Quantum methods for machine learning and classical dynamics

Barthe, A.M.

CHAPTER 2

Background

2.1. Statistical Learning Theory

2.1.1. Concept classes and PAC learning

Defining success for a computational task, such as factoring a number or solving an optimization problem, is relatively easy. However, defining success for learning tasks is more subtle. In this section, we introduce the *Probably Approximately Correct (PAC)* learning framework [27], which is a formal mathematical framework for learning tasks, which we will heavily rely on in Chapter 5.

PAC learning theory provides a formal mathematical framework for learning tasks, it revolves around so-called concept classes $\mathcal{C}$.

Definition 2.1 (Concept class)
A concept class is a set of functions called concepts c_α*.*

$$\mathcal{C} = \{c_\alpha : x \in \mathcal{X} \to y = c_\alpha(x) \in \mathcal{Y}\}_{\alpha \in \mathcal{A}} \,. \tag{2.1}$$

The goal in PAC learning is to approximate an unknown concept from a specified class given access to a dataset of examples labeled by the concept. That is, for a fixed unknown concept c_{α_0} in the concept class, we are given T input-output pairs. The inputs x_t are sampled from a probability

distribution D over the input domain $\mathcal{X}$, and the outputs are $c_{\alpha_0}(x_t)$. This dataset is denoted by $\mathcal{S}$.

$$\mathcal{S} \coloneqq \{(x_t, y_t = c_{\alpha_0}(x_t))\}_{t\in[1,T]},\ x_t \overset{\text{i.i.d.}}{\sim} D\,. \tag{2.2}$$

Using such a dataset, the goal of a learning algorithm is to output a *hypothesis* function $h : \mathcal{X} \to \mathcal{Y}$ from a set of possible hypotheses $\mathcal{H}$, called the hypothesis class. This hypothesis function is used to assign labels to unseen data x_{new}. When the learning is successful, the assigned labels $\hat{y}_{\text{new}} = h(x_{\text{new}})$ are ε-close to the true label with high probability $1-\delta$. A pair $(\mathcal{C}, D)$ is PAC learnable if there exists an algorithm that produces a hypothesis h given only polynomially many examples, that is T scales at most polynomially with δ^{-1}, ε^{-1} and the size of the input. Adding to PAC learnability a notion of computational costs, a concept class $\mathcal{C} = \{\mathcal{C}_n\}_n$[1] increasing in size n, which yields the concept of *efficient* PAC learning, defined as follows.

Definition 2.2 (Efficient PAC learnability)
The concept class $\mathcal{C} = \{\mathcal{C}_n\}_n$ is efficiently PAC learnable if for all $\varepsilon \geq 0$, all $0 \leq \delta \leq 1$, all $n \in \mathbb{N}$ there exists a $\text{poly}(\varepsilon^{-1}, \delta^{-1}, n)$-time algorithm $\mathcal{A}$ such that for any target concept $c : \mathcal{X}_n \to \mathcal{Y}_n$ in $\mathcal{C}_n$ and any target distribution $\mathcal{D}_n$ on $\mathcal{X}_n$, if $\mathcal{A}$ receives in input a training dataset $\{(x_t, c(x_t))\}_{t\in[0,T]}$ of $\text{poly}(\varepsilon^{-1}, \delta^{-1}, n)$-size T, then with probability at least $1-\delta$ over the random datasets, the learning algorithm $\mathcal{A}$ outputs a specification of a hypothesis function $h = \mathcal{A}(T, \varepsilon, \delta)$ running in $\text{poly}(\varepsilon^{-1}, \delta^{-1}, n)$-time that satisfies

$$\Pr_{x\sim\mathcal{D}_n}(|h(x) - c(x)| \leq \varepsilon) \geq 1 - \delta. \tag{2.3}$$

The average error of the hypothesis h is also referred to as the true risk.

We say that a concept class is *classically efficiently learnable*, if it is efficient PAC learnable with both $\mathcal{A}$ and h running in polynomial time on a classical computer. Conversely, we say that a concept class is *quantum efficiently learnable*, if it is efficient PAC learnable with both $\mathcal{A}$ or h running in polynomial time on either a quantum or a classical computer. In Chapter 5, we will prove that a concept class based on quantum dynamics is quantum efficiently learnable, while it is not classically efficiently learnable.

[1] Here the concept class is divided into sub-classes specific to input size n to make the dependence more explicit.

2.1.2. Covering numbers

In this section, we introduce the concept of covering numbers, which is a measure of the complexity or the size of an infinite set of functions, and play a crucial role in PAC learning theory.

Definition 2.3 (Covering number)
The covering number of a set of functions $\mathcal{C}$ with respect to a distance d is the smallest number of balls of radius ε needed to cover the set $\mathcal{C}$, denoted as $N(\mathcal{C}, d, \varepsilon)$.

$$N(\mathcal{C}, d, \varepsilon) = \min_{\mathcal{H}} |\mathcal{H}| \text{ such that } \forall f \in \mathcal{C}, \exists h \in \mathcal{H} : d(f, h) < \varepsilon \,. \tag{2.4}$$

The logarithm of the covering number of a concept class can be used to analyze generalization performance, as it appears in upper bounds on the number of samples needed to learn a concept class. We provide bounds on the covering numbers of the proposed concept classes in Chapter 5.

2.2. Quantum Computing

This section introduces the minimal fundamentals of qubit-based quantum computing needed for the rest of this thesis. We cover quantum states, quantum gates, and measurements as the three pillars of quantum computation. We refer the reader to [43] for a more complete introduction to quantum computing.

2.2.1. Quantum states

The fundamental unit of quantum information is the *qubit*, a two-level quantum system represented by a vector in a two-dimensional Hilbert space $\mathbb{C}^2$. A qubit's state is a unit vector $|\psi\rangle \in \mathbb{C}^2$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad \alpha, \beta \in \mathbb{C},\ |\alpha|^2 + |\beta|^2 = 1\,. \tag{2.5}$$

Here, $|0\rangle$ and $|1\rangle$ form the so-called computational basis. All quantum computations ultimately refer to measurement outcomes in this basis, while the global phase of a state is unobservable.

For systems of n qubits, the joint state lives in a 2^n-dimensional Hilbert space formed by tensor products of single-qubit spaces. $\otimes$ denotes the tensor product. This operation defines the structure of multi-qubit systems as follows,

$$\otimes : \mathbb{C}^n \times \mathbb{C}^m \to \mathbb{C}^{mn}\,. \tag{2.6}$$

For example, if $|\psi_1\rangle$ and $|\psi_2\rangle$ are single-qubit states, the joint state of the two-qubit system is given by

$$\begin{aligned}|\psi_1\rangle \otimes |\psi_2\rangle &= (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma\,|0\rangle + \delta\,|1\rangle) \\ &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle\,, \end{aligned} \tag{2.7}$$

In general, a full n-qubit system can be described as

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i\,|i\rangle\,, \quad \sum_i |c_i|^2 = 1\,, \tag{2.8}$$

where $|i\rangle$ runs over all the n-bitstrings, each corresponding to a computational basis state $|b_1 b_2 \cdots b_n\rangle$ with $b_j \in \{0, 1\}$.

While many quantum algorithms assume *pure states*, noise and decoherence often require the more general framework of *mixed states*, described by a density matrix ρ. Mixed states form a classical mixture of pure states $|\psi_k\rangle$ according to a probability distribution $\{p_k\}$. ρ can be expressed as the linear combination of their projectors $|\psi_k\rangle\langle\psi_k|$, as follows,

$$\rho = \sum_k p_k\,|\psi_k\rangle\langle\psi_k|\,, \quad \mathrm{Tr}(\rho) = 1,\ \rho \geq 0\,. \tag{2.9}$$

2.2.2. Quantum gates

Quantum gates are unitary operators U acting on the Hilbert space. They satisfy

$$U^\dagger U = U U^\dagger = I, \tag{2.10}$$

and evolve quantum states according to

$$|\psi\rangle \mapsto U\,|\psi\rangle\,. \tag{2.11}$$

The most commonly used quantum gates are listed in Table 2.1, along with their matrix representations.

A gate set is *universal* if it can approximate any unitary U on n qubits to arbitrary precision. Clifford gates alone, generated by H, S, and CNOT, are not universal and are classically simulable [12]. Adding a non-Clifford gate, such as T, yields a universal gate set.

The time evolution of a closed quantum system is governed by the

Table 2.1.: Common single-qubit and two-qubits quantum gates

Gate	Symbol	Matrix Representation
Pauli-X	X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y	Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-Z	Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hadamard	H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Phase	S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
T gate	T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
CNOT	CX	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Schrödinger equation,

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H\left|\psi(t)\right\rangle, \tag{2.12}$$

where H is the system Hamiltonian (a Hermitian operator). The time evolution corresponds to a unitary evolution operator as follows,

$$U(t) = e^{iHt}. \tag{2.13}$$

Parameterized quantum gates arise naturally when simulating Hamiltonian evolution for fixed time. These take the form $R_\alpha(\theta) = e^{i\theta H/2}$ where H is a Hermitian operator called the generator. These are most commonly some Pauli operators, such as:

$$R_x(\theta) = e^{i\theta X/2}, \tag{2.14}$$

$$R_z(\theta) = e^{i\theta Z/2}. \tag{2.15}$$

Such gates are called *parameterized quantum gates* and are central to variational quantum circuits.

2.2.3. Measurements

Measurements extract classical information from quantum states. In the computational basis, the probability $p(i)$ of observing outcome $|i\rangle$ is given by the Born rule, that is, the norm of the projection of the state onto the computational basis state $|i\rangle$. This can be expressed with a dot product $\langle\cdot|\cdot\rangle$ as follows,

$$p(i) = |\langle i|\psi\rangle|^2. \tag{2.16}$$

More generally, measurements are described by a set of operators $\{M_i\}$ forming a *positive operator-valued measure* (POVM) [43], satisfying:

$$\sum_i M_i^\dagger M_i = I. \tag{2.17}$$

If outcome i is observed, the post-measurement state is

$$|\psi'\rangle = \frac{M_i |\psi\rangle}{\sqrt{p(i)}}, \quad p(i) = \langle\psi| M_i^\dagger M_i |\psi\rangle . \tag{2.18}$$

In quantum algorithms, a frequent task is estimating the expectation value of an observable O. An observable is a Hermitian operator, and therefore has real eigenvalues $\{\lambda_k\}_k$ and its eigenvectors $\{|\lambda_k\rangle\}_k$ form an orthogonal basis and are the possible measurement outcomes. An observable O can be expressed as follows,

$$O = \sum_k \lambda_k |\lambda_k\rangle\langle\lambda_k| . \tag{2.19}$$

The expectation value is given by

$$\langle O\rangle = \langle\psi| O |\psi\rangle = \sum_k \lambda_k |\langle\lambda_k|\psi\rangle|^2 . \tag{2.20}$$

This value may be estimated by running the circuit multiple times (shots) and computing the empirical average of the eigenvalues corresponding to the measurement outcomes. The uncertainty of this estimate scales as $\sigma/\sqrt{N}$ where σ^2 is related to the spectrum of O and N is the number of shots [5].

2.3. Parameterized Quantum Circuits

In this section, we provide more details on concepts related to concepts related to the expressivity, trainability, and hardness of Parameterized Quantum Circuits (PQC), which are the main focus of the first part of this thesis.

2.3.1. Expressivity

Expressivity in machine learning and computer science characterizes the ability of a model class $\mathcal{H}$ to represent functions within a target space $\mathcal{C}$. Formally, $\mathcal{H}$ is considered more expressive relative to $\mathcal{C}$ when it can approximate more elements of $\mathcal{C}$. The highest level of expressivity, universality, occurs when $\mathcal{H}$ is dense in $\mathcal{C}$, meaning any target in $\mathcal{C}$ can be approximated to arbitrary precision by some element in $\mathcal{H}$. Universality has been proven for several classes of classical machine learning models, including neural networks with the Universal Approximation Theorem [23, 24].

Parameterized Quantum Circuits (PQCs) form the computational foundation for diverse quantum models and variational algorithms [4], central to hybrid quantum-classical approaches in the current noisy intermediate-scale quantum (NISQ) era. Here, we formalize their structure, applications, and expressivity across diverse tasks. First, we highlight that depending on the context of their usage, the expressivity of Parameterized Quantum Circuits does not always have the same meaning. More precisely, expressivity is relative to the nature of the target space $\mathcal{C}$, which can be, for some, the set of all quantum states, while for others it is a functional space.

We distinguish between two primary uses of parameterized quantum circuits (PQCs): input-free (used as ansatzes for variational problems) and data-dependent (used in learning tasks). This distinction is fundamental to understand Chapter 3, as we transfer results for input-free Parameterized Quantum Circuits to data-dependent ones. We use universality results in a supervised context to transfer them to a generative context in Chapter 4. Finally, we use the connection between Parameterized Quantum Circuits and Fourier analysis in Chapter 5 to design a quantum learning algorithm.

Input-free Parameterized Quantum Circuits

A Parameterized Quantum Circuit is a sequence of fixed and parameterized quantum gates applied to an initial state (typically $|0\rangle^{\otimes n}$). We focus on

PQC such that their unitary evolution is defined as:

$$|\psi(\theta)\rangle = U(\theta)\,|0\rangle = \prod_{j=1}^{M} W_j e^{iV_j\theta_{m_j}}\,, \tag{2.21}$$

where $\{V_j\}_j$ are Hermitian generators (e.g. Pauli Operators), $\{W_j\}_j$ are fixed unitaries (e.g. entangling gates), and $\theta \in \mathbb{R}^M$ are trainable real parameters. The output is often the expectation value of an observable O:

$$h(\theta) = \langle 0|\, U^\dagger(\theta) O U(\theta)\,|0\rangle\,. \tag{2.22}$$

Such parameterized circuits can be used in a variety of contexts. In *Variational Quantum Eigensolver* [7], O is chosen as the Hamiltonian we seek to find the ground energy of. This is done by finding the parameters of the state $|\psi(\theta)\rangle$, also called an ansatz minimizing the function $h(\theta)$. Similarly, PQCs may also be used to solve optimization problems, with *Quantum Approximate Optimization Algorithm* [8], where the function to minimize is mapped to an observable O. PQC are also used in a generative modeling context, with *Quantum Circuit Born Machines* [44, 45] (QCBM), where they are used to sample bitstrings based on the Born rule. The goal is to find a set of parameters θ such that sampling bitstrings from the state $|\psi(\theta)\rangle$ yields a distribution over n-bitstrings as close as possible to a target distribution. In all of these cases, the expressivity of the PQC is relative to the Hilbert space. The closer the set $\{|\psi(\theta)\rangle\}_{\theta\in\mathbb{R}^M}$ is to the set of all possible quantum states, the more expressive the model is considered (to the exception that local phases do not matter for QCBM).

Finally, for *Variational Quantum Compilation*, given a target unitary V the goal is to optimize the parameters θ such that the parameterized quantum circuit $U(\theta)$ approximates V. This may be used to compile shallower circuits when the target unitaries, such as time evolution [46], or also approximating a diagonalization such as in variational fast forwarding [47]. In that case, the expressivity is relative to the set of all n-qubits unitaries and sometimes is measured relative to unitary t-designs [48].

Parameterized Quantum Circuits with inputs

Parameterized quantum circuits can, in general, have both *inputs* and *parameters*. So far, we considered PQC with only parameters, serving as ansatzes for optimization problems rather than machine learning in the sense of learning from data. In this subsection, we consider PQC with inputs, which are used to upload data from machine learning tasks. In

this thesis, we consider two different machine learning contexts, supervised learning and generative modeling. In supervised learning, given pairs of input-output $(x_t, y_t)_t$, the goal is to return a function h that mimics the underlying relationship f between the input and the output $y = f(x)$. In generative modeling, given a set of inputs sampled from an unknown distribution p, the goal is to return a sampling algorithm whose underlying distribution q approximates the target distribution.

It is possible to make a quantum state that is input-dependent, or, in other words, onto which data is encoded. We discuss different architectures of input-dependent Parameterized Quantum Circuits in the case where inputs are D-dimensional real vectors $x \in \mathbb{R}^D$, as follows,

$$|\psi(x,\theta)\rangle = U(x,\theta)\,|0\rangle\ . \tag{2.23}$$

We do not cover all architectures exhaustively, but we cover some of the most commonly used ones. A first natural encoding is the so-called amplitude encoding, where the overlap of the state with the k-th computational basis state is the k-th coordinate of the normalized data as follows, resulting in a qubit-dense encoding, where $n = \log(D)$ qubits are needed.

$$U(x)\,|0\rangle = \|x\|_2^{-1/2} \sum_{d=0}^{D-1} x_d\,|d\rangle\ . \tag{2.24}$$

Another option to encode data onto a quantum state yields product states, and requires significantly more qubits with $n = D$.

$$U(x)\,|0\rangle = \otimes_k R_y(x_k)\,|0\rangle = \otimes_k(\cos(x_k)\,|0\rangle + \sin(x_k)\,|1\rangle)\,. \tag{2.25}$$

However, such models have limited expressivity with respect to the set of all possible functions. So-called *quantum reuploading models* were introduced by [25, 49] to propose a richer expressivity. Such circuits interleave data-dependent gates and parameterized gates. The most common form for such models is

$$|\psi(x,\theta)\rangle = U(x,\theta)\,|0\rangle = \prod_{j=1}^{L} W_j(\theta) e^{iV_j x_{m_j}}\ , \tag{2.26}$$

where each $W_j(\theta)$ is a parameterized quantum circuit and $\{V_j\}_j$ are Hermitian generators. The terms of this product are often called layers. Other forms of quantum reuploading models with alternative data encoding gates are considered in e.g. [41].

Usually, the expectation value of an observable O is returned, and the parameterized quantum circuit is the key component of a parameterized function, as follows

$$h_\theta : x \to \langle 0| U^\dagger(x,\theta) O U(x,\theta) |0\rangle \ . \tag{2.27}$$

Typically, this parameterized set of functions is used in supervised learning, where the goal is to approximate the relation between inputs and labels. That is, for a dataset is $\{(x_t, y_t = f(x_t))\}$ for an unknown function f we optimize the parameters θ such that h_θ approximate f.

The expressivity of such circuits is well-understood, and their universality has been proven for several architectures [25, 26, 41]. In general the function they represent can be expressed as generic trigonometric polynomials [26] (GTP), that is there exist a finite set of d-dimensional real vectors $\Omega = \{\omega_l \in \mathbb{R}^d\}_l$ that are usually called frequencies, and a vector of complex numbers $x \in \mathbb{C}^{|\Omega|}$ such that

$$h_\theta(x) = \sum c_l(\theta) e^{i\omega_l \cdot x} \ . \tag{2.28}$$

Leaving the generators V_j free, with, for example, the possibility to have scaling parameters (e.g. $V_j = \alpha Z$), yields universality on a single qubit [25].

A version of this model restricted to Pauli encodings, that is, all V_j must be Pauli strings, is often used, as in the definition below.

Definition 2.4 (Pauli encoding)
A Pauli-encoded circuit is a parameterized quantum circuit on n qubits $U : x \in [0,1]^D \to \mathcal{U}(2^n)$. It is composed of $N_f \in \text{poly}(n)$ fixed unitary gates and $L \in \text{poly}(n)$ parameterized gates $\{V_l(x) \coloneqq e^{i\pi P_l x_{i_l}}\}_{1 \leq l \leq L}$ where P_l are Pauli strings.

When generators are restricted to the set of Pauli strings, the set of frequencies is restricted to a finite set of integers $\Omega = [-L, +L]^d$, and the GTP is de facto a finite Fourier decomposition and a universal model [26].

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

$$f(x) = \langle 0| U^\dagger(x) O U(x) |0\rangle \ . \tag{2.29}$$

In the case of quantum reuploading models with Pauli encodings as in

Definition 2.4, we have that

$$f(x) = \sum_{l \in [-2L, 2L]^D} b_l e^{i\pi x \cdot l} \,. \tag{2.30}$$

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

So far, we have seen ways to use parameterized quantum circuits as explicit models, where the labels are specified by the output of the circuit, but it is also possible to use them as implicit models, such as in kernel methods. In this other option, the function $x \to |\psi(x)\rangle$ becomes a feature map, and a quantum computer is used to compute the corresponding kernel values as $k(x, x') = |\langle \psi(x) \mid \psi(x') \rangle|^2$. The difference between three different models, linear models (where $U(x, \theta) = U'(x)U''(\theta)$), Quantum Reuploading models, and quantum kernels have been studied in [50]. This work also shows equivalences between these models at the cost of more computational resources.

It is crucial to note that the notion of expressivity is very different between input-free PQC and data-dependent PQC. In the first case, the expressivity is relative to the Hilbert space, while in the data-dependent it is relative to a set of functions for supervised learning. We explore this distinction deeper in Chapter 3. For generative modeling, the expressivity is relative to the set of multivariate distributions, we explore this further in Chapter 4 and establish universality results. Finally, we use the connection between PQC and Fourier analysis to prove quantum advantage in learning quantum dynamics in Chapter 5.

2.3.2. Trainability

The training of parameterized quantum circuits (PQCs) involves an optimization procedure that searches for parameter sets minimizing a cost function $\mathcal{L}(\theta)$. The feasibility of this optimization task is captured by the concept of *trainability*, which has been extensively studied in quantum machine learning contexts [4, 11].

In typical implementations, this is done in a hybrid architecture with calculation taking place alternatively in quantum and classical computers. A quantum computer is used to estimate $\mathcal{L}(\theta)$ and its gradients $\partial_\theta \mathcal{L}(\theta)$. A classical computer then updates the parameters using gradient descent, iterating until convergence.

In quantum machine learning, trainability challenges emerge from characteristics of the cost landscape, including non-convex landscapes with

numerous local minima [51]. A fundamental limitation is the *barren plateau (BP) phenomenon* [11], where gradients vanish exponentially with the system size

$$\mathbb{E}_\theta[\partial_\theta \mathcal{L}(\theta)] \in \mathcal{O}(e^{-n}) \tag{2.31}$$

Other formulations were found to be equivalent, such as the variance of the cost function itself vanishing exponentially.

As the estimation of gradients is affected by shot noise, when they are exponentially small, they provide no useful information for parameter updates, and this leads to a situation where the optimization becomes ineffective. This phenomenon is prevalent in many PQC architectures, particularly those with high expressivity [52], entanglement [53], global observables [54], and noise in quantum hardware [55]. Gradient-free optimization methods (e.g., COBYLA, SPSA) also fail in BP regimes due to exponentially small cost function variance [56]. Therefore, understanding how the gradients' magnitude scales with the number of qubits is crucial; we address ways to bound them in Chapter 3.

To address the BP problem, several architectures have been proposed that are provably BP-free, meaning they can be trained efficiently without encountering barren plateaus. Among others, there are *Symmetry-preserving ansatzes* [57], *Shallow circuits* with $\mathcal{O}(\log n)$ depth with local observables [54] and PQCs with polynomially sized *Dynamical Lie Algebras* [58]. Initialization strategies have also been proposed to mitigate the effects of BP in PQCs, a strategy also called warm start [59].

However, these BP-resistant architectures were matched with classical strategies to simulate them efficiently. In Section 2.3.3 we provide examples of such classical algorithms that are able to replace PQCs in some cases. We review existing results about the classical hardness, or lack thereof, of several learning tasks.

2.3.3. Hardness of learning

Trainability vs simulability

Recent theoretical work [18] has established a connection between barren plateau (BP) mitigation strategies and classical simulability in parameterized quantum circuits. It presents evidence that architectures with provable BP absence are classically simulable with polynomial-time classical algorithms. This follows from the observation that BP avoidance strategies inherently confine computations to polynomially-sized subspaces of the full Hilbert space, which can be efficiently characterized and simulated classically. This pattern holds across several major BP-

mitigation strategies including shallow circuits [54], geometrically local observables [60], symmetry-preserving ansatzes [57], and tensor-network based designs [61, 62]. This work highlighted the thin gap between trainability and simulability, and pushed the community to find alternative approaches to quantum machine learning.

Random Fourier Features

The concept of finding efficient classical alternatives to quantum algorithms is known as dequantization. In addition to the aforementioned results on trainability and simulability, another classical learning algorithm was shown to be able to dequantize quantum machine learning in some cases. The central idea of this dequantization method we present now revolves around the fact that most used PQC architectures can be expressed as generic trigonometric polynomials (GTP). This makes them a natural candidate for classical learning algorithms based on so-called Random Fourier Features. The idea is to move away from trying to simulate PQCs, but rather to mimic the associated hypothesis class. Random Fourier Features are defined for a probability distribution over a set of frequencies $p_\omega : \Omega \to [0, 1]$. This yields a random feature map Φ, as a stack of L trigonometric functions for frequencies sampled from the distribution p_ω, as follows,

$$x \xrightarrow{\Phi} [e^{i\omega_l \cdot x}]_{l\in[1,L]}\,, \omega_l \overset{i.i.d}{\sim} p_\omega\,. \tag{2.32}$$

A number of sufficient conditions were found for RFF-based algorithms to have a performance matching or exceeding that of PQC-based algorithms in terms of true risk (definition in Section 2.1). Several theorems [40, 63] have been found to cover most cases: explicit PQC based expression as well as implicit models based on kernels, for classification and regression tasks. Overall, the sufficient conditions are properties of the Fourier transform of the optimal PQC function, normalized to be a distribution q_ω, and the distribution of the RFF p_ω.

Theorem 2.1 (RFF dequantization theorem, informal)
For any learning task, the performance in terms of true risk of an RFF-based algorithm with a probability distribution p_ω exceeds or matches that of any PQC machine learning scheme outputting the optimal hypothesis function, for which we write the associated frequency probability q_ω, if all the following conditions are met:

1. *p_ω is easy to sample from,*

2. *q_ω and p_ω are aligned, that is, their dot product is at least inverse polynomial,*

3. *p_ω is concentrated, that is, its max norm is at least inverse polynomial.*

Dequantization is usually defined as the existence of a p_ω that satisfies such conditions. But then the question of how easily one may find such a distribution remains open [63]. Leaving the domain of provable performance, this algorithm can still be a good heuristic in its own right.

Provable learning separations

In the light of these dequantization results, it is crucial to identify regimes where quantum computers are not merely *used* for machine learning tasks, but truly *necessary*. These situations are called *learning separations* and are characterized by the existence of learning tasks that classical learners cannot address efficiently, but quantum algorithms can, under widely believed complexity assumptions.

Some examples of learning separation based on cryptographic tasks were found. These famously include a learning problem based on the *discrete log* [20], which goes as follows. Given an n-bit prime number p and a generator a of the multiplicative group of integers modulo p, written as $\mathbb{Z}_p^*$, define for each fixed unknown integer $i \in [0, p-1]$ the function:

$$c_i : x \to (\log_a(x) \mod p) \in [i, i + \frac{p-3}{2}]\,. \tag{2.33}$$

The goal is to predict the labels of unseen outputs given access to a dataset of input-output pairs $(x_t, c_i(x_t))$ for T samples. Another notable learning exponential separation has been proven for Boolean functions [64], and a sub-exponential learning separation for a sequence modeling task was found in [65].

In [21], the authors propose that learning separations can be constructed based on BQP-complete problems. Consider a polynomially sized concept class (definition in Section 2.1) constituted of concepts that are all in BQP and at least one being BQP complete. Using a result from [66] that states that the learnability of a concept class implies the efficient evaluation of all the concepts, the considered concept class is hard to learn for a classical learner under widely believed complexity assumptions. It can also be learned efficiently with a quantum computer, by brute force, testing all possible concepts.

In [22], the authors propose a concept class that does not need to be polynomially sized to exhibit a quantum learning separation. We provide more details on how to define the size of a set of functions in Section 2.1.2. Consider a quantum state that depends on the input data $\rho(x)$, and an Observable O with an unknown sparse representation in the Pauli basis as follows

$$O(\alpha) = \sum_{i=1}^{m} \alpha_i P_i \,. \tag{2.34}$$

Define the functions $c_\alpha(x) = \mathrm{Tr}[\rho(x)O(\alpha)]$, and let them constitute the concept class $\mathcal{C} = \{c_\alpha\}_{\alpha\in[-1,1]^m}$. This concept class was proven to exhibit an exponential learning separation.

In Chapter 5, we will prove a quantum advantage for a concept class based on unknown Hamiltonian dynamics.

2.4. Bosonic Hamiltonians and Continuous Variables Quantum Computing

In Chapter 6 and Chapter 7 we will consider quantum circuits based on bosonic modes, with infinite-dimensional Hilbert spaces. In this section, we introduce the necessary background on bosonic Hamiltonians and continuous variable quantum information.

2.4.1. Continuous Variable Quantum Information

We start by introducing the central ideas of Continuous Variable Quantum Information [67]. In contrast to qubit-based computation, where observables can only take a finite discrete set of values upon measurement, Continuous Variable (CV) observables can take a value from an infinite number of values over a continuous interval. Quantum states can be expressed in terms of creation $\hat{a}^\dagger$ and annihilation $\hat{a}$ operators defined such that $\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle$, where $|n\rangle$ are the Fock states in a countably infinite-dimensional Hilbert space. The quadrature operators are the position operator $\hat{q}$ and its conjugate operator is the momentum operator $\hat{p}$,

defined as follows,

$$\hat{q} = \frac{\hat{a} + \hat{a}^\dagger}{\sqrt{2}}\,, \tag{2.35}$$

$$\hat{p} = \mathbf{i}\frac{\hat{a} - \hat{a}^\dagger}{\sqrt{2}}\,. \tag{2.36}$$

Their commutation relation is given by $[\hat{q}, \hat{p}] = i$. The quadrature operators are self-adjoint and have a continuous spectrum with eigenvalues x spanning $\mathbb{R}$, and their corresponding eigenvectors $|x\rangle_{\hat{q}}$ for the position operator and $|x\rangle_{\hat{p}}$ for the momentum operator. These eigenvectors form a full basis of the Hilbert space:

$$I = \int_{-\infty}^{+\infty} |x\rangle_{\hat{q}} \langle x|_{\hat{q}}\, dx = \int_{-\infty}^{+\infty} |x\rangle_{\hat{p}} \langle x|_{\hat{p}}\,. \tag{2.37}$$

The conditions for a set of bosonic gates to be universal, that is, to be able to approximate any unitary transformation, were derived in [29]. The first set of gates of interest is the Gaussian gates, under which the set of Gaussian states is invariant. A Gaussian state is completely specified by its first and second moments of the quadrature operators. We provide more details on Gaussian states and Gaussian gates in Section 2.4.2. Gaussian gates applied to Gaussian states can be efficiently classically simulated [68] for a polynomial number of modes. This result is in some regard analogous to the Gottesman-Knill theorem on efficient simulation of Clifford circuits [12]. We present complexity results on the simulation of Gaussian circuits on exponentially many modes in Chapter 7. The generator of any Gaussian gate can be expressed as a quadratic polynomial in the quadrature operators (e.g. $\hat{p}^2 + \hat{q}$).

In order to reach universality, a non-Gaussian gate with a higher degree in quadrature operator has to be added to the model of computation [67], the most common non-Gaussian gates being either the cubic gate $e^{it\hat{q}^3}$ and the Kerr gate $e^{it\left(\hat{q}^2+\hat{p}^2\right)^2}$. These non-Gaussian gates are very difficult to implement in practice, and can therefore be considered "expensive". For the purpose of Chapter 6, we choose our model of computation as being the following set of gates, defined by their corresponding generators:

- the Displacement Gate: $H = \hat{p}$
- the Squeezing Gate: $H = \hat{p}\hat{q} + \hat{q}\hat{p}$
- the Quadratic Gate: $H = \hat{p}^2$

- the Cubic Gate: $H = \hat{q}^3$
- the Beam-splitter: $H = \hat{q}_1\hat{p}_2 - \hat{q}_2\hat{p}_1$

2.4.2. Gaussian states

In what follows, we will consider systems composed of M bosonic modes (with $M = 2^n$ in Chapter 7). Let $\hat{a}_m^\dagger$ and $\hat{a}_m$, with $m = 1, \ldots, M$, respectively denote the creation and annihilation operators for the m-th mode [67]. We consider the standard Hermitian *quadrature operators* as defined in Section 2.4.1, position $\hat{q}_m = \frac{1}{\sqrt{2}}(\hat{a}_m + \hat{a}_m^\dagger)$ and momentum $\hat{p}_m = \frac{i}{\sqrt{2}}(\hat{a}_m^\dagger - \hat{a}_m)$. They satisfy the canonical commutation relations $[\hat{q}_m, \hat{p}_{m'}] = i\delta_{mm'}$. Furthermore, in Chapter 7, we will focus on the case where an M-mode bosonic state ρ_0 evolves under the action of a Gaussian Bosonic (GB) circuit whose gates are generated by time-independent Hamiltonians that are quadratic in the position and momentum operators[2].

These GB generators, also known as free-bosonic generators, are arbitrary real-valued degree-two homogeneous polynomials on the quadrature operators

$$\hat{H} = \frac{1}{2}\hat{\vec{z}}^T K \hat{\vec{z}}, \text{ with } \hat{\vec{z}} = (\hat{q}_1, \ldots, \hat{q}_M, \hat{p}_1, \ldots, \hat{p}_M)^T, \tag{2.38}$$

where K is a real $2M \times 2M$ symmetric matrix. The vector $\hat{\vec{z}}$ allows us to express the commutation relations in the compact form $[\hat{\vec{z}}_\alpha, \hat{\vec{z}}_\beta] = i\Omega_{\alpha\beta}$, where $\Omega = iY \otimes I_M$. Here, Y is the usual 2×2 Pauli matrix and I_M the $M \times M$ identity matrix.

Next, we collect the expectation values of the position and momentum operators in a vector $\langle\hat{\vec{z}}\rangle = (\langle\hat{q}_1\rangle, \ldots, \langle\hat{q}_M\rangle, \langle\hat{p}_1\rangle, \ldots, \langle\hat{p}_M\rangle)^T \in \mathbb{R}^{2M}$, with $\langle\hat{x}\rangle$ the expectation value of $\hat{x}$ over ρ_0. As shown in the Supplemental Information (SI), evolving ρ_0 with a unitary generated by a GB generator $\hat{H}$ for a time t induces the evolution of $\langle\hat{\vec{z}}\rangle$ as

$$\frac{\partial\langle\hat{\vec{z}}\rangle}{\partial t} = \Omega K \langle\hat{\vec{z}}\rangle\,, \text{ so that } \langle\hat{\vec{z}}\rangle(t) = e^{t\Omega K}\langle\hat{\vec{z}}\rangle(0)\,. \tag{2.39}$$

Here, $\langle\hat{\vec{z}}\rangle(t)$ denotes the vector containing the expectation values of positions and momenta at time t, and thus $\langle\hat{\vec{z}}\rangle(0)$ represents the initial condition. Since the canonical commutation relations must be preserved,

[2] A generalization to time-dependent Hamiltonians is direct using standard Hamiltonian-simulation techniques [69].

the propagator $e^{t\Omega K}$ is a $2M \times 2M$ symplectic matrix with real entries belonging to the Lie group $\mathbb{SP}(M, \mathbb{R})$ [67], which in turn implies that ΩK is an operator in the symplectic Lie algebra $\mathfrak{sp}(M, \mathbb{R})$.

Note that while, in general, $e^{t\Omega K}$ is not unitary, we characterize the quadratic Hamiltonians leading to unitary evolutions of the vector $\langle \hat{\vec{z}} \rangle$, when a gate generator is of the form $\hat{H} = \sum_{m,m'=1}^{M} h_{mm'}\, \hat{a}_m^\dagger \hat{a}_{m'} + \frac{\mathrm{Tr}[h]}{2} I_{2M}$, with h a Hermitian matrix. Then $[\Omega, K] = 0$, and the propagator $e^{t\Omega K}$ is the real-time evolution of the Hermitian $i\Omega K$. Hamiltonians of this form are known as *particle preserving* [70] since the hopping terms $\hat{a}_m^\dagger \hat{a}_{m'}$ move a boson from mode m' to mode m. We also characterize *non-particle-preserving Hamiltonians* of the form $\hat{H} = \sum_{m,m'=1}^{M} \Delta_{mm'}^\dagger\, \hat{a}_m \hat{a}_{m'} + \text{h.c.}$, whose propagator corresponds to the imaginary-time evolution of $\langle \hat{\vec{z}} \rangle$ under the effective Hamiltonian $-\Omega K$.

In addition to $\langle \hat{\vec{z}} \rangle$, we also collect the expectation value of products of quadrature operators over ρ_0 in the $2M \times 2M$ positive-definite covariance matrix $\vec{\sigma}$ whose entries are given by $\vec{\sigma}_{\alpha\beta} = \frac{1}{2}\langle \hat{\vec{z}}_\alpha \hat{\vec{z}}_\beta + \hat{\vec{z}}_\beta \hat{\vec{z}}_\alpha \rangle - \langle \hat{\vec{z}}_\alpha \rangle \langle \hat{\vec{z}}_\beta \rangle$ [71]. Analogously to Equation (2.39), we find (see the SI) that

$$\frac{\partial \vec{\sigma}}{\partial t} = \Omega K \vec{\sigma} - \vec{\sigma} K \Omega\,, \;\; \text{so that} \;\; \vec{\sigma}(t) = e^{\Omega K t} \vec{\sigma}(0) e^{\mp \Omega K t}\,,$$

where the $-\,(+)$ sign corresponds to particle (non-particle) preserving Hamiltonians, as defined above.

The previous insights pave the way to simulate the action of GB circuits on a gate-based quantum computer in Chapter 7.

2.5. Complexity Theory and Quantum Classes

Complexity theory provides a framework for classifying computational problems according to the resources required to solve them. It is fundamental for understanding the capabilities and limits of both classical and quantum computers. Two commonly studied classical complexity classes are:

- **P**: problems solvable by a deterministic Turing machine in polynomial time.
- **NP**: problems for which a candidate solution can be verified in polynomial time.

The question of whether P = NP is still open and lies at the heart of classical computational complexity.

Quantum computing introduced new models of computation, leading to a rich hierarchy of complexity classes capturing the power of quantum algorithms and quantum verification. This section defines the most relevant quantum classes, BQP, QMA, PostBQP, and DQC1 and explains their significance. But first we recall some basic definitions from complexity theory.

2.5.1. Inclusion, hardness, and completeness

A problem L is said to be *in* a complexity class $\mathcal{C}$ if there exists an algorithm in $\mathcal{C}$ that solves L correctly on all inputs.

A problem H is *$\mathcal{C}$-hard* if every problem $L \in \mathcal{C}$ reduces to H under polynomial-time reductions. In other words, an efficient solution for H would yield efficient solutions for all the problems in $\mathcal{C}$.

A problem is *$\mathcal{C}$-complete* if it is both in $\mathcal{C}$ and $\mathcal{C}$-hard. Such problems are the most representative of their class $\mathcal{C}$ and the hardest within it: solving any $\mathcal{C}$-complete problem efficiently implies solving all problems in $\mathcal{C}$ efficiently.

2.5.2. The class BQP

Definition 2.5 (BQP)
The class Bounded-error Quantum Polynomial time *(BQP) contains all decision problems solvable by a polynomial-size quantum circuit in polynomial time, with error probability at most* $1/3$ *for all instances. That is, for input* x*, the quantum algorithm outputs the correct answer with probability* $\geq 2/3$.

The choice of $1/3$ is arbitrary and could be any constant $< 1/2$: by repeating the algorithm and taking a majority vote, the probability of error can be made exponentially small. Examples of PromiseBQP-complete problems include:

- **Simulation of quantum circuits** Given a quantum circuit decide whether it outputs $|1\rangle$ with probability at least $2/3$ or at most $1/3$ when measuring the first qubit, promised that either is true.

- **Simulation of exponentially many coupled oscillators** as in [42].

2.5.3. The class QMA

The class *Quantum Merlin-Arthur* (QMA) generalizes NP to the quantum setting.

Definition 2.6 (QMA)
A decision problem L is in QMA if there exists a quantum verifier $|\psi\rangle$ running in polynomial-time such that:

- *For every $x \in L$, the quantum verifier accepts with probability $\geq 2/3$.*
- *For every $x \notin L$, all quantum proofs $|\psi\rangle$ are rejected with probability $\geq 2/3$.*

Here, Merlin provides the quantum proof, and Arthur verifies it with a quantum computer. Examples of QMA-complete problems include:

- **Local Hamiltonian problem**: Given a $k \geq 2$-local Hamiltonian $H = \sum_j H_j$, decide whether the ground state energy is below a or above b, promised that $b - a \geq 1/\text{poly}(n)$ [72].
- **Quantum k-SAT**: Decide whether there exists a quantum state satisfying all given quantum constraints [73].
- **Betti numbers**: Decide whether the k-th Betti number of a simplicial complex is zero or non-zero [74].

2.5.4. The class PostBQP

Definition 2.7 (PostBQP)
The class Post-selected BQP *(PostBQP) consists of problems solvable by a polynomial-time quantum algorithm that is allowed to condition on measurement outcomes (post-selection). Formally, the algorithm may discard runs where a certain measurement outcome does not occur, even if the probability of success is arbitrarily small.*

Remark: Aaronson showed that PostBQP coincides with the classical class PP (Probabilistic Polynomial time) [34], showing how much post-selection increases the computational power of quantum computers.

Examples of PostBQP-complete problems include:

- **Majority Boolean Formula Problem** Given a Boolean formula F on n variables, decide whether strictly more than half of all 2^n assignments satisfy F.

- **Simulation of post-selected quantum circuits** Given a quantum circuit post-selected on the first qubit measured to be $|1\rangle$ (promised that the probability is non-zero) decide whether it outputs $|1\rangle$ with probability at least 2/3 or at most 1/3 when measuring the second qubit, promised that either is true.

2.5.5. The class DQC1

Definition 2.8 (DQC1)
The class Deterministic Quantum Computation with One Clean Qubit *(DQC1) consists of decision problems solvable by a polynomial-time quantum algorithm that starts with only one qubit in a pure state (the "clean qubit") and all other qubits in the maximally mixed state. The algorithm then applies a polynomial-size quantum circuit and measures the clean qubit to determine the output.*

This model, introduced by Knill and Laflamme [75], was designed to probe the computational capabilities of highly noisy quantum systems. Despite its severe restrictions, essentially a quantum computer with only minimal quantum resources, it can efficiently solve certain problems that are believed to be classically intractable [76]. Positioned between P and BQP in computational power, DQC1 serves as a natural complexity class for describing near-term quantum devices operating with limited coherence and entanglement.

Examples of DQC1 problems include:

- **Normalized trace estimation**: Given a unitary matrix U (described as a quantum circuit), estimate $\mathrm{Tr}(U)/2^n$ to additive inverse polynomial accuracy.
- **Approximating the Jones polynomial at certain roots of unity**: restricted to links presented as the trace closure of braids [77].