1\}^n} \sum_{l\in\mathcal{L}'} a_{l,k}\,|l\rangle\,|k\rangle\ . \tag{5.6}$$

with $n' = n + d\lceil\log(2L+1)\rceil + 1$ qubits and $L' = N_f + L(2n + d\lceil\log(2L+1)\rceil)$ gates.

The detailed algorithm and the proof for this theorem are provided in the Section 5.7.3. We provide a high-level explanation of this algorithm. The fixed gates (the ones without data uploading) are left unchanged by the algorithm $\mathcal{A}$. For the reuploading gates, we note that any data Pauli-uploading gates can be transformed into a Z string by adding appropriate basis change gates. The algorithm $\mathcal{A}$ transforms a data-encoding gate with a Z Pauli on the bitstring x into an increment (decrement) gate on the frequency register controlled on the even (odd) parity of x. This is illustrated in Figure 5.1.

5.2.2. Fourier representation of expectation values

As seen in Lemma 5.1, when the inputs α are Pauli-encoded, the PQC function has a finite Fourier decomposition. We can connect Theorem 5.1 with this representation to obtain a sampling algorithm for the coefficients b_l as given in Equation (5.4). First, we construct a circuit that returns the Fourier decomposition of such a quantum function amplitude encoded on a quantum state.

The key to do so is to realise that as $f(\alpha) = \langle 0| U(\alpha)^\dagger P U(\alpha) |0\rangle$, applying the algorithm $\mathcal{A}$ to the state $U(\alpha)^\dagger P U(\alpha) |0\rangle$ and then post-selecting the all-zero state yields the Fourier decomposition of the quantum function f as a quantum state. The full proof is in the Section 5.7.3.

Corollary 5.1
For every circuit U as defined in Definition 5.1 and Pauli observable P, we define the quantum function f:

$$\alpha \in \mathbb{R}^d \xrightarrow{f} \langle 0| U(\alpha)^\dagger P U(\alpha) |0\rangle \ . \tag{5.7}$$

f has a finite Fourier representation, there exists a vector $b \in \mathbb{C}^{d(4L+1)}$ indexed by $l \in \mathcal{L} \coloneqq \{-2L, \cdots, +2L\}^d$ such that

$$f(\alpha) = \sum_{l \in \mathcal{L}} b_l e^{i\pi\alpha \cdot l} \, . \tag{5.8}$$

Then there is a quantum algorithm $\mathcal{A}$ with complexity $O(\mathrm{poly}(n, d))$ that retrieves the state amplitude encoding of the Fourier coefficients of f on the $|0\rangle$ subspace.

$$\mathcal{A}(U) |0\rangle |0\rangle = \frac{1}{\|b\|_2} \sum_{l \in \mathcal{L}} b_l |l\rangle |0\rangle + |\cdots\rangle \left|0^\perp\right\rangle \ . \tag{5.9}$$

Note that the probabilistic component here is unavoidable, as in general there is no reason for the function f to have a Fourier decomposition which is a unit vector. Note that the theorem above is specific to observables that are Pauli strings P. In the Section 5.7.3, we prove that this result can efficiently be extended to a broader range of observables O, such as linear combinations of Pauli terms with a polynomial number of terms and polynomial spectral norm and local projectors.

Using $\mathcal{A}$ as a subroutine, we can perform *Fourier coefficient extraction*, i.e., extract any coefficient b_l up to additive error, by using controlled versions of $\mathcal{A}(U)$ in a Hadamard test.

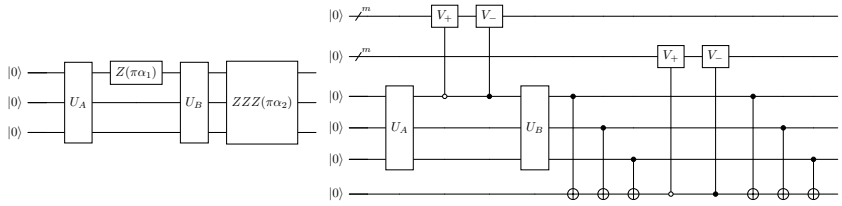

Figure 5.1.: (left) A parameterized circuit $U(\alpha)$ yielding a state $\sum_k \sum_l a_{l,k} e^{i\pi l\cdot\alpha} |k\rangle$. (right) The corresponding circuit $\mathcal{A}(U)$ returning a state amplitude-encoding Fourier coefficients as $\sum_k \sum_l a_{l,k} |l\rangle |k\rangle$
. .

Corollary 5.2
With the same premise as in Corollary 5.1, there exists a $\mathrm{poly}(n, \varepsilon^{-1})$ *quantum algorithm that retrieves* b_l *up to additive error* ε *for every* l*.*

Furthermore, the Fourier amplitude-encoded state realizes a particular quantum feature map, which can be used to construct kernel-based machine learning methods even when the spectrum is exponentially large.

5.3. Learning Parameterized Quantum Circuits

In this section, we study a concept class based on parameterized quantum circuits, which we call the PQC-based functions concept class. Then we describe an efficient quantum learner, based on the *Fourier coefficient extraction* algorithm proposed in the previous section. Finally, we show that this concept class is hard to learn classically, effectively proving a learning separation.

5.3.1. Concept Class Definition

We consider a family of parameterized circuits with Pauli encodings, as in Definition 5.1. We define x as the bitstring describing the fixed gates of the circuit. The parameterized gates have a known nature (P_l) but are parameterized by unknown parameters (α_l). This gives rise to the following concept class, which we call *PQC-based functions*.

Definition 5.2 (PQC-based functions concept class)
Consider a parameterized circuit U *on* n *qubits as defined in Definition 5.1,*

we note as x the bit string describing the fixed gates, and $\alpha \in [0,1]^d$ the input parameters of the circuit. We have:

$$U(x,\alpha)\left|0\right\rangle = \sum_{k\in\{0,1\}^n} \sum_{l\in\mathcal{L}'} a_{l,k}(x) e^{i\pi l\cdot\alpha} \left|k\right\rangle . \tag{5.10}$$

For a given observable O, we define the concept class $\mathcal{C}^{PQC}_{n,d}$, for a qubit number n and a number of unknown parameters d, which may scale with n.

$$\mathcal{C}^{PQC}_{n,d} \coloneqq \{c_\alpha : x \in \{0,1\}^n \to \left\langle 0\right| U(x,\alpha)^\dagger O U(x,\alpha) \left|0\right\rangle\}_{\alpha\in[0,1]^d} . \tag{5.11}$$

This concept class has been studied, in particular, its covering number has been formally upper-bounded [157], yielding a good generalization performance. As a consequence, as long as $d \in \text{poly}(n)$, this class is PAC-learnable. Here we will be focusing on the possibility and impossibility of *efficient* PAC learning.

5.3.2. Efficient Quantum Learner

We consider the concept class as defined in Definition 5.2. In this subsection, we aim to prove that $\mathcal{C}^{PQC}_{n,\log n}$ is quantum efficiently PAC learnable as in Definition 2.2. To do so, we present an efficient quantum learning algorithm that uses the Fourier sampling procedure described in Section 5.2.

We are given a T sized dataset $\{x_t, y_t\}_{t\in\{1,\cdots,T\}}$ for an unknown fixed concept c_α for data sampled over $\mathcal{D}$, a maximum probability of failure as δ and the required accuracy as ε. The problem size is the number of qubits n, and we fix the number of parameters as $d = \log(n)$, each uploaded a constant number of times L. This setup yields a number of frequencies $m = |\mathcal{L}| = (4L+1)^d \in \Theta(\text{poly}(n))$. We describe the training and inference stage of the proposed algorithm.

Training: First we choose the measurement accuracy ε_b for the Fourier coefficients. For each Fourier coefficient, a simple sample average strategy is used, requiring ε_b^{-2} shots (a quadratic improvement may be possible here). For each bitstring x_t specifying a circuit and for each frequency l, the coefficient $b_l(x_t)$ is retrieved using Corollary 5.2 up to additive error ε_b. This procedure requires the execution of mT/ε_b^2 poly-depth quantum circuits. The retrieved Fourier coefficients are stacked in the matrix $\hat{B}$ where the rows correspond to the frequencies indexed by l and

the columns to the samples indexed by t. Similarly, we define the exact Fourier coefficients as B and the shot noise $E = B - \hat{B}$.

We have that $y = b(x) \cdot w(\alpha)$, where w is an unknown weight vector with $w_l(\alpha) \coloneqq e^{i\pi l \cdot \alpha}$. In other words, the concept is linear with respect to B, and stacking all labels in a vector Y, we have $Bw = Y$. Our goal is to minimize the mean square error on unseen data. We look for a hypothesis function h that minimizes the error defined as.

$$\mathcal{L}_c(h) = \mathbb{E}_{x\sim D}[|h(x) - c_\alpha(x)|^2]\,. \tag{5.12}$$

We use a regression technique (LASSO, see details in Section 5.7.1) that yields a mean square error minimizer, with an additional constraint on the 1-norm of the weight vectors, which should be upper bounded by Λ_1. Minimizers returned by this regression technique belong to the following set of functions, which we define as our hypothesis class:

$$\mathcal{H} = \{b \xrightarrow{h_w} b \cdot w \mid \|w\|_1 \leq \Lambda_1\} \tag{5.13}$$

Using Theorem 5.6 with $\Lambda_1 = \|w\|_1 = m$ and $r_\infty = 1$ we know that in order to reach ε accuracy on the mean square error with $1-\delta$ accuracy we can choose the parameters as follows, (noting that $\varepsilon_y = 0$).

$$T = \frac{16m^4\sqrt{2\log\left(\frac{2m}{\delta}\right)}}{\varepsilon^2},\ \varepsilon_b = \frac{0.2\varepsilon}{m} \tag{5.14}$$

We note that $T \in \tilde{\Theta}(\text{poly}(n, \varepsilon^{-1}, \delta^{-1}))$. In addition, the training stage of our algorithm outputs a weight vector $\hat{w}$ executing of $\tilde{\Theta}(m^7/\varepsilon^4) \subset \tilde{\Theta}(\text{poly}(n, \varepsilon^{-1}, \delta^{-1}))$ poly-depth quantum circuits. Therefore, both the number of samples and the runtime required to output a model are polynomial in n, ε^{-1}, and δ^{-1}.

Inference: Given a new datapoint $x_{t'}$, one retrieves the Fourier coefficients $b_{t'}$ using the algorithm in Section 5.2 execution of $m/\varepsilon_b^2 \in \tilde{\Theta}(m^3/\varepsilon^2) \subset \tilde{\Theta}(\text{poly}(n, \varepsilon^{-1}, \delta^{-1}))$ poly-depth circuits. Then using the weight vector $\hat{w}$ derived in the training phase, the model returns $\hat{y}_{t'} = \hat{w} \cdot \hat{b}(x_{t'})$. Theorem 5.6 guarantees that $|\hat{y}_{t'} - y_t| \leq \varepsilon$ with probability $1-\delta$. The runtime of the model is polynomial in n, ε^{-1} and δ^{-1}.

This concludes the proof of our first main result, which we summarize in the following theorem.

Theorem 5.2
The PQC-based functions concept class $\mathcal{C}_{\log}^{PQC}$ as in Definition 5.2 is effi-

ciently quantum PAC learnable as in Definition 2.2.

5.3.3. Hardness of the learning problem

In this subsection, we prove that the PQC-based functions concept class is not classically efficient PAC learnable unless $\mathsf{BQP} \subset \mathsf{P/poly}$, for more on this class see Section 5.7.2.

We choose a promise BQP language $\mathcal{L}$, solved by $U \in \mathcal{U}(2^n)$, meaning that the sign of $\mathrm{Tr}\left[OU\left|x\right\rangle\!\left\langle x\right|U^\dagger\right]$ decides $\mathcal{L}$, where O is some simple observable. We define the concept class as in Definition 5.2 as the circuit U with added parameterized gates P_l at fixed locations with unknown parameters α_l.

The concept class contains at least one concept that is BQP-hard, for $\alpha = 0$. Lemma 2 in [22] (originally found in [66]) states that the efficient learnability of a concept class implies efficient evaluation of all concepts. However, unless $\mathsf{BQP} \subset \mathsf{P/poly}$, no classical algorithm can execute c_0 as it is BQP-hard. Therefore the concept class is not classically learnable if Conjecture 5.1 is true. This yields the following theorem.

Theorem 5.3
If $\mathsf{BQP} \not\subset \mathsf{P/poly}$ *then the PQC-based functions concept class Definition 5.2 is not classically efficient PAC learnable as in Definition 2.2.*

Theorem 5.3 and Theorem 5.2 together prove a separation between quantum and classical learners for the PQC-based functions problem. However, this case is rather contrived and does not have any direct physical application. In the next section, we adapt this result to when a circuit compiles a time evolution, which yields a more practical learning problem.

5.4. Learning Time Evolution

5.4.1. Concept Class Definition

Our main result is a learning separation for the Hamiltonian dynamics learning problem.

We consider a parameterized Hamiltonian $H(x, \alpha)$ the input is $x \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}^d$ are unknown parameters. We are interested in its time evolution for a fixed time τ, as $U_n(x, \alpha) = e^{i\tau(H(x,\alpha))}$.

This gives rise to the following concept class, which we call *Hamiltonian dynamics*.

Definition 5.3 (Hamiltonian dynamics concept class)
Consider a sequence of parameterized Hamiltonian $\{H_n(x,\alpha)\}_n$*, where each* H_n *operates on* n *qubits and is described by continuous parameters* $\alpha \in [0,1]_n^d$ *and bitstrings* x *of length* s_n*. For a given sequence of observables* $\{O_n\}_n$*, we define the concept class* $\mathcal{C}_{n,d}^H$*, for* n *qubits,* d *parameters that may scale with* n*, and a fixed real number* τ *, resulting in a time evolution* $U_n(x,\alpha) = e^{i\tau H_n(x,\alpha)}$*, as follows,*

$$\mathcal{C}_{n,d}^H \coloneqq \{c_\alpha : x \in \{0,1\}^{s_n} \to \langle 0| U_n(x,\alpha)^\dagger O U_n(x,\alpha) |0\rangle\}_{\alpha\in[0,1]^d}. \quad (5.15)$$

5.4.2. Connection between the two concept classes

The Hamiltonian dynamics concept class can be related to the PQC-based functions concept class via Hamiltonian simulation. Given the class $\mathcal{C}_{n,\log n}^H$, by using Hamiltonian simulation on the underlying Hamiltonians (taking into account the parametrizations), we obtain parameterized circuits. The precision we use in Hamiltonian simulation dictates how closely functions in one class approximate the functions in the other in a precisely quantified way, and at the same time, the depth of the parameterized circuit.

We illustrate this with an example. Consider the Ising Hamiltonian on an arbitrary graph with a transverse field of unknown strength α. The arbitrary graph is described by bitstrings indicating whether an edge exists or not, $x_{i,j} = 1$ if $(i,j) \in E$.

$$H(x,\alpha) = \sum_{i,j} x_{i,j} Z_i Z_j + \alpha \sum_i X_i \quad (5.16)$$

The first order Trotterization of r steps, yields the following parameterized quantum circuit for the parameter $\alpha\tau/r$.

$$U_r(x,\alpha) = \left(\prod_{i,j} Z_i Z_j (x_{i,j}\tau/r) \prod_i X_i(\alpha\tau/r) \right)^r \quad (5.17)$$

This is illustrated in Figure 5.2.

We use this connection to show that the learning algorithm that learns $\mathcal{C}_{n,\log n}^{PQC}$ can also learn the time-dynamics class $\mathcal{C}_{n,\log n}^H$. In the rest of this section, we will adapt the results of the previous section to a quantum circuit approximating this evolution. As the quantum learner we devised previously is only efficient for polynomially sized spectrum, we investigate

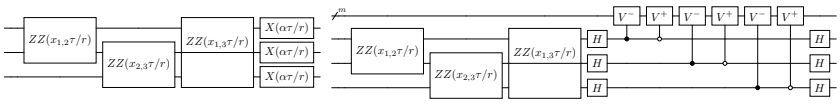

Figure 5.2.: Example of the Trotterization of the τ-time evolution of $H(x,\alpha) = \sum_{i,j} x_{i,j} Z_i Z_j + \alpha \sum_i X_i$ for three qubits in r step. The circuit presented is just one Trotter step and shall be repeated r times. (left) Compilation as a Pauli encoded parameterized quantum circuit as in Definition 5.1. (right) Fourier coefficient extraction applied to the original circuit.

circuit compilation such that this is guaranteed.

With the same strategy as that of the previous section, we show that the learning algorithm for the PQC-based functions concept class also works as an efficient learning algorithm for the Hamiltonian dynamics concept class. For completeness, we have derived the full proof in Section 5.7.6 for first order Trotterization. The key idea is that Hamiltonian simulation allows us to construct a good hypothesis class for our learning algorithm based on the concept class. In fact, as we explain later, all Hamiltonian simulation methods lead to function families with essentially the same learning characteristics.

Theorem 5.4
The Hamiltonian dynamics concept class $\mathcal{C}^H_{\log}$ as in Definition 5.3 is efficiently quantum PAC learnable as in Definition 2.2.

5.4.3. Hardness of the learning problem

In this subsection, we prove that the Hamiltonian dynamics concept class is not classically efficient PAC learnable unless $\mathsf{BQP} \subset \mathsf{P/poly}$.

Specifically, we will be proving that the special case of the Hamiltonian dynamics concept class, where the input is a bit string that specifies the initial state, and the Hamiltonian is parameterized only by α is hard.

For this, we use Lemma 3 in [22], based on [158], which is restated here.

Lemma 5.3
[from [22]] For any k-gate quantum circuit $U = U_k \cdots U_2 U_1$ acting on n qubits there exists a local Hamiltonian H such that for any n qubit initial state $|\psi\rangle$, we have

$$e^{iH\pi} |\psi\rangle |0\rangle = U |\psi\rangle |k\rangle \tag{5.18}$$

We choose a promise BQP language $\mathcal{L}$, solved by $U \in \mathcal{U}(2^n)$, meaning that the sign of $\mathrm{Tr}\left[OU\left|x\right\rangle\!\left\langle x\right|U^\dagger\right]$ decides $\mathcal{L}$, where O is some simple observable. We define the $V(x)$ that prepares $\left|x\right\rangle$ from $\left|0\right\rangle$. We apply Lemma 5.3 and get $H(x)$ such that

$$e^{iH(x)\pi}\left|0\right\rangle\left|0\right\rangle = UV(x)\left|0\right\rangle\left|k\right\rangle = U\left|x\right\rangle\left|k\right\rangle \tag{5.19}$$

Choosing the observable $O' = O \otimes I$, we define the concept class as in Definition 5.3 for the hamiltonian $H(x,\alpha)$ with added parameterized Pauli terms P_l with unknown parameters α_l.

$$H(x,\alpha) = H(x) + \sum \alpha_l P_l \tag{5.20}$$

The concept class contains at least one concept that is BQP-hard, for $\alpha = 0$. Therefore, by the same argument as for the PQC-based functions concept class, using Lemma 2 in [22], this yields the following theorem.

Theorem 5.5
If BQP $\not\subset$ P/poly *then the Hamiltonian dynamics concept class Definition 5.3 is not classically efficiently PAC learnable as in Definition 2.2.*

5

5.5. Beyond log-many parameters

5.5.1. Exponentially large spectrum

So far we have considered concept classes where the number d_n of α parameters scales logarithmically with n, making them restricted. When instead of d_n scales polynomially, the cardinality of the spectrum $|\mathcal{L}|$ scales exponentially. The proposed algorithm is no longer efficient in general, as it would require performing regression in an exponentially large space. In addition, for complexity-theoretic reasons, devising a scheme that provably efficiently PAC-learns any setting with an exponentially large spectrum is not possible.

Specifically, in [159] it was proven that the learning of shallow classical circuits, even in the quantum PAC model (strictly stronger model than ours as the data in terms of a purification rather than samples from a distribution) is impossible under common complexity-theoretic assumptions, e.g. that ring-learning with errors cannot be done in polynomial time on a quantum computer. Polynomially-sized PQCs with polynomially many parameters can encode shallow classical circuits, and thus their efficient

learnability would also imply the unlikely efficient quantum algorithms for learning with errors.

However, giving up on provable bounds, we can still devise a heuristic algorithm making use of the proposed feature map.

5.5.2. Kernel approach

We propose a kernel method approach as a heuristic. Consider the PQC-based functions concept class, for which we have $c_\alpha(x) = \sum_{l\in\mathcal{L}} b_l e^{i\pi\alpha\cdot l}$, or equivalently, for any $l \in \mathcal{L}$:

$$b_l(x) = \int_{(0,1)^d} c_\alpha(x) e^{-i\pi\alpha\cdot l} d\alpha \tag{5.21}$$

We define the kernel as,

$$k(x, x') = b(x) \cdot b(x')\,, \tag{5.22}$$

and make use of the quantum algorithm proposed in Section 5.2 to estimate kernel values. We provide more details on the circuit we use to do so efficiently in Section 5.7.7. Based on this, we build the $T \times T$ Gram matrix with $O(T^2)$ evaluations on a quantum computer. Finally, we can perform traditional kernel methods on a classical computer, for example, kernel ridge regression.

We discuss the limitations of this kernel approach and describe conditions under which they can be mitigated. In general, the dimension of the feature map of the resulting kernel is exponentially large. Therefore, the generalization performance is not guaranteed unless we have an exponential number of training samples. In fact, quantum kernels famously suffer from problems of exponential concentration [160]. A number of known factors yield exponential concentration, such as the expressivity of the data embedding, or global measurements. However, looking at the circuit producing the kernel evaluation in Figure 5.8, we argue that the kernel we propose is not immediately concerned by any of the known causes of exponential concentration in quantum kernels.

In fact, we prove that if the spectrum is sparse, with polynomially large support, the kernel ridge regression yields an efficient PAC learning algorithm in Section 5.7.7. It is possible to construct an artificial case where we get a frequency support that is only polynomially large for an a priori exponentially large spectrum. Consider the concept class Definition 5.2 with $d \in O(\text{poly}(n))$, but such that most α parameters cancel each other

out by construction, with for example $R_z(\alpha_s)YR_z(\alpha_s)$. Suppose that only logarithmically many α survive these cancellations. While the spectrum is a priori exponentially large, it is in reality only polynomially large. The proposed kernel approach could efficiently learn this concept class, while no classical learner could unless $\mathsf{BQP} \subset \mathsf{P/poly}$.

5.5.3. Properties of the feature map

By Mercer's theorem, any kernel has a feature map associated with it. In the kernel approach that we propose, the feature map is simply $x \to b(x)$. That is, the feature map is directly the complex vector that represents the Fourier coefficients with respect to α of the function described by a bitstring x.

In this subsection, we argue that for the Hamiltonian dynamics concept class, the feature map associated with the kernel is in some sense equivalent for any circuit approximating the function. In particular, this means that as long as a given precision level is reached, the feature map is invariant with respect to the Hamiltonian time evolution technique. For simplicity, we propose to use Trotterization, where the depth scales polynomially with the error, but, for more optimal schemes [161, 162], the depth scales logarithmically with the error. We explain this in more detail below.

Given a bounded function $f : [0,1]^d \to \mathbb{C}$, it is always possible to define its Fourier coefficients as $b_l = \int_{(0,1)^d} f(\alpha) e^{i2\pi l\cdot\alpha} d\alpha$ for any $l \in \mathbb{Z}^d$. That is, any such function can be represented as a complex vector in an infinite-dimensional space. We have also seen that when a parameterized circuit is Pauli-encoded, it has a finite Fourier representation, that is, b is a finite-dimensional vector. We argue that if a Pauli-encoded parameterized quantum circuit approximates a function, its finite-dimensional vector approximates the infinite-dimensional one in the 2-norm. The Hausdorff-Young inequality [163] states that, for any two integer $p \in [1,2]$ and p' such that $\frac{1}{p} + \frac{1}{p'} = 1$, we have,

$$\Big(\sum_{l\in\mathbb{Z}^d} \big|b_l\big|^{p'} \Big)^{1/p'} \leq \Big(\int_{(0,1)^l} |f(\alpha)|^p \, d\alpha \Big)^{1/p}. \tag{5.23}$$

Consider an exact time evolution yielding a concept c_α (see Definition 5.3). It is approximated up to $\varepsilon/2$ by a first circuit yielding the functions c' and another circuit yielding the function c''. We note their Fourier decomposition b' and b'' respectively. Then, using the Hausdorff-Young inequality with $p = p' = 2$ we get an upper bound of the 2-norm

of the Fourier decomposition of the difference $c' - c''$ as its infinite norm, itself bounded by ε, as follows,

$$\|b'_l - b''_l\|_2 \leq \Big(\int_{(0,1)^d} |(c' - c'')(\alpha)|^2 \, d\alpha \Big)^{1/2} \leq \varepsilon \,. \tag{5.24}$$

This concludes that feature maps approximating the same quantum function are ε-close in the 2-norm metric space. This analysis implies that the learning and generalization properties of the kernel will not in any significant way depend on which Hamiltonian simulation technique is used, i.e. which hypothesis function is chosen for the quantum learning algorithm.

5.6. Discussion

5.6.1. Cardinality of the Concept class

In the first work demonstrating quantum-classical learning separations for quantum functions the concept classes were polynomially sized, which meant a brute-force algorithm checking all hypotheses is efficient [21]. In [22, 164] first examples of learning with an exponentially-sized concept class were introduced.

The concept classes we study here, the PQC-based functions and the Hamiltonian dynamics concept classes, are indexed by continuous parameters, and each parameter setting mathematically specifies a different function (up to periodicity concerns), and thus the class is a continuum. However, we note that we are interested in approximations, that is, finding functions which agree with the true function on a $1 - \delta$ fraction of the inputs when sampled from the input distribution. The question then arises of what the effective size of this function family is. Precisely, we are interested in the hypothesis family $\mathcal{H}$, such that for each $c \in \mathcal{C}$ there exists $h_c \in H$ such that h_c is ε-close to c in the PAC sense as in Definition 2.2. H can depend on ε and the input distribution $\mathcal{D}$.

If $\mathcal{H}$ is polynomially sized in n and its elements can be efficiently constructed and evaluated, then this would allow for a brute-force type algorithm for our learning task in the log-parameter case. We note that demanding that $\mathcal{H}$ is a subset of $\mathcal{C}$ we obtain the notion of ε-packing of the concept class, which is closely related to covering numbers, and both are quite well understood as quantities governing, for example, generalization bounds [157].

For our concept class PQC-based functions, to the best of our effort to obtain a tight bound (see Section 5.7.4) we can find a super-polynomial grid

of functions which attains epsilon-packing, upper bounding the smallest effective hypothesis size to $O(n^{\log\log(n)})$. We conjecture that this log log scaling in the exponent may be an artefact of the bounding method and that, in fact, the concept classes with d in $O(\log(n))$ allow for a polynomial-sized effective class and a more direct learning algorithm. However the key question of whether this is true and whether these hypotheses can be efficiently found and evaluated remains open.

Therefore, the method we provide is not brute force, but the arguments above raise the question of whether a brute force method could also be possible for our learning task. Either way, we emphasize that a learning separation persists.

5.6.2. Conclusions

In this chapter, we have proposed an algorithm that efficiently prepares a state representing the Fourier decomposition of some quantum functions. We have defined a concept class based on parameterized quantum circuits that can be efficiently PAC learned using a quantum computer and standard regression techniques. As this first concept class is not physically relevant, we built on it and propose a concept class based on a Hamiltonian time evolution and show that it is also quantum-efficient PAC learnable. We do so by applying the previous result to the Trotterized time evolution, but later discuss that any quantum simulation technique would yield similar performance. We also show that both classes cannot be PAC-learned efficiently by any classical algorithm unless $\mathsf{BQP} \subset \mathsf{P/poly}$, effectively proving a learning separation. Both concept classes have polynomially sized feature space, which yields a weaker learning separation in comparison with brute force approaches [21]. We finally discuss regimes in which a priori exponentially large feature space could remain efficiently PAC-learnable and avoid exponential concentration issues [160].

5.7. Appendix

5.7.1. LASSO regression

LASSO (Least Absolute Shrinkage and Selection Operator) regression is a linear regression method [165] with an added regularization on the 1-norm of the weight vector to the loss function. This enforces sparsity in the estimated coefficients. Given an input matrix $X \in \mathbb{R}^{m\times T}$ and label vector $y \in \mathbb{R}^m$, the LASSO estimator solves a mean square error minimization

with a constraint of the norm 1 of the weight vector.

$$\hat{w} = \arg\min_w \|y - Xw\|_2^2 \text{ subject to } \|w\|_1 \leq \Lambda_1,$$

where $\Lambda_1 > 0$ controls the trade-off between sparsity and model fit. This property makes LASSO particularly useful in high-dimensional settings where many predictors may be irrelevant. In Section 5.7.1 we prove the following theorem about the robustness of the LASSO regressor in the presence of perturbations.

Theorem 5.6
Consider a dataset of size T, denoted $\{(\hat{b}_t, \hat{y}_t)\}_{1 \leq t \leq T}$, generated from a linear model with an unknown weight vector $w \in \mathbb{R}^m$, and affected by perturbations, as follows:

$$b_t \cdot w = y_t, \tag{5.25}$$

$$\hat{b}_t = b_t + \eta_{b,t}, \tag{5.26}$$

$$\hat{y}_t = y_t + \eta_{y,t}, \tag{5.27}$$

where $\|\eta_{b,t}\|_\infty$ and $\|\eta_{y,t}\|_\infty$ are bounded by ε_b and ε_y, respectively. Here, m denotes the dimension of b and we know r_∞ such that $\|b\|_\infty \leq r_\infty$. Running the LASSO algorithm with the constraint that $\|w\|_1 \leq \Lambda_1$ on this dataset, with probability at least $1 - \delta$, yields a model h^* such that the true risk (training error + generalisation error) is at most ε, provided that

$$T \geq \frac{(2\Lambda_1 r_\infty)^4 \sqrt{2 \log\left(\frac{2m}{\delta}\right)}}{\varepsilon^2}. \tag{5.28}$$

$$\Lambda_1 \varepsilon_b \leq 0.2\varepsilon \tag{5.29}$$

$$\varepsilon_y \leq 0.5\varepsilon \tag{5.30}$$

We prove Theorem 5.6 below. First, we state the well-known generalization bound for the LASSO algorithm. Then we prove a lemma that bounds the empirical risk, and then we combine these two to bound the true risk.

Theorem 5.7
Let $\mathcal{X} \subseteq \mathbb{R}^m$ and

$$\mathcal{H} = \{b \in \mathcal{X} \mapsto w \cdot b : \|w\|_1 \leq \Lambda_1\}.$$

Let $S = ((b_1, y_1), \ldots, (b_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$. Let $\mathcal{D}$ denote a distribution over $\mathcal{X} \times \mathcal{Y}$ according to which the training data S is drawn. Assume that there exists $r_\infty > 0$ such that for all $b \in \mathcal{X}$, $\|b\|_\infty \leq r_\infty$, and $M > 0$ such that $|h(b) - y| \leq M$ for all $(b, y) \in \mathcal{X} \times \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following inequalities holds for all $h \in \mathcal{H}$:

$$\mathcal{R}(h) \leq \hat{\mathcal{R}}_S(h) + 2r_\infty \Lambda_1 M \sqrt{\frac{2\log(2m)}{T}} + M^2 \sqrt{\frac{\log(\delta^{-1})}{2T}},$$

where $\mathcal{R}(h) = \mathbb{E}_{(b,y)\sim\mathcal{D}}\left[|h(b) - y|^2\right]$ is the prediction error for the hypothesis h, and $\hat{\mathcal{R}}_S(h)$ is the training error of h on the training data S.

Lemma 5.4 (Upper Bound on Empirical Risk under Bounded Perturbations)
Let $h^*(x) = \mathbf{w}^* \cdot \hat{\mathbf{b}}(x)$ denote the hypothesis returned by the LASSO algorithm, trained on a dataset with perturbed features $\hat{\mathbf{b}}(x)$ and noisy labels $\hat{y} = y + \eta_y$. Suppose the true labeling function is linear, i.e., $y = \mathbf{w} \cdot \mathbf{b}(x)$, for some weight vector $\mathbf{w} \in \mathbb{R}^m$, and assume that $\|\mathbf{w}\|_1 \leq \Lambda_1$. Furthermore, assume that the unperturbed features satisfy $\|\mathbf{b}(x)\|_\infty \leq r_\infty$, the perturbation on the features is bounded by $\|\hat{\mathbf{b}}(x) - \mathbf{b}(x)\|_\infty \leq \varepsilon_b$, and the additive noise on the labels is bounded as $|\eta_y| \leq \varepsilon_y$. If the LASSO optimization problem is solved approximately such that the empirical risk of h^* is within $\varepsilon_3/2$ of the optimal empirical risk over all weight vectors with ℓ_1-norm at most Λ_1, then the empirical risk of the resulting hypothesis satisfies

$$\hat{\mathcal{R}}_S(h^*) = \frac{1}{T}\sum_{t=1}^{N} \left(h^*(x_t) - y_t\right)^2 \leq \left(\Lambda_1 \varepsilon_b + \varepsilon_y\right)^2 + \frac{\varepsilon_3}{2}.$$

Proof. For any function g, the training error is defined as

$$\hat{\mathcal{R}}(g) = \frac{1}{T}\sum_{t=1}^{T} |g(x_t) - y_t|^2.$$

Now let $\mathbf{w}^*$ be the optimal vector that the LASSO algorithm outputs, i.e.,

$$\mathbf{w}^* = \arg\min_{\|\mathbf{w}\|_1 \leq \Lambda_1} \left(\frac{1}{T}\sum_{t=1}^{T} \left| \mathbf{w} \cdot \hat{\mathbf{b}}(x_t) - y_t \right|^2 \right).$$

It is clear that for any other $\mathbf{w}' \neq \mathbf{w}^*$ with bounded norm (i.e. $\|\mathbf{w}'\|_1 \leq \Lambda_1$),

$$\frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{w}^* \cdot \hat{\mathbf{b}}(x_i) - y_i\right|^2 \leq \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{w}' \cdot \hat{\mathbf{b}}(x_i) - y_i\right|^2.$$

Let $\mathbf{w}$ be the true weight vector in the true labeling function $h(x) = \mathbf{w}\cdot\mathbf{b}(x)$. Using this, we can bound the training error for the function $h'(x) = \mathbf{w}'\cdot\hat{\mathbf{b}}(x)$. Let

$$t^* = \arg\max_{0\leq t\leq T}|\mathbf{w}\cdot\hat{\mathbf{b}}(x_t) - y_t|^2$$

be the index of the training data point that maximizes the loss. Then:

$$\begin{aligned}
\hat{\mathcal{R}}(h') &= \frac{1}{T}\sum_{t=1}^{T}|\mathbf{w}'\cdot\hat{\mathbf{b}}(x_t) - y_t|^2 \\
&\leq \frac{1}{T}\sum_{t=1}^{T}|\mathbf{w}\cdot\hat{\mathbf{b}}(x_t) - y_t|^2 \\
&\leq |\mathbf{w}\cdot\hat{\mathbf{b}}(x_{t^*}) - y_{t^*}|^2 \\
&\leq \Big(|\mathbf{w}\cdot\hat{\mathbf{b}}(x_{t^*}) - \mathbf{w}\cdot\mathbf{b}(x_{t^*})| + |\mathbf{w}\cdot\mathbf{b}(x_{t^*}) - y_{t^*}|\Big)^2 \\
&\leq (\Lambda_1\varepsilon_b + \varepsilon_y)^2.
\end{aligned}$$

Now consider the function $\hat{h}(x) = \hat{\mathbf{w}}\cdot\hat{\mathbf{b}}(x)$, where $\hat{\mathbf{w}}$ is obtained by minimizing the training error such that its empirical risk is at most $\varepsilon_3/2$ worse than the minimum. That is, we allow for a suboptimal solution. Then the empirical risk of $\hat{h}$ satisfies:

$$\hat{\mathcal{R}}(\hat{h}) \leq (\Lambda_1\varepsilon_b + \varepsilon_y)^2 + \frac{\varepsilon_3}{2}.$$

□

The proof of Theorem 5.7 follows from the prediction error of the LASSO algorithm and the bound on empirical risk derived in Lemma 5.4. First, we address M, which is the upper bound on the absolute prediction error. In the noisy case where LASSO receives $\hat{b}$ as input and the labels are also noisy, we have:

$$|\hat{h}(\hat{b}) - \hat{y}| \leq M.$$

Now:

$$\begin{aligned}|\hat{h}(\hat{b}) - \hat{y}| &= |\hat{b} \cdot \hat{w} - \hat{y}| \\ &= |\hat{b} \cdot \hat{w} - y - \eta_y| \\ &\leq |\hat{b} \cdot \hat{w}| + |y + \eta_y| \\ &\leq \|\hat{b}\|_\infty \|w\|_1 + |y| + |\eta_y|,\end{aligned}$$

where the last inequality follows from Hölder's inequality. The error on the labels is bounded, so $|\eta_y| \leq \varepsilon_y$, and $\|w\|_1 \leq \Lambda_1$. Since the true label is $y = b \cdot w$, and under the assumptions $\|b\|_\infty \leq r_\infty$ and $\|w\|_1 \leq \Lambda_1$, we get $|y| \leq \|b\|_\infty \|w\|_1 \leq r_\infty \Lambda_1$. Similarly, since $\|\hat{b}\|_\infty \leq r_\infty + \varepsilon_b$, and assuming $\|\hat{w}\|_1 \leq \Lambda_1$, we obtain the bound:

$$M := \Lambda_1(2r_\infty + \varepsilon_b) + \varepsilon_y.$$

Using this lemma, we can rewrite the generalization bound as:

$$\mathcal{R}(\hat{h}) \leq (\Lambda_1 \varepsilon_b + \varepsilon_y)^2 + \frac{\varepsilon_3}{2} + 2r_\infty \Lambda_1 M \sqrt{\frac{2\log(2m)}{T}} + M^2 \sqrt{\frac{\log(\delta^{-1})}{2T}},$$

where $r_\infty = r_\infty + \varepsilon_b$. To bound the prediction error above by

$$\varepsilon = (\Lambda_1 \varepsilon_b + \varepsilon_y)^2 + \varepsilon_3,$$

it suffices to choose T such that:

$$2r_\infty \Lambda_1 M \sqrt{\frac{2\log(2m)}{T}} + M^2 \sqrt{\frac{\log(\delta^{-1})}{2T}} \leq \frac{\varepsilon_3}{2},$$

and substituting for M and r_∞, solving for T, gives the sample complexity:

$$T \geq \frac{(\Lambda_1(2r_\infty + \varepsilon_b) + \varepsilon_y)^4 \sqrt{2\log(2m/\delta)}}{\varepsilon_3^2}.$$

By setting $\Lambda_1 \varepsilon_b = 0.2\varepsilon$, $\varepsilon_y = 0.5\varepsilon$, and $\varepsilon_3 = 0.4\varepsilon$, the prediction error is bounded by

$$(\Lambda_1 \varepsilon_b + \varepsilon_y)^2 + \varepsilon_3 \leq \varepsilon,$$

provided that

$$T > \frac{(2\Lambda_1 r_\infty)^4 \sqrt{2\log(2m/\delta)}}{\varepsilon^2}.$$

which concludes the proof of Theorem 5.6.

5.7.2. Complexity assumption

In this chapter, we present learning separation statements that rely on the following widely believed conjecture.

Conjecture 5.1
$\mathsf{BQP} \not\subset \mathsf{P/poly}$.

This is a weaker conjecture than the following one used in previous works.

Conjecture 5.2
There exists a distribution $\mathcal{D}$ *such that* $\mathsf{BQP} \not\subset \mathsf{HeurP}^{\mathcal{D}}/\mathsf{poly}$.

Nonetheless, this too is believed to be true: the discrete logarithm problem or factoring are not believed to be in $\mathsf{HeurP/poly}$. For these problems, the average-case to worst-case reductions imply that if either problem is efficiently solvable heuristically, then it is also efficiently solvable in the worst case as well. This does not hold for BQP functions in general, so Conjecture 1 is indeed weaker. Still, in general, by Lemma 3 in [21] if there exists a single $\mathcal{L} \in \mathsf{BQP}$ that is not in $\mathsf{HeurP/poly}$ under some distribution, then for every BQP-complete problem, there exists a distribution under which the problem is not in $\mathsf{HeurP/poly}$.

Lemma 5.5 (from [21])
If there exists a $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$ *with* $L \in \mathsf{BQP}$*, then for every* $L' \in \mathsf{BQP\text{-}complete}$ *there exists a family of distributions* $\mathcal{D}' = \{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ *such that* $(L', \mathcal{D}') \notin \mathsf{HeurP/poly}$.

5.7.3. Fourier coefficient extraction algorithm

Fourier representation of parameterized circuits

In this section, we prove the Theorem 5.1 and describe the algorithm $\mathcal{A}$. As mentioned in the main text, without loss of generality, we consider that every circuit as defined in Definition 5.1 uses strings of identity and Z matrices encoding. Indeed, if encoding with X or Y appear they can be changed to Z with local unitary gates, which can be absorbed in the set of fixed gates.

Description of the algorithm

The input to the algorithm $\mathcal{A}$ is a description of the n-qubit circuit as an alternating sequence between sets of fixed gates resulting in unitaries $\{U_s\}_{0\leq s\leq dL}$ and encoding gates

$$\{U_{s,j}(\alpha_j) \coloneqq \exp(i\pi\alpha_j \prod_{1\leq k\leq n} Z_k^{b_{s,j,k}})\}_{s,j\in[1,L]\times[1,d]}$$

where j describe the index of the dimension of the data that is encoded, and $b_{s,j}$ is the bitstring of the qubits affected by the encoding gate.

The output of the algorithm $\mathcal{A}$ is a description of a circuit on a larger number of qubits. Registers are added to the existing circuit register to keep track of frequencies of α on each of the d dimensions, each uploaded L times. The number of qubits needed are:

$$n = d\lceil\log(1+2L)\rceil \tag{5.31}$$

For simplicity we assumed that all variables are uploaded the same number of times but, it is possible that it is not the case, in which variables the number of qubits required are $n = \sum_j \lceil\log(1+2L_j)\rceil$.

There is also a single additional ancillary qubit that is used to compute parities. This ancillary qubit is not necessary, not having it occurs an overhead exponential in the locality of Pauli strings in the data encoding gates. Therefore, in total, there are n_T qubits as follows:

$$n_T = n + d\lceil\log(1+2L)\rceil \tag{5.32}$$

We name the registers $f = f_0 \cdots f_j$ for the frequency registers, a for the ancillary qubit, and c for the circuit register. A gate G applied to the register r will be written as $G^{(r)}$. Control on the $|1\rangle$ ($|0\rangle$) state are written as C ($\bar{C}$). The frequency registers are also indexed by negative numbers. For the frequency register we define the increment (decrement) gate with unitary matrix V_+ ($V_-{=}V_+^\dagger$) ensuring unitarity with a circular condition as such:

$$V_+ |k\rangle = |k+1\rangle, \forall -2L \leq k < 2L \tag{5.33}$$

$$V_+ |2L\rangle = |-2L\rangle \tag{5.34}$$

The algorithm $\mathcal{A}$ returns a sequence of gates that follows that of the original circuit, where fixed gates are unchanged and applied to the c register, and encoding gates $\{j, b_{s,j}\}$ are transformed as follows. For each

qubit affected non trivially by the encoding gate (that is, with a Z_k generator when $b_{s,j,k} = 1$) the parity is computed on the ancillary qubit, with a sequence of CNOT gates as follows:

$$D(s,j) = \prod_{k|b_{s,j,k}=1} C^{(c_k)} X^{(a)} \tag{5.35}$$

At this stage, the ancillary encodes the parity of the sub-bitstring for indexes $b_{s,j}$. Then the j-th frequency register, corresponding to the dimension of the input being encoded, is acted on based on the parity encoded in the ancillary as such:

$$G(j) = \left(C^{(a)} V_+^{(f_j)}\right) \left(\bar{C}^{(a)} V_-^{(f_j)}\right) \tag{5.36}$$

Effectively, the subspace where the parity of the substring is even sees an increment in the frequency of the input being encoded, while the other subspace sees a decrement. Finally, the ancillary qubit is reset to be reused later with $D(s,j)^\dagger = D(s,j)$.

The output of the algorithm is an alternating sequence between sets of gates being unchanged from fixed gates resulting in unitaries $\{U_{s,j}^{(c)}(\alpha_j)\}_{s,j}$ and gates replacing encoding gates as $\{D(s,j)G(j)D(s,j)\}_{s,j}$.

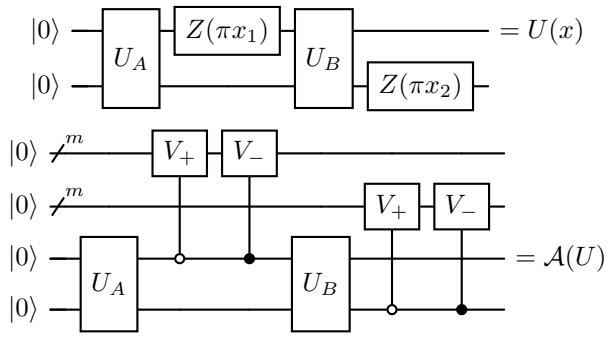

Figure 5.3.: Illustration of the *Fourier coefficient extraction* algorithm for multiple inputs

Proof of the algorithm

Finally, we prove that the algorithm $\mathcal{A}$ indeed outputs what it is promised to return. That is, given U such that

$$|\phi\rangle = U|0\rangle = \sum_{k\in\{0,1\}^n} \sum_{l\in\mathcal{L}'} a_{l,k} e^{i\alpha\cdot l} |k\rangle \ , \tag{5.37}$$

then it returns,

$$|\phi'\rangle = \mathcal{A}(U)|0\rangle = \sum_{k\in\{0,1\}^n} \sum_{l\in\mathcal{L}'} a_{l,k} |l\rangle^{(f)} |0\rangle^{(a)} |k\rangle^{(c)} \ . \tag{5.38}$$

For the rest of the proof, we shorten the indices k and l for easier notation. This is an induction proof in three steps:

1. Initial step : This is trivial, if $|\phi\rangle = |0\rangle$ then $|\phi'\rangle = \mathcal{A}(I)|0\rangle = |0\rangle^{(f)} \otimes |0\rangle^{(a)} \otimes |0\rangle^{(c)}$.
2. Fixed gate step: Suppose that the algorithm works as intended, before the application of a set of fixed gates with unitary V. Let us show that as algorithm $\mathcal{A}$ applies $V' = I^{(f)} \otimes I^{(a)} \otimes V^{(c)}$, it yields the correct state. It is easy to see that $V|\phi\rangle = \sum_l (\sum_k a_{l,k} V|k\rangle) e^{i\alpha\cdot l}$, and $V'|\phi\rangle = \sum_l (\sum_k a_{l,k} V|k\rangle^{(c)}) |0\rangle^{(a)} |l\rangle^{(f)}$.
3. Encoding gate step: Suppose that the algorithm works as intended before the application of an encoding gate $\{j, b_{s,j}\}$. Let us prove that applying $D(s,j)G(j)D(s,j)$ yields the correct state. We prove this below.

If these three steps are true, then by induction, the algorithm is correct.

Proof of step 3. The effect of the encoding gate $\{s, b_{s,j}\}$ on the state $|\phi\rangle$ is as follows

$$e^{i\alpha_j \prod_{k'} Z_{k'}^{b_{s,j,k'}}} |\phi\rangle = \sum_k \sum_l a_{l,k} e^{i\alpha_j p(k,b_{s,j})} |k\rangle \, e^{i\alpha\cdot l} \tag{5.39}$$

$$= \sum_k \sum_l a_{l,k} |k\rangle \, e^{i\alpha\cdot(l + e_j p(k,b_{s,j}))} \tag{5.40}$$

Where $p(k, b_{s,j})$ is the parity of the substring of k with indices $b_{s,j}$ as follows $p(k, b_{s,j}) = (-1)^{\sum_{k'} k_{b_{s,j,k'}}}$. For example if $k = 010110$ and $b_{s,j} = 015$, the substring is 010, with odd parity and $p = -1$. We also write e_r as the vector such that only the r-th element is 1 and the rest is 0.

On the other hand, for the $\mathcal{A}(U)$ computation we have:

$$D(s,j)\,|\psi'\rangle \quad = \quad \sum_k \sum_l a_{l,k}\,|l\rangle^{(f)}\,(X^{\sum_{k'} k_{b_{s,j,k'}}}\,|0\rangle^{(a)})\,|k\rangle^{(c)} \tag{5.41}$$

$$G(j)D(s,j)\,|\psi'\rangle = \sum_k \sum_l a_{l,k}\,|l + e_j p(k, b_{s,j})\rangle^{(f)} \\ (X^{\sum_{k'} k_{b_{s,j,k'}}}\,|0\rangle^{(a)})\,|k\rangle^{(c)} \tag{5.42}$$

$$D(s,j)G(j)D(s,j)\,|\psi'\rangle \;=\; \sum_k \sum_l a_{l,k}\,|l + e_j p(k, b_{s,j})\rangle^{(f)}\,|0\rangle^{(a)}\,|k\rangle^{(c)} \tag{5.43}$$

The states correspond to each other, which concludes the proof to the iterative step 3.

Frequency representation of expectation values

So far, we have seen how to get the Fourier representation of a quantum reuploading circuit, but what about quantum function, that is, the expectation values of some observable? In this section, we prove Corollary 5.1 and Corollary 5.2. We explain how to do this for Pauli observables and extend this result to linear combinations of Pauli observables and projectors.

Pauli observable

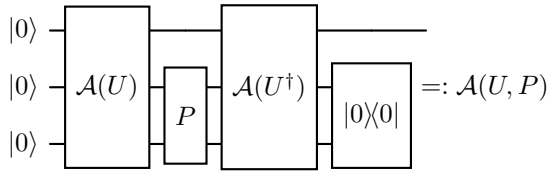

Figure 5.4.: Illustration of the quantum function evaluation algorithm

Without loss of generality (using local changes of basis), the Pauli string measurement can be considered a Z string and therefore $|\phi(\alpha)\rangle$ has the

following form (where $p(P,k)$ is a parity function).

$$f(\alpha) = \langle P \rangle(\alpha) \tag{5.44}$$
$$= \sum_{-L \leq l', l \leq L} \sum_{k} (-1)^{p(P,k)} a_{l,k} a^*_{l',k} e^{i(l-l')\cdot\alpha} \tag{5.45}$$
$$= \sum_{-2L \leq l \leq 2L} b_l e^{il\cdot\alpha} \tag{5.46}$$

We define the state $|\phi(\alpha), P\rangle \coloneqq U(\alpha)^\dagger P U(\alpha) |0\rangle$. It has the following amplitude for $|0\rangle$.

$$\langle 0|\phi(\alpha), P\rangle = \langle 0| U(\alpha)^\dagger P U(\alpha) |0\rangle = \sum_{-2K \leq k \leq 2K} b_k e^{ik\alpha} \tag{5.47}$$

Using the algorithm $\mathcal{A}$ on the state $|\phi(\alpha), P\rangle$ we can lose the dependence on α and efficiently get the Fourier representation state:

$$|\phi, P\rangle = \sum_{-2K \leq k \leq 2K} b_k |k\rangle |0\rangle + \text{trash}\,. \tag{5.48}$$

Post-selecting on $|0\rangle$ has a success probability $\sum |b_k|^2$. The full algorithm is illustrated in Figure 5.4.

5

Linear combination of Pauli observables

In this subsection, we extend the procedure above to a more generic observable, more specifically, a linear combination of polynomially many Pauli strings. This can be done using a Linear Combination of Unitaries approach. Supposing that the observable of interest may be decomposed as follows

$$O = \sum_h \beta_h P_h \tag{5.49}$$

Following the procedure described previously, one may prepare the states $|\phi, P_h\rangle$ for each P_h. By the linearity of the expectation value, adding them yields the expectation value of the full observable.

$$|\phi, O\rangle = \sum_h \beta_h |\phi, P_h\rangle \tag{5.50}$$

The addition may be done using a linear combination of unitary approach, which requires an additional register with a number of qubits logarithmic

in the number of terms in the observable, and the preparation of the following state.

$$V_\beta |0\rangle = \frac{1}{\|\beta\|} \sum_h \beta_h |h\rangle \tag{5.51}$$

It also requires post-selection, which causes overhead in complexity (polynomial for reasonable observables). Once the state is prepared, the same sampling and post-processing may be implemented.

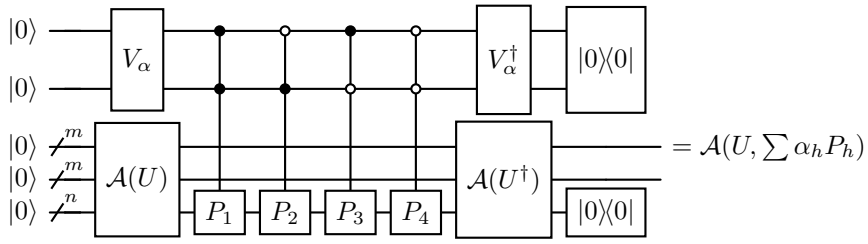

Figure 5.5.: Illustration of the quantum function evaluation algorithm for arbitrary observables

Probabilities, or projectors observables

Consider a circuit U yielding a pure state $U|0\rangle = |\phi\rangle = \sum_k \phi_k |k\rangle$. Its density matrix is $\rho = \sum_{k,l} \phi_k \phi_l^* |k\rangle\langle l|$. Defining the conjugate circuit as U^*, we have

$$(U \otimes U^*)(|0\rangle \otimes |0\rangle) = \sum_{k,l} \phi_k \phi_l^* |k\rangle \otimes |l\rangle \tag{5.52}$$

Therefore, if one wanted to retrieve the probability of measuring $|0\rangle$ of circuit U one could retrieve the amplitude as above. This yields the procedure illustrated in Figure 5.6 to retrieve the coefficients for the observable $|0\rangle\langle 0|$, that is the probability of measuring $|0\rangle$ on the original circuit.

Extracting Fourier coefficients

Once the Fourier decomposition is available, one may be interested in retrieving the value coefficients. Suppose we have the following state as

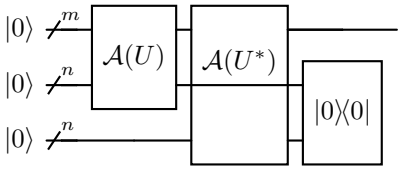

Figure 5.6.: Illustration of the quantum function evaluation algorithm for a projector

an output.

$$\mathcal{A}(U,O)\left|0\right\rangle\left|0\right\rangle = \left|\phi, O\right\rangle = \sum_{-2K\leq k\leq 2K} b_k \left|k\right\rangle\left|0\right\rangle + \text{trash} \tag{5.53}$$

Suppose one is interested in the coefficient of frequency l. We call V_l any unitary such that $\left|l\right\rangle = V_l\left|0\right\rangle$, we have

$$b_l = \left\langle 0\right|\left\langle 0\right|(V_l^\dagger \otimes I)\mathcal{A}(U)\left|0\right\rangle\left|0\right\rangle \tag{5.54}$$

We use a Hadamard test (which can be improved as in [166]) to extract the real and imaginary part of this coefficient as in Figure 5.7. In addition, because the function is real, we have $b_k = b_{-k}$. Therefore, in case there is a polynomial number of frequencies, one may retrieve all of them efficiently, and create a classical surrogate for the quantum function.

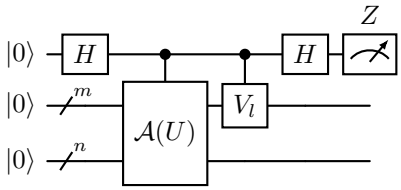

Figure 5.7.: Illustration of the *Fourier coefficient extraction* algorithm

5.7.4. Cardinality of the concept class

First we will find a relatively easy bound for the covering number of the following set of functions:

$$\mathcal{C} = \{c_\alpha : x \to \sum_{-L\leq l\leq L} b_l(x)e^{il\cdot\alpha}, \|c_\alpha\|_\infty \leq 1\}_{\alpha\in[0,1]^d} \tag{5.55}$$

First we construct a covering net using the smoothness of the function with respect to its index α. The partial derivative with respect to a single parameter is as follows,

$$|\partial_{\alpha_k} c_\alpha(x)| \leq |\sum_l il_k b_l(x)e^{il\cdot\alpha}| \leq L\sum_l |b_l(x)e^{il\cdot\alpha}| \leq L\|b(x)\|_1 \,. \tag{5.56}$$

We can use it to bound the 1-norm of the gradient as,

$$\|\nabla_\alpha c_\alpha(x)\|_1 \leq dL\|b(x)\|_1 \,. \tag{5.57}$$

We consider a regular grid with M divisions for each of the d dimensions. We call this grid $\mathcal{A} = \{\alpha_g\}_g$, with $|\mathcal{A}| = M^d$. Therefore, each α will be at least $\|\delta_\alpha\| \coloneqq 1/M$ close to a grid point. We have

$$\begin{aligned} \|c_\alpha - c_{\alpha_g}\|_\infty &\leq \nabla_\alpha c_\alpha \cdot (\alpha - \alpha_g) && (5.58)\\ &\leq \|\nabla_\alpha c_{\alpha_g}\|_1 \|\delta_\alpha(\alpha)\|_\infty && (5.59)\\ &\leq dL\|b(x)\|_1/M \,. && (5.60) \end{aligned}$$

To guarantee the error is lower than ε for all α it is sufficient to choose

$$M = dL\|b(x)\|_1/\varepsilon \,. \tag{5.61}$$

We are interested in the scaling of the grid when $d \in O(\log(n))$ and $L \in O(1)$. For all x we have $\|b(x)\|_2 \leq 1$ and therefore $\|b(x)\|_1 \leq (2L+1)^{d/2} \in O(\text{poly}(n))$. This yields a number of grid points scaling as

$$G = (dL\|b(x)\|_1/\varepsilon)^d \in O(n^{\log(n)}) \,. \tag{5.62}$$

If we leave this function family $\mathcal{C}$, we can apply the approach of [157] states bounds on covering numbers of shot-based expectation values of parameterized quantum circuits. This allow us to get a smaller hypothesis class of proxy functions approximating the functions of the concept class. Theorem 3 in the supplementary material of [157] upper bounds the covering number for a PQC where d parameters (T in the original paper)

are reuploaded L times (M in the original paper). In order to get the expectation values up to η additive accuracy, the number of reuploadings is multiplied by η^{-2} Using $d \in O(\log n)$ and $L \in O(1)$ we get the following scaling:

$$G \in O\left(\left(\frac{dL}{\varepsilon\eta^2}\right)^d\right) \subseteq O(n^{\log\log n})\,. \tag{5.63}$$

Although this scaling is closer to a polynomial bound, it is not polynomial, but this might be an artefact of the proof. The key question of whether there exist a polynomial and constructable set of of hypothesis functions $\mathcal{H}$ that ε-covers $\mathcal{C}$ remains open.

5.7.5. Alternative Oracle-based algorithms

Considering the settings of Section 5.2, with a $U(\alpha)$ defined as above, suppose that instead of the specification of the parameterized quantum circuit U as a sequence of gates, we are given an oracle O_U such that for a m-binary decomposition over the d-dimensional parameter α, we have

$$|\alpha\rangle \otimes |\psi\rangle \xrightarrow{O_U} |\alpha\rangle \otimes U(\alpha)\,|\psi\rangle\ .$$

Applying this oracle to the equal superposition state over the frequency registers, followed by a Quantum Fourier Transform applied to each coordinate frequency register, yields the same result as the algorithm described previously. We define a regular grid over the inputs $\{\alpha_l\}$ such that $\frac{1}{\sqrt{|\mathcal{L}|}}\sum_{l\in\mathcal{L}}\alpha_l = |+\rangle$

$$|0\rangle\,|0\rangle \xrightarrow{H^{\otimes dm}\otimes I} \sum_{l\in\mathcal{L}} |\alpha_l\rangle \otimes |0\rangle \tag{5.64}$$

$$\xrightarrow{O_U} \sum_{l\in\mathcal{L}} |\alpha_l\rangle \otimes U(\alpha_l)\,|0\rangle \tag{5.65}$$

$$\xrightarrow{\mathrm{QFT}_m^{\otimes d}\otimes I} \sum_{l\in\mathcal{L}} \sum_{k\in[1,n]} a_{l,k}\,|l\rangle\,|k\rangle\ . \tag{5.66}$$

The above result provides an analogue for states decomposition as in Theorem 5.1; similarly, we can obtain the analogue for PQC-functions as in Corollary 5.1. For functions, an amplitude oracle can be defined as follows. For a function $f : \{0,1\}^* \to [0,1]$ that takes a binary decomposition over

α, an amplitude oracle O_f yields:

$$|\alpha\rangle \otimes |0\rangle \xrightarrow{O_f} |\alpha\rangle \otimes \left(f(\alpha)\,|0\rangle + \sqrt{1-f(\alpha)^2}\,|1\rangle \right) .$$

As in the previous case, we initialize the α register in the equal superposition state, or equivalently in a superposition on the regular grid. Then we apply the oracle and then the QFT on the frequency registers to finally retrieve the Fourier decomposition of f as an amplitude-encoded state.

5.7.6. Proof of the PAC learnability of the Hamiltonian dynamics concept class

Training: We consider quantum circuits realized by using r Trotter steps of the time evolution of the following parameterized Hamiltonian

$$H(x,\alpha) = H'(x) + \sum_{1\leq s\leq d} \alpha_s P_s \,, \tag{5.67}$$

where P_s are Pauli strings. Such circuits have a frequency space of $m = |\mathcal{L}| = (4r+1)^d$. We implement the circuit learning algorithm exactly as in Section 5.3.2. The difference is that now the labels are off by the Trotter error, that is

$$\varepsilon_y < \frac{t^2 A}{2r} \,, \tag{5.68}$$

where A is the sum of the spectral norm of all pairs of commutators. Using Theorem 5.6 we are guaranteed that the labelling error on a new data point will be smaller ε with probability $1-\delta$ if we choose the following learning parameters

$$T > \frac{16 m^4 \sqrt{2\log\left(\frac{2m}{\delta}\right)}}{\varepsilon^2},\ \varepsilon_b < \frac{0.2\varepsilon}{m},\ \varepsilon_y < 0.5\varepsilon \,. \tag{5.69}$$

The condition on ε_y yields a requirement on the number of Trotter steps as $r > t^2 A/\varepsilon$ which in turn yields the feature space dimension as $m > (4t^2 A/\varepsilon + 1)^d$. We therefore require the number of training data points to scale as

$$T \in \tilde{\Theta}\left(\frac{\sqrt{d}}{\varepsilon^{4d+2}} \right) \subset \tilde{\Theta}(\mathrm{poly}(n, \varepsilon^{-1}, \delta^{-1})) \,, \tag{5.70}$$

so this method is sample-efficient. Regarding computational complexity, the training state requires the execution of $K = mT/\varepsilon_b^2$ poly-depth

quantum circuits with the condition of $\varepsilon_b = 0.2\varepsilon/m$. Recalling that $d \in O(\log(n))$,

$$K \in \tilde{\Theta}\left(\frac{\sqrt{d}}{\varepsilon^{7d+4}}\right) \subset \tilde{\Theta}(\text{poly}(n, \varepsilon^{-1}, \delta^{-1}))\,. \tag{5.71}$$

Therefore, the overall training process is efficient on a quantum computer.

Inference: The inference process takes place analogously to the one in Section 5.3.2. Given a new datapoint $x_{t'}$, one retrieves the Fourier coefficients $b_{t'}$ of the approximate Trotter function. This requires the execution of $m/\varepsilon_b^2 \in \tilde{\Theta}(m^3/\varepsilon^2) \subset \tilde{\Theta}(\text{poly}(n, \varepsilon^{-1}, \delta^{-1}))$ poly-depth circuits. Then using the weight vector $\hat{w}$ derived in the training phase, the model returns $y_{t'} = \hat{w} \cdot \hat{b}(x_{t'})$.

5.7.7. PAC efficient Kernel-based algorithm

In this section, we propose a kernel based approach and proves that it efficiently PAC learns the parameterized circuit concept class as in Definition 5.2 if the spectrum has a polynomially large spectrum.

5

Concept class and noisy data

Consider a U_{hard} that decides a BQP-complete language with the function c_0 defined as follows

$$c_0(x) \coloneqq \langle x|\, U_{\text{hard}}^\dagger Z_0 U_{\text{hard}}\, |x\rangle \tag{5.72}$$

We define the concept class where gates parameterized by an unknown vector $\alpha \in \mathbb{R}^d$ are added to the circuit implementing U_{hard}, such that the concept have a finite Fourier decomposition, as follows

$$c_\alpha(x) = \langle b(x) \mid w(\alpha)\rangle, w(\alpha) = [e^{i\alpha\cdot l}]_{l\in[-L,+L]^d}, b(x), w(\alpha) \in \mathbb{C}^m\,. \tag{5.73}$$

We have access to a dataset $\{(x_t, y_t = c_\alpha(x_t)\}_{t\in[1,T]}$. We define the feature space representation of the data $B = [b(x_t)] \in \mathbb{C}^{T\times m}$, and the Gram matrix $K = [\langle b(x_t) \mid b(x_{t'})\rangle]_{t,t'} \in \mathbb{C}^{T\times T}$. We have access to an approximation of the Gram matrix with $\hat{K} = K + E$ with E p.s.d. and each element is bounded by ε_k, which depends on the number of shots we use to measure the overlap, see Figure 5.8.

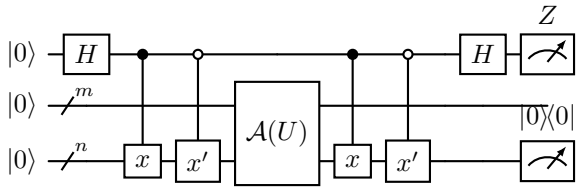

Figure 5.8.: Evaluation of the kernel overlap, we have that $\langle Z \otimes I \otimes |0\rangle\langle 0| \rangle = b(x) \cdot b(x')$

Getting the overlap

We have a circuit such that

$$\mathcal{A}(U) |0\rangle |x\rangle = |b(x)\rangle |x\rangle + |\cdots\rangle |x\rangle^{\perp} \tag{5.74}$$

The goal is to find a circuit such that the expectation of an observable yields $\langle b(x)|b(x')\rangle$. We present such a circuit in Figure 5.8, and prove below that it yields the desired outputs. The application of U and then of the last Hadamard gate yields the following states:

$$\frac{1}{\sqrt{2}}(|0\rangle |0\rangle |x\rangle + |1\rangle |0\rangle |x'\rangle) \xrightarrow{I\otimes U} \tag{5.75}$$

$$\frac{1}{\sqrt{2}}(|0\rangle |b(x)\rangle |0\rangle + |1\rangle |b(x')\rangle |0\rangle + \cdots) \xrightarrow{H\otimes I} \tag{5.76}$$

$$\frac{1}{2}(|0\rangle (|b(x)\rangle + |b(x')\rangle) |0\rangle + |1\rangle (|b(x)\rangle - |b(x')\rangle) |0\rangle + \cdots) \tag{5.77}$$

Finally the expectation value of $Z \otimes I \otimes |0\rangle\langle 0|$ (note it has unit spectral norm) is

$$\frac{1}{4}(||b(x)\rangle + |b(x')\rangle|^2 - ||b(x)\rangle - |b(x')\rangle|^2) = \mathrm{Re}(\langle b(x)|b(x')\rangle)\,. \tag{5.78}$$

5.7.8. Noisy Kernel Ridge Regression

In this subsection we derive a bound on the squared prediction error of kernel ridge regression when both the Gram matrix and the labels are observed with noise. For consistency with earlier sections, we temporarily simplify notation and write $x = b(x)$, $w = w(\alpha)$ and $y = c_\alpha(x)$.

Problem setting: We consider the linear model with input $x \in \mathbb{R}^d$

and unknown parameter $w \in \mathbb{R}^d$:

$$y = x^\top w. \tag{5.79}$$

We observe T training samples collected in a data matrix $X \in \mathbb{R}^{T\times d}$, yielding the Gram matrix and label vector

$$K := XX^\top \in \mathbb{R}^{T\times T}, \qquad Y := Xw \in \mathbb{R}^T. \tag{5.80}$$

Instead of (K, Y) we are given noisy observations, where the noisy matrix is potentially corrected to be positive semi-definite [167].

$$\hat{K} = K + E_K, \qquad \hat{Y} = Y + E_Y, \tag{5.81}$$

with entrywise bounds

$$\|E_K\|_\infty \le \varepsilon_k, \qquad \|E_Y\|_\infty \le \varepsilon_y. \tag{5.82}$$

For a new test point $x' \in \mathbb{R}^d$ we define the kernel evaluation vector

$$F := Xx' \in \mathbb{R}^T, \tag{5.83}$$

and we observe a noisy version

$$\hat{F} = F + E_F, \qquad \|E_F\|_\infty \le \varepsilon_k. \tag{5.84}$$

Assume

$$\|w\|_2 \le B, \qquad k(x,x) \le \kappa, \qquad \|Y\|_\infty \le M \tag{5.85}$$

where k is the kernel used by the Kernel Ridge Regression (KRR). Define $\lambda = T\lambda_0 > 0$ and the KRR solution

$$a_{K,Y} = (K + \lambda I)^{-1} Y \tag{5.86}$$

and predict with test evaluations via

$$h_{K,Y}(x') = a_{K,Y} \cdot F \tag{5.87}$$

In reality we only have access to noisy evaluation and get

$$h_{\hat{K},\hat{Y}}(x') = a_{\hat{K},\hat{Y}} \cdot \hat{F} \tag{5.88}$$

Proposition 1 of [168] gives (for exact test evaluation)

$$|h_{\hat{K},Y}(x) - h_{K,Y}(x)| \leq \frac{\kappa M}{\lambda_0^2 T} \|\hat{K} - K\|_2 \tag{5.89}$$

and since $\|\hat{K} - K\|_2 \leq T\|\hat{K} - K\|_\infty \leq T\varepsilon_k$ we obtain the bound

$$|h_{\hat{K},Y}(x) - h_{K,Y}(x)| \leq \frac{\kappa M}{\lambda_0^2} \varepsilon_k \tag{5.90}$$

Using $\hat{K} \succeq 0$ so $\|a\|_2 \leq \|y\|_2/\lambda \leq \sqrt{T}M/(T\lambda_0)$ and $\|a\|_1 \leq \sqrt{T}\|a\|_2$, the noisy evaluation for new data contributes

$$|a \cdot (\hat{F} - F)| \leq \|a\|_1 \, \varepsilon_k \leq \frac{M}{\lambda_0} \varepsilon_k \tag{5.91}$$

Finally, we address the noise on the labels, which impacts the output of the KRR as follows:

$$a_{K,\hat{Y}} - a_{K,Y} = (K+\lambda I)^{-1} E_Y, \qquad h_{K,\hat{Y}}(x') - h_{K,Y}(x') = F^\top (K+\lambda I)^{-1} E_Y. \tag{5.92}$$

Using $\|(K + \lambda I)^{-1}\|_2 \leq 1/\lambda$, $\|E_Y\|_2 \leq \sqrt{T}\|E_Y\|_\infty \leq \sqrt{T}\varepsilon_y$, and $\|F\|_2 \leq \sqrt{T}\kappa$ (since $|F_i| = |k(x_i, x')| \leq \kappa$), we obtain

$$|h_{K,\hat{Y}}(x') - h_{K,Y}(x')| \leq \|F\|_2 \, \|(K + \lambda I)^{-1}\|_2 \, \|E_Y\|_2 \leq \frac{\kappa}{\lambda_0} \, \varepsilon_y. \tag{5.93}$$

Combining the effect of all noise sources yields

$$|h_{\hat{K},\hat{Y}}(x') - h_{K,Y}(x')| \leq \frac{\kappa M}{\lambda_0^2} \varepsilon_k + \frac{\kappa}{\lambda_0} \varepsilon_y + \frac{M}{\lambda_0} \varepsilon_k. \tag{5.94}$$

Moreover, noiseless KRR with regularization $\lambda = T\lambda_0$ is uniformly stable; hence by [169], for bounded loss, its population risk concentrates around its empirical risk at rate $O(1/T)$, implying PAC learnability, while the additional degradation due to kernel/label noise is controlled by the additive drift bound above polynomial in all relevant quantities.

5.7.9. Conclusion

The proposed kernel-based learning algorithm is able to PAC learn the concept Definition 5.2 under some conditions. Because we have $\kappa = 1$, when B is at most polynomial, we can find parameters $\varepsilon_k, \varepsilon_y$ and T that

scale polynomially, guaranteeing the efficiency of the learning algorithm. In particular, if the spectrum is sparse, then there exist a polynomially sized subset of indices $L' = \{l'\}$ such that any x the $b_{l \notin L'}(x) = 0$, then $B \in \text{poly}(n)$.

5.7.10. Flipped concept and connection to RFF

The flipped concept, where the input and the index of the concept class are inverted as, is in contrast to the concept class $\mathcal{C}^U$ we study easy to learn. We define it as follows,

$$\bar{\mathcal{C}} \coloneqq \{c_x : \alpha \in \mathbb{R}^d \to \sum_l b_l(x) e^{il\cdot\alpha}\}_{x\in\{0,1\}^*} .$$

This corresponds to a scenario where a quantum circuit has some fixed, potentially unknown gates and some parameterized gates. Although this concept class looks similar to the one we studied earlier, for this one it is easy to see that unlike $\mathcal{C}^U_{n,\log n}$, the concepts here are in $\mathsf{P/poly}$, where the advice is the polynomially sized list of $b_l(x)$. Therefore, the arguments we make about the hardness of $\mathcal{C}_{n,\log n}$ do not apply.

In fact, this concept is classically efficiently learnable by a simple Fourier analysis of the data. This is typically the scenario of quantum neural networks and quantum kernels, which have been dequantized by techniques like Random Fourier Features [40, 63, 170]. In fact, using the circuit proposed and sampling from the frequency register after post-selection on the circuit register, yields the optimal distribution to approximate the circuit using RFF, as it satisfies perfectly the alignment condition.