Computational speedups and learning separations in quantum machine learning

Gyurik, C.

Computational speedups and learning separations in quantum machine learning

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof. dr. ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op donderdag 4 april 2024
klokke 11.15 uur

door

Casper Ferenc Sàndor Gyurik

geboren te Elst

in 1994

Promotores:

Prof. dr. V. Dunjko
Prof. dr. A. Plaat

Promotiecomissie:

Prof. dr. T.H.W Bäck	
Prof. dr. M.M. Bonsangue	
Prof. dr. J. Calsamiglia Costa	(Universitat Autònoma de Barcelona)
Dr. S. Jeffery	(Centrum Wiskunde & Informatica)
Dr. A.W. Laarman	

This work was supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.03).

Contents

Acknowledgements

Firstly, I want to express my gratitude to my supervisor Vedran Dunjko for being a continuous driving force in my development. Every interaction I had with Vedran made me feel that his sole purpose was to help me improve in everything I wanted to achieve. Our meetings ranged from challenging to enlightening and roundabout fun, and I will truly miss these interactions.

After work, I was able to fully unwind and enjoy my off hours, thanks to the incredible warmth I felt at home from my dear partner Sophie. Being able to share my life with such a driven and outgoing person who also knows how to appreciate things beyond work was both inspiring and enabled me to enjoy so much of what life has to offer.

I want to express my gratitude to my parents, Andor & Marjolijne, and my brother Lucas for providing me with a solid foundation of love and support. Without the love and effort my parents put into providing me with the opportunities that were given to me, none of the life I currently enjoy would have been possible. It was great experiencing how proud my parents and my brother always were of me, which always motivated me to be even better.

I want to say thank you to my friends Marty, Coenen, Sjawty, and Spitter for our wonderful adventures, from Saint Petersburg to Lake Balaton and the Ruk & Pluk. Also, I want to say thank you to the players and staff of AFC Ajax (with a special thanks to Dušan Tadić, Hakim Ziyech, and Erik ten Hag) for winning the Eredivisie all four years of my PhD and for their 2018/2019 Champions League run, providing me with blissful distractions from work.

Chapter 1

Introduction

In this chapter, we provide an introduction to the topics discussed in this thesis. We start by discussing the problem statement and research questions covered by this thesis, which can be found in Section 1.1. Afterwards, in Section 1.2, we give a detailed overview of the contributions of this thesis.

1.1 Problem statement and research questions

Quantum machine learning (QML) is a rapidly growing field that has brought forth numerous proposals regarding ways for quantum computers to help analyze data. Several of these proposals involve using quantum algorithms for linear algebra – most notably Harrow, Hassidim and Lloyd's matrix inversion algorithm [100] – to exponentially speed up tasks in machine learning. Other proposals involve using parameterized quantum circuits [102, 167, 31] to identify novel quantum learning models that are better suited to the limitations of near-term quantum computing, rather focussing on improving established classical methods. These QML proposals have all been hailed as possible examples of quantum computing's "killer application": genuinely and broadly useful quantum algorithms that (superpolynomially) outperform their best known classical counterparts. In this thesis, we study several of these QML proposals and we investigate whether and how they are able to provide (superpolynomial) speedups over their classical counterparts, and how to get the best possible performance out of these proposals. Specifically, the problem statement of this thesis is:

Problem statement. *Can we provide evidence that various QML proposals can (superpolynomially) outperform their classical counterparts, and what methods can we devise to attain their best possible performance?*

The first QML proposal we study concerns quantum algorithms for topological data analysis, as first introduced by Lloyd, Garnerone and Zanardi [130]. Specifically, in Chapter 3, we study the potential for quantum algorithms to achieve superpolynomial speedups for linear-algebraic problems in topological data analysis. An important linear-algebraic problem that arises in topological data analysis is that of computing

so-called *Betti numbers*, which can be formulated as computing the dimension of the kernel of a set of matrices whose entries encode the connectivity of cliques (i.e., complete subgraphs) in a given graph. While Lloyd, Garnerone and Zanardi developed a quantum algorithm that in certain regimes is able to solve this problem superpolynomially faster than the best-known classical algorithm [130], it is unclear whether this speedup will persist with the development of new classical algorithms. In particular, for several other linear-algebraic QML proposals the previously speculated superpolynomial speedups were revealed to actually be at most polynomial speedups, as exponentially faster classical algorithms were devised that operate under analogous assumptions [186, 60] (an event called "dequantizations"). While polynomial speedups have appeal on paper, an analysis involving near-term device properties revealed that low-degree polynomial improvements are not expected to translate to real-world advantages due to various overheads [27]. Thus, finding superpolynomial speedups is of great importance, especially in the early days of practical quantum computing. Therefore, we examine the linear-algebraic QML algorithms for Betti numbers and study whether speculated superpolynomial quantum speedups will not be lost due to development of better classical algorithms.

Research question 1. *Can the linear-algebraic QML algorithms for Betti numbers maintain their speculated superpolynomial quantum speedups, even with the development of better classical algorithms?*

Next, we turn our focus to QML proposals that involve the use of parameterized quantum circuits [102, 167, 31] to build genuinely new quantum machine learning models. Specifically, in Chapter 4, we study how to optimally tune a family of these new quantum machine learning models based on the principles of *structural risk minimization*. In the context of structural risk minimization, it is crucial to identify the tunable aspects a model, i.e., hyperparameters, that impact both its performance on training data and its generalization performance. An important part of this investigation involves characterizing complexity measures, such as the VC-dimension or fat-shattering dimension, associated with these models and understanding how hyperparameters influence these complexity measures. Consequently, we focus on characterizing complexity measures of novel quantum learning models based in parameterized quantum circuits. In particular, we aim to identify the hyperparameters that exert influence over these complexity measures, a critical step in effectively implementing structural risk minimization.

Research question 2. *Can we identify hyperparameters within novel quantum learning models based on parameterized quantum circuits that impact both complexity measures and performance on training data, as is crucial for the successful implementation of structural risk minimization?*

Afterwards, in Chapter 5, we study how these new quantum machine learning models based on parameterized quantum circuits [102, 167, 31] can be used in the field of reinforcement learning. Reinforcement learning is a flavour of learning where one has to learn through interacting with an environment and adjusting its behaviour based on rewards obtained (i.e., there is no large amount of labeled data available). Arguably, the largest impact quantum computing can have is by providing enhancements to the

hardest learning problems. From this perspective, reinforcement learning stands out as a field that can greatly benefit from a powerful hypothesis family. Nonetheless, the true potential of near-term quantum approaches in reinforcement learning remains very little explored. The few existing works [58, 131, 202, 110] have failed so far at solving classical benchmarking tasks using PQCs and left open the question of their ability to provide a learning advantage. Consequently, we focus on designing quantum machine learning models based on parameterized quantum circuits that are on par with the best classical models in standard benchmarking tasks, but also outperform all classical models in certain other tasks.

Research question 3. *How can new quantum machine learning models based on parameterized quantum circuits be effectively leveraged within the realm of reinforcement learning? Specifically, can these quantum approaches demonstrate the potential to be on par with classical models in standard benchmarking tasks and outperform them in novel specific scenarios?*

Finally, in Chapter 6, we tackle the challenge of identifying learning problems that exhibit provable exponential speedup for quantum learning algorithms compared to classical learning algorithms. The first thing we address is that there is no single definition of what precisely constitutes a *learning* separation. In particular, when trying to come up with a definition there are many choices to be made, and various choices make sense depending on the particular settings. This ambiguity can lead to conflating the task of learning in an intuitive sense with a purely computational task. Moreover, we study existing learning separations [126, 173] and carefully delineate where the classical hardness of learning lies and the types of learning separations they achieve. Next, we set out to find new examples of learning separations where the classical hardness lies more in learning in an intuitive sense rather than evaluating the functions to be learned. Finally, we turn our attention to the folklore in the community that states that quantum machine learning is most likely to have advantages when the data is quantum-generated. However, it is not immediately clear how quantum-generated data can give rise to learning separations. We address this question by exploring the additional complexity-theoretic assumptions required to build such a learning separation.

Research question 4. *How can we identify learning problems that exhibit a provable exponential speedup for quantum learning algorithms compared to their classical counterparts, and can we confirm the validity of the folklore that quantum machine learning excels when handling quantum-generated data?*

1.2 List of contributions

- **Chapter 3:** We show that the linear-algebraic methods underlying the algorithm of Lloyd et al. are "safe" against general dequantization approaches of the type introduced in [186, 60], and that the corresponding computational problem (i.e., a generalization of Betti numbers) is classically intractable under widely-believed assumptions. Specifically, we show that a natural generalization of the Betti number problem is hard for the complexity class DQC1 (see Section 3.2).

Since the hard problems in DQC1 are widely-believed to require superpolynomial time on a classical computer, this ensures that our generalization also most likely requires superpolynomial time on a classical computer. This further establishes the potential of these methods to be a source of useful quantum algorithms with superpolynomial speedups over classical methods. We concretely demonstrate the potential of these methods by connecting them to practical problems in machine learning and complex network analysis (see Section 3.3). Finally, we provide examples of instances that satisfy the requirements for the quantum algorithm to be efficient, while making sure the best-known classical algorithms are not efficient (see Section 3.2.4).

- **Chapter 4:** By exploiting a connection between the new quantum learning models and certain established classical learning models (i.e., linear-classifiers) we characterize some complexity measures of the new quantum learning models (i.e., their VC- and fat-shattering dimension). Since these complexity measures characterize the generalization performance of the machine learning model, our results give rise to ways to fine-tune your model such that you perform well on training data, while ensuring that the generalization performance remains good enough (i.e., the principle of structural risk minimization).

- **Chapter 5:** We exhibit how to use quantum machine learning models based on parameterized quantum circuits to solve problems in reinforcement learning using the policy-gradient algorithm. Next, we build reinforcement learning settings where we (*i*) can prove that the quantum learning model performs much better than any classical learner, and (*ii*) can provide numerical evidence that the quantum learning model performs much better than any deep neural network based learner (i.e., the current classical state of the art).

- **Chapter 6:** We delve into the nuances of computational learning theory and highlight how subtle variations in definitions lead to distinct requirements and tasks for learners. Next, we examine existing learning problems demonstrating provable quantum speedups [126, 173] and observe their reliance on the classical complexity of evaluating the data-generating function rather than its identification. To address this limitation, we present two novel learning scenarios where the primary classical challenge lies in identifying the underlying function generating the data. Additionally, we explore computational hardness assumptions that can be utilized to establish quantum speedups in situations where the data is quantum-generated. This implies quantum advantages in various natural settings such as condensed matter and high-energy physics.

This thesis is based on the following papers:

[1] Casper Gyurik, Chris Cade, and Vedran Dunjko. Towards quantum advantage via topological data analysis. *Quantum*, 6, 2022.

[2] Dominic W Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. Analyzing prospects for quantum advantage in topological data analysis. *PRX Quantum*, 5, 2024.

[3] Casper Gyurik, Dyon van Vreumingen, and Vedran Dunjko. Structural risk minimization for quantum linear classifiers. *Quantum*, 7, 2023.

[4] Casper Gyurik and Vedran Dunjko. Exponential separations between classical and quantum learners. arXiv:2306.16028, 2023.

[5] Sofiene Jerbi, Casper Gyurik, Simon Marshall, Hans Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

In the course of their PhD, the author has additionally co-authored the following articles that are not included in this thesis:

[6] Simon Marshall, Casper Gyurik, and Vedran Dunjko. High dimensional quantum machine learning with small quantum computers. *Quantum*, 7, 2023.

[7] Sofiene Jerbi, Casper Gyurik, Simon Marshall, Riccardo Molteni, and Vedran Dunjko. Shadows of quantum machine learning. arXiv:2306.00061, 2023. *Submitted to Nature Physics.*

Chapter 2

Background and definitions

In this chapter, we introduce the required background and definitions that will be used throughout this thesis. We begin with an explanation of the basics of quantum computing in Section 2.1. Following that, we delve into the basics of topological data analysis in Section 2.2, which will serve as the underpinning for Chapter 3. Next, Section 2.3 provides an in-depth look into structural risk minimization, a topic discussed in Chapter 4. Subsequently, in Section 2.4, we introduce the field of reinforcement learning, which will be central to our discussions in Chapter 5. Finally, in Section 2.5, we explore the basics of computational learning theory, which we further study in Chapter 6.

2.1 Quantum computing

In this section we will go over the basics of quantum computing. We will not cover all aspects of quantum computing, for more details we refer to [145, 69]. We will assume a basic understanding of linear algebra (for the required linear algebra see Appendix A of [69]). First, we will introduce the fundamental way quantum mechanics can be used to store, manipulate, and extract information. Next, in Section 2.1.1, we will discuss how these fundamental building blocks can be brought together to become quantum algorithms and we discuss two examples (i.e., quantum phase estimation and Hamiltonian simulation). Afterwards, in Section 2.1.2, we will give an introduction to complexity theory, and how quantum computing fits in that field. Finally, in Section 2.1.3, we will discuss how quantum computing can be used to evaluate certain families of linear classifiers (i.e., families of functions used in machine learning that separate classes of data by drawing hyperplanes between them).

From bits to qubits

In classical computing, the basic units of information are bits (i.e., $\{0, 1\}$). On the other hand, in quantum computing the basic units of information are *qubits*, which

are described by unit vectors $|\psi\rangle \in \mathbb{C}^2$, i.e.,

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle , \tag{2.1}$$

where $|\alpha_0|^2 + |\alpha_1|^2 = 1$, $|0\rangle = [1,0]^T$, and $|1\rangle = [0,1]^T$. The normalization constraint turns out to be important later on when we discuss how to extract classical information from qubits through measurements. Analogous to the classical case, we typically gather n qubits into a single register. Following the postulates of quantum mechanics, an n-qubit register is described by a unit vector $|\psi\rangle \in \left(\mathbb{C}^2\right)^{\otimes n} \simeq \mathbb{C}^{2^n}$, i.e.,

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle , \tag{2.2}$$

where $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$, and $|i\rangle$ denotes the ith canonical basis vector (with zeroes everywhere except the ith entry). More generally, if an n_1-qubit register is in a state $|\psi\rangle \in \mathbb{C}^{2^{n_1}}$, and another n_2-qubit register is in a state $|\phi\rangle \in \mathbb{C}^{2^{n_2}}$, then their joint register is in the state $|\gamma\rangle$ given by

$$|\gamma\rangle = |\psi\rangle \otimes |\phi\rangle \in \mathbb{C}^{2^{n_1}} \otimes \mathbb{C}^{2^{n_2}}, \tag{2.3}$$

which is often referred to as the composition postulate.

From classical to quantum circuits

In classical computing, when performing computations the units of information (i.e., bits) are typically manipulated using Boolean circuits[1] that are build up from a basic set of logical gates. In fact, for any function manipulating bitstrings (i.e., so-called Boolean functions) one can build a Boolean circuit using just the logic gate set $\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$ that implements the given Boolean function. In the quantum setting, Boolean functions are replaced by unitary transformations, Boolean circuits are replaced by quantum circuits, and logic gates are replaced by quantum gates. Specifically, instead of the logic gate set $\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$ one typically consider the quantum gate set $\{X, S, T, H, \text{CNOT}\}$[2], which that are linear transformations described by the

[1] or Turing machines, but since they are in some sense equivalent to Boolean circuits, we will stick to the latter since it they are easier to translate to the quantum setting.

[2] There exists many different sets of quantum gates that are universal (two quantum gates can even be enough), though for our purposes we simply fix this to be our quantum gate set. Moreover, this set is "overcomplete" in the sense that the subset $\{\text{CNOT}, H, S, T\}$ is already universal, but we choose to introduce the X-gate anyways for future purposes.

matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.4}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \tag{2.5}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \tag{2.6}$$

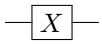
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \tag{2.7}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.8}$$

Note that CNOT is a 2-qubit quantum gate, whereas the other quantum gates X, Y, Z and H act on a single qubit. Sometimes the quantum gate can have a dependence on some parameter $\theta \in \mathbb{R}$, in which case they are refered to as *paramaterized quantum gates*. For instance, one could consider the parameterized X-gate, which is described by the matrix

$$X(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{2.9}$$

These quantum gates can be concatenated to build a quantum circuit. Here a quantum circuit is a collection of wires (each representing a qubit) and quantum gates (to be applied to the qubits) that one reads from left to right. If one of the quantum gates has a free parameter, then the resulting circuit is often called a *parameterized quantum circuit*. The single qubit gates are typically represented by boxes containing the letter corresponding to the gate respective gate, whereas the CNOT has a different notation. Specifically, the X-gate (and any other single qubit gate) is represented by



and the CNOT gate is represented by

Having fixed our notation for individual quantum gates, we can construct larger quantum circuit that specify how to manipulate an n-qubit register by concatenating individual quantum gates. For example, consider the 2-qubit quantum circuit in Figure 2.1 below.

We see that the circuit in Figure 2.1 first applies an H-gate to the first qubit, and afterwards we apply a CNOT to the output state. More precisely, the circuit in

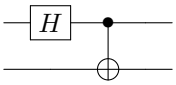


Figure 2.1: An example of a quantum circuit

Figure 2.1 corresponds to the unitary transformation

$$|\psi\rangle \mapsto U\,|\psi\rangle\,,$$

where U is given by

$$U = \mathrm{CNOT} \cdot (H \otimes I).$$

Another important quantum circuit is that of the *quantum Fourier transform*. For the quantum Fourier transform the goal is to construct an n-qubit quantum circuit U_n that implements the unitary transformation

$$|j\rangle \mapsto \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n}\,|k\rangle\,. \tag{2.10}$$

By exploiting a clever rewriting of Eq. (2.10), it turns out that for any given $n \in \mathbb{N}$ we can construct a quantum circuit U_n consisting of $\mathcal{O}(\mathrm{poly}(n))$ quantum gates that implements the map in Eq. (2.10). The quantum Fourier transform is a pivotal building block for many important quantum algorithms (such as quantum phase estimation discussed in Section 2.1.1). For more details on the quantum Fourier transform see [69, 145].

Measurements

Having discussed what the basic units of information are in quantum computing, and how to manipulate them, all that remains is how to extract classical information afterwards (i.e., extracting the output) through *measurements*. There are many different ways in which one can measure qubits to extract classical information, but we will focus on only two forms of measurements. First, we discuss what happens when we *measure in the computational basis*. Suppose we are given some n-qubit quantum state $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle$, then if we measure this state in the computational basis, we obtain the outcome i with probability $|\alpha_i|^2$. Next, we discuss what happens when one measures an *observable*, which is strictly more general than measuring in the computational basis. An n-qubit observable is an operator $O \in \mathbb{C}^{2^n \times 2^n}$ such that $O^\dagger = O$ (i.e., it is *Hermitian*). The possible outcomes when measuring O are its eigenvalues $\{\lambda_j\}_{j=0}^{2^n-1}$. When we measure the observable O on an n-qubit state $|\psi\rangle$ we obtain outcome λ_j with probability

$$|\,\langle \phi_j \mid \psi\rangle\,|^2, \tag{2.11}$$

where $|\phi_j\rangle$ denotes an eigenvector of O with eigenvalue λ_j, and $\langle v| := |v\rangle^\dagger$. Also, the expectation value of O on an n-qubit state $|\psi\rangle$ is given by

$$\langle\psi| O |\psi\rangle . \tag{2.12}$$

Mixed states

Previously we have restricted ourselves to *pure states* (i.e., unit vectors), where we are certain about the state our qubit register is in. However, there can sometimes be uncertainty regarding the state of a qubit register in a classical sense. For example, one could have a register that is in some state $|\phi\rangle$ with probability p, and it is in some other state $|\psi\rangle$ with probabilty $1-p$. To model this, we consider *mixed states*, which are probability distributions (i.e., "mixtures") over pure states. Note that being in a superposition between basis states is sometimes misrepresented as a case of "uncertainty", even though we know for certain the precise superposition the quantum state is in. It is convenient to write down mixed states using *density matrices*, which are positive semidefinite operators (i..e, all eigenvalues are nonnegative) with trace 1. In particular, suppose we know that our qubit register is in one of the states $|\psi_1\rangle, \ldots, |\psi_r\rangle$ each with a respective probability p_j, then we write down this mixed states as a density matrix

$$\rho = \sum_{j=1}^{r} p_j |\psi_j\rangle \langle\psi_j| . \tag{2.13}$$

Following the conventions for pure states, we find that an n-qubit mixed state ρ is manipulated by an n-qubit quantum circuit U as follows

$$\rho \mapsto U\rho U^\dagger. \tag{2.14}$$

Moreover, when measuring an observable O with orthonormal eigenvectors $\{|\psi_j\rangle\}_{j=0}^{2^n-1}$ and corresponding eigenvalues $\{\lambda_j\}_{j=0}^{2^n-1}$, the probability p_j of outcome λ_j is given by

$$p_j = \mathrm{Tr}\left[|\psi_j\rangle \langle\psi_j| \rho\right] \tag{2.15}$$

and the expectation value $\langle O\rangle$ (i.e., the probabilistic expected value of the measurement outcome) is given by

$$\langle O\rangle = \mathrm{Tr}\left[O\rho\right] . \tag{2.16}$$

2.1.1 Basics of quantum algorithms

Having discussed the basics of quantum computing, we are ready to define what a *quantum algorithm* is. In short, a quantum algorithm is a routine that gets a specification of an instance of a problem and efficiently (i.e., in time polynomial in the instance size) constructs a quantum circuit and measurement. This quantum circuit is then applied to the state $|0^n\rangle$, after which the measurement is preformed, and based on the outcomes of the measurement the quantum algorithm decides its

output. In the remainder of this section we will discuss two concrete examples of quantum algorithms: quantum phase estimation (QPE) and Hamiltonian simulation (HS).

Quantum phase estimation

Quantum phase estimation is an important building block for the algorithms discussed in Chapter 3. The input to the problem is an n-qubit unitary U and an eigenvector $|\psi\rangle$ of U. The required output is the *eigenphase* of $|\psi\rangle$ with respect to U, i.e., $\phi \in [0, 1)$ such that

$$U |\psi\rangle = e^{2\pi i \phi} |\psi\rangle .$$

As originally proposed by Kitaev [120] and put in a broader context by Cleve et al. [64], given access to a black box capable of controlled-U^{2^j} operations for $j = 1, \ldots, n$, we can construct a quantum circuit that uses $\mathcal{O}(\text{poly}(n))$ single-qubit gates together with $\mathcal{O}(\text{poly}(1/\delta, 1/\epsilon))$ calls to the black box such that with probability at least $1-\epsilon$ measuring the output state will produce an estimate $\tilde{\phi}$ such that $|\phi - \tilde{\phi}| < \delta$. Note that the use of black boxes suggests that quantum phase estimation by itself is not a complete algorithm in its own right. Rather, one should think of it as a kind of subroutine that, together with a realization of the unitary U (capable of implementing the required black box) and a suitable quantum state $|\psi\rangle$, can perform interesting computational tasks.

To get some intuition how quantum phase estimation works, consider the case where the eigenphase can be expressed as a n-bit string j (i.e., $\phi = \sum_{j=1}^{n} j_i 2^{-i}$). Then, quantum phase estimation works as follows

1. Start with $|0^n\rangle |\psi\rangle$.
2. Apply the layer of gates $(H^{\otimes n} \otimes I)$.
3. Use access to U to apply the map $|j\rangle |\psi\rangle \mapsto |j\rangle U^j |\psi\rangle = e^{2\pi i \phi j} |j\rangle |\psi\rangle$.
4. Apply the inverse quantum Fourier transform to the first n qubits.
5. Measure the first n qubits.

After Step 3., the first n qubits are in the state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} e^{2\pi i \phi j} |j\rangle$ which is the same state obtained by applying the quantum Fourier transform to the state $|\phi\rangle = |j_1, \ldots, j_n\rangle$. Since the quantum Fourier transform is self-inverse, we find that after Step 4. the first n qubits must be in the state $|\phi\rangle$ which when measured in the computational basis will return ϕ. For more details on quantum phase estimation (e.g., what happens in the case when the eigenphase cannot be expressed as a bitstring) we refer to [69, 145]. For our purposes, we highlight that quantum phase estimation can, with probability at least $1 - \epsilon$, obtain an estimate of the eigenphase up to additive error δ using $\mathcal{O}(\text{poly}(n))$ single-qubit gates and $\mathcal{O}(\text{poly}(1/\delta, 1/\epsilon))$ calls to the black box controlled-U^{2^j} operations.

Hamiltonian Simulation

Hamiltonian simulation is another important building block for the algorithms discussed in Chapter 3. Here the input is a specification of a sparse Hermitian operator H together with some time $t \in \mathbb{R}_{>0}$, and the goal is to construct a circuit U that implements the unitary transformation e^{iHt}. We call a $2^n \times 2^n$ matrix *sparse* if at most $\mathcal{O}(\text{poly}(n))$ entries in each row are nonzero. A special class of sparse positive semidefinite matrices that we consider is the class of *log-local* Hamiltonians, i.e., n-qubit Hermitian operators that can be written as a sum

$$H = \sum_{j=1}^{m} H_j, \tag{2.17}$$

where each H_j acts on at most $\mathcal{O}(\log n)$ qubits and $m \in \mathcal{O}(\text{poly}(n))$. We specify the the input matrix H using either of the following two standard cases. First, the input matrix can be specified in terms of *sparse access*. That is, the input matrix $H \in \mathbb{C}^{2^n \times 2^n}$ is specified by quantum circuits that let us query the values of its entries, and the locations of the nonzero entries. More precisely, we assume that we are given classical descriptions of $\mathcal{O}(\text{poly}(n))$-sized quantum circuits that implement the oracles O_H and $O_{H,\text{loc}}$, which map

$$O_H : |i,j\rangle\,|0\rangle \mapsto |i,j\rangle\,|H_{i,j}\rangle\,,$$
$$O_{H,\text{loc}} : |j,\ell\rangle\,|0\rangle \mapsto |j,\ell\rangle\,|\nu(j,\ell)\rangle\,,$$

where $0 \leq i,j,\ell \leq 2^n - 1$, and $\nu(j,\ell) \in \{0,\ldots,2^n-1\}$ denotes the location of the ℓ-th nonzero entry of the j-th column of H. Secondly, for log-local Hamiltonians, we also consider specifying the input matrix H by its *local-terms* $\{H_j\}$ as in Eq. (2.17). Note that any specification in terms of local-terms can be turned into a specification in terms of a sparse access oracle.

Depending on the type of access you have to the input matrix H, different quantum algorithms exist to turn this specification of H into a quantum circuit U that implements the unitary transformation e^{iHt}. If the access is in terms of the local-terms in Eq. (2.17) (with respect to a family of local terms H_j that allow for Hamiltonian simulation in an efficient way, such as the family of Pauli-strings), then one can use Trotterization [129] to construct a quantum circuit U consisting of $\mathcal{O}(\text{poly}(n, 1/\epsilon))$ many two- and single-qubit quantum gates that satisfies

$$||e^{iHt} - U||_2 < \epsilon.$$

On the other hand, if the access is in terms of sparse access oracles then more advanced methods such as [133] are required. These methods use sparse access to H to produce a quantum circuit U of size $\mathcal{O}(\text{poly}(n, \log(1/\epsilon)))$ such that[3]

$$||U - e^{iHt}||_2 < \epsilon.$$

[3]The runtime of certain more advanced methods also depend on the largest entry of H and its sparsity, but we choose to omit these dependencies from the runtime since they are not relevant for the purposes of Chapter 3

2.1.2 Quantum complexity theory

In the previous section we discussed how classical and quantum computers perform computations in a fundamentally different way. In this section we will discuss some of the consequences that this fundamentally different way of computing has on the field of *complexity theory*. In short, complexity theory studies how much of a certain resource is required to solve a given problem as a function of the instance size. We will stick to an intuitive explanation of some classes of problems as categorized by complexity theory, and for the formal definitions we refer to [23]. The problems that one typically studies in complexity theory are so-called *decision problems*. In such problems, one is given some $x \in \{0,1\}^n$ and one is asked to decide whether the input encoded by x satisfies a certain property. For instance, we could use $x \in \{0,1\}^n$ to encode a Boolean formula, and ask to decide whether x has any satisfying assignment or not.

In complexity theory, the class P denotes all problems that are solvable by a classical circuit that is allowed a number of gates that is at most polynomial in the input length (with the caveat that there must exist an efficient procedure that generates these circuits for a given input size). Additionally, the class BPP denotes all problems that are solvable by a classical circuit that is allowed a number of gates that is at most polynomial in the input length and is additionally allowed to use some randomness (e.g., by providing it with a random number generator). One of the questions studied in complexity theory is what additional problems one could solve by allowing this additional internal randomness, if any such problems exist. This question can be reformulated as asking whether or not $\mathsf{BPP} = \mathsf{P}$, which sparked the study of *derandomization* techniques [57].

With the addition of quantum computing one can define a whole new family of complexity classes. For instance, the class BQP denotes all problems that are solvable by a quantum circuit that is allowed a number of quantum gates that is at most polynomial in the input length. Another important question in complexity theory is whether there exists problems that can be solved with polynomial-size quantum circuits, but which cannot be solved with polynomial-size (randomized) classical circuits. This question can be reformulated as asking whether or not $\mathsf{BPP} = \mathsf{BQP}$. Since quantum circuits can simulate any classical circuit (with a negligible increase in circuit size), it holds that $\mathsf{BPP} \subseteq \mathsf{BQP}$. Even though it is not formally proven that $\mathsf{BPP} \subsetneq \mathsf{BPP}$ (i.e., that BQP is strictly larger than BPP), it is it is still widely-believed to be the case. For example, the problem of factoring lies in BQP due to the famous Shor's algorithm [176], but it is widely-believed to be outside of BPP.

If we allow ourselves to operate under the assumption that $\mathsf{BQP} \neq \mathsf{BPP}$, then how could one use complexity theory to argue that a certain problem is likely to exhibit a (superpolynomial) quantum advantage? Intuitively, what we mean by a "quantum advantage" is that the classical resources required to solve this problem drastically outscale the quantum resources required to solve the problem. Under the assumption that $\mathsf{BQP} \neq \mathsf{BPP}$, one way to show that a problem exhibits a quantum advantage is to show that it is among the hardest problems in BQP. More precisely, we would like to show that if we could solve this problem using certain resources, then we could solve any problem in BQP using comparable resources. In complexity theory, the above is

captured using the notion of *reductions*. In the broadest sense, we say that a problem A is reducible to a problem B if an algorithm for solving B with a certain set of resources (if it were to exist) can be turned into an algorithm for solving A that uses a comparable amount of resources (e.g., both algorithms use polynomial-size circuits). Now if any problem in a complexity class is reducible to a problem A, then we say that A is *hard* for that class. If additionally A is also contained in that complexity class, we say that A is *complete* for that class.

If we were to show that a given problem A is BQP-complete[4], then this is solid evidence that A exhibits a quantum advantage under the assumption that $\mathsf{BPP} \neq \mathsf{BQP}$. In particular, suppose we could solve A using a polynomial-size randomized classical circuit (i.e., it is in BPP). Since A is among the hardest problems in BQP, through the reductions we find that any other problem is BQP can also be solved using the resources of BPP. This clearly contradicts the assumption that $\mathsf{BPP} \neq \mathsf{BQP}$, which establishes by contradiction that A is not in BPP. Moreover, since A was in fact complete for BQP, we know that is in BQP and that it is thus solvable using a polynomial-depth quantum circuit. Using a completeness-result to give evidence for quantum advantage is not limited to the class BQP. In fact, any class that contains problems in BQP that are widely-believed to be outside BPP would work for this purpose. In Section 3.2.1, we discuss another example of a complexity class whose completeness result provide evidence for quantum advantage called DQC1, and we use it to argue that certain problems in topological data analysis (or machine learning more general) are likely to exhibit a quantum advantage.

To end this section, we introduce two more complexity classes. In particular, we introduce two classes of problems whose solutions are *verifiable* using a given amount of resources. For instance, if our input $x \in \{0,1\}^n$ encodes a Boolean formula, then how much resources does it require to check whether a given assignment makes the formula evaluate to 1? In the classical setting, the class NP denotes all problems whose solutions are verifiable in time polynomial in the input length on a classical computer. Note that deciding whether a Boolean formula has a satisfying assignment is in NP, since the evaluation of the Boolean formula for a given assignment can be done using in polynomial-time on a classical computer. In the quantum setting, the class QMA denotes the set of problems whose solutions are verifiable in time polynomial in the input length on a quantum computer. It is a long-standing open question in complexity theory whether $\mathsf{NP} = \mathsf{P}$, i.e., whether problems whose solution can be *verified* in polynomial-time on a classical computer can also be *solved* using the same resources. However, it is widely-believed that $\mathsf{P} \subsetneq \mathsf{NP}$, though there is no proof (yet). Similarly, it is widely-believed that $\mathsf{BQP} \subsetneq \mathsf{QMA}$, though also here there is no proof (yet).

[4]Strictly speaking, when we say that problem is BQP-complete, we actually mean that it is complete for the class PromiseBQP (the class BQP is not known to have any complete problems). The difference between BQP and PromiseBQP is that in the latter one only needs to be correct on a subset of instances (i.e., those that satisfy a certain "promise").

2.1.3 Quantum linear classifiers

A fundamental family of classifiers (i.e., binary-valued functions) used throughout machine learning are those constructed from *linear functions*. Specifically, they are constructed from the family of real-valued functions on $\mathbb{R}^\ell$

$$\mathcal{F}_{\text{lin}} = \Big\{ f_w(x) = \langle w, x \rangle \ : \ w \in \mathbb{R}^\ell \Big\}, \tag{2.18}$$

where $\langle ., . \rangle$ denotes an inner product on the input space $\mathbb{R}^\ell$. These linear functions are turned into classifiers by adding an offset and taking the sign, i.e., the classifiers are given by

$$\mathcal{C}_{\text{lin}} = \Big\{ c_{w,d}(x) = \text{sign}\big(\langle w, x \rangle - d\big) \ : \ w \in \mathbb{R}^\ell, \ d \in \mathbb{R} \Big\}. \tag{2.19}$$

While this family of classifiers is relatively limited (e.g., it cannot solve the well-known `XOR` problem), it becomes powerful when introducing a *feature map*. Specifically, a feature map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^N$ is used to (non-linearly) map the data to a (much) higher-dimensional space – called the *feature space* – in order to make the data more linearly-separable. We let $\mathcal{C}\left(\Phi\right) = \{c \circ \Phi \mid c \in \mathcal{C}\}$ denote the family of classifiers on $\mathbb{R}^\ell$ obtained by combining a family of linear classifiers $\mathcal{C} \subseteq \mathcal{C}_{\text{lin}}$ on $\mathbb{R}^N$ with a feature map Φ. If the feature map is clear from the context we will omit it in the notation and just write $\mathcal{C}$. A well known example of a model based on linear classifiers is the *support vector machine* (SVM), which aims to finds the hyperplane that attains the maximal perpendicular distance to the two classes of points in the two distinct half-spaces (assuming the feature map makes the data linearly separable, though one could relax this using so-called soft-margin SVMs [66] in the non-separable case).

The linear-algebraic nature of linear classifiers makes them particularly well-suited for quantum treatment. In the influential works of Havlíček et al. [103], and Schuld & Killoran [168], the authors propose a model where the space of n-qubit Hermitian operators – denoted $\text{Herm}\left(\mathbb{C}^{2^n}\right)$ – takes the role of the feature space. Note that $\text{Herm}\left(\mathbb{C}^{2^n}\right)$ is a 4^n-dimensional real vector space equipped with the Frobenius inner product $\langle A, B \rangle = \text{Tr}\big[A^\dagger B\big]$. Their feature map maps classical inputs x to n-qubit density matrices $\Phi(x) := \rho_\Phi(x)$ (i.e., positive semi-definite matrices of trace one). Finally, the hyperplanes that separates the states $\rho_{\Phi(x)}$ corresponding to the different classes are given by n-qubit observables. In short, the family of functions their model uses is given by

$$\mathcal{F}_{\text{qlin}} = \Big\{ f_{\mathcal{O}}(x) = \text{Tr}\left[\mathcal{O}\rho_\Phi(x)\right] \ : \ \mathcal{O} \in \text{Herm}\big(\mathbb{C}^{2^n}\big) \Big\}, \tag{2.20}$$

and the family of classifiers – which we refer to as *quantum linear classifiers* – is given by

$$\mathcal{C}_{\text{qlin}} = \Big\{ c_{\mathcal{O},d}(x) = \text{sign}\big(\text{Tr}\left[\mathcal{O}\rho_\Phi(x)\right] - d\big) \ : \ \mathcal{O} \in \text{Herm}\big(\mathbb{C}^{2^n}\big), \ d \in \mathbb{R} \Big\}. \tag{2.21}$$

We can estimate $f_{\mathcal{O}}(x)$ defined in Equation (2.20) by preparing the state $\rho_{\Phi(x)}$ and

measuring the observable $\mathcal{O}$. In particular, approximating $f_{\mathcal{O}}(x)$ up to additive error ϵ requires only $\mathcal{O}\left(1/\epsilon^2\right)$ samples. While the error creates a fuzzy region around the decision boundary, this turns out to not cause major problems in practical settings [31].

Using parameterized quantum circuits both the preparation of a quantum state that encodes the classical input and the measurement of observables can be done efficiently for certain feature maps and families of observables. We now briefly recap two ways in which parameterized quantum circuits can be used to efficiently implement a family of quantum linear classifiers, as originally proposed by Havlíček et al. [103], and Schuld & Killoran [168]. Both ways use a parameterized quantum circuit to implement the feature map. Specifically, let U_Φ be a parameterized quantum circuit, then we can use it to implement the feature map given by

$$\Phi : x \mapsto \rho_\Phi(x) := |\Phi(x)\rangle \langle \Phi(x)| , \tag{2.22}$$

where $|\Phi(x)\rangle := U_\Phi(x) |0\rangle^{\otimes n}$. The key difference between the two approaches is which observables they are able to implement (i.e., which separating hyperplanes they can represent) and how the observables are actually measured (i.e., how the functions $f_{\mathcal{O}}$ are evaluated). An overview of how the two approaches implement quantum linear classifiers can be found in Figure 2.2, and we discuss the main ideas behind the two approaches below.

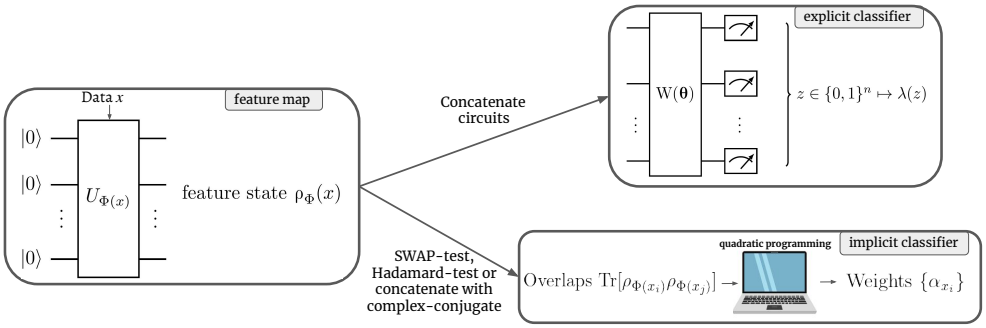


Figure 2.2: An overview of the explicit and implicit quantum linear classifiers defined in Equations (2.24) and (2.25), respectively (adapted from [95]). Note that in the case of the explicit classifier, a universal circuit $W(\theta)$ (specifying the eigenbasis) followed by a computational basis measurement and universal postprocessing λ (specifying the eigenvalues) allows one to measure any observable (albeit not efficiently with respect to the number of qubits).

Explicit quantum linear classifier[5] The observables measured in this approach are implemented by first applying a parameterized quantum circuit $W(\theta)$, followed by a computational basis measurement and postprocessing of the measurement outcome

[5] Also called the *quantum variational classifier* [103].

$\lambda : [2^n] \to \mathbb{R}$. Upon closer investigation, one can derive that the corresponding observable is given by

$$\mathcal{O}_\theta^\lambda = W^\dagger(\theta) \cdot \mathrm{diag}\Big(\lambda(0), \lambda(1), \ldots, \lambda(2^n - 1)\Big) \cdot W(\theta). \tag{2.23}$$

Examples of efficiently computable postprocessing functions λ include functions with a polynomially small support (implemented using a lookup table), functions that are efficiently computable from the input bitstring (e.g., the parity of the bitstring, which is equivalent to measuring $Z^{\otimes n}$), or parameterized functions such as neural networks. Note that the postprocessing function λ plays an important role in how the measurement of the observable in Eq. (2.23) is physically realized. Altogether, this efficiently implements the family of linear classifiers – which we refer to as *explicit quantum linear classifiers* – given by

$$\begin{aligned}\mathcal{C}_{\mathrm{qlin}}^{\mathrm{explicit}} = \Big\{ & c_{\mathcal{O}_\theta^\lambda, d}(x) = \mathrm{sign}\big(\mathrm{Tr}\left[\rho_\Phi(x)\mathcal{O}_\theta^\lambda\right] - d\big) \\ & \qquad : \ \mathcal{O}_\theta^\lambda \text{ as in Equation (2.23)}, \ d \in \mathbb{R} \Big\}.\end{aligned} \tag{2.24}$$

The power of this model lies in the efficient parameterization of the manifold (inside the 4^n-dimensional vector space of Hermitian operators on $\mathbb{C}^{2^n}$) realized by the quantum feature map together with the parameterized separating hyperplanes that can be attained by $W(\theta)$ and λ. Here also lies a restriction of the explicit quantum linear classifier compared to standard linear classifiers, as in the latter all hyperplanes are possible and in the former only the hyperplanes that lie in the manifold parameterized by $W(\theta)$ and λ are possible. Furthermore, explicit quantum linear classifiers can likely not be efficiently evaluated classically, as computing expectation values $\mathrm{Tr}\left[\rho_\Phi(x)\mathcal{O}_\theta^\lambda\right]$ is classically intractable for sufficiently complex feature maps and observables [188, 41].

Implicit quantum linear classifier[6] Another way to implement a linear classifier is by using the so-called *kernel trick* [164]. In short, this trick involves expressing the normal vector of the separating hyperplane, – i.e., the observable $\mathcal{O}$ in the case of quantum linear classifiers – on a set of training examples $\mathcal{D}$ as a linear combination of feature vectors, resulting in the expression

$$\mathcal{O}_\alpha = \sum_{x' \in \mathcal{D}} \alpha_{x'} \rho_{\Phi(x')} = \sum_{x' \in \mathcal{D}} \alpha_{x'} \left|\Phi(x')\right\rangle \left\langle\Phi(x')\right|.$$

Using this expression we can rewrite the corresponding quantum linear classifier as

$$c_{\mathcal{O}_\alpha, d}(x) = \mathrm{sign}\big(\mathrm{Tr}\left[\rho_\Phi(x)\mathcal{O}_\alpha\right] - d\big) = \mathrm{sign}\big(\sum_{x' \in \mathcal{D}} \alpha_{x'} \mathrm{Tr}\left[\rho_\Phi(x)\rho_\Phi(x')\right] - d\big).$$

These type of linear classifiers can also be efficiently realized using parameterized quantum circuits. Using quantum protocols such as the SWAP-test or the Hadamard-

[6] Also called the *quantum kernel estimator* [103].

test, or by using the inverse of the circuit that implements the feature map $U_\Phi(x)^{-1}$, it is possible to efficiently evaluate the overlaps $\mathrm{Tr}[\rho_\Phi(x)\rho_\Phi(x')]$ for the feature map defined in Equation (2.22). Afterwards, the optimal parameters $\{\alpha_{x'}\}_{x'\in\mathcal{D}}$ are obtained on a classical computer, e.g., by solving a quadratic program. Altogether, this allows us to efficiently implement the family of linear classifiers – which we refer to as *implicit quantum linear classifiers* – given by

$$\mathcal{C}_{\mathrm{qlin}}^{\mathrm{implicit}} = \Big\{ c_{\mathcal{O}_\alpha,d}(x) = \mathrm{sign}\big(\mathrm{Tr}\left[\rho_\Phi(x)\mathcal{O}_\alpha\right] - d\big) \\ : \ \mathcal{O}_\alpha = \sum_{x'\in\mathcal{D}} \alpha_{x'}\rho_\Phi(x'),\ \alpha\in\mathbb{R}^{|\mathcal{D}|},\ d\in\mathbb{R}\Big\}. \tag{2.25}$$

The power of this model comes from the fact that evaluating the overlaps $\mathrm{Tr}\left[\rho_\Phi(x)\rho_\Phi(x')\right]$ is likely classically intractable for sufficiently complex feature maps [103], demonstrating that classical computers can likely neither train nor evaluate this quantum linear classifier efficiently. Moreover, from the well-known *Representer theorem* we know any quantum linear classifier that is the minimizer of a loss functions that includes *regularization* of the Frobenius norm of the observable can be expressed as an implicit quantum linear classifier [165]. However, as we indicate later in Section 4.3, this does not mean that we can forego explicit quantum linear classifiers entirely, as in the explicit approach there are unique types of meaningful regularization for which there is no straightforwards correspondence to the implicit approach.

2.2 Topological data analysis

Topological data analysis is a recent approach to data analysis that extracts robust features from a dataset by inferring properties of the *shape* of the data. This is perhaps best explained in analogy to a better-known method: much like how principal component analysis extracts features (i.e., the singular values characterizing the spread of the data in the directions of highest variance) that are invariant under translation and rotation of the data, topological data analysis goes a step further and extract features that are also invariant under bending and stretching of the data (i.e., by inferring properties of its general shape). Because of this invariance of the extracted features, topological data analysis techniques can be inherently more robust to noise in the data.

The theory behind topological data analysis is fairly extensive, but most of it we will not need for our purpose. Namely, we can set most of the topology aside and tackle the issue in linear-algebraic terms, which are well-suited for quantum approaches. In this section we introduce the relevant linear-algebraic concepts, and we briefly review the quantum algorithm for topological data analysis of Lloyd, Garnerone and Zanardi (LGZ) [130].

2.2.1 Betti numbers as features

In topological data analysis the dataset is typically a point cloud (i.e., a collection of points in some ambient space) and the aim is to extract the shape of the underlying

data (i.e., the 'source' of these points). This is done by constructing a connected object – called a *simplicial complex* – composed of points, lines, triangles and their higher-dimensional counterparts, whose shape one can study. After constructing the simplicial complex, features of the shape of the data – in particular, the number of connected components, holes, voids and higher-dimensional counterparts – can be extracted using linear-algebraic computations based on *homology*. An overview of this procedure can be found in Figure 2.4.

Consider a dataset of points $\{x_i\}_{i=1}^n$ embedded in some space equipped with a distance function d (typically $\mathbb{R}^m$ equipped with the Euclidean distance). The construction of the simplicial complex from this point cloud proceeds as follows. First, one constructs a graph by connecting datapoints that are "close" to each other. This is done by choosing a *grouping scale* ϵ (defining which points are considered "close") and connecting all datapoints that are within ϵ distance from each other. This yields the graph $G = ([n], E_\epsilon)$, with vertices $[n] := \{1, \ldots, n\}$ and edges

$$E_\epsilon = \{(i, j) \mid d(x_i, x_j) \leq \epsilon\}.$$

After having constructed this graph, one relates to it a particular kind of simplicial complex called a *clique complex*, by associating its cliques (i.e., complete subgraphs) with the building blocks of a simplicial complex[1]. That is, a 2-clique is considered a line, a 3-clique a triangle, a 4-clique a tetrahedron, and $(k+1)$-cliques the k-dimensional counterparts[2].

To fix the notation, let $\mathrm{Cl}_k(G) \subset \{0,1\}^n$ denote the set of $(k+1)$-cliques in G – where we encode a subset $\{i_1, \ldots, i_k\} \subset [n]$ as an n-bit string where the indices i_k specify the positions of the ones in the bitstring – and let $\chi_k := |\mathrm{Cl}_k(G)|$ denote the number of these cliques. Throughout this thesis, we will discuss everything in terms of clique complexes, as this is sufficient for our purposes and allows us to use the more familiar terminology of graph theory.

The constructed clique complex exhibits the features that we want to extract from our dataset – i.e., the number of k-dimensional holes. For example, in Figure 2.3 we see a clique complex where we can count three 1-dimensional holes. Interestingly, counting these holes can be done more elegantly using linear algebra by employing constructions from homology.

To extract these features using linear algebra, embed the clique complex into a Hilbert space $\mathcal{H}_k^G$, by raising the set of bitstrings that specify $(k+1)$-cliques to labels of orthonormal basis vectors. Let $\mathcal{H}_k$ denote the Hilbert space spanned by computational basis states with Hamming weight[3] $k+1$. Due to the way we encode cliques as bitstrings, we have that $\mathcal{H}_k^G$ is a subspace of $\mathcal{H}_k$. Moreover, each $\mathcal{H}_k$ is an $\binom{n}{k+1}$-dimensional subspace of the entire n-qubit Hilbert space $\mathbb{C}^{2^n}$, and $\mathbb{C}^{2^n} \simeq \bigoplus_{k=-1}^{n-1} \mathcal{H}_k$.

The next step towards extracting features using linear algebra involves studying properties of the *boundary maps* $\partial_k : \mathcal{H}_k \to \mathcal{H}_{k-1}$, which are defined by linearly

[1] The resulting simplicial complex coincides with the Vietoris-Rips complex common in topological data analysis literature [83].

[2] The shift in the indexing is due to different terminologies in graph theory and topology (e.g., in graph theory a triangle is called a 3-clique, whereas in topology it is called a 2-simplex).

[3] The Hamming weight of a bitstring is the number of 1s in it.

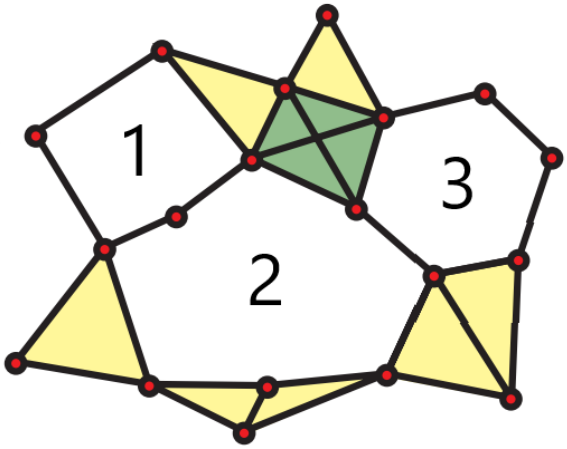


Figure 2.3: Example of a clique complex with three 1-dimensional holes (adapted from [83]). The number of these holes is equal to the first *Betti number*.

extending the action on the basis states given by

$$\partial_k |j\rangle := \sum_{i=0}^{k} (-1)^i \, |\widehat{j(i)}\rangle \,, \tag{2.26}$$

where $\widehat{j(i)}$ denotes the n-bit string of Hamming weight k that encodes the subset obtained by removing the i-th element from the subset encoded by j (i.e., we set the i-th one in the bitstring j to zero). By considering the restriction of ∂_k to $\mathcal{H}_k^G$ – which we denote by ∂_k^G – these boundary maps can encode the connectivities of the graph G, in which case their image and kernel encode various properties of the corresponding clique complex. Intuitively, these boundary maps map a $(k+1)$-clique to a superposition (i.e., a linear combination) of all k-cliques that it contains, as seen in Eq. (2.26).

These boundary maps allow one to extract features of the shape of a clique complex by studying their images and kernels, and in particular their quotients. Specifically, the quotient space

$$H_k(G) := \ker \partial_k^G / \mathrm{Im} \partial_{k+1}^G, \tag{2.27}$$

which is called the k-th *homology group*, captures features of the shape of the underlying clique complex. The main feature is the k-th *Betti number* β_k^G, which is defined as the dimension of the k-th homology group, i.e.,

$$\beta_k^G := \dim H_k(G).$$

By construction, the k-th Betti number is equal to the number of k-dimensional holes in the clique complex.

The main problem in topological data analysis that we study in this thesis is the computation of Betti numbers. To do so, we study the *combinatorial Laplacians* [76], which are defined as

$$\Delta_k^G = \left(\partial_k^G\right)^\dagger \partial_k^G + \partial_{k+1}^G \left(\partial_{k+1}^G\right)^\dagger . \tag{2.28}$$

These combinatorial Laplacians can be viewed as generalized (or rather, higher-order) graph Laplacians in that they encode the connectivity between cliques in the graph as opposed to encoding the connectivity between individual vertices. We study the combinatorial Laplacians because the discrete version of the Hodge theorem [76] tells us that

$$\dim \ker \left(\Delta_k^G\right) = \beta_k^G, \tag{2.29}$$

which is often used as a more convenient way to compute Betti numbers [81], particularly in the case of the quantum algorithm that we discuss in the next section.

In conclusion, if the clique complex is constructed from a point cloud according to the construction discussed above, then computing these Betti numbers can be viewed as a method to extract features of the shape of the data (specifically, the number of holes are present at scale ϵ). By recording Betti numbers across varying scales ϵ in a so-called *barcode* [83], one can discern which holes are "real" and which are "noise", resulting in feature extraction that is robust to noise in the data. Even though barcodes are very important in topological data analysis, we will mostly focus on the problem of estimating the number of holes at a given scale.

2.2.2 Quantum algorithm for Betti number estimation

The algorithm for Betti number estimation of Lloyd, Garnerone and Zanardi (LGZ) [130] utilizes Hamiltonian simulation and phase estimation to estimate the dimension of the kernel (i.e., the *nullity*) of the combinatorial Laplacian (which by Eq. (2.29) is equal to the corresponding Betti number). After the initial algorithm of Lloyd et al. several different improvements were made, focusing either on reducing the number of T-gates [33], making it more amenable to NISQ requirements [189], or on exploiting the quantum singular value transform and achieving an exponential saving in the number of qubits [135]. To make our presentation self-contained, we review the original quantum algorithm for Betti number estimation of Lloyd et al.

Estimating the nullity of a sparse Hermitian matrix can be achieved using some of the most fundamental quantum-algorithmic primitives. Namely, using Hamiltonian simulation and quantum phase estimation one can estimate the eigenvalues of the Hermitian matrix, given that the eigenvector register starts out in an eigenstate. Moreover, if instead the eigenvector register starts out in the maximally mixed state (which can be thought of as a random choice of an eigenstate), then measurements of the eigenvalue register produce approximations of eigenvalues, sampled uniformly at random from the set of all eigenvalues. This routine is then repeated to estimate the nullity by simply computing the frequency of zero eigenvalues (recall that the

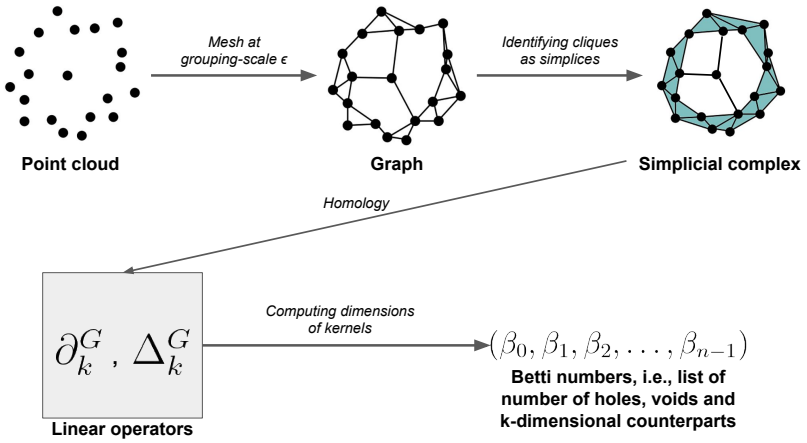


Figure 2.4: The pipeline of topological data analysis (adapted from [90]). First, points within ϵ distance are connected to create a graph. Afterwards, cliques in this graph are identified with simplices to create a simplicial complex. Next, homology is used to construct linear operators that encode the topology. Finally, the dimensions of the kernels of these operators are computed to obtain the Betti numbers (which correspond to the number of holes).

dimension of the kernel is equal to the multiplicity of the zero eigenvalue). Note that this procedure does not strictly speaking estimate the nullity, but rather the number of small eigenvalues, where the threshold is determined by the precision of the quantum phase estimation (see Section 2.2.2 for more details). The steps of the quantum algorithm for Betti number estimation of LGZ are summarized in Figure 2.5.

In Step 1(a), Grover's algorithm is used to prepare the uniform superposition over $\mathcal{H}_k^G$, from which one can prepare the state ρ_k^G by applying a CNOT gate to each qubit of the uniform superposition into some ancilla qubits and tracing those out. When given access to the adjacency matrix of G, one can check in $\mathcal{O}\left(k^2\right)$ operations whether a bitstring $j \in \{0,1\}^n$ encodes a valid k-clique and mark them accordingly in the application of Grover's algorithm. By cleverly encoding Hamming weight k strings we can avoid searching over all n-bit strings, which requires $\mathcal{O}\left(nk\right)$ additional gates per round of Grover's algorithm plus an additional one-time cost of $\mathcal{O}\left(n^2k\right)$ [93]. Hence, the runtime of this first step is

$$\mathcal{O}\left(n^2k + nk^3\sqrt{\binom{n}{k+1}/\chi_k}\right)$$

where χ_k denotes the number of $(k+1)$-cliques. This runtime is polynomial in the

Quantum algorithm for Betti number estimation

1. For $i = 1, \ldots, M$ repeat:
   (a) Prepare the state:
   $$\rho_k^G = \frac{1}{|\dim \mathcal{H}_k^G|} \sum_{j \in \mathrm{Cl}_k(G)} |j\rangle \langle j| . \tag{2.30}$$
   (b) Apply quantum phase estimation to the unitary $e^{i\Delta_k^G}$, with the eigenvector register starting out in the state ρ_k^G.
   (c) Measure the eigenvalue register to obtain an approximation $\widetilde{\lambda}_i$.
2. Output the frequency of zero eigenvalues:
   $$\left|\{i \mid \widetilde{\lambda}_i = 0\}\right| / M.$$

Figure 2.5: Overview of the quantum algorithm of Lloyd, Garnerone and Zanardi (LGZ) [130]

number of vertices n when

$$\binom{n}{k+1} / \chi_k \in \mathcal{O}\left(\mathrm{poly}(n)\right). \tag{2.31}$$

Throughout this thesis we say that a graph is *clique-dense* if it satisfies Eq. (2.31). Note that ρ_k^G can of course also be directly prepared without the use of Grover's algorithm by using rejection sampling: choose a subset uniformly at random and accept it if it encodes a valid clique. This is quadratically less efficient, however it has advantages if one has near-term implementations in mind, as it is a completely classical subroutine. As we will discuss in more detail in Section 2.2.2, this state preparation procedure via Grover's algorithm or uniform clique-sampling is a crucial bottleneck in the quantum algorithm.

In Step 1(b), standard methods for Hamiltonian simulation of sparse Hermitian matrices are used together with quantum phase estimation to produce approximations of the eigenvalues of the simulated matrix. In the original algorithm, the matrix that LGZ simulates (i.e., the matrix it applies Hamiltonian simulation on) in this step is

the *Dirac operator*, which is defined as

$$B_G = \begin{pmatrix} 0 & \partial_1^G & 0 & \dots & \dots & 0 \\ \left(\partial_1^G\right)^\dagger & 0 & \partial_2^G & \dots & \dots & 0 \\ 0 & \left(\partial_2^G\right)^\dagger & 0 & \ddots & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 & \partial_{n-1}^G \\ 0 & 0 & 0 & \dots & \left(\partial_{n-1}^G\right)^\dagger & 0 \end{pmatrix}$$

and satisfies

$$B_G^2 = \begin{pmatrix} \Delta_0^G & 0 & \dots & 0 \\ 0 & \Delta_2^G & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \Delta_{n-1}^G \end{pmatrix}. \tag{2.32}$$

From Eq. (2.32) we gather that the probability of obtaining an approximation of an eigenvalue that is equal to zero is proportional to the nullity of the combinatorial Laplacian. Because B_G is an n-sparse Hermitian matrix with entries $0, -1$ and 1, to which we can implement sparse access using $\mathcal{O}(n)$ gates, we can implement e^{iB} using $\widetilde{\mathcal{O}}\left(n^2\right)$ gates [133] (here $\widetilde{\mathcal{O}}$ suppresses logarithmically growing factors).

We remark that it is also possible to simulate Δ_k^G directly (as depicted in Figure 2.5). Namely, as Δ_k^G is an n^2-sparse Hermitian matrix whose entries are bounded above by n, to which we can implement sparse access using $\mathcal{O}\left(n^4\right)$ gates (e.g., see Theorem 3.3.4 [86]), we can implement $e^{i\Delta_k^G}$ using $\widetilde{\mathcal{O}}\left(n^6\right)$ gates [133].

The disadvantage to directly simulating Δ_k^G is that it requires more gates. However, the advantage is that the Hamiltonian simulation of Δ_k^G requires fewer qubits compared to the Hamiltonian simulation of B_G, namely, $\log\binom{n}{k+1}$ qubits instead of n. Moreover, when the graph is clique-dense one can bypass Step 1(a) by padding Δ_k^G with all-zero rows and columns and letting the eigenvector start out in the maximally mixed state $I/2^n$ (see Section 3.1.1 for more details).

Let $\lambda_{\max}$ denote the largest eigenvalue and let $\lambda_{\min}$ denote the smallest nonzero eigenvalue of Δ_k^G. By scaling down the matrix one chooses to simulate (i.e., either B or Δ_k^G) by $1/\lambda_{\max}$ to avoid multiples of 2π, we can tell whether an eigenvalue is equal to zero or not if the precision of the quantum phase estimation is at least $\lambda_{\max}/\lambda_{\min}$. By the Gershgorin circle theorem (which states that $\lambda_{\max}$ is bounded above by the maximum sum of absolute values of the entries of a column or row) we know that $\lambda_{\max} \in \mathcal{O}(n)$. For the general case not much is known in terms of lower bounds on $\lambda_{\min}$. Nonetheless, even if we do not have such a lower bound, the number of small eigenvalues (as opposed to zero eigenvalues) still conveys topological information about the underlying graph (see Section 2.2.2 for more details). By taking into account the cost of the quantum phase estimation [145], the total runtime of Step 1(b) becomes $\widetilde{\mathcal{O}}\left(n^3/\lambda_{\min}\right)$.

Finally, estimating $\beta_k^G / \dim \mathcal{H}_k^G$ up to additive precision ϵ can be done using $M \in$

$\mathcal{O}\left(\epsilon^{-2}\right)$ repetitions of Step 1(a) through 1(c). This brings the total cost of estimating $\beta_k^G / \dim \mathcal{H}_k^G$ up to additive precision ϵ to

$$\widetilde{\mathcal{O}}\left(\left(nk^3\sqrt{\binom{n}{k+1}/\chi_k} + n^3/\lambda_{\min}\right)/\epsilon^2\right).$$

In conclusion, the quantum algorithm for Betti number estimation runs in time polynomial in n under two conditions. Firstly, the graph has to be clique-dense, i.e., it has to satisfy Eq. (2.31) (see Section 2.2.2 for more details). Secondly, the smallest nonzero eigenvalue $\lambda_{\min}$ has to scale at least inverse polynomial in n (see Section 2.2.2 for more details). If both these conditions are satisfied, then the quantum algorithm for Betti number estimation achieves an exponential speedup over the best known classical algorithms if the size of the combinatorial Laplacian – i.e., the number of $(k+1)$-cliques – scales exponentially in n (see Section 2.2.3 for more details).

Approximate Betti numbers

As mentioned in the previous section, the quantum algorithm for Betti number estimation does not strictly speaking estimate the Betti number (i.e., the nullity of the combinatorial Laplacian), but rather the number of small eigenvalues of the combinatorial Laplacian. This is because little is known in terms of lower bounds for the smallest nonzero eigenvalue of combinatorial Laplacians, and hence it is unclear to what precision one has to estimate the eigenvalues in the quantum phase estimation. In any case, it is conjectured that for high-dimensional simplicial complexes the smallest nonzero eigenvalue will generally be at least inverse polynomial in n [81], which would imply that quantum phase estimation can in time $\mathcal{O}\left(\text{poly}(n)\right)$ determine whether an eigenvalue is exactly equal to zero.

Even without knowing a lower bound on the smallest nonzero eigenvalue of the combinatorial Laplacian, we can still perform quantum phase estimation up to some fixed inverse polynomial precision. The quantum algorithm for Betti number estimation then outputs an estimate of the number of eigenvalues of the combinatorial Laplacian that lie below this precision threshold. Throughout this thesis we will refer to this as *approximate Betti numbers*, which turn out to still convey information about the underlying graph. For instance, Cheeger's inequality – which relates the sparsest cut of a graph to the smallest nonzero eigenvalues of its standard graph Laplacian – turns out to have a higher-order generalization that utilizes the combinatorial Laplacian [92]. Moreover, there are several other spectral properties of the combinatorial Laplacian beyond the number of small eigenvalues that also convey topological information about the underlying graph. Some of these spectral properties can also be efficiently extracted using quantum algorithms (see Section 3.3.2 for more details).

Efficient state preparation

In Section 2.2.2 we saw that the quantum algorithm for Betti number estimation can efficiently estimate approximate Betti numbers if the input graph satisfies certain criteria. In particular, the graph has to be such that one can efficiently prepare the

maximally mixed state over all its cliques of a given size (i.e., the state in Eq. (2.30) in Figure 2.5). In this section we highlight that this state preparation constitutes one of the main bottlenecks in the quantum algorithm for Betti number estimation.

One way to prepare the maximally mixed state over all k-cliques of an n-vertex graph is to sample k-cliques uniformly at random and feed them into the quantum algorithm. For the quantum algorithm for Betti number estimation to run in time sub-exponential in n, we have to be able to sample a k-clique uniformly at random in time $n^{o(k)}$. However, for general graphs finding a k-clique cannot be done in time $n^{o(k)}$ unless the exponential time hypothesis fails [56]. Nonetheless, for certain families of graphs, uniform clique sampling can be done much more efficiently, e.g., in time polynomial in n (in which case the quantum algorithm also runs in time polynomial in n). In particular, the graph's clique-density (i.e., probability that a uniformly random subset of vertices is a clique), or the graph's arboricity (which up to a factor $1/2$ is equivalent to the maximum average degree of a subgraph) are important factors that dictate the efficiency of uniform clique sampling algorithms. In Section 3.2.4 we outline concrete families of graphs (based on their clique-density or arboricity) for which the quantum algorithm achieves a (superpolynomial) speedup over classical algorithms.

2.2.3 Classical algorithms for Betti number estimation

In this section we will closely investigate the state-of-the-art classical algorithms, to analyze whether it is possible to strengthen the argument for quantum advantage (or, to actually find an efficient classical algorithm) for the topological data analysis problem. In particular, we will cover classical algorithms based on numerical linear algebra or random walks and analyze the theoretical hurdles that, at least currently, stymie them from performing equally as well as the quantum algorithm.

To the best of our knowledge, the best known classical algorithms for approximate Betti number estimation is based on a numerical linear algebra algorithm for low-lying spectral density estimation [190, 59, 70, 124]. These algorithms typically run in time linear in the number of nonzero entries. Since combinatorial Laplacians are n-sparse, the number of nonzero entries of the combinatorial Laplacian – and hence also the runtime of the best known classical algorithm for approximate Betti number estimation – scales as

$$\mathcal{O}\left(n \cdot \chi_k\right) \in \mathcal{O}\left(n^{k+1}\right).$$

Recall that the quantum algorithm for Betti number estimation can estimate approximate Betti numbers in time polynomial in n if we can efficiently prepare the maximally mixed state over the cliques of a given size (e.g., if it satisfies Eq. (2.31)). For graphs that satisfy this condition, we conclude that the quantum algorithm for Betti number estimation achieves an exponential speedup over the best known classical algorithms if the size of the combinatorial Laplacian – i.e., the number of $(k+1)$-cliques – scales exponential in n (which requires k to scale with n). For exponential speedups for Betti number estimation, we also require that the smallest nonzero eigenvalue of the combinatorial Laplacian scales at least inverse polynomially in n.

To investigate the actual hardness of approximate Betti number estimation, we go one step further and discuss new possibilities for efficient classical algorithms. In particular, we investigate potential classical algorithms that take into account the specifics of the combinatorial Laplacian by using carefully designed random walks. Firstly, there exists a classical random walk based algorithm that can approximate the spectrum of the 0th combinatorial Laplacian (i.e., the ordinary graph Laplacian) up to ϵ distance in the Wasserstein-1 metric in time $\mathcal{O}\left(\exp(1/\epsilon)\right)$ (i.e., independent of the size of the graph) [65]. To generalize this to higher-order combinatorial Laplacians, one would have to construct an efficiently implementable walk operator whose spectral properties coincide with the higher-order combinatorial Laplacian. While potential candidates for such higher-order walk operators have previously been studied [143, 154], relatively little is known about such higher-order walk operators. Note that such a construction must take into account the specifics of the combinatorial Laplacian, since if the construction would work for arbitrary sparse Hermitian matrices, then this would lead to an efficient classical algorithm for LLSD (which by Theorem 5 is widely-believed to be impossible). Recently, Berry et al. [33] and Apers et al. [22] proposed new classical algorithms for Betti number estimation based on random walks that works best precisely in the regime where the quantum algorithm works best. However, the scaling of these algorithm with respect to the spectral gap is exponentially worse compared to the existing quantum algorithms. Thus, to obtain a quantum speedup, we must ensure that the spectral gap of the combinatorial Laplacian is not too large such that these classical algorithms become efficient.

2.3 Structural risk minimization

When looking for the optimal family of classifiers for a given learning problem, it is important to carefully select the family's *complexity* (also known as expressivity or capacity). For instance, in the case of linear classifiers, it is important to select what kind of hyperplanes one allows the classifier to use. Generally, the more complex the family is, the lower the training errors will be. However, if the family becomes overly complex, then it becomes more prone to worse generalization performance (i.e., due to overfitting). Structural risk minimization is a concrete method that balances this trade-off in order to obtain the best possible performance on unseen examples. Specifically, structural risk minimization aims to saturate well-established upper bounds on the expected error of the classifier that consist of the sum of two inversely related terms: a *training error* term, and a *complexity term* penalizing more complex models.

In statistical learning theory it is generally assumed that the data is sampled according to some underlying probability distribution P on $\mathcal{X} \times \{-1, +1\}$. The goal is to find a classifier that minimizes the probability that a random pair sampled according to P is misclassified. That is, the goal is to find a classifier $c_{f,d}(x) = \operatorname{sign}(f(x) - d)$ that minimize the *expected error* given by

$$\operatorname{er}_P(c_{f,d}) = \Pr_{(x,y)\sim P}\big(c_{f,d}(x) \neq y\big). \tag{2.33}$$

As one generally only has access to training examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ that are sampled according to the distribution P, it is not possible to compute er_P directly. Nonetheless, one can try to approximate Equation (2.33) using *training errors* such as

$$\widehat{\mathrm{er}}_{\mathcal{D}}(c_{f,d}) = \frac{1}{m}\Big|\big\{i \;:\; c_{f,d}(x_i) \neq y_i\big\}\Big|, \tag{2.34}$$

$$\widehat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(c_{f,d}) = \frac{1}{m}\Big|\big\{i \;:\; y_i \cdot \big(f(x_i) - d\big) < \gamma\big\}\Big|, \;\; \gamma \in \mathbb{R}_{\geq 0}. \tag{2.35}$$

Intuitively, $\widehat{\mathrm{er}}_{\mathcal{D}}$ in Equation (2.34) represents the frequency of misclassified training examples, and $\widehat{\mathrm{er}}^{\gamma}_{\mathcal{D}}$ in Equation (2.35) represents the frequency of training examples that are either misclassified or are "within margin γ from being misclassified". In particular, for $\gamma = 0$ both training error estimates are identical (i.e., $\widehat{\mathrm{er}}_{\mathcal{D}} = \widehat{\mathrm{er}}^{0}_{\mathcal{D}}$). When selecting the optimal classifiers from a given model one typically searches for the classifier that minimizes the training error (in practice more elaborate and smooth loss functions are used), which is referred to as *empirical risk minimization*. The problem that structural risk minimization aims to tackle is how to optimally select a model such that one will have some guarantee that the training error will be close to the expected error.

Structural risk minimization uses expected error bounds – two of which we will discuss shortly – that involve a training error term, and a complexity term that penalizes more complex models. This complexity term usually scales with a certain measure of the complexity of the family of classifiers. A well known example of such a complexity measure is the Vapnik-Chervonenkis dimension.

Definition 1 (VC dimension [194])**.** *Let $\mathcal{C}$ be a family of functions on $\mathcal{X}$ taking values in $\{-1, +1\}$. We say that a set of points $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}$ is shattered by $\mathcal{C}$ if for all $y \in \{-1, +1\}^m$, there exists a classifier $c_y \in \mathcal{C}$ that satisfies $c_y(x_i) = y_i$. The VC dimension of $\mathcal{C}$ defined as*

$$\mathrm{VC}\big(\mathcal{C}\big) = \max\big\{m \mid \exists\{x_1, \ldots, x_m\} \subset \mathcal{X} \text{ that is shattered by } \mathcal{C}\big\}.$$

Besides the VC dimension we also consider a complexity measure called the *fat-shattering dimension*, which can be viewed as a generalization of the VC dimension to real-valued functions. An important difference between the VC dimension and the fat-shattering dimension is that the latter also takes into account the so-called *margins* that the family of classifiers can achieve. Here the margin of a classifier $c_{f,d}(x) = \mathrm{sign}\big(f(x) - d\big)$ on a set of examples $\{x_i\}_{i=1}^m$ is given by $\min_i |f(x_i) - d|$. Throughout the literature, this is often referred to as the functional margin.

Definition 2 (Fat-shattering dimension [117])**.** *Let $\mathcal{F}$ be a family of real-valued functions on $\mathcal{X}$. We say that a set of points $X = \{x_1, \ldots, x_m\} \subset \mathcal{X}$ is γ-shattered by $\mathcal{F}$ if there exists an $s \in \mathbb{R}^m$ such that for all $y \in \{-1, +1\}^m$, there exists a function*

$f_y \in \mathcal{F}$ *satisfying*

$$f_y(x_i) \begin{cases} \leq s_i - \gamma & \text{if } y_i = -1, \\ \geq s_i + \gamma & \text{if } y_i = +1. \end{cases}$$

The fat-shattering dimension of $\mathcal{F}$ *is a function* $\mathrm{fat}_{\mathcal{F}} : \mathbb{R} \to \mathbb{Z}_{\geq 0}$ *that maps*

$$\mathrm{fat}_{\mathcal{F}}(\gamma) = \max\big\{m \;\big|\; \exists\{x_1, \ldots, x_m\} \subset \mathcal{X} \text{ that is } \gamma\text{-shattered by } \mathcal{F}\big\}.$$

We will now state two expected error bounds that can be used to perform structural risk minimization. These error bounds theoretically quantify how an increase in model complexity (i.e., VC dimension or fat-shattering dimension) results in a worse expected error (i.e., due to overfitting). First, we state the expected error bound that involves the VC dimension.

Theorem 1 (Expected error bound using VC dimension [201])**.** *Consider a set of functions* $\mathcal{C}$ *on* $\mathcal{X}$ *taking values in* $\{-1, +1\}$*. Suppose* $\mathcal{D} = \big\{(x_i, y_i)\big\}_{i=1}^{m}$ *is sampled using* m *independent draws from* P*. Then, with probability at least* $1-\delta$*, the following holds for all* $c \in \mathcal{C}$*:*

$$\mathrm{er}_P(c) \leq \widehat{\mathrm{er}}_{\mathcal{D}}(c) + 62\sqrt{\frac{k}{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}} \tag{2.36}$$

where $k = \mathrm{VC}\big(\mathcal{C}\big)$*.*

Next, we state the expected error bound that involves the fat-shattering dimension. One possible advantage of using the fat-shattering dimension instead of the VC dimension is that it can take into account the margin that the classifier achieves on the training examples. This turns out to be useful since this margin can be used to more precisely fine-tune the expected error bound.

Theorem 2 (Expected error bound using fat-shattering dimension [28])**.** *Consider a set of real-valued functions* $\mathcal{F}$ *on* $\mathcal{X}$*. Suppose* $\mathcal{D} = \big\{(x_i, y_i)\big\}_{i=1}^{m}$ *is sampled using* m *independent draws from* P*. Then, with probability at least* $1 - \delta$*, the following holds for all* $c(x) = \mathrm{sign}\big(f(x) - d\big)$ *with* $f \in \mathcal{F}$ *and* $d \in \mathbb{R}$*:*

$$\mathrm{er}_P(c) \leq \widehat{\mathrm{er}}^{\gamma}_{\mathcal{D}}(c) + \sqrt{\frac{2}{m}\Big(k \log(34em/k) \log_2(578m) + \log(4/\delta)\Big)}. \tag{2.37}$$

where $k = \mathrm{fat}_{\mathcal{F}}(\gamma/16)$*.*

Remark(s). *If the classifier can correctly classify all examples in* $\mathcal{D}$*, then the optimal choice of* γ *in the above theorem is the margin achieved on the examples in* $\mathcal{D}$*, i.e.,* $\gamma = \min_{x_i \in \mathcal{D}} \big|f(x_i) - d\big|$*.*

Generally, the more complex a family of classifiers is, the larger its generalization errors are. This correlation between a family's complexity and its generalization errors is theoretically quantified in Theorems 1 and 2. Specifically, the more complex the family is the larger its VC dimension will be, which strictly increases the second

term in Equation 2.36 that corresponds to the generalization error. Note that for the fat-shattering dimension in Theorem 2 this is not as obvious. In particular, a more complex model could achieve a larger margin γ, which actually decreases the second term in Equation 2.37 that corresponds to the generalization error.

Theorems 1 and 2 establish that in order to minimize the expected error, we should aim to minimize either of the sums on the right-hand side of Equations (2.36) or (2.37) (depending on which complexity measure one wishes to focus on). Note that in both cases the first term corresponds to a training error and the second term corresponds to a complexity term that penalizes more complex models. Crucially, the effect that the complexity measure of the family of classifiers has on these terms is inversely related. Namely, a large complexity measure generally gives rise to smaller training errors, but at the cost of a larger complexity term. Balancing this trade-off is precisely the idea behind structural risk minimization. More precisely, structural risk minimization selects a classifier that minimizes either of the expected error bounds stated in Theorem 1 or 2, by selecting the classifier from a family whose complexity measure is fine-tuned in order to balance both terms on the right-hand side of Equations (2.36) or (2.37). Note that limiting the VC dimension and fat-shattering dimension does not achieve the same theoretical guarantees on the generalization error, and it will generally give rise to different performances in practice (as also discussed Section 4.2). An overview of the trade-off in the error bounds stated in Theorems 1 and 2 is depicted in Figure 2.6.

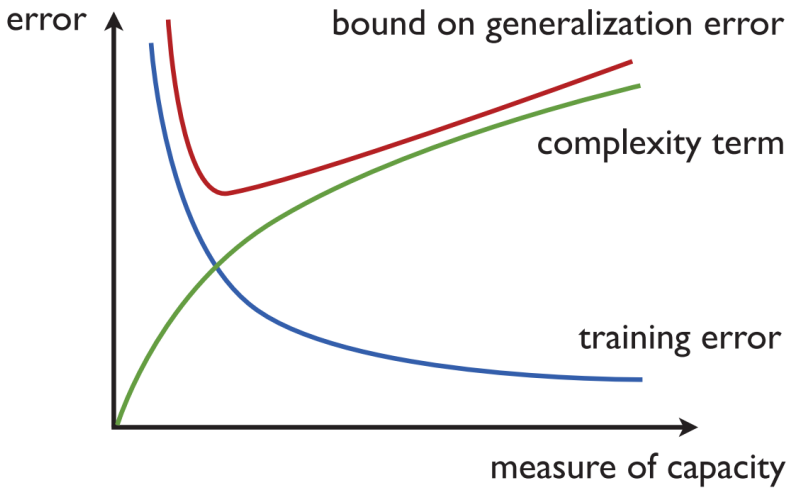


Figure 2.6: Illustration of structural risk minimization taken from [139]. Increasing the complexity causes the training error (blue) to decrease, while it increases the complexity term (green). Structural risk minimization selects the classifier minimizing the expected error bound in Eqs. (2.36) and (2.37) given by the sum of the training error and the complexity term (red).

2.4 Reinforcement learning

In this section, we introduce the basic ideas of reinforcement learning (for a more in-depth treatment we refer to [182]). Intuitively, in reinforcement learning the problem is to learn from interactions to achieve a goal. The way this is generally modelled is through the *agent-environment* interaction setup depicted in Figure 2.4. In this setup, the learner takes the role of the agent which continually interacts with the environment by performing *actions*. After every action from the agent, the environment responds by presenting the agent with a new *state*, and some numerical value called the *reward* that the agent tries to maximize over time.

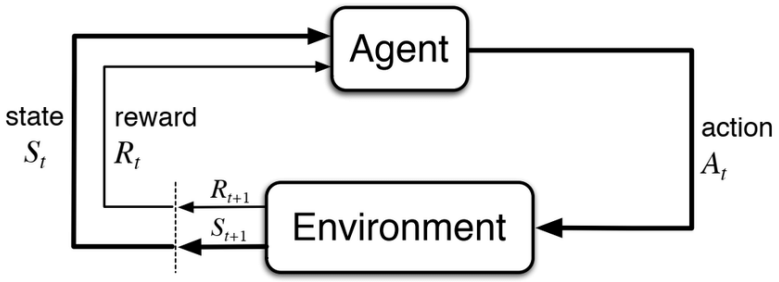


Figure 2.7: The agent-environment interaction setup (taken from [182]).

More precisely, the agent and the environment interact with each other for a number of discrete time steps $t = 0, 1, 2, \ldots$. At every time step t, the agent is presented with a state S_t from the environment, at which point it has to select an action A_t, after which the environment updates its state to a new state S_{t+1} that it sends to the agent together with some reward R_{t+1} Note that the actions the agent can choose from can depend on the current state of the environment. The way an action A_t causes the environment to change $S_t \to S_{t+1}$ (together with the associated reward R_{t+1}) is often modelled using a Markov Decision Process (MDP). We choose not to discuss the MDP-setup in detail here, and instead provide a more high-level overview of reinforcement learning (see [182] for more details).

Generally, the way the agent chooses its actions based on the state of the environment is modelled by what is called its *policy* $\pi(. \mid .)$. The agent's policy maps the current state to a probability distribution over all possible actions, and the agent simply samples from it to selects its next step. More precisely, we let $\pi(a \mid s)$ denote the probability that the agent selects action $A_t = a$ given that the current state of the environment is $S_t = s$.

A standard way of training a reinforcement learning agent is through *policy iteration*, which consists of two consecutive steps: *policy evaluation* and *policy improvement*. First, in the policy evaluation step, the agent uses a policy to interact with

the environment and collect rewards for a given number of steps. Afterwards, in the policy improvement step, the agent updates its policy based on the interactions it had gained in the policy evaluation step. There are many different ways to perform the policy improvement step, e.g., in state of the art deep-learning methods one often uses Q-learning. In short, in Q-learning the agent trains a model (e.g., a deep neural network) that learns what the expected rewards are for a given action in a given state, and uses this to selects its next action. For more details on Q-learning or other policy improvement methods see [182]. Next, we focus on a different method called the *policy gradient method*, which we will use throughout Chapter 5.

In the policy gradient method, the agent is equipped with a differential policy π_θ, where $\theta \in \mathbb{R}^\ell$ denote the differential parameters. For instance, π_θ could be implemented using a neural network, in which case the θ correspond to the weights of the neural network. In Chapter 5 we show how one can use parameterized quantum circuits to encode a differential policy, and how these policies are able to outperform classical policies. The main idea behind policy-gradient methods is to use a scalar performance measure $J(\theta)$, and to update the parameters using a gradient-ascent step

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta)}, \tag{2.38}$$

where $\widehat{\nabla J(\theta)}$ denotes a (stochastic) estimate of the gradient of J with respect to θ. In our case, our performance measure $J(\theta)$ will be the *value function* of the initial state of the environment with respect to the current policy $v_{\pi_\theta}(s_0)$. The value function $v_\pi(s)$ denotes the expected discounted rewards obtained by an agent following policy π when starting in state s. More precisely, we set $J(\theta) = v_{\pi_\theta}(s_0)$ where s_0 denotes the initial state of the environment and

$$v_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \tag{2.39}$$

where $\gamma \in [0, 1)$ is some discount factor chosen beforehand. What remains is to find a way to obtain a (stochastic) estimate of the gradient of $v_{\pi_\theta}(s_0)$ in order to implement the update step in Eq. (2.38). A priori, when computing the gradient of $v_{\pi_\theta}(s_0)$, we see that it depends on the effect of the policy on the state distribution of the environment. But, since we do not know what the effect of the policy is on the state distribution of the environment, how are we able to compute the gradient of $v_{\pi_\theta}(s_0)$? Fortunately, there is a nice theoretical answer to this question in the form of the *policy gradient theorem*, which provides an analytic expression for the gradient of $v_{\pi_\theta}(s_0)$ that does *not* involve the derivative of the state distribution of the environment.

For our purposes, we will stick to the vanilla version of the policy-gradient algorithm called REINFORCE [182], as used throughout Chapter 5. In this setting, the policy-gradient theorem tells us that

$$\nabla v_{\pi_\theta}(s_0) = \mathbb{E}_\pi\left[G_t \frac{\nabla \pi_\theta(A_t \mid S_t)}{\pi_\theta(A_t \mid S_t)}\right], \tag{2.40}$$

where $G_t = \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$ is called the *return*. The expression on the right

hand side of Eq. (2.40) is exactly what we need, a quantity that we can sample (i.e., it does not require us to know the state distribution of the environment) whose expectation value is equal to the gradient. In other words, we can use the right hand side of Eq. (2.40) to obtain a (stochastic) estimate of $\nabla v_{\pi_\theta}(s_0)$. In particular, we can let the agent interact with the environment for a given number of steps T and collect the interactions

$$\{(s_0, a_0, r_1), (s_1, a_1, r_2), \ldots, (s_T, a_T, r_{T+1})\}.$$

Next, we loop over each step in the interactions $t = 0, \ldots, T$, and at each step t we compute $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_t$, and we update the parameters

$$\theta = \theta + \alpha \gamma^t G_t \nabla \frac{\nabla \pi_\theta(a_t \mid a_t)}{\pi_\theta(a_t \mid a_t)}. \tag{2.41}$$

This is all summarized in Algorithm 1 in Figure 5.1.2.

2.5 Computational learning theory

Quantum machine learning (QML) [35, 26] is a bustling field with the potential to deliver quantum enhancements for practically relevant problems. An important goal of the community is to find practically relevant learning problems for which one can prove that quantum learners have an exponential advantage over classical learners. In Chapter 6 of this thesis, we study how to achieve such exponential separations between classical and quantum learners for problems with classical data in the efficient probably approximately correct (PAC) learning framework. But before we do so, we will first introduce the required background and definitions from computational learning theory together with some more computational complexity theory.

2.5.1 Learning separations in the PAC learning framework

As already mentioned, we use the standard terminology of the efficient *probably approximately correct* (PAC) learning framework, and we focus on the supervised learning setting (for an overview of the generative modelling setting see [185]). In this framework a learning problem is defined by a *concept class* $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{C}_n$ is a set of *concepts*, which are functions from some *input space* $\mathcal{X}_n$ (in this thesis we assume $\mathcal{X}_n$ is either $\{0,1\}^n$ or $\mathbb{R}^n$) to some *label set* $\mathcal{Y}_n$ (in this thesis we assume $\mathcal{Y}_n$ is $\{0,1\}$, with the exception of Section 6.1.3 where it is $\{0,1\}^n$)[7]. As input the learning algorithm has access to a procedure $EX(c, \mathcal{D}_n)$ (sometimes called an *example oracle*) that runs in unit time, and on each call returns a labeled *example* $(x, c(x))$, where $x \in \mathcal{X}_n$ is drawn according to *target distributions* $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$. Finally, the learning algorithm has associated to it a hypothesis class $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$, and its goal is to output a *hypothesis* $h \in \mathcal{H}_n$ – which is another function from $\mathcal{X}_n$ to $\mathcal{Y}_n$– that is in some sense "close" to the concept $c \in \mathcal{C}_n$ generating the examples (we will make this more precise shortly).

[7]Since our focus is on the computational complexity of the learner, we choose to explicitly highlight the relevance of the instance size n in our notation.

In the statistical version of the PAC learning framework the learning algorithm has to identify (and/or evaluate) a good hypothesis using $\mathcal{O}(\text{poly}(n))$ many queries to $EX(c, \mathcal{D}_n)$, and the computational complexity (i.e., "runtime") of the learning algorithm is not considered. In this thesis however, we focus on the *efficient* PAC learning framework, where the learning algorithm must output such a good hypothesis in *time* $\mathcal{O}(\text{poly}(n))$ (note that this also implies that the learning algorithm can only use $\mathcal{O}(\text{poly}(n))$ many queries to $EX(c, \mathcal{D}_n)$). Moreover, in this thesis, we study exponential separations specifically with respect to the time complexity of the learning algorithms.

The PAC learning framework formalizes supervised learning. For instance, in the learning scenario where one wants to detect a specific object in an image, the concepts are defined to attain the value 1 when the object is present and 0 otherwise. Moreover, the oracle represents the set of training examples that is available in supervised learning. We formally define efficient PAC learnability as follows.

Definition 3 (Efficient PAC learnability)**.** *A concept class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ is efficiently PAC learnable under target distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ if there exists a hypothesis class $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ and a (randomized) learning algorithm $\mathcal{A}$ with the following property: for every $c \in \mathcal{C}_n$, and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, if $\mathcal{A}$ is given access to $EX(c, \mathcal{D}_n)$ and ϵ and δ, then with probability at least $1 - \delta$, $\mathcal{A}$ outputs a specification[8] of some $h \in \mathcal{H}_n$ that satisfies*

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\big[h(x) \neq c(x)\big] \leq \epsilon.$$

Moreover, the learning algorithm $\mathcal{A}$ must run in time $\mathcal{O}(\text{poly}(n, 1/\epsilon, 1/\delta))$.

In the above definition, the probability $1 - \delta$ is over the random examples drawn from $EX(c, \mathcal{D}_n)$ and over the internal randomness of $\mathcal{A}$. If the learning algorithm is a polynomial-time classical algorithm (or, a quantum algorithm), we say that the concept class is *classically learnable* (or, *quantumly learnable*, respectively). An important thing to note in the above definition is that the learner itself consists of two parts: a hypothesis class, and a learning algorithm. More precisely, the learner consists of a family of functions that it will use to approximate the concepts (i.e., the hypothesis class), and of a way to select which function from this family is the best approximation for a given concept (i.e., the learning algorithm). This is generally not very different from how supervised learning is done in practice. For example, in deep learning the hypothesis class consists of all functions realizable by a deep neural network with some given architecture, and the learning algorithm uses gradient descent to find the best hypothesis (i.e., the best assignment of weights).

To solve a given learning problem according to Definition 3, one thus needs to construct a learner, which consists of both a learning algorithms as well as a hypothesis class. Specifically, one is thus able to specifically tailor the hypothesis class to the learning problem that one is trying to solve. Another way to define a learning problem and how to solve it, would be to constrain the learner to only output hypotheses from

[8]The hypotheses (and concepts) are specified according to some enumeration $R : \cup_{n\in\mathbb{N}}\{0,1\}^n \to \cup_n \mathcal{H}_n$ (or, $\cup_n \mathcal{C}_n$) and by a "specification of $h \in \mathcal{H}_n$" we mean a string $\sigma \in \{0,1\}^*$ such that $R(\sigma) = h$ (see [116] for more details).

a given hypothesis class (which would then be part of the specification of the learning problem). In this thesis, our main focus is on investigating the limitations of *all* possible classical learners for a given task. To do so, we primarily focus on setting where constructing the right hypothesis class is also part of the learning problem. Our goal is to demonstrate separations that establish the inability of classical learners, regardless of the hypotheses they use, to efficiently solve a learning problem that can be solved by a quantum learner. Nonetheless, in certain cases, it is more natural to consider the setting where the learner is constrained to output hypotheses from a given hypothesis class. We explore this setting, along with an instance of it which is called *proper PAC learning*, in Sections 2.5.1 and 6.3.3.

When one is able to specifically tailor the hypothesis class to a given learning problem (as is the case in Definition 3), it turns out to be necessary to limit the computational power of the hypotheses. Constraining the learning algorithm to run in polynomial-time turns out to be pointless if one allows arbitrary superpolynomial-time hypotheses. More precisely, if we allow superpolynomial-time hypotheses, then any concept class that can be learned by a superpolynomial-time learning algorithm, can also be learned by a polynomial-time learning algorithm (see Appendix D.1.1 for more details). Intuitively, this is because by tailoring the hypothesis class one can "offload" the learning algorithm onto the evaluation of the hypotheses, which makes any constraints on the learning algorithm pointless. This is different when we constrain the learner to only be able to output hypotheses from a given hypothesis class, in which case it can be meaningful and natural to consider hypotheses with superpolynomial runtimes (see also Sections 2.5.1 and 6.3.3). Additionally, because we are studying separations between classical and quantum learners, we make the distinction whether the hypotheses are efficiently evaluatable classically or quantumly.

Definition 4 (Efficiently evaluatable hypothesis class)**.** *A hypothesis class* $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ *is* classically (quantumly) efficiently evaluatable *if there exists a classical (respectively quantum) polynomial-time evaluation algorithm* $\mathcal{A}_{\text{eval}}$ *that on input* $x \in \mathcal{X}_n$ *and a specification of a hypothesis* $h \in \mathcal{H}_n$*, outputs* $\mathcal{A}_{\text{eval}}(x, h) = h(x)$*.*

For example, the hypotheses could be specified by a polynomial-sized Boolean circuit, in which case they are *classically* efficiently evaluatable. On the other hand, the hypotheses could also be specified by polynomial-depth quantum circuits, in which case they are *quantumly* efficiently evaluatable. If the family of quantum circuits that make up the hypothesis class is BQP-complete, then the hypothesis class will be quantumly efficiently evaluatable, but not classically efficiently evaluatable (assuming $\mathsf{BPP} \neq \mathsf{BQP}$). In this thesis we will drop the "efficiently" and simply call a hypothesis class classically- or quantumly evaluatable.

Given the definitions above one may assume that there is only one way to define a learning separation in the PAC learning framework. However, it is in fact more subtle, and there are various definitions that each have operationally different meanings. In particular, one needs to differentiate whether the learning algorithm is classical or quantum, and whether the hypothesis class is classically- or quantumly- evaluatable. As a result, we define four categories of learning problems: concept classes that are either *classically-* or *quantumly- learnable* (i.e., whether the learning algorithm is classical or quantum), using a *classically-* or *quantumly-* evaluatable hypothesis class.

We denote these categories by $\mathsf{CC}, \mathsf{CQ}, \mathsf{QC}$, and QQ, where the first letter signifies whether the concept class is classically- or quantumly- learnable , and the second letter signifies whether the learner uses a classically- or quantumly- evaluatable hypothesis class. These distinctions are *not* about the nature of the data (i.e., we only consider the setting where the examples are classical) as it often occurs in literature, and even on the Wikipedia-page on quantum machine learning.

Definition 5 (Categories of learning problem)**.**

- *Let* CC *denote the set of tuples* $(\mathcal{C}, \mathcal{D})$ *such that* $\mathcal{C}$ *is* ***classically*** *learnable under target distributions* $\mathcal{D}$ *with a* ***classically*** *evaluatable hypothesis class.*
- *Let* CQ *denote the set of tuples* $(\mathcal{C}, \mathcal{D})$ *such that* $\mathcal{C}$ *is* ***classically*** *learnable under target distributions* $\mathcal{D}$ *with a* ***quantumly*** *evaluatable hypothesis class.*
- *Let* QC *denote the set of tuples* $(\mathcal{C}, \mathcal{D})$ *such that* $\mathcal{C}$ *is* ***quantumly*** *learnable under target distributions* $\mathcal{D}$ *with a* ***classically*** *evaluatable hypothesis class.*
- *Let* QQ *denote the set of tuples* $(\mathcal{C}, \mathcal{D})$ *such that* $\mathcal{C}$ *is* ***quantumly*** *efficiently learnable under target distributions* $\mathcal{D}$ *with a* ***quantumly*** *evaluatable hypothesis class.*

We remark that our definitions do not (yet) talk about the computational tractability of the concepts, the importance of which we will discuss in Section 2.5.2 and throughout Sections 6.1 and 6.2. We now proceed with a few observations. Firstly, since any classical algorithm can be simulated by a quantum algorithm it is clear that $\mathsf{CC} \subseteq \mathsf{CQ}$, $\mathsf{CC} \subseteq \mathsf{QC}$, $\mathsf{CC} \subseteq \mathsf{QQ}$, $\mathsf{CQ} \subseteq \mathsf{QQ}$, and $\mathsf{QC} \subseteq \mathsf{QQ}$. Secondly, we make the non-trivial observation that if the hypothesis class can use a quantum evaluation algorithm (i.e., it is quantumly evaluatable), then it does not matter whether we constrain the learning algorithm to be a classical- or a quantum- algorithm. More precisely, as a first result we show that any learning problem that is quantumly learnable using a quantumly evaluatable hypothesis class is also classically learnable using *another* quantumly evaluatable hypothesis class. This observation is summarized in the lemma below, and we defer the proof to Appendix D.1.2.

Lemma 3. $\mathsf{CQ} = \mathsf{QQ}$.

The above lemma is analogous to why we constrain the hypotheses to be efficiently evaluatable, in the sense that by changing the hypothesis class one can "offload" the *quantum* learning algorithm onto the evaluation of the *quantum* hypotheses. We reiterate that it is critical that one can change the hypothesis class when mapping a learning problem in QQ to CQ. If the learner is constrained to output hypotheses from a fixed hypothesis class, then such a collapse does not happen.

Having studied the relations between the categories learning problems, we can now specify what it means for a learning problem to exhibit a separation between classical and quantum learners.

Definition 6 (Learning separation)**.** *A learning problem* $L = (\mathcal{C}, \mathcal{D})$ *is said to exhibit a*

- CC/QC *separation if* $L \in \mathsf{QC}$ *and* $L \notin \mathsf{CC}$.
- CC/QQ *separation if* $L \in \mathsf{QQ}$ *and* $L \notin \mathsf{CC}$.

Firstly, note that due to the previously listed inclusions any CC/QC separation is also a CC/QQ separation. Secondly, note that by fully relying on the classical intractability of concepts one can construct trivial learning separations that are less about "learning" in an intuitive sense. More precisely, consider the separation exhibited by the concept class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$, where each $\mathcal{C}_n$ consists of a *single* concept that is classically hard to evaluate on a fraction of inputs even in the presence of data, yet it can be efficiently evaluated by a quantum algorithm. This singleton concept class is clearly quantumly learnable using a quantumly evaluatable hypothesis class. Also, it is not classically learnable using any classically evaluatable hypothesis class, since this would violate the classical intractability of the concepts. However, note that the quantum learner *requires no data* to learn the concept class, so it is hard to argue that this is a genuine learning problem. We will discuss how to construct examples of such concept classes in Sections 6.1.1 and 6.1.2.

Observation 1 (Trivial learning separation without data)**.** *Consider a family of concept classes* $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$*, where each* $\mathcal{C}_n = \{c_n\}$ *consists of a single concept that is classically hard to evaluate on a fraction of inputs when given access to examples, yet it can be efficiently evaluated by a quantum algorithm. Then,* $\mathcal{C}$ *exhibits a* CC/QQ *separation which is quantum learnable without requiring data.*

We want to emphasize that some concept classes are *efficiently evaluatable on a classical computer*, yet they are *not classically learnable*. One such example is the class of polynomially-sized logarithmic-depth Boolean circuits [116]. Moreover, in Section 6.1.3, we provide an example of concept class which (assuming a plausible but relatively unexplored hardness assumption) exhibits a CC/QC separation where the concepts are efficiently evaluatable on a classical computer.

Learning separations with a fixed hypothesis class and proper PAC learning

In some practical settings, it can be natural to constrain the learner to only output hypotheses from a fixed hypothesis class. To give a physics-motivated example, when studying phases of matter one might want to identify what observable properties characterize a phase. One can formulate this problem as finding a specification of the correct hypothesis selected from a hypothesis class consisting of possible *order parameters*. More precisely, we fix the hypotheses to be of a particular form, e.g., those that compute certain expectation values of ground states given a specification of a Hamiltonian[9]. We further discuss this setting of characterizing phases of matter in Section 6.3.3, where we also discuss Hamiltonian learning as a natural setting in which the learner is constrained to output hypotheses from a fixed hypothesis class.

[9]Note the computation of these hypotheses can be QMA-hard, as it involves preparing ground states. Nonetheless, we can still study whether a learner is able to *identify* which of these hypotheses matches the data.

Recall that in the standard PAC learning framework discussed in the previous section, the learner is free to output arbitrary hypotheses (barring tractability constraints discussed in Appendix D.1.1). It therefore fails to capture the setting where one aims to characterize phases of matter, as the learner might output hypotheses that are not order parameters, which will not allow one to identify physical properties that characterize a phase. To remedy this, one could consider the setting where the learner is constrained to output hypotheses from a fixed hypothesis class.

Definition 7 (Efficient PAC learnability with fixed hypothesis class)**.** *A concept class $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ is efficiently PAC learnable with a fixed hypothesis class $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ under target distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ if there exists a (randomized) learning algorithms $\mathcal{A}$ with the following property: for every $c \in \mathcal{C}_n$, and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, if $\mathcal{A}$ is given access to $EX(c, \mathcal{D}_n)$ and ϵ and δ, then with probability at least $1 - \delta$, $\mathcal{A}$ outputs a specification of some $h \in \mathcal{H}_n$ that satisfies*

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\big[h(x) \neq c(x)\big] \leq \epsilon.$$

Moreover, the learning algorithm $\mathcal{A}$ must run in time $\mathcal{O}(\mathrm{poly}(n, 1/\epsilon, 1/\delta))$.

In the above definition, the probability $1 - \delta$ is over the random examples from $EX(c, \mathcal{D}_n)$ and over the internal randomization of $\mathcal{A}_n$. If the learning algorithm is a polynomial-time classical algorithm (or, a quantum algorithm), we say that the concept class is *classically learnable with fixed hypothesis class* (or, *quantumly learnable with fixed hypothesis class*, respectively). An example of learning with a fixed hypothesis class is that of *proper PAC learning*. In proper PAC learning the learner is constrained to only output hypothesis from the concept class it is trying to learn.

We emphasize again that if the learner is constrained to output hypotheses from a fixed hypothesis class, then it is allowed and reasonable for the hypothesis class to be (classically- or quantumly-) intractable. In particular, doing so will not trivialize the definitions as it did in the standard PAC learning framework (see Appendix D.1) as this requires one to be able to change the hypotheses.

In the setting where the learner is constrained to output hypotheses from a fixed hypothesis class, it is relatively clear how to define a learning separation. In particular, one only has to distinguish whether the learning algorithm is an efficient classical- or quantum- algorithm, which we capture by defining the following categories of learning problems.

Definition 8 (Categories of learning problem – fixed hypothesis class $\mathcal{H}$)**.**

- *Let $\mathsf{C}_{\mathcal{H}}$ denote the set of tuples $\big(\mathcal{C}, \mathcal{D}\big)$ such that $\mathcal{C}$ is **classically** learnable with fixed hypothesis class $\mathcal{H}$ under target distributions $\mathcal{D}$.*
- *Let $\mathsf{Q}_{\mathcal{H}}$ denote the set of tuples $\big(\mathcal{C}, \mathcal{D}\big)$ such that $\mathcal{C}$ is **quantumly** learnable with fixed hypothesis class $\mathcal{H}$ under target distributions $\mathcal{D}$.*

We can now specify what it means for a learning problem to exhibit a separation between classical and quantum learners in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class.

Definition 9 (Learning separation – fixed hypothesis class $\mathcal{H}$)**.**
A learning problem $L = (\mathcal{C}, \mathcal{D}) \in \mathsf{Q}_{\mathcal{H}}$ is said to exhibit a $\mathsf{C}_{\mathcal{H}}/\mathsf{Q}_{\mathcal{H}}$ separation if $L \notin \mathsf{C}_{\mathcal{H}}$.

In Sections 6.1.3 and 6.1.4, we provide examples of learning separations in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class. Moreover, in Section 6.3.3, we further discuss the practical relevance of this setting by discussing how it captures certain physics-motivated examples of learning settings.

Identification versus Evaluation

An important difference in what exactly entails a learning task in practice is whether the learner has to only *identify* a hypothesis that is close to the concept generating the examples, or whether the learner also has to *evaluate* the hypothesis on unseen examples later on. Moreover, these differences in tasks have implications for the role of quantum computers in achieving separations. This difference in tasks is reflected in two aspects within the definitions discussed in this section.

Firstly, this difference in tasks is reflected in the difference between CC/QQ and CC/QC separations. In particular, it is reflected in the task that requires a quantum computer (i.e., what task needs to be classically intractable yet efficient on a quantum computer). On the one hand, for a CC/QC separation, one has to show that only a quantum algorithm can *identify* how to label unseen examples using a classical algorithm. On the other hand, for a CC/QQ separation, one also needs to show that only a quantum algorithm can *evaluate* the labels of unseen examples. In Section 6.1.3, we provide an example of a CC/QC separation (contingent on a plausible though relatively unexplored hardness assumption), where the classical hardness lies in *identifying* an hypothesis matching the examples, since the concepts are efficiently evaluatable classically.

Secondly, the difference in tasks is also reflected in the difference between the setting where the learner is allowed to output arbitrary hypothesis, or whether it can only output hypotheses from a fixed hypothesis class. In the arbitrary hypothesis class setting, one has to demand that the hypotheses are efficiently evaluatable (i.e., see Appendix D.1), which allows the learner to efficiently *evaluate* the hypotheses on unseen examples. In the fixed hypothesis class setting, the hypotheses need not be efficiently evaluatable, and the learner is only required to *identify* the correct hypothesis without having to evaluate it on unseen examples. In Sections 6.1.3 and 6.1.4, we provide examples of separation in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class. Note that the classical hardness in these separation lies identifying the hypotheses, as we do not require the learner to evaluate the hypothesis on unseen examples afterwards.

2.5.2 Complexity theory

In this section we provide a short overview of the areas of complexity theory that we will refer to when discussing separations in the PAC learning framework. In particular, we focus on the computational hardness assumptions that one can leverage to establish a learning separation.

We turn our attention to the definition of the PAC learning framework (see Definition 3) and make some observations that will be relevant later. First, we note that the hypothesis that the learning algorithm outputs is only required to be correct with probability ϵ over the target distribution. In complexity theory, this is related to the notion of heuristic complexity classes (for more details see [37]). To define heuristic complexity classes, we first need to incorporate the target distribution as a part of the problem, which is done by considering distributional problems.

Definition 10 (Distributional problem [37])**.** *A distributional problem is a tuple* $(L, \mathcal{D})$*, where* $L \subseteq \{0,1\}^*$ *is a language*[10] *and* $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ *is a family of distributions such that* $\mathrm{supp}(\mathcal{D}_n) \subseteq \{0,1\}^n$*.*

A distributional problem at its core remains a decision problem, wherein the inputs follow a specific distribution. It is important not to confuse this with a *sampling problem*, in which the goal is to generate samples from a designated distribution. Having defined distributional problems, we now define the relevant heuristic complexity classes.

Definition 11 (Heuristic complexity [37])**.** *A distributional problem* $(L, \mathcal{D})$ *is in* HeurBPP *if there exists a polynomial-time randomized classical algorithm* $\mathcal{A}$[11] *such that for all* n *and* $\epsilon > 0$[12]*:*

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\Big[\mathsf{Pr}\big(\mathcal{A}(x, 0^{\lfloor 1/\epsilon\rfloor}) = L(x)\big) \geq \frac{2}{3}\Big] \geq 1-\epsilon, \tag{2.42}$$

where the inner probability is taken over the internal randomization of $\mathcal{A}$*.*

Analogously, we say that a distributional problem $(L, \mathcal{D})$ *is in* HeurBQP *if there exists a polynomial-time quantum algorithm* $\mathcal{A}$ *that satisfies the property in Eq.* (2.42)*.*

A related and perhaps better known area of complexity theory is that of *average-case* complexity. The main difference between average-case complexity and heuristic complexity, is that in the latter one is allowed to err, whereas in the former one can never err but is allowed to output "don't know". Note that an average-case algorithm can always be converted into a heuristic algorithm by simply outputting a random result instead of outputting "don't know". Similarly, if there is a way to efficiently check if a solution is correct, any heuristic algorithm can be turned into an average-case algorithm by outputting "don't know" when the solution is incorrect. Even though they are closely related, in the PAC learning framework one deals with heuristic complexity.

While heuristic-hardness statements are not as common in quantum computing literature, many cryptographic security assumptions (such as that of RSA and Diffie-Hellman) are in fact examples of heuristic-hardness statements. These heuristic-hardness statements are generally derived from *worst-case to average-case reductions*,

[10] Throughout this thesis, we also use an equivalent definition of a language $L \subseteq \{0,1\}^*$ by instead calling it a *problem* and defining it as a function $L : \{0,1\}^* \to \{0,1\}$ such that $L(x) = 1$ if and only if $x \in L$.

[11] More precisely, a Turing machine.

[12] Here $0^{\lfloor 1/\epsilon\rfloor}$ denotes the bitstring consisting of $\lfloor 1/\epsilon\rfloor$ zeroes (i.e., it is a unary specification of the precision ϵ).

which show that being correct with a certain probability over a specific input distribution is at least as difficult as being correct on all inputs. Problems that admit a worst- to average-case reduction are called *random self-reducible* (for a formal definition see [79]). It is worth noting that despite the term "average-case", these reductions can also yield heuristic hardness statements. Specifically, if one can efficiently check whether a solution is correct, then a worst-case to average-case reduction also results in a heuristic hardness statement when the worst-case is hard. For instance, a worst-case to average-case reduction by Blum and Micali [36] demonstrates that for the discrete logarithm problem being correct on any $\frac{1}{2} + \frac{1}{\text{poly}(n)}$ fraction of inputs is as difficult as being correct for all inputs (notably, modular exponentiation allows for efficient checking of the correctness of a discrete logarithm solution).

Finally, there is the notion of the example oracle. The fact that access to the example oracle radically enhances what can be efficiently evaluated is related (though not completely analogous, as we will explain below) to the notion of "advice" complexity classes such as $\mathsf{P/poly}$.

Definition 12 (Polynomial advice [24])**.** *A problem $L : \{0,1\}^* \to \{0,1\}$ is in $\mathsf{P/poly}$ if there exists a polynomial-time classical algorithm $\mathcal{A}$ with the following property: for every n there exists an advice bitstring $\alpha_n \in \{0,1\}^{\text{poly}(n)}$ such that for all $x \in \{0,1\}^n$:*

$$\mathcal{A}(x, \alpha_n) = L(x). \tag{2.43}$$

Analogously, we say that a problem L is in $\mathsf{BQP/poly}$ if there exists a polynomial-time quantum algorithm $\mathcal{A}$ with the following property: for every n there exists an advice bitstring $\alpha_n \in \{0,1\}^{\text{poly}(n)}$ such that for all $x \in \{0,1\}^n$:

$$\mathsf{Pr}\big(\mathcal{A}(x, \alpha_n) = L(x)\big) \geq \frac{2}{3}, \tag{2.44}$$

where the probability is taken over the internal randomization of $\mathcal{A}$.

Equivalently, $\mathsf{P/poly}$ can also be defined as the class of problems that can be solved by a *non-uniform* family of polynomial-size Boolean circuits. Specifically, for each instance size, a polynomially-sized circuit that solves the problem exists, though there does not need to be a polynomial-time algorithm that constructs these circuits from the instance size. Since in the PAC learning framework we deal with randomized learning algorithms one may want to consider $\mathsf{BPP/poly}$ instead, however by [15] we have that $\mathsf{BPP} \subseteq \mathsf{P/poly}$, and so $\mathsf{BPP/poly} = \mathsf{P/poly}$. On the other hand, from the perspective of the PAC learning framework, it is both natural and essential to allow the algorithm that uses the advice to err on a fraction of inputs, which is captured by the complexity class $\mathsf{HeurP/poly}$.

Definition 13 (Heuristic complexity with polynomial advice)**.** *A distributional problem $(L, \mathcal{D})$ is in $\mathsf{HeurP/poly}$ if there exists a polynomial-time classical algorithm $\mathcal{A}$ with the following property: for every n and $\epsilon > 0$ there exists an advice string $\alpha_{n,\epsilon} \in \{0,1\}^{\text{poly}(n,1/\epsilon)}$ such that:*

$$\mathsf{Pr}_{x \sim \mathcal{D}_n}\left[\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n,\epsilon}) = L(x)\right] \geq 1 - \epsilon. \tag{2.45}$$

Analogously, we say that $(L, \mathcal{D})$ is in HeurBQP/poly *if there exists a polynomial-time quantum algorithm $\mathcal{A}$ such that: for every n and $\epsilon > 0$ there exists an advice string $\alpha_{n,\epsilon} \in \{0,1\}^{\mathrm{poly}(n,1/\epsilon)}$ with the following property:*

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathsf{Pr}\left(\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n,\epsilon}) = L(x)\right) \geq \frac{2}{3}\right] \geq 1 - \epsilon, \tag{2.46}$$

where the inner probability is taken over the internal randomization of $\mathcal{A}$.

Note that in the PAC learning framework, the advice that the learning algorithm gets is of a specific form, namely that obtained through queries to the example oracle. This is more closely related to the notion of "sampling advice" complexity classes such as BPP/samp [109] defined below. In [109] it is shown that BPP/samp $\subseteq$ P/poly, i.e., sampling advice is not more powerful than the standard notion of advice.

Definition 14 (Sampling advice [109])**.** *A problem $L : \{0,1\}^* \to \{0,1\}$ is in* BPP/samp *if there exists polynomial-time classical randomized algorithms $\mathcal{S}$ and $\mathcal{A}$ such that for every n:*

- *$\mathcal{S}$ generates random instances $x \in \{0,1\}^n$ sampled from the distribution $\mathcal{D}_n$.*
- *$\mathcal{A}$ receives as input $\mathcal{T} = \{(x_i, L(x_i)) \mid x_i \sim \mathcal{D}_n\}_{i=1}^{\mathrm{poly}(n)}$ and satisfies for all $x \in \{0,1\}^n$:*

$$\mathsf{Pr}\big(\mathcal{A}(x, \mathcal{T}) = L(x)\big) \geq \frac{2}{3}, \tag{2.47}$$

  where the probability is taken over the internal randomization of $\mathcal{A}$ and $\mathcal{T}$.

Having related notions in the PAC learning framework to different areas of complexity theory, we are now ready to determine what computational hardness assumptions one can leverage to establish that no classical learner is able to learn a given concept class. More specifically, how hard must evaluating the concepts be for the concept class to not be classically learnable? Since the learning algorithm is a *randomized* algorithm that *heuristically* computes the concepts when provided with *advice* in the form of samples from the example oracle, the existence of a polynomial-time learning algorithm puts the concepts in a complexity class that we call HeurBPP/samp.

Definition 15. *A distributional problem $(L, \mathcal{D})$ is in* HeurBPP/samp *if there exists classical randomized algorithms $\mathcal{S}$ and $\mathcal{A}$ such that for every n:*

- *$\mathcal{S}$ generates random instances $x \in \{0,1\}^n$ sampled from the distribution $\mathcal{D}_n$.*
- *$\mathcal{A}$ receives as input $\mathcal{T} = \{(x_i, L(x_i)) \mid x_i \sim \mathcal{D}_n\}_{i=1}^{\mathrm{poly}(n)}$ and for every $\epsilon > 0$ satisfies:*

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathsf{Pr}\big(\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \mathcal{T}) = L(x)\big) \geq \frac{2}{3}\right] \geq 1 - \epsilon, \tag{2.48}$$

  where the inner probability is taken over the internal randomization of $\mathcal{A}$ and $\mathcal{T}$.

More precisely, if the concepts lie outside of HeurBPP/samp, then the concept class is not classically learnable. We can connect the class HeurBPP/samp to other complexity classes by adopting a proof strategy similar to that of [109] (we defer the proof to Appendix D.1.3).

Lemma 4. $\mathsf{HeurBPP/samp} \subseteq \mathsf{HeurP/poly}$.

By the above lemma, we find that any problem not in HeurP/poly is also not in HeurBPP/samp. Consequently, to show the non-learnability of a concept class, it is sufficient to show that the concept class includes concepts that are not in HeurP/poly.

Having discussed the related notions from computational learning theory and complexity theory, we are set to investigate how one establishes learning separations. First, in Section 6.1, we will analyze how existing learning separations have used efficient data generation, and we generalize this construction to (*i*) establish a learning separation (contingent on a plausible though relatively unexplored hardness assumption) with efficiently evaluatable concepts, and (*ii*) establish a learning separation in the setting where the learner is constrained to output an hypothesis from a fixed hypothesis class. Afterwards, in Section 6.2, we discuss the additional constructions required to prove separations in tune with the folklore that quantum machine learning is most likely to have it advantages when the data generated by a "genuine quantum process". For an overview of the learning separations discussed throughout this thesis see Table 2.1.

First proposed	**Concepts based on**	**Separation**	**Complexity concepts**
[126]	Discrete logarithm	CC/QQ	$\notin \mathsf{BPP}$
[173]	Discrete cube root	CC/QC	$\in (\mathsf{P/poly}) \setminus \mathsf{BPP}$
Section 6.1.3	Modular exponentiation	CC/QC	$\in \mathsf{P}$
Section 6.1.4	Discrete cube root	$\mathsf{C}_{\mathcal{H}}/\mathsf{Q}_{\mathcal{H}}$	$\in \mathsf{P}$
Section 6.2.1	Quantum process	CC/QQ	$\notin \mathsf{HeurP/poly}$ but $\in \mathsf{BQP}$

Table 2.1: The learning separations discussed in Sections 6.1 and 6.2.

Chapter 3

Towards quantum advantage via topological data analysis

In this chapter we discuss the advantages that the quantum algorithm for Betti number estimation can achieve over classical algorithms. Firstly, in Section 3.1, we formally define the computational problems that the quantum algorithm for Betti number estimation can (efficiently) solve. In particular, it is clear that the techniques used in the quantum algorithm for Betti number estimation can also be used to estimate the number of small eigenvalues of arbitrary sparse Hermitian matrix, not just of combinatorial Laplacians. We take this as the starting point to define our natural generalization, which is called *low-lying spectral density* estimation (a version of which was also studied by Brandão [40]). Next, in Section 3.2, we show that this generalization is DQC1-hard, which suggests that the quantum-algorithmic methods behind the quantum algorithm for Betti number estimation may be a source of exponential separation between quantum and classical computers. We also discuss how to potentially close the gap between the topological data analysis problem of Betti number estimation and its generalization, which would show that the topological data analysis problem is itself classically intractable. Setting aside the complexity theory, in Section 2.2.3 we discuss the state-of-the-art classical algorithms for Betti number estimation and compare them with the quantum algorithms for Betti number estimation. We also discuss promising approaches for developing novel more efficient classical algorithms that take into account the specifics of the combinatorial Laplacian and we clearly delineate the theoretical hurdles that, at least currently, stymie such classical approaches. After discussing the strengths and weaknesses of the classical algorithms, we identify graphs for which the quantum algorithm can achieve (superpolynomial) speedups over the best known classical algorithms in Section 3.2.4.

3.1 Problem definitions

In this section we formally define the computational problems whose hardness we will study. We begin by defining the problems that capture the key steps of the quantum

algorithm for Betti number estimation. Afterwards, we define the problems related to topological data analysis that the quantum algorithm for Betti number estimation aims to solve. We end this section by discussing the precise relationships between these problems.

The input matrices that we consider are sparse positive semidefinite matrices. We call a $2^n \times 2^n$ positive semidefinite matrix *sparse* if at most $\mathcal{O}(\mathrm{poly}(n))$ entries in each row are nonzero. A special class of sparse positive semidefinite matrices that we consider is the class of *log-local* Hamiltonians, i.e., n-qubit Hamiltonians that can be written as a sum

$$H = \sum_{j=1}^{m} H_j, \tag{3.1}$$

where each H_j acts on at most $\mathcal{O}(\log n)$ qubits and we assume that $m \in \mathcal{O}(\mathrm{poly}(n))$.

Our problems take as input a specification of a sparse positive semidefinite matrix, and we consider the following two standard cases. First, we consider the case where the input matrix is specified in terms of *sparse access*. That is, the input matrix $H \in \mathbb{C}^{2^n \times 2^n}$ is specified by quantum circuits that let us query the values of its entries, and the locations of the nonzero entries. More precisely, we assume that we are given classical descriptions of $\mathcal{O}(\mathrm{poly}(n))$-sized quantum circuits that implement the oracles O_H and $O_{H,\mathrm{loc}}$, which map

$$O_H : |i,j\rangle\,|0\rangle \mapsto |i,j\rangle\,|H_{i,j}\rangle\,,$$
$$O_{H,\mathrm{loc}} : |j,\ell\rangle\,|0\rangle \mapsto |j,\ell\rangle\,|\nu(j,\ell)\rangle\,,$$

where $0 \leq i,j,\ell \leq 2^n - 1$, and $\nu(j,\ell) \in \{0,\ldots,2^n-1\}$ denotes the location of the ℓ-th nonzero entry of the j-th column of H. Secondly, for log-local Hamiltonians, we also consider specifying the input matrix H by its *local-terms* $\{H_j\}$ as in Eq. (3.1).

In order to define the problem of generating approximations of eigenvalues that are sampled uniformly at random, we fix a suitable notion of an approximation of a probability distribution. In particular, this notion needs to take into account that the algorithm may err on both the estimation of the eigenvalue, and on the probability with which it provides such an estimation. For this we use the following definition presented in [200]. Let p be some probability distribution over the eigenvalues of a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$. That is, sampling according to p will output an eigenvalue λ_k with probability $p(\lambda_k)$, and $\sum_{k=0}^{2^n-1} p(\lambda_k) = 1$. In this context, a probability distribution q with finite support $Y_q \subset \mathbb{R}$ is said to be an (δ,μ)-*approximation* of p if it satisfies

$$\sum_{y \in Y_q \,:\, |y-\lambda_k|<\delta} q(y) \geq (1-\mu)p(\lambda_k), \quad \forall k \in \{0,\ldots,2^{n-1}\}.$$

Intuitively, this means that if we draw a sample according to q, then this sample will be δ-close to an eigenvalue λ_k with probability at least $(1-\mu)p(\lambda_k)$[1] Using this

[1]This definition captures the distribution generated by quantum phase estimation: the eigenvector is chosen according to the distribution p specified by the input state, and the output is δ-close to

definition, we define the problem of generating approximations of eigenvalues that are sampled uniformly at random from the set of all eigenvalues as follows.

Sparse uniform eigenvalue sampling (SUES)[2]
Input:
1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq \text{poly}(n)$.
2) An estimation precision $\delta \in \Omega\left(1/\text{poly}(n)\right)$.
3) An error probability $\mu \in \Omega\left(1/\text{poly}(n)\right)$.

Output: A sample drawn according to a (δ, μ)-approximation of the uniform distribution over the eigenvalues of H.

In the quantum algorithm for Betti number estimation, samples from SUES are used to estimate the number of eigenvalues of the combinatorial Laplacian that are close to zero. Clearly, this same idea can be used to estimate the number of eigenvalues that lie in some given interval for arbitrary sparse positive semidefinite matrices. This is called the *eigenvalue count* [40], which for a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$ and eigenvalue thresholds $a, b \in \mathbb{R}_{\geq 0}$ is given by

$$N_H(a, b) = \frac{1}{2^n} \sum_{k \,:\, a \leq \lambda_k \leq b} 1,$$

where $\lambda_0 \leq \cdots \leq \lambda_{2^n - 1}$ denote the eigenvalues of H. For a threshold $b \in \Omega\left(1/\text{poly}(n)\right)$, we shall refer to the quantity $N_H(0, b)$ as *low-lying spectral density*. This precisely captures our notion of the number of eigenvalues close to zero as discussed before. We define the problem of estimating the low-lying spectral density as follows.

Low-lying spectral density estimation (LLSD)[3]
Input:
1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq \text{poly}(n)$.
2) A threshold $b \in \Omega\left(1/\text{poly}(n)\right)$.
3) Precision parameters $\delta, \epsilon \in \Omega\left(1/\text{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0, 1]$ that, with probability at least μ, satisfies

$$N_H\left(0, b\right) - \epsilon \leq \chi \leq N_H(0, b + \delta) + \epsilon.$$

To provide some intuition behind this definition, note that it is supposed to precisely capture the problem that is solved by repeatedly sampling from SUES and computing the frequency of the eigenvalues that lie below the given threshold. We therefore require the precision parameter δ due to the imprecisions in the quantum phase estimation algorithm. Moreover, the precision parameter ϵ is necessary due to the sampling error we incur by estimating a probability by a relative frequency.

Now that we have formally defined the problems that capture the key steps of the

the corresponding eigenvalue with probability at least $(1 - \mu)$.

[2] In view of noisy quantum computers, it is interesting to consider distributions that are close to these (δ, μ)-approximation in total variation distance. Sampling such distributions can be less demanding, however, the precise hardness remains to be analyzed.

[3] The exact version of this problem is closely related to #P [45].

quantum algorithm for Betti number estimation, we define the problems related to topological data analysis that they allow us to solve. For these problems we consider the adjacency matrix of the graph to be the input, as this is usually the input to the quantum algorithm for Betti number estimation. We define the problem of estimating Betti numbers as follows.

Betti number estimation (BNE)[4]
Input:
1) The adjacency matrix of a graph $G = ([n], E)$.
2) An integer $0 \leq k \leq n-1$.
3) A precision parameter $\epsilon \in \Omega\left(1/\text{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$\left| \chi - \frac{\beta_k^G}{\dim \mathcal{H}_k^G} \right| \leq \epsilon.$$

As discussed in Section 2.2.2, the quantum algorithm for Betti number estimation does not precisely solve the above problem. Namely, due to the lack of knowledge regarding lower bounds on the smallest nonzero eigenvalue of the combinatorial Laplacian, we are not always able to estimate the number of eigenvalues that are exactly equal to zero. Nonetheless, the quantum algorithm for Betti number estimation is still able to estimate the number of eigenvalues of the combinatorial Laplacian that are close to zero, which we called approximate Betti numbers. We define the problem of estimating approximate Betti numbers as follows.

Approximate Betti number estimation (ABNE)
Input:
1) The adjacency matrix of a graph $G = ([n], E)$.
2) An integer $0 \leq k \leq n-1$.
3) Precision parameters $\delta, \epsilon \in \Omega\left(1/\text{poly}(n)\right)$.
4) A success probability $\mu > 1/2$.

Output: An estimate $\chi \in [0,1]$ that, with probability at least μ, satisfies

$$\frac{\beta_k^G}{\dim \mathcal{H}_k^G} - \epsilon \leq \chi \leq N_{\Delta_k^G}(0, \delta) + \epsilon.$$

We are now set to outline the problem that the quantum algorithm for Betti number estimation can efficiently solve. As discussed in Section 2.2.2, the quantum algorithm for Betti number estimation can efficiently solve ABNE, but only in certain regimes. In particular, one has to be able to efficiently prepare the maximally mixed state over all cliques of a given size from the adjacency matrix of the graph. As mentioned in Section 2.2.2, the efficiency of this state preparation depends on the graph's clique-density (i.e., probability that a uniformly random subset of vertices is a clique), or the graph's arboricity (which up to a factor 1/2 is equivalent to the

[4]The exact version of this problem is NP-hard [14]

maximum average degree of a subgraph). In short, the problem that the quantum algorithm for Betti number estimation can efficiently solve is a restriction of ABNE, where one is promised that the input graph is such that one can efficiently prepare the maximally mixed state over all cliques of a given size from the adjacency matrix (e.g., if the graph is sufficiently clique-dense or if it has a sufficiently bounded arboricity). We discuss this in more detail in Section 3.2.4, where we outline sufficient conditions on the graph's clique-density or arboricity that allow the quantum algorithm to efficiently solve ABNE.

Next, we will study the complexity of LLSD as it is a generalization of the problem that the quantum algorithm for Betti number estimation efficiently solves. Namely, as we will show in the following section, we can use LLSD to directly solve the problem that the quantum algorithm for Betti number estimation efficiently solves. Note that the input to the quantum algorithm for Betti number estimation is the adjacency matrix, and not the combinatorial Laplacian. Therefore, in order to use LLSD to solve the problem that the quantum algorithm for Betti number estimation efficiently solves, one first has to construct the appropriate input to LLSD. As it is computationally too expensive to enumerate all cliques in your graph, we cannot take the straightforward approach of first computing the combinatorial Laplacian to construct the desired input to LLSD. Fortunately, we can still use LLSD to efficiently solve the problem that the quantum algorithm for Betti number estimation efficiently solves by simulating sparse access to a matrix that is obtained by padding the combinatorial Laplacian with all-zeros columns and rows (see Section 3.1.1 for more details).

3.1.1 Relationships between the problems

In the previous section we have formally defined the computational problems whose complexity we will study. In this section we examine the reductions between LLSD and the problems related to topological data analysis in order to elucidate the precise relationships. An overview of the reductions can be found in Figure 3.1.

First, we discuss the relationship between LLSD and ABNE. It is clear that LLSD with a combinatorial Laplacian as input produces a solution to the corresponding instance of ABNE. It is also clear that LLSD can be used to solve ABNE if given the input of ABNE (i.e, the adjacency matrix), we can efficiently implement sparse access to a matrix such that an estimate of its low-lying spectral density allows us to recover an estimate of the low-lying spectral density of the combinatorial Laplacian. Interestingly, it turns out that we can do so if the input graph is clique-dense (i.e., in precisely the regime that is efficiently solvable by the quantum algorithm for Betti number estimation). Namely, we can efficiently implement sparse access to the $\binom{n}{k+1} \times \binom{n}{k+1}$-sized matrix Γ_k^G whose columns and rows are indexed by $(k+1)$-subsets of vertices, and whose entries are given by

$$\left(\Gamma_k^G\right)_{i,j} = \begin{cases} (\Delta_k^G)_{i,j} & \text{if } i \text{ and } j \text{ are } (k+1)\text{-cliques,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

In other words, the entries of the columns and rows that correspond to $(k+1)$-cliques are equal to the corresponding entries of the combinatorial Laplacian, and all other

entries are equal to zero. After subtracting the extra nullity caused by adding the $\binom{n}{k+1} - \chi_k$ all-zeros columns and rows, and renormalizing the eigenvalue count by a factor $\binom{n}{k+1}/\chi_k$, the low-lying spectral density of this Γ_k^G is equal to the low-lying spectral density of the combinatorial Laplacian. In equation form, we have that

$$N_{\Delta_k^G}(0,b) = \frac{\binom{n}{k+1}}{\chi_k} N_{\Gamma_k^G}(0,b) - \frac{\binom{n}{k+1} - \chi_k}{\chi_k}. \tag{3.3}$$

From Eq. (3.3), it is clear that an estimate of $N_{\Gamma_k^G}(0,b)$ up to additive inverse polynomial precision allows us to obtain an estimate of $N_{\Delta_k^G}(0,b)$ up to additive inverse polynomial precision, assuming indeed that the graph is clique-dense – i.e., that $\chi_k/\binom{n}{k+1} \in \Omega\left(1/\mathrm{poly}(n)\right)$. Note that this also requires us to have an estimate of $\chi_k/\binom{n}{k+1}$. Since the graph is clique-dense, it suffices to estimate $\chi_k/\binom{n}{k+1}$ up to additive inverse polynomial precision. An estimate of $\chi_k/\binom{n}{k+1}$ up to additive error ϵ can be obtained by drawing $\mathcal{O}(\epsilon^{-2})$ many k-subsets of vertices uniformly at random, and computing the fraction of these subsets that constitute an actual k-clique.

We emphasize that the above reduction works in precisely the regime where the quantum algorithm for Betti number estimation can efficiently solve ABNE. In other words, LLSD can be used to directly solve the problem that the quantum algorithm for Betti number estimation can efficiently solve. In this regard, LLSD is indeed a generalization of the problem that the quantum algorithm for Betti number estimation can efficiently solve.

Finally, let us discuss the reductions between ABNE and BNE. It is clear that BNE is reducible to ABNE if the size of the smallest nonzero eigenvalue of the combinatorial Laplacian is at least inverse polynomial in n. The reverse direction is unclear, as for BNE the threshold on the eigenvalues is fixed to be exactly zero. A possible approach would be to first project the eigenvalues that lie below the given threshold to zero and then count the zero eigenvalues. However, using techniques inspired by ideas from [85, 121], we have only been able to project these eigenvalues close to zero, as opposed to exactly equal to zero, and we are not aware of any way to circumvent this.

3.2 Classical intractability of LLSD

To show that quantum computers have an advantage over classical computers in topological data analysis, one would have to show that Betti number estimation requires exponential time on a classical computer. In this section we study the classical hardness of the problem efficiently solved by the quantum algorithm for Betti number estimation. In particular, we show that the natural generalization of this problem (which we called low-lying spectral density estimation) is classically intractable under widely-believed complexity-theoretic assumptions by showing that it is hard for the one clean qubit model of computation. Afterwards, we discuss how to potentially close the gap between the classical intractability of low-lying spectral density and (approximate) Betti number estimation in order to show that the topological data analysis problem is itself classically intractable.

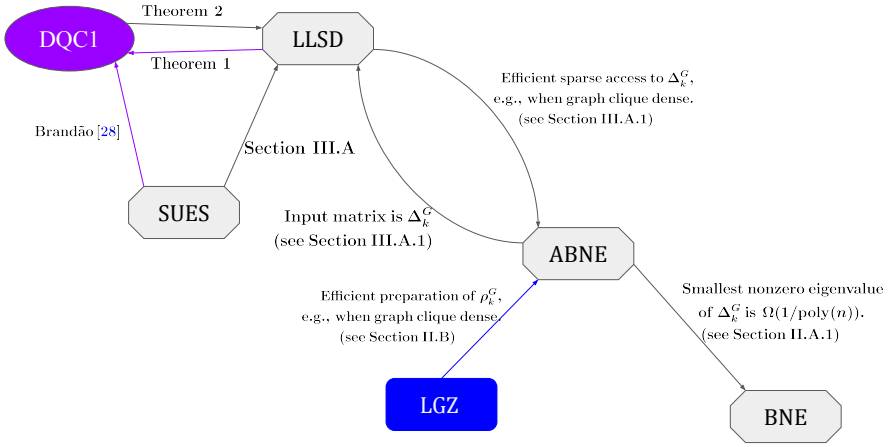


Figure 3.1: Overview of the relations between the problems (octagons), algorithm (rectangle) and complexity class (ellipse) studied in this thesis. A $\xrightarrow{\text{C}}$ B stand for: "A can efficiently solve B if condition C holds". The algorithm studied is that by Lloyd, Garnerone and Zanardi (LGZ) as described in Figure 2.5. The problems are sparse uniform eigenvalue sampling (SUES), low-lying spectral density estimation (LLSD), approximate Betti number estimation (ABNE), and Betti number estimation (BNE) as defined in Section 3.1 The class DQC1 is defined in Section 3.2.1.

3.2.1 The one clean qubit model of computation

In the next section we will show that the complexity of the problems defined in Section 3.1 are closely related to the one clean qubit model of quantum computation [122]. In this model we are given a quantum register that is initialized in a state consisting of a single 'clean' qubit in the state $|0\rangle$, and $n-1$ qubits in the maximally mixed state. We can then apply any polynomially-sized quantum circuit to this register, and measure only the first qubit in the computational basis. Following [122], we will refer to the complexity class of problems that can be solved in polynomial time using this model of computation as DQC1 – "deterministic quantum computation with a single clean qubit".

We will refer to a problem as DQC1-hard if any problem in DQC1 can be reduced to it under polynomial time truth-table reductions. That is, a problem L is DQC1-hard if we can solve any problem in DQC1 using polynomially many nonadaptive queries to an oracle for L, together with polynomial time preprocessing of the inputs and postprocessing of the outcomes. Technically, instead of containing the problem of estimating a given quantity up to additive inverse polynomial precision, DQC1 contains the decision problem of deciding whether this quantity is greater than $1/2+\sigma$ or less than $1/2-\sigma$, where σ is some inverse polynomial gap. However, as the estimation versions of these problems are straightforwardly reduced to their decision version using binary search, we will bypass this point from now on and only consider the problems of estimating a given quantity up to inverse polynomial precision [178].

It is widely believed that the one clean qubit model of computation is more powerful than classical computation. For instance, estimating quantities that are supposedly hard to estimate classically, such as the normalized trace of a unitary matrix corresponding to a polynomial-depth quantum circuit and the evaluation of a Jones polynomial at a root of unity, turn out to be complete problems for DQC1 [178]. Moreover, it has been shown that classical computers cannot efficiently sample from the output distribution of the one clean qubit model up to constant total variation distance error, provided that some complexity theoretic conjectures hold [141, 142].

3.2.2 Hardness of LLSD for the one clean qubit model

Recall that in order to show that quantum computers have an advantage over classical computers in topological data analysis, one would have to show that the problem that the quantum algorithm for Betti number estimation can efficiently solve is hard for classical computers. In Section 3.1, we pointed out that the problem that the quantum algorithm for Betti number estimation can efficiently solve is a restriction of ABNE to clique-dense graphs (i.e., graphs which satisfy Eq. (2.31)). Moreover, we showed in Section 3.1.1 that LLSD is a generalization of this version of ABNE. This motivates us to study the classical hardness of LLSD. In this section we present our results, which show that the complexity of LLSD is intimately related to the one clean qubit model.

Our first and main result is that LLSD is hard for the class DQC1[5], even when the input is restricted to log-local Hamiltonians. As the one clean qubit model of

[5]Throughout this section we mean DQC1-hardness with respect to *Turing reductions*. We believe that our approach could be modified to a Karp reduction, but since this reduction is not vital for our claim, we leave this question open for future work.

computation is widely believed to be more powerful than classical computation, this shows that LLSD is likely hard for classical computers. We discuss the implications of this result on the classical hardness of the problem that the quantum algorithm for Betti number estimation can efficiently solve in Section 3.2.3.

Theorem 5. LLSD *is* DQC1*-hard. Moreover,* LLSD *with the input restricted to log-local Hamiltonians remains* DQC1*-hard.*

We now give a sketch of our proof of the above theorem, the complete proof can be found in Appendix A.1. The main idea behind the proof is to show that we can use LLSD to estimate a quantity similar to a normalized subtrace – or more precisely, a normalized sum of eigenvalues below a given threshold – which has been shown to be DQC1-hard by Brandão [40]. We estimate this normalized subtrace by constructing a histogram approximation of the low-lying eigenvalues, and afterwards computing the mean of this histogram. To construct this histogram, we use LLSD to estimate the number of eigenvalues that lie in each bin. To avoid double counting of eigenvalues due to imprecisions around the thresholds of the bins, we subtract the output of LLSD with the eigenvalue threshold set to the lower-threshold of the bin from the output of LLSD with the eigenvalue threshold set to the upper-threshold of the bin. By doing so, we obtain an estimate of the number of eigenvalues within the bin, and misplace eigenvalues by at most one bin.

Our second result shows that the complexity of LLSD is more closely related to DQC1 than just hardness. Namely, we point out that if the input to LLSD is restricted to log-local Hamiltonians (or more generally, any type of Hamiltonian that allows for efficient Hamiltonian simulation using $\mathcal{O}(\log(n))$ ancilla qubits), then it can be solved using the one-clean qubit model. From this it follows that LLSD is DQC1-complete if the input is restricted to log-local Hamiltonians. The main idea behind why we can solve these instances of LLSD using the one clean qubit model is that the one-clean qubit model can simulate having access to up to $\mathcal{O}(\log(n))$ pure qubits [178]. These pure qubits allow for Hamiltonian simulation techniques based on the Trotter-Suzuki formula [129] and for quantum phase estimation up to the required precision. We summarize this in the following theorem, the proof of which can be found in Appendix A.2.

Theorem 6. LLSD *with the input restricted to log-local Hamiltonians is* DQC1*-complete.*

As an added result, we find that the complexity of SUES with the input restricted to log-local Hamiltonians is also closely related to DQC1. The complexity of this instance SUES was stated as an open problem by Wocjan and Zhang [200]. Moreover, we believe that it is interesting to study the complexity of SUES, as this problem can potentially find practical applications beyond both LLSD and Betti number estimation. We remark that SUES with the input restricted to log-local Hamiltonians was already shown to be DQC1-hard by Brandão [40]. Here we point out that the complexity of this instance of SUES is more closely related to the one clean qubit model than just hardness, as it can also be solved using $\mathsf{DQC1}_{\log n}$ circuits, that is, DQC1 circuits where we are allowed to measure logarithmically many of the qubits in the computational

basis at the end (to read out the encoding of the eigenvalue). The proof of the following proposition can be found in Appendix A.2.

Proposition 7. SUES *with the input restricted to log-local Hamiltonians can be solved in polynomial time by the one clean qubit model with logarithmically many qubits measured at the end.*

3.2.3 Closing the gap for classical intractability of ABNE

The results discussed in the previous section are not sufficient to conclude that ABNE and BNE are hard for classical computers, because for these problems the family of input matrices is restricted to combinatorial Laplacians. Nonetheless, because LLSD is a generalization of the problem that the quantum algorithm for Betti number estimation can efficiently solve, our result shows that – aside from the matter regarding the restriction to combinatorial Laplacians – the quantum algorithm for Betti number estimation solves a classically intractable problem which in some cases captures interesting information concerning an underlying graph. Moreover, our result eliminates the possibility of certain routes for dequantization, namely those that are oblivious to the particular structure of the input matrix, which in particular eliminates the approaches of Tang et al. [60].

The open question regarding the classical hardness of ABNE and the problem that the quantum algorithm for Betti number estimation can efficiently solve is whether LLSD remains classically hard when restricted to combinatorial Laplacians of arbitrary or clique-dense graphs, respectively. Even though these restrictions on the input seem quite stringent, note that our result shows that LLSD is already DQC1-hard for the restricted family of log-local Hamiltonians obtained from Kitaev's circuit-to-Hamiltonian construction[6]. Moreover, there exists a family of combinatorial Laplacians that can encode DQC1-hard Hamiltonians, but not all of those are combinatorial Laplacians of clique complexes [48]. One way we tried to close this gap was by investigating whether we could encode Hamiltonians obtained from Kitaev's circuit-to-Hamiltonian construction into combinatorial Laplacians of sufficiently large graphs. While indeed various matrices related to quantum gates can be found as submatrices of combinatorial Laplacians, we did not succeed in finding an explicit embedding. In our view, this remains a promising way of showing that LLSD remains classically hard when restricted to combinatorial Laplacians (if indeed this claim is true at all).

Besides the above approach based on the Kitaev circuit-to-Hamiltonian construction, there are many other constructions that could potentially be used to show that LLSD remains classically hard when restricted to combinatorial Laplacians (again, if indeed this claim is true at all). In particular, there are several constructions used to prove QMA-hardness of the ground-state energy problem for certain families of Hamiltonians (i.e., deciding if the smallest eigenvalue lies above or below some thresholds)[7]. All of these constructions typically take as input a (verification) circuit and produce a Hamiltonian that has a small eigenvalue if and only if there exists a quantum state (also called a witness) that makes the circuit accept (i.e., if on this input it is more

[6] In [51, 40] and our case it is unclear whether this holds for k-local Hamiltonians with constant k, as the standard constructions of these local Hamiltonians involve a clock register that is too large.

[7] For an overview of circuit-to-Hamiltonian constructions see [39].

likely to output 1 on the first qubit). A special property of the Kitaev construction is that for every input to the circuit, there exists a state whose energy with respect to the corresponding Hamiltonian is close to the acceptance probability of the circuit (i.e., not just that there exists small eigenvalue if and only if there exists a state that makes the circuit accept). This property allowed Brandão to prove that normalized sub-trace estimation for these Hamiltonians is DQC1-hard [40], which is at the core of our proof of DQC1-hardness of LLSD. Hence, a promising approach to show DQC1-hardness of LLSD for a family of Hamiltonians is to look at existing circuit-to-Hamiltonian constructions used to prove QMA-hardness of versions of the ground-state energy problem and investigate whether they also have this special property that the Kitaev construction has (or to see if they can be equipped with it). This is particularly interesting for the constructions used to show QMA-hardness of the Bose-Hubbard model [61], or the Fermi-Hubbard model [149]. The reason for this is that both of these Hamiltonians exhibit similarities to the Hamiltonian of the hardcore fermion model, which is equal to the combinatorial Laplacian of a clique complex [48]. Specifically, the Hamiltonian H_G of the fermion hardcore model on a graph $G = ([n], E)$ is given by

$$H_G = \sum_{(i,j)\in E} P_i a_i a_j^\dagger P_j + \sum_{i\in V} P_i, \tag{3.4}$$

where $P_i = \prod_{(i,j)\in E}(I - n_j)$, a_i denotes the fermionic annihilation operator, and n_j denotes the fermionic number operator [48]. For this Hamiltonian H_G it holds that

$$H_G = \bigoplus_{k=0}^{n-1} \Delta_k^{\bar{G}}, \tag{3.5}$$

where $\bar{G}$ denotes the complement graph of G, and Δ_k^G denotes the k-th combinatorial Laplacian. Continuing along these lines, the authors of [67] established a circuit-to-Hamiltonian mapping onto combinatorial Laplacians that allowed them to show that deciding whether a Betti number is zero or not is QMA_1[8]-hard (though it has not yet lead to DQC1-hardness of ABNE).

Finally, instead of trying to show that the family of combinatorial Laplacians is sufficiently rich, we could also generalize this family of matrices while still remaining relevant to topological data analysis. For example, one could consider generalizations of combinatorial Laplacians, such as weighted combinatorial Laplacians [105] or persistent combinatorial Laplacians [195], and show that these generalized families are sufficiently rich as to contain DQC1-hard instances. Besides all the approaches discussed above, other routes such as proving classical hardness of LLSD when restricted to other sets of matrices such as $\{0, \pm 1\}$-matrices, or by going through the discrete structures related to Tutte and Jones polynomials [16, 178] could all be possible as well.

The open questions regarding the classical hardness of BNE are the same as those regarding the classical hardness of ABNE, except that there is one additional open

[8] QMA_1 is the one-sided error version of QMA.

question. Namely, assuming that ABNE is classically hard, the remaining open question regarding the classical hardness of BNE is whether estimating the number of eigenvalues exactly equal to zero is at least as hard as estimating the number of eigenvalues below a given inverse polynomially small threshold. This question was already addressed in Section 3.1.1 when we examined the reductions between ABNE and BNE. As discussed there, one approach would be to project the eigenvalues below the given threshold to zero, and afterwards count only the zero eigenvalues.

Regardless, even if LLSD does not remain classically hard when restricted to combinatorial Laplacians, we can envision practical generalizations of the quantum-algorithmic methods used by the algorithm for Betti number estimation that go beyond Betti numbers, as we will discuss in more detail in Section 3.3. Specifically, in Section 3.3 we provide efficient quantum algorithms for two concrete examples of such practical generalizations, together with complexity-theoretic evidence of their classical hardness. The first example we discuss is numerical rank estimation, an important problem in machine learning, data analysis and many other applications. The second example is spectral entropy estimation, which can be used as a tool in complex network analysis.

3.2.4 Graphs with quantum speedup

In Section 2.2.2, we outlined criteria that the graph has to satisfy in order for the quantum algorithm to be able to efficiently estimate (approximate) Betti numbers. Specifically, the graph has to be such that one can efficiently prepare the input state in Eq. (2.30), e.g., by sampling uniformly at random from cliques of a given size. Afterwards, in Section 2.2.3, we discussed the best known classical algorithms and we outlined the regimes in which they require superpolynomial runtimes. In this section we put these two considerations together and we concretely characterize families of graphs for which the quantum algorithm achieves either a high-degree polynomial, or even a superpolynomial speedup over the best known classical algorithm. In particular, we identify families of graphs for which the quantum algorithm is efficient and for which the best known classical algorithms are unable to achieve competitive runtimes.

As discussed in Section 2.2.2, one way to efficiently prepare the input state is to use Grover's algorithm or rejection sampling to sample uniformly at random from cliques of a given size. Recall that for this to be efficient the graph has to be clique dense, i.e., it has to satisfy Eq. (2.31). To identify a family a clique-dense graphs, let us consider clique sizes $k \geq 3$, let $\gamma > \frac{k-2}{2(k-1)}$ be a constant, and consider any graph on n vertices with at least γn^2 edges. Suppose we want to estimate the k-th approximate Betti number of this graph, where k and the precision parameters are constant. The quantum algorithm for Betti number estimation can do so in time

$$\widetilde{\mathcal{O}}\left(\sqrt{n^{k+1}/\chi_k} + n^3\right),$$

where χ_k denotes the number of $(k+1)$-cliques. Having chosen the graph the way we

did, the clique density theorem [162] now directly guarantees that our graph satisfies

$$\chi_k \in \Omega(n^{k+1}),$$

which is a phenomenon known as "supersaturation". In particular, this implies that our graph is clique-dense and that the quantum algorithm for Betti number estimation estimates the required approximate Betti number in time

$$\widetilde{\mathcal{O}}\left(n^3\right).$$

Moreover, as discussed in Section 2.2.3, the best known classical algorithm requires time

$$\mathcal{O}\left(n^{k+1}\right),$$

as the number of nonzero entries of the corresponding combinatorial Laplacian is at least χ_k. We conclude that in these instances the quantum algorithm for Betti number estimation achieves a $(k-2)$-degree polynomial speedup over the best known classical methods, which for large enough k might allow for runtime advantages on prospective fault-tolerant computers, even when all overheads are accounted for [27].

We can push the above separation between the best known classical algorithm and the quantum algorithm even further. Consider the same setting as above, but with $\gamma = \frac{k-1}{k}$ and we allow k to scale with n. Using a result of Moon and Moser [140, 132, 191], we can derive that in this setting the graph satisfies

$$\binom{n}{k+1}/\chi_k \in \mathcal{O}\left(k^k\right).$$

Therefore, the quantum algorithm can estimate the k-th approximate Betti number in time

$$\mathcal{O}\left(k^{2+k/2} + n^3\right).$$

On the other hand, the best known classical algorithm runs in time

$$\mathcal{O}\left(n^{k+1}/k^{2k}\right),$$

as the number of nonzero entries of the corresponding combinatorial Laplacian is at least $\chi_k \geq n^{k+1}/k^{2k}$. In particular, if we let k scale with n in an appropriate way, then the quantum algorithm achieves a superpolynomial speedup over the best known classical method. For example, if we let the clique size scale as $k \sim \log n$, then the quantum algorithm runs in time

$$2^{\mathcal{O}(\log n \log\log n)},$$

whereas the best known classical algorithm runs in time

$$2^{\mathcal{O}\left((\log n)^2\right)},$$

giving rise to a superpolynomial quantum speedup. Note that the graphs in the previous two settings are rather edge-dense (which occurs in topological data analysis

if the grouping-scale ϵ approaches the maximum distance between two datapoints), and it is unknown whether better classical algorithms are possible in this regime.

Next, we construct a family of graphs where (i) the Betti numbers are large, (ii) the clique-density is high, and (iii) the spectral gaps the combinatorial Laplacian are sufficiently large. Due to properties (i)-(iii) this family of graphs provides a great example of a family of graphs on which the quantum algorithm outperforms its classical counterpart. Let $K(m,k)$ be the k-partite complete graph, where each partition contains m vertices. That is, $K(m,k)$ consists of k clusters, each with m vertices; there are no edges within clusters, but all edges between clusters are included. Note $K(m,1)$ is a collection of m points with no edges. $K(m,k)$ gives a useful example of a clique complex with a high Betti number [13]. It also has a Laplacian with a large spectral gap.

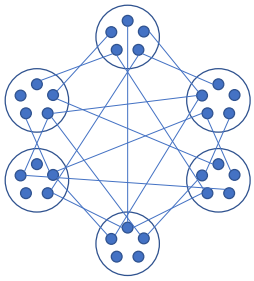


Figure 3.2: The graph $K(5,6)$.

Proposition 8. *The $(k-1)^{th}$ Betti number of the clique complex of $K(m,k)$ is*

$$\beta_{k-1} = (m-1)^k. \tag{3.6}$$

Proposition 9. *The combinatorial Laplacian $\Delta_{k-1}^G = (\partial_{k-1}^G)^\dagger \partial_{k-1}^G + \partial_k^G (\partial_k^G)^\dagger$ of the clique complex of $K(m,k)$ has spectral gap*

$$\lambda_{\min} = m. \tag{3.7}$$

We prove these in A.3 using techniques from simplicial homology. A further useful fact is that

$$|\mathrm{Cl}_k(K(m,k))| = m^k. \tag{3.8}$$

Standard classical approaches need to at least store a vector of this length, so we can give a classical complexity T_c of estimating normalized Betti numbers

$$T_c \sim e^{k \ln m}. \tag{3.9}$$

As a first approximation for the quantum cost T_q, we use the formula

$$T(G,k,r,\delta) = 3\pi|E|\frac{\ln(1/\delta)}{r}\sqrt{\frac{\binom{n}{k}}{\beta_{k-1}^G}}. \tag{3.10}$$

and consider just the square root factor and $|E|$. Stirling's approximation gives $\binom{n}{k} \sim \left(\frac{m^{1+1/m}}{m-1}\right)^n$, and Proposition 8 gives $\beta_{k-1} = (m-1)^{n/m}$, giving a quantum complexity scaling as

$$T_q \sim |E|\left(\frac{m^{1+1/m}}{(m-1)^{1+1/m}}\right)^{n/2} \sim n^2 e^{(k/2)(1+1/m)}. \tag{3.11}$$

Therefore, for constant m, there is a polynomial speedup by a $2\ln m$ root (ignoring n^2 and the $1/m$ term). Alternatively, taking k constant, the above formulae give

$$T_c = \mathcal{O}(n^k), \tag{3.12}$$

$$T_q = \mathcal{O}(n^2). \tag{3.13}$$

Then there is a polynomial speedup by a $k/2$ root. To obtain a superpolynomial speedup, m can be taken to increase close to linear in n, but k can be taken to also increase with n. Close to the best result is obtained for $k = c\ln^2 n$ with some constant c. Then the logs of the complexities are approximately

$$\ln T_c \sim c\ln^3 n, \tag{3.14}$$

$$\ln T_q \sim 2\ln n + (c/2)\ln^2 n. \tag{3.15}$$

That implies a speedup by a $2\ln n$ root, which is superpolynomial.

This is still not an exponential speedup, but as far as the graph is concerned this is the best speedup that could be obtained from this type of approach. This is because, with k constant, the quantum complexity ignoring the $|E|$ factor is $\mathcal{O}(1)$. The Betti number is already scaling the same as $\binom{n}{k}$, but the overhead from $|E|$ means that the speedup is not exponential.

As also discussed in Section 2.2.2, besides clique-density another important graph parameter that dictates the runtimes of specialized algorithms for uniform clique sampling is the so-called *arboricity*. The arboricity of a graph is equivalent (up to a factor 1/2) to the maximum average degree of a subgraph. For a graph with n vertices and arboricity α, near-optimal classical algorithms sample a k-clique uniformly at random in time [77]

$$\widetilde{\mathcal{O}}\left(k^k \cdot \max\left\{\left(\frac{(n\alpha)^{k/2}}{\chi_k}\right)^{\frac{1}{k-1}},\ \min\left\{n\alpha, \frac{n\alpha^{k-1}}{\chi_k}\right\}\right\}\right). \tag{3.16}$$

By also considering the algorithm of [77] (i.e., instead of rejection sampling or Grover's algorithm) we strictly expand the family of graphs for which the quantum algorithm achieves a superpolynomial speedup for ABNE. In particular, there exists a

family of graphs for which the algorithm of [77] is superpolynomially more efficient[9] than Grover's algorithm and rejection sampling for the problem of uniform clique sampling. An example of such a family is as follows: consider the n-vertex graphs consisting of n/r cliques of size r (for simplicity we assume that n is a multiple of r), where each r-clique is fully-connected with d other r-cliques (i.e., all edges between the $2r$ vertices are present). In other words, consider a d-regular graph on n/r vertices, and replace each vertex with an r-clique and fully-connect all r-cliques that were connected according to the d-regular graph we started with. Now if we set $d, r = \log n$ and $k = \log\log n$, then the number of k-cliques (and thus also the runtime of the best known classical algorithm for ABNE) scales like $\log(n)^{\log\log(n)}$. Moreover, the clique-density (and thus also the runtime of rejection sampling and Grover's algorithm) scales like $n^{\log\log(n)}$. Finally, the runtime of the algorithm of [77] scales like $\log\log(n)^{\log\log(n)}$. In conclusion, for these graphs the algorithm of [77] is superpolynomially more efficient than rejection sampling and Grover's algorithm for the problem of uniform clique sampling. Moreover, for these graphs the quantum algorithm for ABNE achieves a superpolynomial speedup over the best-known classical algorithm for ABNE, but only if one uses the algorithm of [77] (i.e., this speedup goes away if one uses rejection sampling or Grover's algorithm). We again remark that we are dealing with special types of graphs, and it is unknown whether better classical algorithms are possible in this regime.

3.3 Quantum speedups beyond Betti numbers

In the previous section we provided evidence that the computational problems tackled by the quantum algorithm for Betti number estimation are likely hard for classical computers. Even though we fell short of showing that the topological data analysis problem of estimating (approximate) Betti number is classically intractable, we did provide evidence that the quantum algorithmic methods that underlie the quantum algorithm for Betti number estimation could give rise to a potential source of practical quantum advantage. In this section we demonstrate this by discussing extensions of the quantum-algorithmic methods behind the algorithm for Betti number estimation that go beyond Betti numbers. In particular, we provide efficient quantum algorithms for numerical rank estimation (an important problem in machine learning and data analysis) and spectral entropy estimation (which can be used to compare complex networks), together with complexity-theoretic evidence of their classical hardness.

3.3.1 Numerical rank estimation

In this section we identify a practically important application of the problem of estimating the number of small eigenvalues (which we called LLSD). Specifically, we consider the problem of *numerical rank estimation*. The numerical rank of a matrix $H \in \mathbb{C}^{2^n \times 2^n}$ is the number of eigenvalues that lie above some given threshold b, i.e.,

[9] We say that a runtime $t_1(n)$ is *superpolynomially more efficient* than a runtime $t_2(n)$ if $\log t_2(n)/\log t_1(n) \to \infty$ when $n \to \infty$.

it is defined as

$$r_H(b) = \frac{1}{2^n} \sum_{k \,:\, \lambda_k > b} 1,$$

where $\lambda_1 \leq \cdots \leq \lambda_{2^n-1}$ denote the eigenvalues of H. By the rank-nullity theorem we have that

$$r_H(b) = 1 - N_H(0, b),$$

which shows that we can estimate the numerical rank using low-lying spectral density estimation and that the error scaling is the same.

Many machine learning and data analysis applications deal with high-dimensional matrices whose relevant information lies in a low-dimensional subspace. To be specific, it is a standard assumption that the input matrix is the result of adding small perturbations (e.g., noise in the data) to a low-rank matrix. This small perturbation turns the input matrix into a high-rank matrix, that can be well approximated by a low-rank matrix. Techniques such as principle component analysis [111] and randomized low-rank approximations [99] are able exploit this property of the input matrix. However, these techniques often require as input the dimension of this low-dimensional subspace, which is often unknown. This is where numerical rank estimation comes in, as it can estimate the dimension of the relevant subspace by estimating the number of eigenvalues that lie above the "noise-threshold". In addition, being able to determine whether the numerical rank of a matrix is large or small enables one to assert whether the above low-rank approximation techniques is applicable at all, or not.

From Theorem 5 it directly follows that quantum computers achieve an exponential speedup over classical computers for numerical rank estimation of matrices specified via sparse access (unless the one clean qubit model can be efficiently simulated on a classical computer). Still, it is also interesting to consider settings where the matrix is specified via a different input model. In the remainder of this section we study two examples of different input models. Firstly, motivated by a more practical perspective we consider a seemingly weaker input model that is more closely related to the input models that appear in classical data analysis settings. Secondly, we consider a likely stronger input model that appears throughout quantum machine learning literature, which is more informative from a complexity-theoretic perspective.

In typical (classical) applications, matrices are generally not specified via sparse access. Here we consider an input model that is more closely related to what is encountered in a typical classical setting. Specifically, we consider the case where a sparse matrix A of size $2^n \times 2^n$ is specified as a list of triples

$$\big\{(i_k, j_k, A_{i_k,j_k}) \mid A_{i_k,j_k} \neq 0\big\},$$

which is sorted lexicographically by column and then row. Storing matrices in this type of memory structure is very natural when dealing with matrices with a limited number of nonzero entries (which we denote by nnz). Now, for the quantum analogue we consider the same specification but we suppose that it is stored in a QRAM-type

memory, only additionally allowing us to query it in superposition as follows:

$$\sum_k \alpha_k |k\rangle |0\rangle \mapsto \sum_k \alpha_k |k\rangle |i_k, j_k, A_{i_k,j_k}\rangle .$$

Since the list is sorted, and since A is sparse, we can still simulate column-wise sparse access in $\mathcal{O}(\log \mathsf{nnz})$ queries, essentially by using binary search. Therefore, if A is Hermitian, then the quantum algorithm can estimate its numerical rank in time $\mathcal{O}(\mathrm{poly}(n, \log \mathsf{nnz}))$. On the other hand, the best known classical algorithms run in time $\mathcal{O}(\mathsf{nnz})$ [190, 59, 70, 124]. Consequently, the quantum algorithm achieves a speedup over the best known classical algorithm if nnz is at least a high-enough degree polynomial in n (and it achieves an exponential speedup if nnz is itself exponential). For the case where A is not Hermitian, recall that we also need sparse access to $A^\dagger$. For this issue we found no general method that can do so in time less than $\mathcal{O}(\mathsf{nnz})$, without assuming a high sparsity. However, the high sparsity then exactly offsets any potential quantum advantage in the full algorithm complexity.

Next, we consider a likely stronger input model which is widely-studied in the quantum machine learning literature. Specifically, we study the quantum-accessible data structure introduced in [118, 119], which can generate quantum states proportional to the columns of the input matrix, together with a quantum state whose amplitudes are proportional to the 2-norms of the columns. When the input matrix is provided in this quantum-accessible data structure, the quantum-algorithmic methods of [85, 55] can be used to estimate its numerical rank in time $\mathcal{O}(\mathrm{poly}(A_{\max}, n))$, where $A_{\max} = \max_{i,j} |A_{ij}|$.

The classical analogue of this quantum-accessible data structure is the sampling and query access model introduced in [186], which brought forth the "dequantization" methods discussed in [60]. At present it is not clear whether assuming sampling and query access allows us to efficiently estimate the numerical rank using dequantizations, or other methods. Here both possibilities are interesting. Firstly, if numerical rank estimation remains equally hard with sampling and query access, then it shows that quantum algorithms relying on the methods of LGZ have a chance of maintaining their exponential advantage in more general scenarios. Secondly, if an efficient classical algorithm for numerical rank estimation is possible with sampling and query access, then this leads to new insights regarding the hardness of the one clean qubit model. Recall that we have shown that estimating the numerical rank of matrices specified via sparse access is DQC1-hard (in the sense that, if a classical algorithm could do so efficiently given analogous access, then it can be used to efficiently solve all problems in DQC1). Now for the sparse matrix case, the only difference between sparse access and sampling and query access is that the latter allows one to sample from a distribution whose probabilities are proportional to the 2-norms of the columns. Indeed, the other part (i.e., sampling from distributions whose probabilities are proportional to the squared entries of the columns) is straightforward when the matrix is specified via sparse access. This implies that, if sampling and query access allows us to efficiently estimate the numerical rank of sparse matrices, then producing samples according to the 2-norms of the columns of a sparse matrix is DQC1-hard. This also holds for the log-local Hamiltonian setting, so it would also follow that sampling from a distribution

proportional to the 2-norms of the columns of log-local Hamiltonians is DQC1-hard. We summarize this observation in the proposition below.

Proposition 10. *Suppose there exists an efficient classical algorithm for numerical rank estimation (or, equivalently* LLSD*) for matrices provided by sampling and query access. Then, sampling from a distribution whose probabilities are proportional to the 2-norms of the columns of a sparse Hermitian matrix is* DQC1*-hard (with respect to Turing reductions).*

3.3.2 Combinatorial Laplacians beyond Betti numbers

In the previous section we discussed a practical application of the quantum-algorithmic methods behind the algorithm for Betti number estimation by using the same methods, but changing the family of input matrices (i.e., going beyond combinatorial Laplacians). In this section we take a different approach, namely we again consider the combinatorial Laplacians, but investigate applications beyond Betti number estimation (i.e., beyond estimating its nullity) relying on different algorithms than the one for low-lying spectral density estimation. Moreover, we will again find regimes where the same type of evidence of classical hardness can be provided, further motivating investigations into quantum algorithms that operate on the combinatorial Laplacians.

The eigenvalues and eigenvectors of the combinatorial Laplacian have many interesting graph-oriented applications beyond the applications in topological data analysis discussed in Section 2.2. The intuition behind this is that the combinatorial Laplacian can be viewed as a generalization of the standard graph Laplacian. For example, there exist generalizations of spectral clustering and label propagation (important techniques in machine learning that are used for dimensionality reduction and classification) which utilize the eigenvalues and eigenvectors of the combinatorial Laplacians [152]. Moreover, the eigenvalues of a normalized version of the combinatorial Laplacian convey information about the existence of circuits of cliques (i.e., ordered lists of adjacent cliques that cover the whole graph) and about the chromatic number [105]. Lastly, Kirchhoff's matrix tree theorem – which relates the eigenvalues of the standard graph Laplacian to the number of spanning trees – turns out to have a generalization to higher-order combinatorial Laplacians [75].

The specific problem that we study in this section is that of sampling from a distribution over the eigenvalues whose probabilities are proportional to the magnitude of the eigenvalues. In particular, we give a quantum algorithm that efficiently samples from an approximation of these distributions. Moreover, we show that sampling from these distributions for arbitrary sparse Hermitian matrices is again as hard as simulating the one clean qubit model, which shows that it is classically intractable (unless the one clean qubit model can be efficiently simulated on a classical computer). Finally, we discuss how this quantum algorithm can speed up spectral entropy estimation, which when applied to combinatorial Laplacians can be used to compare complex networks.

We define the problem that we study in this section as follows.

Sparse weighted eigenvalue sampling (SWES)
Input:

1) A sparse positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$, with $||H|| \leq 1$ and $\mathrm{tr}\{H\}/2^n \in \mathcal{O}(\mathrm{poly}(n))$.
2) An estimation precision $\delta \in \Omega(1/\mathrm{poly}(n))$.
3) A sampling error probability $\mu \in \Omega(1/\mathrm{poly}(n))$.

Output: A sample drawn from a (δ, μ)-approximation of the distribution $p(\lambda_j) = \lambda_j / \mathrm{tr}\{H\}$.

Using the subroutines of the quantum algorithm for Betti number estimation (i.e., Hamiltonian simulation and quantum phase estimation), we can efficiently sample from an approximation of the distribution of SWES defined above. In fact, we can efficiently implement *purified quantum query-access* to $p(\lambda_j)$ [84]. To be precise, we can implement an approximation of the unitary U_H (and its inverse) which acts as

$$U_H |0\rangle_A |0\rangle_B = |\psi_H\rangle = \sum_{j=0}^{2^n-1} \sqrt{p(\lambda_j)} |\psi_j\rangle_A |\phi_j\rangle_B , \tag{3.17}$$

such that $\mathrm{Tr}_B(|\psi_H\rangle\langle\psi_H|) = H/\mathrm{tr}\{H\}$. Purified quantum query-access has been shown to be more powerful than standard classical sampling access, as it can speedup the postprocessing of the samples when trying to find out properties of the underlying distribution [84].

We implement an approximation of the purified quantum-query access defined in Eq. (3.17) as follows:

1. Prepare the following input state by taking a maximally entangled state (which can always be expressed in the eigenbasis of H in one of its subsystems) and adding two ancillary registers

$$|\psi\rangle_{in} = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\psi_k\rangle |\phi_k\rangle \otimes |0^t\rangle \otimes |0\rangle_{flag} ,$$

where $\{|\psi_k\rangle\}_{k=0}^{2^n-1}$ are orthonormal eigenvectors of H and $\{|\phi_k\rangle\}_{k=0}^{2^n-1}$ is an orthonormal basis of $\mathbb{C}^{2^n}$.

2. Use Hamiltonian simulation on H, and apply quantum phase estimation of the realized unitary to the first register to prepare the state

$$\begin{aligned} &\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} |\psi_k\rangle |\phi_k\rangle \otimes |\widetilde{\lambda_{k,j}}\rangle \otimes |0\rangle_{flag} \\ &\quad \approx \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} |\psi_k\rangle |\phi_k\rangle \otimes |\widetilde{\lambda_k}\rangle \otimes |0\rangle_{flag} , \end{aligned}$$

where the $\widetilde{\lambda_{k,j}}$ are t-bit strings, $|\alpha_{k,j}|^2$ is close to 1 if and only if $\lambda_k \approx \widetilde{\lambda_{k,j}}$, and $\widetilde{\lambda_k}$ denotes the best t-bit approximation of λ_k.

3. Use controlled rotations to "imprint" the t-bit approximations of the eigenvalues into the amplitudes of the flag-register to prepare the state

$$\begin{aligned}
&\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_{k,j}}\right\rangle \\
&\qquad \otimes \left(\sqrt{\widetilde{\lambda_{k,j}}} \left|0\right\rangle_{flag} + \sqrt{1-\widetilde{\lambda_{k,j}}} \left|1\right\rangle_{flag} \right) \\
&\approx \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_k}\right\rangle \\
&\qquad \otimes \left(\sqrt{\widetilde{\lambda_k}} \left|0\right\rangle_{flag} + \sqrt{1-\widetilde{\lambda_k}} \left|1\right\rangle_{flag} \right).
\end{aligned}$$

4. Use fixed point amplitude amplification to amplify states whose flag-register is in the state $|0\rangle$ to prepare an approximation of the state

$$\begin{aligned}
&\frac{1}{\sqrt{\mathrm{Tr}(H)}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \sqrt{\widetilde{\lambda_{k,j}}} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_{k,j}}\right\rangle \otimes \left|0\right\rangle_{flag} \\
&\quad \approx \frac{1}{\sqrt{\mathrm{Tr}(H)}} \sum_{k=0}^{2^n-1} \sqrt{\widetilde{\lambda_k}} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \otimes \left|\widetilde{\lambda_k}\right\rangle \otimes \left|0\right\rangle_{flag}.
\end{aligned}$$

5. Finally, uncompute and discard the eigenvalue- and flag-register to prepare the state

$$\begin{aligned}
\left|\psi_H\right\rangle &= \frac{1}{\sqrt{\mathrm{Tr}(H)}} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^t} \alpha_{k,j} \sqrt{\widetilde{\lambda_{k,j}}} \left|\psi_k\right\rangle \left|\phi_k\right\rangle \\
&\approx \frac{1}{\sqrt{\mathrm{Tr}(H)}} \sum_{k=0}^{2^n-1} \sqrt{\widetilde{\lambda_k}} \left|\psi_k\right\rangle \left|\phi_k\right\rangle.
\end{aligned}$$

Looking at the cost of the above algorithm, we note that Steps 2 and 3 can be implemented up to polynomial precision in time $\mathcal{O}\left(\mathrm{poly}(n)\right)$. Also, note that Step 4 can be implemented up to polynomial precision in time $\mathcal{O}\left(\sqrt{2^n/\operatorname{tr}\{H\}}\right)$, which brings the total runtime to

$$\mathcal{O}\left(\mathrm{poly}(n) + \sqrt{2^n/\operatorname{tr}\{H\}}\right).$$

Besides being able to efficiently sample from an approximation of SWES on a quantum computer, we show that SWES requires superpolynomial time on a classical computer (unless the one clean qubit model can be efficiently simulated on a classical computer). To be precise, we show that sampling from SWES allows us to efficiently estimate the normalized subtrace discussed in Section 3.2.2, which is known to be

DQC1-hard [40]. We gather this in the following theorem, the proof of which can be found in the Supplementary Material.

Theorem 11. SWES *is* DQC1*-hard. Moreover,* SWES *with the input restricted to log-local Hamiltonians remains* DQC1*-hard.*

The above theorem motivates us to look for practical applications of SWES, or more specifically, of the purified quantum query-access described in Eq. (3.17). We end this section by discussing such an application called spectral entropy estimation, which when applied to combinatorial Laplacians can be used to compare complex networks. The classical hardness of SWES opens up another road towards practical quantum advantage, as it could be that combinatorial Laplacians arising in complex network analysis form a rich enough family for which SWES remains classically hard when restricted to them.

Spectral entropy estimation of the combinatorial Laplacian

Recently, several quantum information-inspired entropic measures for complex network analysis have been proposed [34, 68]. One example of these are spectral entropies of the combinatorial Laplacian, which measure the degree of overlapping of cliques within the given complex network [155, 179, 134]. Specifically, it has been shown that these entropic measures can be used to measure network centralization (i.e., how central is the most central node in relation to all other nodes) [179], network regularity (i.e., the difference in degrees among nodes) [155], and clique connectivity (i.e., the overlaps between communities in the network) [134].

If $\lambda_0, \ldots, \lambda_{d_k^G-1}$ denote the eigenvalues of a combinatorial Laplacian Δ_k^G (i.e., $d_k^G = \dim \mathcal{H}_k^G$), then its *spectral entropy* is defined by

$$S(\Delta_k^G) = -\sum_{j=0}^{d_k^G-1} p(\lambda_j) \log(p(\lambda_j)), \tag{3.18}$$

where we define $p(\lambda_j) = \lambda_j / \left(\sum_k \lambda_k\right)$. This spectral entropy coincides with the von Neumann entropy of $\Delta_k^G / \operatorname{tr}\{\Delta_k^G\}$. Equivalently, it coincides with the Shannon entropy of the distribution $p(\lambda_j)$. Another entropy that is used in complex network analysis is the α*-Renyi spectral entropy*, which is given by

$$S_\alpha(\Delta_k^G) = \frac{1}{1-\alpha} \log\left(\sum_{j=0}^{d_k-1} p(\lambda_j)^\alpha\right), \tag{3.19}$$

where $\alpha \geq 0$ and $\alpha \neq 1$. The limit for $\alpha \to 1$ is the spectral entropy as defined in Eq. (3.18).

To estimate the spectral entropy defined in Eq. (3.18), one can use techniques from [11, 192] to classically postprocess samples from $p(\lambda_j)$ that one obtains from the quantum algorithm for SWES described in the previous section. However, since we can implement purified quantum query-access using the algorithm described in the previous section, the postprocessing can be sped up quadratically using quantum

methods [84]. This idea of speeding up the postprocessing of samples using quantum methods also holds for the α-Renyi entropy defined in Eq. (3.19), where one can either classically postprocess the samples [12], or use faster quantum methods [181].

Because we have shown that sampling from SWES is DQC1-hard, the above approach to spectral entropy estimation can not be done efficiently on a classical computer – i.e., it cannot be dequantized – when generalized to arbitrary sparse matrices (unless the one clean qubit model can be efficiently simulated on a classical computer). Moreover, as the α-Renyi entropy is the logarithm of the Schatten p-norm, and it is known that estimating Schatten p-norms is DQC1-hard [51], we find that computing α-Renyi entropy is classically intractable (again, unless the one clean qubit model can be efficiently simulated on a classical computer).

3.4 Possibilities and challenges for implementations

As near-term quantum devices are still limited, it is crucial to make sure to use them to their fullest extent when implementing a quantum algorithm. Near-term devices are limited in size, gates are error prone, qubits decohere, and their architectures are limited [160]. We are therefore interested in algorithms that require few gates (to minimize the effect of decoherence and gate errors), that are not too demanding regarding architecture, while achieving advantages with few qubits and being tolerant to noise (which will inevitably be present in the system regardless of the depth and gate count). The quantum algorithms we consider use Hamiltonian simulation and quantum phase estimation. Fortunately, both resource optimization [30] and error-mitigation [187, 38, 78, 136, 147] for these routines are important topics for the broadly investigated field of quantum algorithms for quantum chemistry and many-body physics, and any progress achieved for those purposes can be readily applied. Moreover, recent work has focused on reducing the depth of the quantum circuit required to implement the algorithm for (approximate) Betti number estimation [189]. In this section we will focus on the issues of size and noise. First, we investigate the required number of qubits and we propose methods on how to reduce this. Based on these methods, we provide an estimate of the number of qubits required to challenge classical methods. Finally, we discuss issues regarding robustness of the algorithm to noise in the quantum hardware.

To analyze the number of qubits required to implement Hamiltonian simulation of a $2^n \times 2^n$-sized input matrix, we consider two possible scenarios: the input matrix is either given to us as local terms, or it is specified via sparse access. If the input matrix is given to us as local terms, then we can implement Hamiltonian simulation based on the Trotter-Suzuki formula [129]. As this Hamiltonian simulation technique does not require ancillary qubits (assuming the available gate set can implement each of the Trotter steps without ancillary qubits) [51], we can implement it using only n qubits. On the other hand, if the input matrix is specified via sparse access, then we have to use more intricate Hamiltonian simulation techniques (e.g., based on quantum signal processing [133]). The downside of these methods is that they require an ancillary register to 'load' the queries to the sparse-access oracles onto. By having to add this ancillary register, the total number of qubits required to implement these Hamiltonian

simulation techniques becomes $2n+r+1$, where r is the number of bits used to specify the entries of the input matrix. In other words, sparse-access oracles more than double the required number of qubits.

When possible it is therefore advantageous to avoid using sparse access when having first proof-of-principle demonstrations of quantum advantage in mind. One way of doing so is to add an extra precompilation step that finds a suitable decomposition of the input matrix. In particular, one can trade-off the required number of ancilla qubits for some amount of precompilation and some extra depth of the precompiled circuit, in the following two ways. First, one could decompose the input matrix in terms of a linear combination of unitaries, and use related techniques for Hamiltonian simulation of such input matrices [32]. This brings the required number of qubits down from $2n + r + 1$ to $n + \log(m)$, where m is the number of terms in the linear combination of unitaries. Secondly, one could decompose the input matrix in terms of a sum of local Hamiltonians and use Hamiltonian simulation based on the Trotter-Suzuki formula. This brings the required number of qubits down from $2n+r+1$ to n. Thus, both approaches can halve the number of required qubits, however, one has to be careful as finding such decompositions may constitute a dominating overhead.

In case of Betti number estimation, we note that such precompilation is in fact feasible and meaningful. This is due to the fact that in this case there is a direct way to decompose input matrix (i.e., the combinatorial Laplacian) as a sum of Pauli-strings in order to implement Hamiltonian simulation based on the Trotter-Suzuki formula. Specifically, due to the close relationship between combinatorial Laplacians and Hamiltonians of the fermion hardcore model (as described in Section 3.2.3) [48] we can decompose the combinatorial Laplacian into a sum of Pauli-strings by applying a fermion to qubit mapping such as the Jordan-Wigner or Bravyi-Kitaev transformations to Eq (3.4). Note however that this does not guarantee that Hamiltonian simulation based on the Trotter-Suzuki formula will be efficient as the decomposition might require exponentially many terms and the locality of the individual terms could be large. As can be seen in Eq. (3.4), the number of terms in the decomposition scales with the degree of the vertices in the complement of the graph. In particular, if the graph is such that any vertex is connected to all other vertices except for a constant number of them, then the number of terms in the decomposition scales polynomially. As discussed in Section 3.2.4, these are exactly the type of graphs where the quantum algorithm for Betti number estimation achieves a speedup over the best known classical algorithms, since these types of graphs are clique-dense (i.e., they satisfy Eq. (2.31)). The locality of the Pauli-strings in the decomposition can however not be guaranteed to be small, but this fortunately has less effect on the depth of the circuit. Finally, we remark that this decomposition also gives rise to a technique that allows one to control the depth of the circuit required for the Hamiltonian simulation. Namely, by dropping certain terms from the decomposition (e.g., terms with a small coefficient) one could reduce the depth of the circuit required for Hamiltonian simulation, while making sure to not perturb the matrix too much as to drastically change the low-lying spectral density.

Next, we focus on the number of qubits required for the quantum phase estimation. Standard quantum phase estimation requires an eigenvalue register of t qubits to estimate the eigenvalues up to t-bits of precision (which consequently determines the

threshold in low-lying spectral density estimation). Fortunately, much improvement is possible in terms of the size of this eigenvalue register. First, as low-lying spectral density is only concerned with whether the t-bit approximation of an eigenvalue is zero or not, we can bring the size the of eigenvalue register down to $\log(t)$ by using a counter [163]. Moreover, we can bring the size of this eigenvalue register down to a single qubit at the expense of classical post-processing and qubit reinitialization methods [74, 148, 180].

We can now give the brief estimate of the number of qubits needed for demonstrations of quantum advantage (i.e., sizes needed to go beyond the best known classical methods). The best known classical methods for low-lying spectral density estimation, to our knowledge, are able to estimate the rank of a matrix in time linear in the number of nonzero entries [190, 59, 70, 124]. These methods are at most quadratically faster than exact diagonalization, which tends to hit a practical wall around matrices of size 2^{40}. We therefore look at how many qubits are required to estimate the low-lying spectral density below a threshold of about 10^{-9} (i.e., $t \approx \log(10^9) < 30$) of matrices of size around 2^{80} (i.e., $n \approx 80$). In this case, the required number of qubits for standard implementations is approximately

$$2n + r + 1 + t \approx 200.$$

If we precompile the input matrix through finding a decomposition in terms of local Hamiltonians, this can be reduced to

$$n + t \approx 110.$$

This can be further reduced to $n + \log(t)$ by using a counter in the eigenvalue register. Lastly, by using a single-qubit eigenvalue register (at the cost of classical postprocessing and qubit reinitialization) we bring the number of required qubits in the optimal case down to

$$n + 1 \approx 80,$$

which is tantalizingly close to what leading teams are expected to achieve in the immediate future in terms of qubit numbers alone.

When it comes to the robustness to noise in the hardware, we need to consider the type of algorithm that is being applied (i.e., how noise affects this algorithm in general) together with the specifics of the application. The algorithm we consider involves many iterations of Hamiltonian simulation and quantum phase estimation, where we are interested in the expected value of a two outcome measurement (designating the zero eigenvalues). As noted earlier, these routines are also crucial for quantum algorithms for quantum chemistry and many-body physics, and consequently, all error-mitigation methods developed for these purposes can be readily applied [187, 38, 78, 136, 147]. However, as in quantum chemistry and many-body physics one extracts the entire eigenvalues, as opposed to just the frequency of the zero eigenvalue, the application we consider is less demanding. Additional robustness properties van be inferred from the nature of the particular problem solved. For instance, in machine learning and data analysis applications, the fact that the algorithm serves the purpose of dealing with noise in the data might make noise in the hardware

less detrimental compared to when solving more exact problems [73].

Unfortunately, this argument cannot be as readily applied to Betti number estimation, as noise in the data does not correspond to small perturbations of the simulated matrix (i.e., the combinatorial Laplacian), but rather to a completely different matrix altogether. In turn, small perturbations of the simulated matrix do not corresponds to any meaningful perturbation of the input data. However, we can still identify certain robust features by considering what perturbations of the combinatorial Laplacian entail for the final output, i.e., the low-lying spectral density. Specifically, if the combinatorial Laplacian is perturbed by a small enough matrix (e.g., in terms of operator norm or rank), then the low-lying spectral density remains largely unchanged as such perturbations will not push the low-lying eigenvalues above the threshold. These settings are often studied in the field of perturbation theory [114], which would allow us to make these arguments completely formal. Moreover, as a random matrix is likely of full rank [80], the perturbed combinatorial Laplacian is also likely of full rank, indicating that in the noisy setting we should focus on approximate Betti number estimation methods, as opposed to exact ones. Finally, there has been work verifying the robustness of the quantum algorithm for Betti number estimation in an experimental setting [106].

Chapter 4

Structural risk minimization for quantum linear classifiers

In this chapter we theoretically analyze and quantify the influence that model parameters of quantum linear classifiers have on the trade-off in structural risk minimization. We first analyze the effect that model parameters have on the complexity term (i.e., the green line in Figure 2.6) and afterwards we analyze their effect on the training error (i.e., the blue line in Figure 2.6). Specifically, in Section 4.1 we analyze the complexity term by establishing analytic upper bounds on complexity measures (i.e., the VC dimension and fat-shattering dimension) of quantum linear classifiers. In Section 4.2 we study the influence that model parameters which influence the established complexity measure bounds have on the training error term. Finally, in Section 4.3, we discuss how to implement structural risk minimization of quantum linear classifiers based on the obtained results.

4.1 Complexity of quantum linear classifiers

In this section we determine the two complexity measures defined in the previous section – i.e., the fat-shattering dimension and VC dimension – for families of quantum linear classifiers. As a result, we identify model parameters that allow us to control the complexity term in the expected error bounds of Theorems 1 and 2. These bounds upper bound the expected error by a sum of a training error and a complexity term that we would like to trade-off to achieve the best possible bound. Using the model parameters that we identify, we can balance this trade-off to construct the best possible model. In short, these model parameters can be used to balance the trade-off considered by structural risk minimization, as depicted in Figure 2.6. Throughout this section we fix the feature map to be the one defined Equation (2.22) and we allow our separating hyperplanes to come from a family of observables $\mathbb{O} \subseteq \mathrm{Herm}\left(\mathbb{C}^{2^n}\right)$ (e.g., the family of observables implementable using either the explicit or implicit realization of quantum linear classifiers). Our goal is to determine analytical upper bounds on complexity measures of the resulting family of quantum linear classifiers.

First, we show that the VC dimension of a family of quantum linear classifiers is upper bounded by the dimension of the span of the observables that it uses. This in turn is upper bounded by the square of the dimension of the space upon which the observables act nontrivially. We remark that while the VC dimension of quantum linear classifiers also has a clear dependence on the feature map, we chose to focus on the observables because the resulting upper bounds give rise to more explicit guidelines on how to tune the quantum model to perform structural risk minimization (as we discuss in more detail in Section 4.3). We defer the proof to Appendix B.1.1.

Proposition 12. *Let $\mathbb{O} \subseteq \mathrm{Herm}\big(\mathbb{C}^{2^n}\big)$ be a family of n-qubit observables with $r = \dim\big(\sum_{\mathcal{O}\in\mathbb{O}} \mathrm{Im}\,\mathcal{O}\big)$[1]. Then, the VC dimension of*

$$\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}} = \Big\{ c(x) = \mathrm{sign}\big(\mathrm{Tr}\left[\mathcal{O}\rho_\Phi(x)\right] - d\big) \;\Big|\; \mathcal{O} \in \mathbb{O},\, d \in \mathbb{R} \Big\} \tag{4.1}$$

satisfies

$$\mathrm{VC}\big(\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}\big) \leq \dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1 \leq r^2 + 1. \tag{4.2}$$

Remark(s). *The quantity r in the above proposition is related to the ranks of the observables. Specifically, note that for any two observables $\mathcal{O}, \mathcal{O}' \in \mathrm{Herm}\big(\mathbb{C}^{2^n}\big)$ we have that*

$$\dim\big(\mathrm{Im}\,\mathcal{O} + \mathrm{Im}\,\mathcal{O}'\big) = \mathrm{rank}\big(\mathcal{O}\big) + \mathrm{rank}\big(\mathcal{O}'\big) - \dim\big(\mathrm{Im}\,\mathcal{O} \cap \mathrm{Im}\,\mathcal{O}'\big).$$

The above proposition implies the (essentially obvious) result that VC dimension of a family of implicit quantum linear classifiers is upper bounded by the number of training examples (i.e., the operators $\{\rho_\Phi(x)\}_{x\in\mathcal{D}}$ span a subspace of dimension at most $\big|\mathcal{D}\big|$). We are however more interested in the application of the above proposition to explicit quantum linear classifiers. In this case, we choose to focus on the upper bound $r^2 + 1$ because it has interpretational advantages as to what parts of the model one has to tune from the perspective of structural risk minimization (i.e., recall from Section 2.3 that one way to perform structural risk minimization is to tune the VC dimension). Moreover, in the case of explicit quantum linear classifiers, the bound $r^2 + 1$ is only quadratically worse than the bound $\dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1$. To see this, we consider a family of explicit quantum linear classifiers with observables $\mathbb{O}_{\mathrm{explicit}} = \big\{\mathcal{O}^\lambda_\theta\big\}$, where

$$\mathcal{O}^\lambda_\theta = W^\dagger(\theta) \cdot \mathrm{diag}\big(\lambda(0), \ldots, \lambda(2^n - 1)\big) \cdot W(\theta)$$

and we denote $W(\theta)\,|i\rangle = |\psi_i(\theta)\rangle$. Next, suppose that $\lambda(j) = 0$ for all $j > L$ and

[1] Here $\sum$ denotes the sum of vector spaces and $\mathrm{Im}\,\mathcal{O}$ denotes the image (or column space) of the operator $\mathcal{O}$.

define

$$H = \mathrm{Span}_{\mathbb{C}}\Big\{ |\psi_0(\theta)\rangle, \ldots, |\psi_L(\theta)\rangle \;\; : \;\; \theta \in \mathbb{R}^m \Big\}, \tag{4.3}$$

$$V = \mathrm{Span}_{\mathbb{R}}\Big\{ \sum_{i=0}^{L} \lambda(i) \, |\psi_i(\theta)\rangle \langle\psi_i(\theta)| \;\; : \;\; \theta \in \mathbb{R}^m \Big\}, \tag{4.4}$$

Then, Proposition 12 states that

$$\mathrm{VC}\big(\mathcal{C}_{\mathrm{qlin}}^{\mathbb{O}_{\mathrm{explicit}}}\big) \leq \dim\big(V\big) + 1 \leq \dim(H)^2 + 1.$$

Now, by the following lemma, we indeed find that the bound $r^2 + 1$ is only quadratically worse than the bound $\dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1$. We again defer the proof to Appendix B.1.1.

Lemma 13. *The vector spaces defined in Eq.* (4.3) *and Eq.* (4.4) *satisfy*[2]

$$\dim(H) \leq \dim(V) \leq \dim(H)^2.$$

Therefore, if we sufficiently limit $r = \dim(H)$, then this also limits $\dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) = \dim(V)$. Moreover, even though $\dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1$ can provide a tighter bound, it can still be advantageous to study the bound $r^2 + 1$ because it might have interpretational advantages. Specifically, it might be easier to construct cases of ansatze where the latter bound allows us to identify a controlable hyperparameter that controls the VC dimension (as we discuss in more detail in Section 4.3).

Note that the quantity r defined in the above proposition, depends on both the structure of the ansatz W as well as the post-processing function λ. One way to potentially limit r is by varying the rank of the final measurement (i.e., the value L defined above). However, for several ansatzes in literature, having either a low-rank or a high-rank final measurement will not make a difference in terms of the VC dimension bound $r^2 + 1$[3]. To see this, consider an ansatz consisting of a single layer of parameterized X-rotations on all qubits, where each rotation is given a separate parameter. Already for this simple ansatz even the first columns $\{\bigotimes_{i=1}^{n} X_i(\theta_i) \, |0\rangle \mid \theta \in [0, 2\pi)^n\}$ span the entire n-qubit Hilbert space. In particular, the above proposition gives the same VC dimension upper bound for the cases where the final measurement is of rank $L = 1$, and where it is of full rank $L = 2^n$ (i.e., we have no guarantee that limiting L limits the VC dimension). This motivates us to design ansatzes for which subsets of columns do not span the entire Hilbert space when varying the variational parameter θ. On the other hand, to exploit the bound $\dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1$ one needs to consider the span of the projectors onto the first L columns in the vector space of Hermitian operators. This quantity can be slightly less intuitive than the span of the first L columns in the n-qubit Hilbert space, and in Section 4.3 we show that this

[2]Note that there exists ansatzes for which the inequalities are strict, i.e., $\dim(H) < \dim(V) < \dim(H)^2$ (e.g., see the first example discussed in Section 4.3).

[3]The relationship between the quantity r and the ranks of the observable can be made explicit by considering the overlaps between the images of the observables. A more detailed explanation of this can be found in Appendix B.1.2.

latter quantity can already be used to affirm the effectiveness of certain regularization techniques. Specifically, in Section 4.3 we discuss examples of ansatzes for which subsets of columns do not span the entire Hilbert space when varying the variational parameter, and we explain how they allow for structural risk minimization by limiting the rank of the final measurement.

Next, we show that the fat-shattering dimension of a family of quantum linear classifiers is related to the Frobenius norm of the observables that it uses. In particular, we show that we can control the fat-shattering dimension of a family of quantum linear classifiers by limiting the Frobenius norm of its observables. We defer the proof to Appendix B.1.3, where we also discuss the implications of this result in the probably approximately correct (PAC) learning framework.

Proposition 14. *Let $\mathbb{O} \subseteq \mathrm{Herm}\left(\mathbb{C}^{2^n}\right)$ be a family of n-qubit observables with $\eta = \max_{\mathcal{O}\in\mathbb{O}} \|\mathcal{O}\|_F$. Then, the fat-shattering dimension of*

$$\mathcal{F}^{\mathbb{O}}_{\mathrm{qlin}} = \left\{ f_{\mathcal{O},d}(x) = \mathrm{Tr}\left[\mathcal{O}\rho_\Phi(x)\right] - d \;\middle|\; \mathcal{O} \in \mathbb{O},\ d \in \mathbb{R} \right\} \tag{4.5}$$

is upper bounded by

$$\mathrm{fat}_{\mathcal{F}^{\mathbb{O}}_{\mathrm{qlin}}}(\gamma) \leq O\left(\frac{\eta^2}{\gamma^2}\right). \tag{4.6}$$

Remark(s). *The upper bound in the above proposition matches the result discussed in [127]. This was derived independently by one of the authors of [95] in [193], and we include it here for completeness.*

The above proposition shows that the fat-shattering dimension of a family of explicit quantum linear classifiers can be controlled by limiting $||\mathcal{O}^\lambda_\theta||_F = \sqrt{\sum_{i=1}^{2^n} \lambda(i)^2}$. In particular, it shows that the selection of the postprocessing function λ is important when tuning the complexity of the family of classifiers. Furthermore, the above proposition shows that the fat-shattering dimension of a family of implicit quantum linear classifiers can be controlled by limiting $||\mathcal{O}_\alpha||_F \leq ||\alpha||_1$. It is important to note that the Frobenius norm itself does not fully characterize the generalization performance of a family of quantum linear classifiers. Specifically, plugging Theorem 14 into Proposition 2 we find that the generalization performance bounds depend on both the Frobenius norm as well as the functional margin on training examples[4]. Therefore, to optimize the generalization performance bounds one has to minimize the Frobenius norm, while ensuring the functional margin on training examples stays large. Note that one way to achieve this is by maximizing the so-called geometric margin, which on a set of example $\{x_i\}$ is given by $\min_i \left|\mathrm{Tr}\left[\mathcal{O}\rho_\Phi(x_i)\right] - d\right| / ||\mathcal{O}||_F$.

[4]Recall that the functional margin of $c_{f,d}(x) = \mathrm{sign}\big(f(x) - d\big)$ on a set of examples $\{x_i\}$ is $\min_i |f(x_i) - d|$.

4.2 Expressivity of quantum linear classifiers

Having established that the quantity r defined in Proposition 12 and the Frobenius norms of the observables influence the complexity of the family of quantum linear classifiers (i.e., the green line in Figure 2.6), we will now study the influence of these parameters on the training errors that the classifiers can achieve (i.e., the blue line in Figure 2.6). First, we study the influence of these model parameters on the ability of the classifiers to correctly classify certain sets of examples. Afterwards, we study the influence of these model parameters on the margins that the classifiers can achieve.

Recall from the previous section that the VC dimension of certain families of quantum linear classifiers depends on the rank of the observables that it uses. For instance, if the observables are such that their images are (largely) overlapping, then the quantity r defined in Proposition 12 can be controlled by limiting the ranks of all observables. In Section 4.3 we use this observation to construct ansatzes for which the VC dimension bound can be tuned by varying the rank of the observable measured on the output of the circuit. Since the VC dimension is only concerned with whether an example is correctly classified (and not what margin it achieves), we choose to investigate the influence of the rank on being able to correctly classify certain sets of examples. In particular, we show that any set of examples that can be correctly classified using a low-rank observable, can also be correctly classified using a high-rank observable. Moreover, we also show that there exist sets of examples that can only be correctly classified using observables of at least a certain rank. We defer the proof to Appendix B.2.1.

Proposition 15. *Let $\mathcal{C}_{\text{qlin}}^{(r)}$ denote the family of quantum linear classifiers corresponding to observables of exactly rank r, that is,*

$$\mathcal{C}_{\text{qlin}}^{(r)} = \left\{ c(\rho) = \operatorname{sign}\big(\operatorname{Tr}\left[\mathcal{O}\rho\right] - d\big) \;\middle|\; \mathcal{O} \in \operatorname{Herm}\big(\mathbb{C}^{2^n}\big),\, \operatorname{rank}\big(\mathcal{O}\big) = r,\, d \in \mathbb{R} \right\} \tag{4.7}$$

Then, the following statements hold:

(i) For every finite set of examples $\mathcal{D}$ that is correctly classified by a quantum linear classifier $c \in \mathcal{C}_{\text{qlin}}^{(k)}$ with $0 < k < 2^n$, there exists a quantum linear classifier $c \in \mathcal{C}_{\text{qlin}}^{(r)}$ with $r > k$ that also correctly classifies $\mathcal{D}$.

(ii) There exists a finite set of examples that can be correctly classified by a classifier $c \in \mathcal{C}_{\text{qlin}}^{(r)}$, but which no classifier $c' \in \mathcal{C}_{\text{qlin}}^{(k)}$ with $k < r$ can classify correctly.

Note that in the above proposition we define our classifiers in such a way that high-rank classifiers do not subsume low-rank classifiers. In particular, the family of observables that $\mathcal{C}_{\text{qlin}}^{(r)}$ and $\mathcal{C}_{\text{qlin}}^{(k)}$ use are completely disjoint for $k \neq r$. The construction behind the proof of the above proposition is inspired by tomography of observables. Specifically, we construct a protocol that queries a quantum linear classifier and based on the assigned labels checks whether the underlying observable is approximately equal to a fixed target observable of a certain rank. In particular, we can use this to test whether the underlying observable is really of a given rank, as no low-rank observable can agree with a high-rank observable on the assigned labels during this

protocol. Note that if we could query the expectation values of the observable, then tomography would be straightforward. However, the classifier only outputs the sign of the expectation value, which introduces a technical problem that we circumvent. Our protocol could be generalized to a more complete tomographic-protocol which uses queries to a quantum linear classifier in order to find the spectrum of the underlying observable.

Next, we investigate the effect that limitations of the rank of the observables used by a family of quantum linear classifier have on its ability to implement certain families of standard linear classifiers. In particular, assuming that the feature map is bounded (i.e., all feature vectors have finite norm), then the following proposition establishes the following chain of inclusions:

$$\mathcal{C}_{\text{lin}} \text{ on } \mathbb{R}^{2^n} \subseteq \mathcal{C}_{\text{qlin}}^{(\leq 1)} \text{ on } n+1 \text{ qubits} \subseteq \ldots \tag{4.8}$$

$$\subseteq \mathcal{C}_{\text{qlin}}^{(\leq r)} \text{ on } n+1 \text{ qubits} \subseteq \cdots \subseteq \mathcal{C}_{\text{lin}} \text{ on } \mathbb{R}^{4^n}, \tag{4.9}$$

where $\mathcal{C}_{\text{qlin}}^{(\leq r)}$ denotes the family of quantum linear classifiers using observables of rank at most r. Note that $\mathcal{C}_{\text{qlin}}^{(\leq r)} \subsetneq \mathcal{C}_{\text{qlin}}^{(\leq r+1)}$ is strict due to Proposition 15. We defer the proof to Appendix B.2.2.

Proposition 16. *Let $\mathcal{C}_{\text{lin}}(\Phi)$ denote the family of linear classifiers that is equipped with a feature map Φ. Also, let $\mathcal{C}_{\text{qlin}}^{(\leq r)}(\Phi')$ denote the family of quantum linear classifiers that uses observables of rank at most r and which is equipped with a quantum feature map Φ'. Then, the following statements hold:*

(i) For every feature map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^N$ with $\sup_{x\in\mathbb{R}^\ell} ||\Phi(x)|| = M < \infty$, there exists a feature map $\Phi' : \mathbb{R}^\ell \to \mathbb{R}^{N+1}$ such that $||\Phi'(x)|| = 1$ for all $x \in \mathbb{R}^\ell$ and the families of linear classifiers satisfy $\mathcal{C}_{\text{lin}}(\Phi) \subseteq \mathcal{C}_{\text{lin}}(\Phi')$.

(ii) For every feature map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^N$ with $||\Phi(x)|| = 1$ for all $x \in \mathbb{R}^\ell$, there exists a quantum feature map $\Phi' : \mathbb{R}^\ell \to \text{Herm}\left(\mathbb{C}^{2^n}\right)$ that uses $n = \lceil \log N + 1 \rceil + 1$ qubits such that the families of linear classifiers satisfy $\mathcal{C}_{\text{lin}}(\Phi) \subseteq \mathcal{C}_{\text{qlin}}^{(\leq 1)}(\Phi')$.

(iii) For every quantum feature map $\Phi : \mathbb{R}^\ell \to \text{Herm}\left(\mathbb{C}^{2^n}\right)$, there exists a classical feature map $\Phi' : \mathbb{R}^\ell \to \mathbb{R}^{4^n}$ such that the families of linear classifiers satisfy $\mathcal{C}_{\text{qlin}}(\Phi) = \mathcal{C}_{\text{lin}}(\Phi')$.

Recall from the previous section that the fat-shattering dimension of a family of linear classifiers depends on the Frobenius norm of the observables that is uses. In the following proposition we show that tuning the Frobenius norm changes the margins that the model can achieve, which gives rise to better generalization performance (as discussed in Section 2.3). In particular, we show that there exist sets of examples that can only be classified with a certain margin by a classifier that uses an observable of at least a certain Frobenius norm. We defer the proof to Appendix B.2.3.

Proposition 17. *Let $\mathcal{C}^{(\eta)}_{\text{qlin}}$ denote the family of quantum linear classifiers corresponding to all n-qubit observables of Frobenius norm η, that is,*

$$\mathcal{C}^{(\eta)}_{\text{qlin}} = \Big\{ c(\rho) = \text{sign}\big(\text{Tr}\left[\mathcal{O}\rho\right] - d\big) \;\Big|\; \mathcal{O} \in \text{Herm}\big(\mathbb{C}^{2^n}\big) \text{ with } ||\mathcal{O}||_F = \eta,\, d \in \mathbb{R} \Big\}. \tag{4.10}$$

Then, for every $\eta \in \mathbb{R}_{>0}$ and $0 < m \leq 2^n$ there exists a set of m examples consisting of binary labeled n-qubit pure states that satisfies the following two conditions:

(i) There exists a classifier $c \in \mathcal{C}^{(\eta)}_{\text{qlin}}$ that correctly classifies all examples with margin $\eta/\sqrt{m}$.

(ii) No classifier $c' \in \mathcal{C}^{(\eta')}_{\text{qlin}}$ with $\eta' < \eta$ can classify all examples correctly with margin $\geq \eta/\sqrt{m}$.

In conclusion, in Proposition 12 we showed that in certain cases the rank of the observables control the model's complexity (e.g., if the observables have overlapping images), and in Proposition 15 we showed that the rank also controls the model's ability to achieve small training errors. Moreover, in Proposition 17 we similarly showed that the Frobenius norm not only controls the model's complexity (see Proposition 14), but that it also controls the model's ability to achieve large functional margins. However, note that tuning each model parameter achieves a different objective. Namely, increasing the rank of the observable increases the ability to correctly classify sets of examples, whereas increasing the Frobenius norm of the observable increases the margins that it can achieve. For example, one can increase the Frobenius norm of an observable by multiplying it with a positive scalar which increases the margin it achieves, but in order to correctly classify the sets of examples discussed in Proposition 15 one actually has to increase the rank of the observable.

4.3 Structural risk minimization in practice

Having established how certain model parameters of quantum linear classifiers influence both the model's complexity and its ability to achieve small training errors, we now discuss how to use these results to implement structural risk minimization of quantum linear classifiers in practice. In particular, we will discuss a common approach to structural risk minimization called *regularization*. In short, what regularization entails is instead of minimizing only the training error E_{train}, one simultaneously minimizes an extra term $h(\omega)$, where h is a function that takes larger values for model parameters ω that correspond to more complex models. In this section, we discuss different types of regularization (i.e., different choices of the function h) that can be performed in the context of quantum linear classifiers based on the results of the previous section. These types of regularization help improve the performance of quantum linear classifiers in practice, without putting more stringent requirements on the quantum hardware and are thus NISQ-suitable.

To illustrate how Proposition 12 can be used to implement structural risk minimization in the explicit approach, consider the setting where we have a parameterized

quantum circuit $W(\theta)$ (with $\theta \in \mathbb{R}^p$) followed by a fixed measurement that projects onto the first ℓ computational basis states. To use the bound $r^2 + 1$ from Proposition 12 one has to compute the quantity

$$\dim_{\mathbb{C}} \Big(\mathrm{Span}_{\mathbb{C}} \big\{ |\psi_i(\theta)\rangle \;\; : \;\; i = 1, \dots \ell, \;\; \theta \in \mathbb{R}^p \big\} \Big), \tag{4.11}$$

where $|\psi_i(\theta)\rangle$ denotes the ith column of $W(\theta)$. To use the other bound $\dim\big(\mathrm{Span}(\mathbb{O})\big) + 1$ from Proposition 12 one has to compute the quantity

$$\dim_{\mathbb{R}} \Big(\mathrm{Span}_{\mathbb{R}} \big\{ \sum_{i=1}^{\ell} |\psi_i(\theta)\rangle \langle \psi_i(\theta)| \;\; : \;\; \theta \in \mathbb{R}^p \big\} \Big), \tag{4.12}$$

Although both are of course possible, in some cases it is slightly easier to see how the quantity in Eq. (4.11) scales with respect to ℓ. Specifically, utilizing the quantity in Eq. (4.11) already leads to interesting ansatze that allow for structural risk minimization by limiting ℓ. As discussed below Proposition 12, setting ℓ to be either large or small will not influence the upper bound on the VC dimension independently of the structure of the parameterized quantum circuit ansatz W. The proposition therefore motivates the design of ansatzes whose first ℓ columns define a manifold when varying the variational parameter that is contained in a relatively low-dimensional linear subspace. Specifically, in this case Proposition 12 results in nontrivial bounds on the VC dimension that we aim to control by varying ℓ. We now give three examples of ansatzes that allow one to control the upper bound on the VC dimension by varying ℓ. In particular, these ansatzes allow structural risk minimization to be implemented by regularizing with respect to the rank of the final measurement.

Example 1 For the first example, split up the qubits up in a "control register" of size c and a "target register" of size t (i.e., $n = t + c$). Next, let $C{-}U_i(\theta_i)$ denote the controlled gate that applies the t-qubit parameterized unitary $U_i(\theta_i)$ to the target register if the control register is in the state $|i\rangle$. Finally, consider the ansatz

$$W(\theta) = \big[C{-}U_{2^c}(\theta_{2^c})\big] \cdot \ldots \cdot \big[C{-}U_1(\theta_1)\big].^{5}$$

Note that the matrix of $W(\theta)$ is given by the block matrix

$$W(\theta) = \begin{pmatrix} U_1(\theta_1) & & & \\ & U_2(\theta_2) & & \\ & & \ddots & \\ & & & U_{2^c}(\theta_{2^c}) \end{pmatrix}.$$

For this choice of ansatz, if the final measurement projects onto $\ell = m2^t$ ($m < 2^c$) computational basis states, then by Proposition 12 the VC dimension is at most $\ell^2 + 1$. Note that t is a controllable hyperparameter that can be used to tune the

[5] We can control the depth of $W(\theta)$ by either limiting the size of the control register or by simply dropping some of the controlled parameterized unitaries (i.e., setting $U_i(\theta_i) = I$).

VC dimension. In particular, we can set it such that the resulting VC dimension is not exponential in n. Let us now consider the other bound $\dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big)+1$ from Proposition 12. For this choice of ansatz, computing the quantity in Eq. (4.12) is also straightforward due to the block structure of the unitary. Moreover, for this choice of ansatz the inequalities in Lemma 13 are strict, which shows why being able to compute the quantity in Eq. (4.11) does not always imply that we can also compute the quantity in Eq. (4.12) (i.e., one is not simply the square of the other).

Example 2 For the second example, consider an ansatz that is composed of parameterized gates of the form $U(\theta)=e^{i\theta P}$ for some Pauli string $P\in\{X,Y,Z,I\}^{\otimes n}$. Specifically, consider the ansatz

$$W(\theta)=e^{i\theta_d P_d}\cdot\ldots\cdot e^{i\theta_1 P_1}.$$

By the bound r^2+1 from Proposition 12, for this choice of ansatz if the final measurement projects onto ℓ computational basis states the VC dimension is at most r^2+1, where $r=\ell\cdot 2^d$. This bound is obtained by computing the quantity in Eq. (4.11), which can be done by noting that a column of the unitary $U(\theta)$ spans a subspace of dimension at most 2 when varying the variational parameter θ. Moreover, subsequent layers of $U(\theta)$ will only increase the dimension of the span of a column by at most a factor 2. Thus, when applying $U(\theta)$ a total of d times, the dimension of the span of any ℓ columns of $W(\theta)$ is at most $r=\ell\cdot 2^d$. Also in this construction we note that d is a controllable hyperparameter that can be used to tune the VC dimension. In particular, we can set it such that the resulting VC dimension is not exponential in n. For this particular choice of ansatze, computing the quantity in Eq. (4.12) might also be possible, but it is a bit more involved and not necessary for our main goal of establishing that ℓ controls the VC dimension. In particular, one might be able to compute the quantity in Eq. (4.12), but the bound r^2+1 from Proposition 12 already suffices to establish that ℓ is a tunable hyperparameter that controls the VC dimension.

Example 3 For the third example, we use symmetry considerations as a tool to control the VC dimension. First, partition the n-qubit register into disjoint subsets $I_1,\ldots,I_k$ of size $|I_j|=m_j$ (i.e., $\sum_j m_j=n$). Next, consider "permutation-symmetry preserving" parameterized unitaries on these partitions, which are defined as

$$S^{+}_{I_j}(\theta)=e^{i\theta\sum_{i\in I_j}P_i},\quad\text{and}\quad S^{\otimes}_{I_j}(\theta)=e^{i\theta\prod_{i\in I_j}P_i},$$

where we have say $P_i=X_i$, $P_i=Y_i$, $P_i=Z_i$ or $P_i=I$ for all $i\in I_j$ (i.e., the same operator acting on all qubits in the partition I_j). Note that if we apply these operators to a permutation invariant state on the m_j-qubits in the jth partition, then it remains permutation invariant (independent of θ). From these symmetric parameterized unitaries we construct parameterized layers $U(\theta_1,\ldots,\theta_k)=\prod_{j=1}^{k}S^{+/\otimes}_{I_j}(\theta_j)$, from which we construct the ansatz as

$$W(\theta)=U(\theta_1^d,\ldots,\theta_k^d)\cdot\cdots\cdot U(\theta_1^1,\ldots,\theta_k^1),\quad\theta\in[0,1\pi)^{dk}.$$

By the bound r^2+1 from Proposition 12, for this choice of ansatz if the final measurement projects onto ℓ computational basis states the VC dimension is at most r^2+1, where

$$r = \ell \cdot \prod_{j=1}^{k} (m_j + 1).$$

This bound is obtained by computing the quantity in Eq. (4.11), which can be done by noting that if we apply a layer U to an n-qubit state that is invariant under permutations that only permute qubits within each partition, then it remains invariant under these permutations (i.e., independent of the choice of θ). In other words, the first column of $W(\theta)$ is always contained in the space of n-qubit states that are invariant under permutations that only permute qubits within each partition. Next, note that the dimension of the space of n-qubit states that are invariant under permutations that only permute qubits within each partition is equal to $\prod_{j=1}^{k}(m_j+1)$. Finally, note that any other column of $W(\theta)$ spans a space whose dimension is at most that of the first column of $W(\theta)$ when varying θ. Thus, any ℓ columns of $W(\theta)$ span a space of dimension is most $r = \ell \cdot \prod_{j=1}^{k}(m_j+1)$ when varying θ. Equivalent to the example above, for this particular choice of ansatze, computing the quantity in Eq. (4.12) might also be possible, but it is again a bit more involved and not necessary for our main goal of establishing that ℓ controls the VC dimension. In particular, one might be able to compute the quantity in Eq. (4.12), but the bound r^2+1 from Proposition 12 again already suffices to establish that ℓ is a tunable hyperparameter that controls the VC dimension.

In all of the above cases we see that we can control the upper bound on the VC dimension by varying the rank of the final measurement ℓ. It is worth noting that in these cases the regularized explicit quantum linear classifiers will generally give rise to a different model then the implicit approach without any theoretical guarantee regarding which will do better, because the standard relationship between the two models [165] will not hold anymore (i.e., the regularized explicit model does not necessarily correspond to a kernel method anymore).

Secondly, recall that by tuning the Frobenius norms of the observables used by a quantum linear classifier, we can balance the trade-off between its fat-shattering dimension and its ability to achieve large margins. In particular, this shows that we can implement structural risk minimization of quantum linear classifiers with respect to the fat-shattering dimension by regularizing the Frobenius norms of the observables. Again, it is important to note that the Frobenius norm itself does not fully characterize the generalization performance, since one also has to take into account the functional margin on training examples. In particular, to optimize the generalization performance one has to minimize the Frobenius norm, while ensuring that the functional margin on training examples stays large. As mentioned earlier, one way to achieve this is by maximizing the geometric margin, which on a set of examples $\{x_i\}$ is given by $\min_i \left|\mathrm{Tr}\left[\mathcal{O}\rho_\Phi(x_i)\right] - d\right| / ||\mathcal{O}||_F$. As before, for explicit quantum linear classifiers, we can estimate the Frobenius norm by sampling random computational basis states and computing the average of the postprocessing function λ on them in order to estimate $||\mathcal{O}_\theta^\lambda||_F = \sqrt{\sum_{i=1}^{2^n} \lambda(i)^2}$ (note that in some cases the Frobenius norm can be

computed more directly). On the other hand, for implicit quantum linear classifiers, we can regularize the Frobenius norm by bounding $||\alpha||_1$ as $||\mathcal{O}_\alpha||_F \leq ||\alpha||_1$. However, if the weights are obtained by solving the usual quadratic program [103, 168], then the resulting observable is already (optimally) regularized with respect to the Frobenius norm [165].

Besides the types of regularization for which we have established theoretical evidence of the effect on structural risk minimization, there are also other types of regularization that are important to consider. For instance, for explicit quantum linear classifiers, one could regularize the angles of the parameterized quantum circuit [153]. Theoretically analyzing the effect that regularizing the angles of the parameterized quantum circuit has on structural risk minimization would constitute an interesting direction for future research. Another example is regularizing circuit parameters such as depth, width and number of gates for which certain theoretical results are known [46, 52]. Finally, it turns out that one can also regularize quantum linear classifiers by running the circuits under varying levels of noise [47]. For these kinds of regularization the relationships between the regularized explicit and regularized implicit quantum linear classifiers are still to be investigated.

Chapter 5

Parametrized quantum policies for reinforcement learning

In this chapter, we demonstrate the potential of policies based on parameterized quantum circuits (PQCs) in solving classical reinforcement learning (RL) environments. Firstly, in Section 5.1 we first propose new model constructions and describe their learning algorithms. Next, in Section 5.2 we show numerically the influence of design choices on their learning performance. Finally, in Section 5.3, inspired by the classification task of Havlíček et al. [104], conjectured to be classically hard by the authors, we construct analogous RL environments where we show an empirical learning advantage of our PQC policies over standard DNN policies used in deep RL. Moreover, we also construct RL environments with a provable gap in performance between a family of PQC policies and any efficient classical learner.

5.1 Parametrized quantum policies

In this section, we give a detailed construction of our parametrized quantum policies and describe their associated training algorithms.

5.1.1 The RAW-PQC and SOFTMAX-PQC policies

At the core of our parametrized quantum policies is a PQC defined by a unitary $U(s, \boldsymbol{\theta})$ that acts on a fixed n-qubit state (e.g., $|0^{\otimes n}\rangle$). This unitary encodes an input state $s \in \mathbb{R}^d$ and is parametrized by a trainable vector $\boldsymbol{\theta}$. Although different choices of PQCs are possible, throughout our numerical experiments (Sec. 5.2 and 5.3.2), we consider so-called hardware-efficient PQCs [113] with an alternating-layered architecture [157, 170]. This architecture is depicted in Fig. 5.1 and essentially consists in an alternation of D_{enc} encoding unitaries U_{enc} (composed of single-qubit rotations R_z, R_y) and $D_{\text{enc}} + 1$ variational unitaries U_{var} (composed of single-qubit rotations R_z, R_y and entangling Ctrl-Z gates).

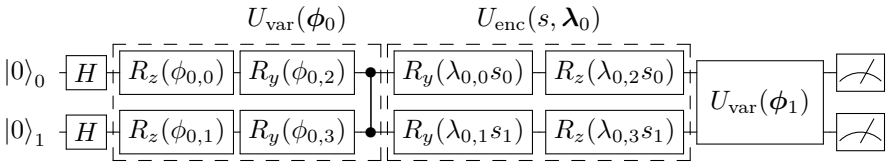


Figure 5.1: **PQC architecture for $n = 2$ qubits and depth $D_{\text{enc}} = 1$.** This architecture is composed of alternating layers of encoding unitaries $U_{\text{enc}}(s, \boldsymbol{\lambda}_i)$ taking as input a state vector $s = (s_0, \ldots, s_{d-1})$ and scaling parameters $\boldsymbol{\lambda}_i$ (part of a vector $\boldsymbol{\lambda} \in \mathbb{R}^{|\boldsymbol{\lambda}|}$ of dimension $|\boldsymbol{\lambda}|$), and variational unitaries $U_{\text{var}}(\boldsymbol{\phi}_i)$ taking as input rotation angles $\boldsymbol{\phi}_i$ (part of a vector $\boldsymbol{\phi} \in [0, 2\pi]^{|\boldsymbol{\phi}|}$ of dimension $|\boldsymbol{\phi}|$).

For any given PQC, we define two families of policies, differing in how the final quantum states $|\psi_{s,\boldsymbol{\theta}}\rangle = U(s, \boldsymbol{\theta})\,|0^{\otimes n}\rangle$ are used. In the RAW-PQC model, we exploit the probabilistic nature of quantum measurements to define an RL policy. For $|A|$ available actions to the RL agent, we partition $\mathcal{H}$ in $|A|$ disjoint subspaces (e.g., spanned by computational basis states) and associate a projection P_a to each of these subspaces. Using the projections P_a, we define our RAW-PQC policy $\pi_{\boldsymbol{\theta}}(a|s) = \langle P_a \rangle_{s,\boldsymbol{\theta}}$. A limitation of this policy family however is that it does not have a directly adjustable "greediness" (i.e., a control parameter that makes the policy more concentrated around certain actions). This consideration arises naturally in an RL context where an agent's policy needs to shift from an exploratory behavior (i.e., close to uniform distribution) to a more exploitative behavior (i.e., a peaked distribution). To remedy this limitation, we define the SOFTMAX-PQC model, that applies an adjustable softmax_β non-linear activation function on the expectation values $\langle P_a \rangle_{s,\boldsymbol{\theta}}$ measured on $|\psi_{s,\boldsymbol{\theta}}\rangle$. Since the softmax function normalizes any real-valued input, we can generalize the projections P_a to be arbitrary Hermitian operators O_a. We also generalize these observables one step further by assigning them trainable weights. The two models are formally defined below.

Definition 16 (RAW- and SOFTMAX-PQC)**.** *Given a PQC acting on n qubits, taking as input a state $s \in \mathbb{R}^d$, rotation angles $\boldsymbol{\phi} \in [0, 2\pi]^{|\boldsymbol{\phi}|}$ and scaling parameters $\boldsymbol{\lambda} \in \mathbb{R}^{|\boldsymbol{\lambda}|}$, such that its corresponding unitary $U(s, \boldsymbol{\phi}, \boldsymbol{\lambda})$ produces the quantum state $|\psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}}\rangle = U(s, \boldsymbol{\phi}, \boldsymbol{\lambda})\,|0^{\otimes n}\rangle$, we define its associated* RAW-PQC *policy as:*

$$\pi_{\boldsymbol{\theta}}(a|s) = \langle P_a \rangle_{s,\boldsymbol{\theta}} \tag{5.1}$$

where $\langle P_a \rangle_{s,\boldsymbol{\theta}} = \langle \psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}} | P_a | \psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}} \rangle$ is the expectation value of a projection P_a associated to action a, such that $\sum_a P_a = I$ and $P_a P_{a'} = \delta_{a,a'}$. $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\lambda})$ constitute all of its trainable parameters.
Using the same PQC, we also define a SOFTMAX-PQC *policy as:*

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{\beta \langle O_a \rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'} e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}}} \tag{5.2}$$

where $\langle O_a \rangle_{s,\boldsymbol{\theta}} = \langle \psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}} | \sum_i w_{a,i} H_{a,i} | \psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}} \rangle$ is the expectation value of the weighted Hermitian operators $H_{a,i}$ associated to action a, $\beta \in \mathbb{R}$ is an inverse-temperature

parameter and $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{\lambda}, \boldsymbol{w})$.

Note that we adopt here a very general definition for the observables O_a of our SOFTMAX-PQC policies. As we discuss in more detail in Appendix C.3, very expressive trainable observables can in some extreme cases take over all training of the PQC parameters $\boldsymbol{\phi}, \boldsymbol{\lambda}$ and render the role of the PQC in learning trivial. However, in practice, as well as in our numerical experiments, we only consider very restricted observables $O_a = \sum_i w_{a,i} H_{a,i}$, where $H_{a,i}$ are (tensor products of) Pauli matrices or high-rank projections on computational basis states, which do not allow for these extreme scenarios.

In our PQC construction, we include trainable *scaling parameters* $\boldsymbol{\lambda}$, used in every encoding gate to re-scale its input components. This modification to the standard data encoding in PQCs comes in light of recent considerations on the structure of PQC functions [166]. These additional parameters allow to represent functions with a wider and richer spectrum of frequencies, and hence provide shallow PQCs with more expressive power.

5.1.2 Learning algorithm

In order to analyze the properties of our PQC policies without the interference of other learning mechanisms [198], we train these policies using the basic Monte Carlo policy gradient algorithm REINFORCE [183, 199] (see Alg. 1). This algorithm consists in evaluating Monte Carlo estimates of the value function $V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{H-1} \gamma^t r_t\right]$, $\gamma \in [0,1]$, using batches of interactions with the environment, and updating the policy parameters $\boldsymbol{\theta}$ via a gradient ascent on $V_{\pi_{\boldsymbol{\theta}}}(s_0)$. The resulting updates (see line 8 of Alg. 1) involve the gradient of the log-policy $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$, which we therefore need to compute for our policies. We describe this computation in the following lemma.

Lemma 18. *Given a* SOFTMAX-PQC *policy* $\pi_{\boldsymbol{\theta}}$*, the gradient of its logarithm is given by:*

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) = \beta\Big(\nabla_{\boldsymbol{\theta}} \langle O_a \rangle_{s,\boldsymbol{\theta}} - \sum\nolimits_{a'} \pi_{\boldsymbol{\theta}}(a'|s) \nabla_{\boldsymbol{\theta}} \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}\Big). \tag{5.3}$$

Partial derivatives with respect to observable weights are trivially given by $\partial_{w_{a,i}} \langle O_a \rangle_{s,\boldsymbol{\theta}} = \langle \psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}} | H_{a,i} | \psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}} \rangle$ *(see Def. 16), while derivatives with respect to rotation angles* $\partial_{\phi_i} \langle O_a \rangle_{s,\boldsymbol{\theta}}$ *and scaling parameters*[1] $\partial_{\lambda_i} \langle O_a \rangle_{s,\boldsymbol{\theta}}$ *can be estimated via the parameter-shift rule [137, 166]:*

$$\partial_i \langle O_a \rangle_{s,\boldsymbol{\theta}} = \frac{1}{2}\big(\langle O_a \rangle_{s,\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e_i}} - \langle O_a \rangle_{s,\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e_i}} \big), \tag{5.4}$$

i.e., using the difference of two expectation values $\langle O_a \rangle_{s,\boldsymbol{\theta}'}$ *with a single angle shifted by* $\pm\frac{\pi}{2}$*.*
For a RAW-PQC *policy* $\pi_{\boldsymbol{\theta}}$*, we have instead:*

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) = \nabla_{\boldsymbol{\theta}} \langle P_a \rangle_{s,\boldsymbol{\theta}} \,/\, \langle P_a \rangle_{s,\boldsymbol{\theta}} \tag{5.5}$$

[1] Note that the parameters $\boldsymbol{\lambda}$ do not act as rotation angles. To compute the derivatives $\partial_{\lambda_{i,j}} \langle O_a \rangle_{s,\boldsymbol{\theta}}$, one should compute derivatives w.r.t. $s_j \lambda_{i,j}$ instead and apply the chain rule: $\partial_{\lambda_{i,j}} \langle O_a \rangle_{s,\boldsymbol{\theta}} = s_j \partial_{s_j \lambda_{i,j}} \langle O_a \rangle_{s,\boldsymbol{\theta}}$.

Algorithm 1: REINFORCE with PQC policies and value-function baselines

Input: a PQC policy $\pi_{\boldsymbol{\theta}}$ from Def. 16; a value-function approximator $\widetilde{V}_{\boldsymbol{\omega}}$

Initialize parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$;

while *True* **do**

  Generate N episodes $\{(s_0, a_0, r_1, \ldots, s_{H-1}, a_{H-1}, r_H)\}_i$ following $\pi_{\boldsymbol{\theta}}$;

  for *episode i in batch* **do**

    Compute the returns $G_{i,t} \leftarrow \sum_{t'=1}^{H-t} \gamma^{t'} r_{t+t'}^{(i)}$;

    Compute the gradients $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)}|s_t^{(i)})$ using Lemma 18;

  Fit $\left\{\widetilde{V}_{\boldsymbol{\omega}}(s_t^{(i)})\right\}_{i,t}$ to the returns $\{G_{i,t}\}_{i,t}$;

  Compute

$$\Delta\boldsymbol{\theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)}|s_t^{(i)}) \left(G_{i,t} - \widetilde{V}_{\boldsymbol{\omega}}(s_t^{(i)})\right);$$

  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}$;

where the partial derivatives $\partial_{\phi_i}\langle P_a\rangle_{s,\boldsymbol{\theta}}$ and $\partial_{\lambda_i}\langle P_a\rangle_{s,\boldsymbol{\theta}}$ can be estimated similarly to above.

In some of our environments, we additionally rely on a linear value-function baseline to reduce the variance of the Monte Carlo estimates [91]. We choose it to be identical to that of Ref. [71].

5.1.3 Efficient policy sampling and policy-gradient evaluation

A natural consideration when it comes to the implementation of our PQC policies is whether one can efficiently (in the number of executions of the PQC on a quantum computer) sample and train them.

By design, sampling from our RAW-PQC policies can be done with a single execution (and measurement) of the PQC: the projective measurement corresponding to the observable $O = \sum_a aP_a$ naturally samples a basis state associated to action a with probability $\langle P_a\rangle_{s,\boldsymbol{\theta}}$. However, as Eq. (5.5) indicates, in order to train these policies using REINFORCE, one is nonetheless required to estimate the expectation values $\langle P_a\rangle_{s,\boldsymbol{\theta}}$, along with the gradients $\nabla_{\boldsymbol{\theta}}\langle P_a\rangle_{s,\boldsymbol{\theta}}$. Fortunately, these quantities can be estimated efficiently up to some additive error ε, using only $\mathcal{O}(\varepsilon^{-2})$ repeated executions and measurements on a quantum computer.

In the case of our SOFTMAX-PQC policies, it is less clear whether similar noisy estimates $\widetilde{\langle O_a\rangle}_{s,\boldsymbol{\theta}}$ of the expectation values $\langle O_a\rangle_{s,\boldsymbol{\theta}}$ are sufficient to evaluate policies of the form of Eq. (5.2). We show however that, using these noisy estimates, we can compute a policy $\widetilde{\pi}_{\boldsymbol{\theta}}$ that produces samples close to that of the true policy $\pi_{\boldsymbol{\theta}}$. We state our result formally in the following lemma, proven in Appendix C.2.

Lemma 19. *For a* SOFTMAX-PQC *policy* $\pi_{\boldsymbol{\theta}}$ *defined by a unitary* $U(s, \boldsymbol{\theta})$ *and observables* O_a*, call* $\langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}}$ *approximations of the true expectation values* $\langle O_a \rangle_{s,\boldsymbol{\theta}}$ *with at most* ε *additive error. Then the approximate policy* $\widetilde{\pi}_{\boldsymbol{\theta}} = \mathrm{softmax}_\beta(\langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}})$ *has total variation distance* $\mathcal{O}(\beta\varepsilon)$ *to* $\pi_{\boldsymbol{\theta}} = \mathrm{softmax}_\beta(\langle O_a \rangle_{s,\boldsymbol{\theta}})$*. Since expectation values can be efficiently estimated to additive error on a quantum computer, this implies efficient approximate sampling from* $\pi_{\boldsymbol{\theta}}$*.*

We also obtain a similar result for the log-policy gradient of SOFTMAX-PQCs (see Lemma 18), that we show can be efficiently estimated to additive error in ℓ_∞-norm (see Appendix C.2 for a proof).

5.2 Performance comparison in benchmarking environments

In the previous section, we have introduced our quantum policies and described several of our design choices. We defined the RAW-PQC and SOFTMAX-PQC models and introduced two original features for PQCs: trainable observables at their output and trainable scaling parameters for their input. In this section, we evaluate the influence of these design choices on learning performance through numerical simulations. We consider three classical benchmarking environments from the OpenAI Gym library [43]: CartPole, MountainCar and Acrobot. All three have continuous state spaces and discrete action spaces (see Appendix C.4 for their specifications). Moreover, simple NN-policies, as well as simple closed-form policies, are known to perform very well in these environments [151], which makes them an excellent test-bed to benchmark PQC policies.

5.2.1 RAW-PQC v.s. SOFTMAX-PQC

In our first set of experiments, presented in Fig. 5.2, we evaluate the general performance of our proposed policies. The aim of these experiments is twofold: first, to showcase that quantum policies based on shallow PQCs and acting on very few qubits can be trained to good performance in our selected environments; second, to test the advantage of SOFTMAX-PQC policies over RAW-PQC policies that we conjectured in the Sec. 5.1.1. To assess these claims, we take a similar approach for each of our benchmarking environments, in which we evaluate the average learning performance of 20 RAW-PQC and 20 SOFTMAX-PQC agents. Apart from the PQC depth, the shared hyperparameters of these two models were jointly picked as to give the best overall performance for both; the hyperparameters specific to each model were optimized independently. As for the PQC depth D_{enc}, the latter was chosen as the minimum depth for which near-optimal performance was observed for either model. The simulation results confirm both our hypotheses: quantum policies can achieve good performance on the three benchmarking tasks that we consider, and we can see a clear separation between the performance of SOFTMAX-PQC and RAW-PQC agents.

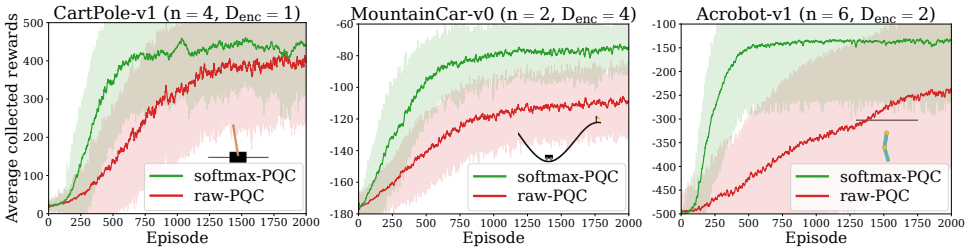


Figure 5.2: **Numerical evidence of the advantage of SOFTMAX-PQC over RAW-PQC in benchmarking environments.** The learning curves (20 agents per curve) of randomly-initialized SOFTMAX-PQC agents (green curves) and RAW-PQC agents (red curves) in OpenAI Gym environments: CartPole-v1, MountainCar-v0, and Acrobot-v1. Each curve is temporally averaged with a time window of 10 episodes. All agents have been trained using the REINFORCE algorithm (see Alg. 1), with value-function baselines for the MountainCar and Acrobot environments.

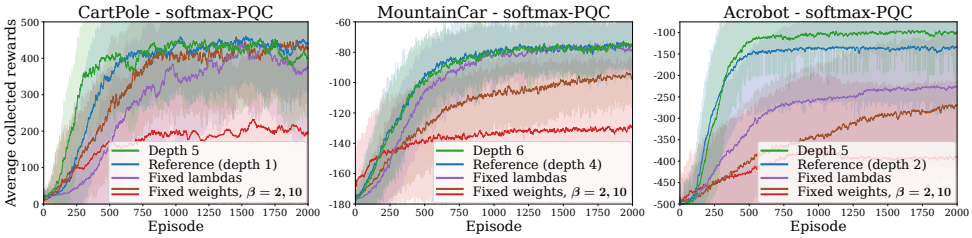


Figure 5.3: **Influence of the model architecture for SOFTMAX-PQC agents.** The blue curves in each plot correspond to the learning curves from Fig. 5.2 and are taken as a reference. Other curves highlight the influence of individual hyperparameters. For RAW-PQC agents, see Appendix C.5.

5.2.2 Influence of architectural choices

The results of the previous subsection however do not indicate whether other design choices we have made in Sec. 5.1.1 had an influence on the performance of our quantum agents. To address this, we run a second set of experiments, presented in Fig. 5.3. In these simulations, we evaluate the average performance of our SOFTMAX-PQC agents after modifying one of three design choices: we either increment the depth of the PQC (until no significant increase in performance is observed), fix the input-scaling parameters $\boldsymbol{\lambda}$ to $\mathbf{1}$, or fix the observable weights $\boldsymbol{w}$ to $\mathbf{1}$. By comparing the performance of these agents with that of the agents from Fig. 5.2, we can make the following observations:

- **Influence of depth:** Increasing the depth of the PQC generally improves (not strictly) the performance of the agents. Note that the maximum depth we tested was $D_{\text{enc}} = 10$.
- **Influence of scaling parameters $\boldsymbol{\lambda}$:** We observe that training these scaling parameters in general benefits the learning performance of our PQC policies, likely due to their increased expressivity.
- **Influence of trainable observable weights $\boldsymbol{w}$:** our final consideration relates to the importance of having a policy with "trainable greediness" in RL scenarios. For this, we consider SOFTMAX-PQC agents with fixed observables βO_a throughout training. We observe that this has the general effect of decreasing the performance and/or the speed of convergence of the agents. We also see that policies with fixed high β (or equivalently, a large observable norm $\beta\|O_a\|$) tend to have a poor learning performance, likely due to their lack of exploration in the RL environments.

Finally, note that all the numerical simulations performed here did not include any source of noise in the PQC evaluations. It would be an interesting research direction to assess the influence of (simulated or hardware-induced) noise on the learning performance of PQC agents.

5.3 Quantum advantage of PQC agents in RL environments

The proof-of-concept experiments of the previous section show that our PQC agents can learn in basic classical environments, where they achieve comparable performance to standard DNN policies. This observation naturally raises the question of whether there exist RL environments where PQC policies can provide a learning advantage over standard classical policies. In this section, we answer this question in the affirmative by constructing: a) environments with a provable separation in learning performance between quantum and any classical (polynomial-time) learners, and b) environments where our PQC policies of Sec. 5.1 show an empirical learning advantage over standard DNN policies.

5.3.1 Quantum advantage of PQC agents over classical agents

In this subsection, we construct RL environments with theoretical guarantees of separation between quantum and classical learning agents. These constructions are predominantly based on the recent work of Liu *et al.* [128], which defines a classification task out of the discrete logarithm problem (DLP), i.e., the problem solved in the seminal work of Shor [177]. In broad strokes, this task can be viewed as an encryption of an easy-to-learn problem. For an "un-encrypted" version, one defines a labeling f_s of integers between 0 and $p-2$ (for a large prime p), where the integers are labeled positively if and only if they lie in the segment $[s, s+(p-3)/2]$ (mod $p-1$). Since this labeling is linearly separable, the concept class $\{f_s\}_s$ is then easy to learn. To make it hard, the input integers x (now between 1 and $p-1$) are first encrypted using modular exponentiation, i.e., the secure operation performed in the Diffie–Hellman key exchange protocol. In the encrypted problem, the logarithm of the input integer $\log_g(x)$ (for a generator g of $\mathbb{Z}_p^*$, see Appendix C.6) hence determines the label of x. Without the ability to decrypt by solving DLP, which is widely believed to be classically intractable, the numbers appear randomly labeled. Moreover, Liu *et al.* show that achieving non-trivial labeling accuracy $1/2 + 1/\text{poly}(n)$ (for $n = \log(p)$, i.e., slightly better than random guessing) with a classical polynomial-time algorithm using $\text{poly}(n)$ examples would lead to an efficient classical algorithm that solves DLP [128]. In contrast, the same authors construct a family of quantum learners based on Shor's algorithm, that can achieve a labeling accuracy larger than 0.99 with high probability.

SL-DLP Our objective is to show that analogous separations between classical and quantum learners can be established for RL environments, in terms of their attainable value functions. We start by pointing out that supervised learning (SL) tasks (and so the classification problem of Liu *et al.*) can be trivially embedded into RL environments [72]: for a given concept f_s, the states x are datapoints, an action a is an agent's guess on the label of x, an immediate reward specifies if it was correct (i.e., $f_s(x) = a$), and subsequent states are chosen uniformly at random. In such settings, the value function is trivially related to the testing accuracy of the SL problem, yielding a direct reduction of the separation result of Liu *et al.* [128] to an RL setting. We call this family of environments SL-DLP.

Cliffwalk-DLP In the SL-DLP construction, we made the environment fully random in order to simulate the process of obtaining i.i.d. samples in an SL setting. It is an interesting question whether similar results can be obtained for environments that are less random, and endowed with temporal structure, which is characteristic of RL. In our second family of environments (Cliffwalk-DLP), we supplement the SL-DLP construction with next-state transitions inspired by the textbook "cliff walking" environment of Sutton & Barto [183]: all states are ordered in a chain and some actions of the agent can lead to immediate episode termination. We keep however stochasticity in the environment by allowing next states to be uniformly sampled, with a certain probability δ (common in RL to ensure that an agent is not simply memorizing a correct sequence of actions). This allows us to show that, as long as

sufficient randomness is provided, we still have a simple classical-quantum separation.

Deterministic-DLP In the two families constructed above, each environment instance provided the randomness needed for a reduction from the SL problem. This brings us to the question of whether separations are also possible for fully deterministic environments. In this case, it is clear that for any given environment, there exists an efficient classical agent which performs perfectly over any polynomial horizon (a lookup-table will do). However, we show in our third family of environments (Deterministic-DLP) that a separation can still be attained by moving the randomness to the choice of the environment itself: assuming an efficient classical agent is successful in most of exponentially-many randomly generated (but otherwise deterministic) environments, implies the existence of a classical efficient algorithm for DLP.

We summarize our results in the following theorem, detailed and proven in Appendices C.7 through C.9.

Theorem 20. *There exist families of reinforcement learning environments which are: i) fully random (i.e., subsequent states are independent from the previous state and action); ii) partially random (i.e., the previous moves determine subsequent states, except with a probability δ at least* 0.86 *where they are chosen uniformly at random), and iii) fully deterministic; such that there exists a separation in the value functions achievable by a given quantum polynomial-time agent and any classical polynomial-time agent. Specifically, the value of the initial state for the quantum agent $V_q(s_0)$ is $\varepsilon-$close to the optimal value function (for a chosen ε, and with probability above 2/3). Further, if there exists a classical efficient learning agent that achieves a value $V_c(s_0)$ better than $V_{\text{rand}}(s_0)+\varepsilon'$ (for a chosen ε', and with probability above 0.845), then there exists a classical efficient algorithm to solve DLP. Finally, we have $V_q(s_0) - V_c(s_0)$ larger than some constant, which depends on the details of the environment.*

The remaining point we need to address here is that the learning agents obtained from the methods of Liu *et al.* do not rely on PQCs but rather support vector machines (SVMs) based on quantum kernels [104, 169]. Nonetheless, using a connection between these quantum SVMs and PQCs [169], we construct PQC policies which are as powerful in solving the DLP environments as the agents obtained from the methods of Liu *et al.* (even under similar noise considerations). We state our result in the following informal theorem, that we re-state formally, along with the details of our construction in Appendices C.10 and C.11.

Theorem 21 (informal version)**.** *Using a training set of size polynomial in $n = \log(p)$ and a number of (noisy) quantum circuit evaluations also polynomial in n, we can train a PQC classifier on the DLP task of Liu* et al. *of size n that achieves a testing accuracy arbitrarily close to optimal, with high probability. This PQC classifier can in turn be used to construct close-to-optimal quantum agents in our DLP environments, as prescribed by Theorem 20.*

5.3.2 Quantum advantage of PQC agents over DNN agents

While the DLP environments establish a proof of the learning advantage PQC policies can have in theory, these environments remain extremely contrived and artificial.

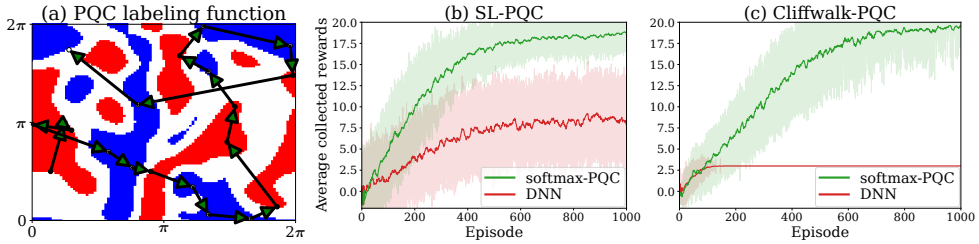


Figure 5.4: **Numerical evidence of the advantage of PQC policies over DNN policies in PQC-generated environments.** (a) Labeling function and training data used for both RL environments. The data labels (red for $+1$ label and blue for -1 label) are generated using a RAW-PQC of depth $D_{\text{enc}} = 4$ with a margin $\Delta = 0.3$ (white areas). The training samples are uniformly sampled from the blue and red regions, and arrows indicate the rewarded path of the cliffwalk environment. (b) and (c) The learning curves (20 agents per curve) of randomly-initialized SOFTMAX-PQC agents and DNN agents in RL environments where input states are (b) uniformly sampled from the dataset and (c) follow cliffwalk dynamics. Each curve is temporally averaged with a time window of 10 episodes.

They are based on algebraic properties that agents must explicitly decrypt in order to perform well. Instead, we would like to consider environments that are less tailored to a specific decryption function, which would allow more general agents to learn. To do this, we take inspiration from the work of Havlíček *et al.* [104], who, in order to test their PQC classifiers, define a learning task generated by similar quantum circuits.

PQC-generated environments

We generate our RL environments out of random RAW-PQCs. To do so, we start by uniformly sampling a RAW-PQC that uses the alternating-layer architecture of Fig. 5.1 for $n = 2$ qubits and depth $D_{\text{enc}} = 4$. We use this RAW-PQC to generate a labeling function $f(s)$ by assigning a label $+1$ to the datapoints s in $[0, 2\pi]^2$ for which $\langle ZZ \rangle_{s,\boldsymbol{\theta}} \geq 0$ and a label -1 otherwise. We create a dataset S of 10 datapoints per label by uniformly sampling points in $[0, 2\pi]^2$ for which $|\langle ZZ \rangle_{s,\boldsymbol{\theta}}| \geq \frac{\Delta}{2} = 0.15$. This dataset allows us to define two RL environments, similar to the SL-DLP and Cliffwalk-DLP environments of Sec. 5.3.1:

- **SL-PQC:** this degenerate RL environment encodes a classification task in an episodic RL environment: at each interaction step of a 20-step episode, a sample state s is uniformly sampled from the dataset S, the agent assigns a label $a = \pm 1$ to it and receives a reward $\delta_{f(s),a} = \pm 1$.
- **Cliffwalk-PQC:** this environment essentially adds a temporal structure to SL-PQC: each episode starts from a fixed state $s_0 \in S$, and if an agent assigns the correct label to a state s_i, $0 \leq i \leq 19$, it moves to a fixed state s_{i+1} and receives a

+1 reward, otherwise the episode is instantly terminated and the agent gets a −1 reward. Reaching s_{20} also causes termination.

Performance comparison

Having defined our PQC-generated environments, we now evaluate the performance of SOFTMAX-PQC and DNN policies in these tasks. The particular models we consider are SOFTMAX-PQCs with PQCs sampled from the same family as that of the RAW-PQCs generating the environments (but with re-initialized parameters $\boldsymbol{\theta}$), and DNNs using Rectified Linear Units (ReLUs) in their hidden layers. In our hyperparameter search, we evaluated the performance of DNNs with a wide range of depths (number of hidden layers between 2 to 10) and widths (number of units per hidden layer between 8 and 64), and kept the architecture with the best average performance (depth 4, width 16).

Despite this hyperparametrization, we find (see Fig. 5.4, and Fig. C.4 in Appendix C.5 for different environment instances) that the performance of DNN policies on these tasks remains limited compared to that of SOFTMAX-PQCs, that learn close-to-optimal policies on both tasks. Moreover, we observe that the separation in performance gets boosted by the cliffwalk temporal structure. This is likely do to the increased complexity of this task, as, in order to move farther in the cliffwalk, the policy family should allow learning new labels without "forgetting" the labels of earlier states. In these particular case studies, the SOFTMAX-PQC policies exhibited sufficient flexibility in this sense, whereas the DNNs we considered did not (see Appendix C.5 for a visualization of these policies). Note that these results do not reflect the difficulty of our tasks at the sizes we consider (a look-up table would perform optimally) but rather highlight the inefficacy of these DNNs at learning PQC functions.

Chapter 6

Exponential separations between classical and quantum learners

In this chapter we address the challenge of finding learning settings where quantum learners can achieve a provable exponential speedup over classical learners in the efficient probably approximately correct (PAC) framework.

First, in Section 6.1, we discuss known learning separations that rely on efficient data generation [126, 173], and we provide a fine-grained analysis of where the classical hardness of learning stems from. While discussing these learning separations, we find that the ones available in literature largely rely on the classical hardness of *evaluating* the function generating the data on unseen points, as opposed to the hardness of *identifying* it. We elaborate how the identification problem can be what is needed in practice, and we address this gap by proving two new learning separations where the classical hardness primarily lies in identifying the function generating the data (see Theorems 24 and 25).

Afterwards, in Section 6.2, we show how leveraging stronger complexity-theoretic assumptions can lead to learning separations where the data is generated by a genuine quantum process. Our main contribution is Theorem 26, which outlines a method of establishing learning separations from BQP-complete functions. We also provide two lemmas, Lemmas 27 and 28, which introduce natural assumptions under which the criteria in Theorem 26 are satisfied. Finally, in Section 6.2.1, we show how Theorem 26 can be used to build learning separations from problems in quantum many-body physics.

To connect our work to some of the related results in the field [107, 109, 146, 97], we discuss selected topics related to learning separations with classical data. In Section 6.3.1 we discuss the milestone work of Huang et al. [107] and how their classical machine learning methods based on the classical shadow framework relate to learning separations with quantum-generated data (i.e., those from Theorem 26). In particular, we highlight their limitations by constructing a family of Hamiltonians whose ground

state properties cannot be predicted based on cryptographic assumptions (see Theorem 30). In Section 6.3.2 we discuss a specific example (i.e., evaluating parameterized quantum circuits) that exemplifies how access to data radically enhances what is efficiently evaluated classically. In Section 6.3.3 we discuss how two physically-motivated problems (i.e., Hamiltonian learning, and identifying order parameters and phases of matter) naturally fit in a learning setting where the learner is constrained to output a hypothesis from a fixed hypothesis class.

6.1 Learning separations with efficient data generation

A commonality between the learning separations of [126, 173] is that the proof of classical non-learnability relies on the fact that the examples can be efficiently generated classically (i.e., the example oracle can be efficiently simulated classically)[1]. This is crucial, since it ensures that access to the example oracle does not enhance what a classical learner can evaluate relative to a conventional (non-learning) classical algorithm. This then allows one to directly deduce classical non-learnability from a complexity-theoretic hardness assumption related to the concepts, since the existence of an efficient classical learner would imply the existence of an efficient classical algorithm. A similar observation was made by the authors of [156] (which came out after our initial observation [94]), where they also study the problem of distribution-independent learning separations.

In this section we study the learning separations of [126, 173], and we characterize them with respect to the type of learning separation they achieve (as discussed in Section 2.5.1), and the kind of hardness assumptions they leverage to obtain classical non-learnability (as discussed in Section 2.5.2). Firstly, in Section 6.1.1, we discuss the CC/QQ separation of the discrete logarithm concept class of [126], whose concepts are believed to be classically intractable, no matter how they are specified. Secondly, in Section 6.1.2, we discuss the CC/QC separation of the cube root concept class of [116, 173], whose concepts are specified in a way that makes them classically intractable, though when specified in a different way they become classically efficient (i.e., the concepts are "obfuscated" versions of classically efficient functions).

While discussing the learning separations of [126, 173], we notice that their proofs largely rely on the classical difficulty of *evaluating* the hypotheses on unseen examples, rather than the difficulty of *identifying* a hypothesis that is close to the concept generating the examples. To complement these works, we present two new examples of learning separations where the classical hardness lies in *identifying* the concept that is generating the examples. Specifically, in Section 6.1.3, we provide an example of a CC/QC separation (contingent on a plausible though relatively unexplored hardness assumption) where the concepts are classically efficiently evaluatable, making it impossible for the classical hardness to come from evaluating them on unseen examples. Afterwards, in Section 6.1.4, we provide an example of a separation in the setting where the learner is constrained to output hypotheses from a fixed hypothesis

[1] The notion of efficiently generatable examples is closely related to the notion of *random verifiability* [25].

class, in which case the learner is only required to identify the concept generating the examples, therefore also eliminating the possibility that the classical hardness comes from evaluating them on unseen examples[2].

6.1.1 A learning separation based on a worst-case to average-case reduction

In this section we discuss the discrete logarithm concept class studied in [126]. In this work, the authors prove that the *discrete logarithm concept class* defined below exhibits a CC/QQ separation.

Definition 17 (Discrete logarithm concept class [126])**.** *Fix an n-bit prime number p and a generator a of $\mathbb{Z}_p^*$ (i.e., the multiplicative group of integers modulo p). We define the* discrete logarithm concept class *as $\mathcal{C}_n^{\mathrm{DL}} = \{c_i\}_{i\in\mathbb{Z}_p^*}$, where*

$$c_i(x) = \begin{cases} 1, & \text{if } \log_a x \in [i, i + \frac{p-3}{2}], \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

Remark(s). *Here $\log_a x$ denotes the discrete logarithm of x with respect to the generator a. That is, the discrete logarithm $\log_a x$ is the smallest positive integer ℓ such that $a^\ell \equiv x \mod p$.*

To see why the examples are efficiently generatable for the discrete logarithm class, first note that the examples are of the form

$$(x, c_i(x)) = (a^y, f_i(y)), \tag{6.2}$$

where $y \in \{1, \ldots, p-1\}$ is the unique integer such that $x \equiv a^y \mod p$, and we let

$$f_i(y) = \begin{cases} 1, & \text{if } y \in [i, i + \frac{p-3}{2}], \\ 0, & \text{otherwise.} \end{cases} \tag{6.3}$$

Secondly, note that $y \mapsto a^y \mod p$ is a bijection from $\{1, \ldots, p-1\}$ to $\mathbb{Z}_p^*$, which implies that sampling $x \in \mathbb{Z}_p^*$ uniformly at random is equivalent to sampling $y \in \{0, \ldots, p-1\}$ uniformly at random and computing $x = a^y \mod p$. By combining this observation with Eq. (6.2), one finds that one can efficiently generate examples of the discrete logarithm concept c_i under the uniform distribution over $\mathbb{Z}_p^*$ by sampling $y \in \{1, \ldots, p-1\}$ uniformly at random, and computing $(a^y, f_i(y))$.

The hardness assumption that one can leverage to obtain classical non-learnability is that the discrete logarithm is classically intractable (i.e., not in BPP)[3]. Namely,

[2]We remark that the concept class of Section 6.1.3 also exhibits a separation in the setting the learner is constrained to output a hypothesis from a fixed hypothesis class. However, we choose to present it as a CC/QC separation to highlight that such separations are still possible if the concepts are classically efficiently evaluatable. Moreover, we still include the separation in the setting the learner is constrained to output a hypothesis from a fixed hypothesis class of Section 6.1.4, because it is not contingent on a relatively unexplored hardness assumption.

[3]For some sequence of primes $\{p_n\}_{n\in\mathbb{N}}$, where $|p_n| = n$ and given n one can efficiently construct p_n.

in [36] it is shown that computing the most-significant bit of the discrete logarithm on any $\frac{1}{2} + \frac{1}{\text{poly}(n)}$ fraction of inputs is at least as hard as computing the discrete logarithm on all inputs. Using the terminology of Section 2.5.2, if one assumes that the discrete logarithm is classically intractable (i.e., not in BPP), then the concepts $c_0 \in \mathcal{C}_n^{\text{DL}}$ lie outside of $\mathsf{HeurBPP}$. In conclusion, to obtain the classical non-learnability it is sufficient to assume that the discrete logarithm is classically intractable.

We remark that one could obtain a similar learning separation for the singleton concept class $\mathcal{C}'_n = \{c\}$ for any choice of $c \in \mathcal{C}_n^{\text{DL}}$. This singleton concept class is quantum learnable without requiring use of the example oracle (i.e. without requiring any data), and one could thus argue that it is not a genuine learning problem anymore (i.e., similar to Observation 1).

Having discussed classical non-learnability, one still needs to ensure quantum learnability. To this end, the authors of [126] show that a general-purpose quantum learning algorithm (i.e., a quantum kernel method) can efficiently learn the discrete logarithm concept class under the uniform distribution. We summarize the result of [126] discussed in this section in the following theorem.

Theorem 22 ([126]). *$L_{\text{DLP}} = \left(\{\mathcal{C}_n^{\text{DLP}}\}_{n\in\mathbb{N}}, \{\mathcal{D}_n^U\}_{n\in\mathbb{N}}\right)$ exhibits a $\mathsf{CC/QQ}$ separation, where $\mathcal{D}_n^U$ denotes the uniform distribution over $\mathbb{Z}_p^*$.*

The hypothesis class the authors of [126] use is quantumly evaluatable, and to the best of our knowledge it is unknown whether the discrete logarithm concepts are quantumly learnable using a classically evaluatable hypothesis class (which would imply a $\mathsf{CC/QC}$ separation).

6.1.2 A learning separation based on obfuscation

The cube root concept class was first studied in [116], and the fact that this concept class exhibits a $\mathsf{CC/QC}$ separation was first observed in [173], albeit using different terminology. We note that there exist many similar concept classes based on public-key cryptosystems such as the RSA cryptosystem that exhibit $\mathsf{CC/QC}$ separations (see [115]). Recall that for $\mathsf{CC/QC}$ separations the hypothesis class has to be classically evaluatable, so the role of the quantum computer is only to identify which hypothesis is close to the concept that is generating the examples.

Definition 18 (Cube root concept class [116]). *Fix an n-bit integer $N = pq$[4], where p and q are two $\lfloor n/2 \rfloor$-bit primes such that $\gcd\big(3, (p-1)(q-1)\big) = 1$. We define the* cube root concept class *as $\mathcal{C}_n^{\text{DCR}} = \{c_i\}_{i\in[n]}$, with*

$$c_i(x) = \text{bin}(f_N^{-1}(x), i),$$

where $\text{bin}(y, i)$ denotes the ith bit of the binary representation of y, and the function f_N^{-1} is the inverse of $f_N(x) = x^3 \mod N$ defined on $\mathbb{Z}_N^*$ (i.e., the multiplicative group of integers modulo N).

Remark(s). *By requiring that p and q satisfy $\gcd\big(3, (p-1)(q-1)\big) = 1$, we ensure that f_N^{-1} exists.*

[4]Throughout this chapter, the integer N is known to the learner beforehand but p and q are not.

To see why the examples are efficiently generatable for the cube root concept class, first note that the examples are of the form

$$(x, c_i(x)) = \left(y^3, \mathrm{bin}(y, i)\right), \tag{6.4}$$

where $y \in \mathbb{Z}_N^*$ is the unique element such that $x \equiv y^3 \mod N$. Secondly, note that $f_N(x) = x^3 \mod N$ is a bijection from $\mathbb{Z}_N^*$ to itself, which implies that sampling $x \in \mathbb{Z}_N^*$ uniformly at random is equivalent to sampling $y \in \mathbb{Z}_N^*$ uniformly at random and computing $x = y^3 \mod N$. By combining this observation with Eq. (6.4), one finds that one can efficiently generate examples of the cube root concept c_i under the uniform distribution over $\mathbb{Z}_N^*$ by first sampling $y \in \mathbb{Z}_N^*$ uniformly at random, and then computing $y^3 \mod N$ together with the ith bit of the binary representation of y.

The hardness assumption that one can leverage to obtain classical non-learnability is what we will call the Discrete Cube Root Assumption (DCRA), which states that computing f_N^{-1} is classically intractable (i.e., not in BPP)[5]. Namely, in [17, 87] it is shown that computing the least-significant bit of f_N^{-1} on any $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ fraction of inputs is at least as hard as computing entire binary representation of $f_N^{-1}(x)$ on all inputs. Using the terminology of Section 2.5.2, if one assumes that computing f_N^{-1} is classically intractable (i.e., not in BPP), then the concepts $c_n \in \mathcal{C}_n^{\mathrm{DCR}}$ lie outside of $\mathsf{HeurBPP}$. In conclusion, analogous to the discrete logarithm concept class, to obtain the classical non-learnability it is sufficient to assume that computing f_N^{-1} is classically intractable.

We remark that also in this case one could obtain a similar learning separation for the singleton concept class $\mathcal{C}'_n = \{c_n\}$ for the concept $c_n \in \mathcal{C}_n^{\mathrm{DCR}}$. This singleton concept class is quantum learnable without requiring the example oracle (i.e. without requiring any data), and one could thus argue that it is not a genuine learning problem anymore (i.e., similar to Observation 1).

However, there is a significant distinction between the learning separations for the discrete logarithm concept class and the cube root concept class that is worth highlighting: the latter is quantumly learnable using a *classically* evaluatable hypothesis class. To see why this is the case, it is important to note that f_N^{-1} is of the form

$$f_N^{-1}(y) = y^{d^*} \mod N, \tag{6.5}$$

for some d^* that only depends on N[6]. The function f_N^{-1} is a type of "trap-door function" in that if one is also given d^*, then computing f_N^{-1} suddenly becomes classically tractable. In other words, there exist polynomially-sized Boolean circuits which evaluate this function, whereas for the discrete logarithm we do not know whether such circuits exist. In this example we thus see the relevance of how the concepts are specified. The specifications "f_N^{-1} where $f(x) = x^3$" and "$f_N^{-1} = x^{d^*}$" refer to the same functions, yet computing them is in one case classically tractable, and in the other case it is classically intractable (under the DCRA). The ideas of concealing (easy) functions in difficult descriptions is reminiscent of the term "obfuscation" in

[5]For some sequence of moduli $\{N_i\}_{i\in\mathbb{N}}$, where $|N_i| = i$ and given i one can efficiently construct N_i.

[6]In cryptographic terms, d^* is the private decryption key corresponding to the public encryption key $e = 3$ and public modulus N in the RSA cryptosystem.

computer science, and we will use this term in this context as well. Specifically, we say that the specification "f_N^{-1} where $f(x) = x^3$" is an obfuscation of the specification "$f_N^{-1} = x^{d^*}$". Using the terminology of Section 2.5.2, this establishes that the problem of evaluating the concepts actually lies inside $\mathsf{P/poly}$ (where the advice string – i.e., d^* – is used to "de-obfuscate" the function).

With regards to quantum learnability, in [173] the authors note that using Shor's algorithm a quantum learning algorithm can efficiently compute d^* following the standard attack on the RSA cryptosystem. The cube root concept class is thus quantumly learnable using the classically evaluatable hypothesis class

$$\left\{ f_{d,i}(x) = \mathrm{bin}(x^d \bmod N, i) \;\middle|\; d \in [N],\ i \in [n] \right\}, \tag{6.6}$$

Another feature of the cube root concept class which warrants a comment is that even though computing d^* does not require access to the example oracle (recall that N is known beforehand), we still have to learn the bit of x^{d^*} that is generating the examples, which does require access to the example oracle (i.e., it requires data). We summarize the results regarding the separation of the cube root concept class in the following theorem.

Theorem 23 ([173, 116])**.** *$L_{\mathrm{DCR}} = \left(\{\mathcal{C}_n^{\mathrm{DCR}}\}_{n\in\mathbb{N}}, \{\mathcal{D}_n^U\}_{n\in\mathbb{N}}\right)$ exhibits a $\mathsf{CC/QC}$ separation, where $\mathcal{D}_n^U$ denotes the uniform distribution over $\mathbb{Z}_N^*$.*

6.1.3 A learning separation with efficiently evaluatable concepts

In this section we establish a learning separation (contingent on a plausible though relatively unexplored hardness assumption) where the concepts do not just admit polynomial-sized Boolean circuits, but are also given in a representation which is efficiently evaluatable on a classical computer. For this concept class, the hardness of learning then cannot stem from the hardness of *evaluating* the concepts, and it thus lies in *identifying* which specific concept is generating the examples. To the best of our knowledge, no such separation was given in the literature before. The concept class that satisfies all of the above is the modular exponentiation concept class defined as follows.

Definition 19 (Modular exponentiation concept class)**.** *Let $N = pq$ be an n-bit 2^c-integer as defined in Definition 20 with* $\gcd\left(3, (p-1)(q-1)\right) = 1$*. We define the* modular exponentiation concept class *as*

$$\mathcal{C}_n^{\mathrm{modexp}} = \left\{ c_d \;\middle|\; d = 1, \ldots, (p-1)(q-1) \text{ and } \gcd(d, (p-1)(q-1) = 1 \right\},$$

where

$$c_d : \mathbb{Z}_N^* \to \mathbb{Z}_N^*, \quad c_d(x) = x^d \mod N, \tag{6.7}$$

and $\mathbb{Z}_N^*$ denotes the multiplicative group of integers modulo N.

Remark(s). *The concepts are not binary-valued, and it is an open question whether and how the separation can be translated to also hold for binary-valued concepts.*

Definition 20 (2^c-integer)**.** *An n-bit integer $N = pq$ is a 2^c-integer if p and q are two $\lfloor n/2 \rfloor$-bit primes such that:*

(i) There exists a constant c (i.e., independent of n) such that $2^c \nmid (p-1)(q-1)$.

(ii) There exists a constant c' (i.e., independent of n) such that $\gcd(p-1, q-1) = 2^{c'}$.

We show that the above concept class is not classically learnable under the assumption that computing f_N^{-1} is classically intractable (i.e., not in BPP) when we restrict N to be a 2^c integer. We will refer to this assumption as the 2^c-discrete cube root assumption (2^c-DCRA). First, note that the modular exponentiation concept class contains the cube root function f_N^{-1} discussed in the previous section (though this time it is not "obfuscated"). Moreover, using the construction from the previous section we can efficiently generate examples $(y, f_N^{-1}(y))$, for $y \in \mathbb{Z}_N^*$ uniformly at random. If we put these examples into an efficient classical learning algorithm for the modular exponentiation concept class, it would with high probability identify a classically efficiently evaluatable hypothesis that agrees with f_N^{-1} on a $1 - \frac{1}{\mathrm{poly}(n)}$ fraction of inputs. Analogous to the previous section, by the worst-case to average-case reduction of [17, 87] this directly violates the 2^c-DCRA.

We note that by imposing that N is a 2^c-integer might cause the 2^c-DCRA to no longer hold, since there could be an efficient classical algorithm for these specific 2^c-integer moduli. However, since 2^c-integers are generally not considered to be unsecure or "weak" moduli for the RSA cryptosystem, and since recently factored RSA numbers[7] are all essentially 2^c-integers, it is plausible that the 2^c-DCRA still holds (see Appendix D.2.1 for more details).

To show that the modular exponentiation concept class is quantumly learnable, we use a combination of the quantum algorithm for order-finding and the quantum algorithm for the discrete logarithm [176]. The key observation is that an example $(x, x^d \bmod N)$ specifies a congruence relation $d \equiv a \bmod r$, where r denotes the multiplicative order of $x \in \mathbb{Z}_N^*$, and a denotes the discrete logarithm of x^d in the subgroup generated by x (i.e., the smallest positive integer ℓ such that $x^\ell \equiv x^d \bmod N$). Next, using the fact that N is a 2^c-number, we show that a polynomial number of these congruences suffices to recover d with high probability. We summarize the learning separation of the modular exponentiation concepts in Theorem 24, and defer the proof to Appendix D.2.

Theorem 24. *If the 2^c-DCRA holds, then the learning problem*

$$L_{\mathrm{modexp}} = \left(\{\mathcal{C}_n^{\mathrm{modexp}}\}_{n\in\mathbb{N}}, \{\mathcal{D}_n^U\}_{n\in\mathbb{N}}\right)$$

exhibits a $\mathsf{CC/QC}$ separation, where $\mathcal{D}_n^U$ denotes the uniform distribution over $\mathbb{Z}_N^*$.

In conclusion, the modular exponentiation concept class exhibits a $\mathsf{CC/QC}$ separation (assuming the 2^c-DCRA hold), where the concepts are classically efficiently

[7] `https://en.wikipedia.org/wiki/RSA_numbers`

evaluatable. Since the concepts are classically efficiently evaluatable, one could argue that the classical hardness lies in *identifying* rather than *evaluating* a hypothesis that is close to the concept generating the examples. We remark that for the modular exponentation concept class, it is not possible to restrict the concept class and obtain a similar learning separation where a quantum learner does not require any data (i.e., similar to Observation 1). In fact, since the concepts are efficiently evaluatable, any polynomially-sized subset of concepts is classically learnable since one can simply do a brute-force search to find the concept that best matches the data.

In the next section, we present an example of a separation in the setting where the learner is constrained to only output hypotheses from a fixed hypothesis class. Since the learner is not required to evaluate the concepts on unseen examples, it can be argued that in this case the classical hardness also lies in identifying rather than evaluating the concept generating the examples.

6.1.4 A learning separation with a fixed hypothesis class

In this section we establish a separation in the setting where the learner is constrained to only output hypotheses from a fixed hypothesis class. Recall that in this setting the learner is not required to be able to evaluate the concepts, so the hardness of learning must stem from the hardness of identifying the hypothesis that is close to the concept generating the data. The main differences compared to the modular exponentiation concept class are that the concepts discussed in this section are binary-valued and that it is unknown whether they exhibit a separation in the setting where the learner is free to output arbitrary hypotheses. The concept class we discuss in this section is defined below, and it is a modification of the cube root concept class from Definition 18.

Definition 21 (Cube root identification concept class)**.** *Fix an n-bit integer $N = pq$, where p and q are two $\lfloor n/2 \rfloor$-bit primes such that* $\gcd\big(3, (p-1)(q-1)\big) = 1$. *We define the* cube root identification concept class *as $\mathcal{C}_n^{\mathrm{DCRI}} = \{c_m\}_{m \in \mathbb{Z}_N^*}$, with*

$$c_m(x) = \mathrm{bin}(m^3 \bmod N,\ \mathrm{int}(x_1 : \cdots : x_{\lfloor \log n \rfloor})),$$

where $\mathrm{bin}(y, k)$ *denotes the kth bit of the binary representation of y, and* $\mathrm{int}(x_1 : \cdots : x_{\lfloor \log n \rfloor})$ *is the integer encoded by the first $\lfloor \log n \rfloor$-bits of $x \in \{0,1\}^n$.*

We show that the cube root identification concept class is not classically learnable with a fixed hypothesis class under the *Discrete Cube Root Assumption* (DCRA) discussed in Section 6.1.2. To show that the existence of an efficient classical learner violates the DCRA, we let $e \in \mathbb{Z}_N^*$ and show we how an efficient classical learner can efficiently compute $m = f_N^{-1}(e)$. First, we generate examples $(x, \mathrm{bin}(e, k))$, where $k = \mathrm{int}(x_1 : \cdots : x_{\lfloor \log n \rfloor})$. When plugging these examples into an efficient classical learner it will with high probability identify an m' such that $(m')^3 \equiv m \mod N$. Since $x \mapsto x^3 \mod N$ is a bijection on $\mathbb{Z}_N^*$ we find that $m = m'$, and thus conclude that an efficient classical learner can indeed efficiently compute the solution to our discrete cube root instance.

To establish that the cube root identification concept class is quantumly proper learnable, we first note that using $\mathcal{O}(\mathrm{poly}(n))$ examples of a concept c_m under the

uniform distribution we can with high probability reconstruct the full binary representation of m^3. Next, since N is known we can use Shor's algorithm [176] to compute d such that $(m^3)^d \equiv m \bmod N$, which allows us to correctly identify the concept c_m. Note that the quantum learner needs access to the data to obtain a full reconstruction of the binary representation of m^3. We summarize the learning separation of the cube root identification concept class in Theorem 25, and we defer the proof to Appendix D.3.

Theorem 25. *$L_{\mathrm{DCRI}} = \left(\{\mathcal{C}_n^{\mathrm{DCRI}}\}_{n\in\mathbb{N}}, \{\mathcal{D}_n^U\}_{n\in\mathbb{N}}\right)$ exhibits a $\mathsf{C}_{\mathcal{H}}/\mathsf{Q}_{\mathcal{H}}$ separation, where $\mathcal{H} = \mathcal{C}^{\mathrm{DCRI}}$ and $\mathcal{D}_n^U$ denotes the uniform distribution over $\mathbb{Z}_N^*$.*

In conclusion, the cube root identification concept class exhibits a separation in the setting where the learner is constrained to only output hypotheses from a fixed hypothesis class. In fact, this is a separation in the *proper* efficient PAC learning framework, since the hypothesis class is the same as the concept class. Since in this setting it is not required to evaluate the concepts on unseen examples, the classical hardness has to lie in *identifying* rather than *evaluating* the concept generating the examples. We remark that for the cube root identification concept class it is not possible to obtain a similar learning separation for a singleton concept class where a learner does not require any data (see also Observation 1).

6.2 Learning separations without efficient data generation

In the quantum machine learning community there is an often-mentioned conjecture that quantum machine learning is most likely to have its advantages for data that is generated by a "genuine quantum process"[1]. We understand this to mean that the concepts generating the data are BQP-complete or perhaps DQC1-complete. It is worth noting that if concepts in BQP or DQC1 that are not in BPP are already considered a "genuine quantum process", then the discrete logarithm concept class discussed in Section 6.1.1 suffices. However, we aim to investigate learning separations beyond these concepts, i.e., where the concepts are BQP-complete.

A natural question that arises is, given a family of BQP-complete concepts, what additional assumptions are sufficient to prove that these concepts exhibit a learning separation? In Section 6.1, we discussed proofs of learning separations that were predicated on the data being efficiently generatable by a classical device. However, since there is no reason to believe that a family of BQP-complete concepts allow for efficient data generation, we will need to adopt a different proof-strategy.

To ensure quantum learnability of a family of BQP-complete concepts $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$, we can simply limit the size of each concept class $\mathcal{C}_n$ to be no more than a polynomial in n. When the size of the concept class is polynomial, a quantum learner can iterate over all concepts and identify the concept that best matches the examples from the oracle. In more technical terms, a quantum learner can efficiently perform empirical risk

[1] Recently, there have been notable developments that have yielded contrasting conclusions. For instance, in [107], surprisingly complex physics problems are efficiently learned by classical learners. We will briefly discuss this in Section 6.3.1.

minimization through brute-force fitting. From standard results in learning theory (e.g., Corollary 2.3 in [174]), it follows that this method results in a learner that satisfies the conditions of the PAC learning framework.

As discussed in Section 2.5.2, assuming that the concepts are not in HeurP/poly is sufficient to ensure that the concept class is not classically learnable. Intuitively, this is because if the concepts were classically learnable, the examples could be used to construct an advice string that, together with an efficient classical learning algorithm, would put the concepts in HeurP/poly. By combining this with our approach to ensure quantum learnability, we can show that if there exists a family of polynomially-sized concept classes consisting of BQP-complete concepts that are not in HeurP/poly, then this family of concept classes exhibits a CC/QQ separation. Moreover, in Section 6.2.1 we discuss how several of these separations can be build around data that is generated by a "genuine quantum process". The following theorem summarizes our findings, and we defer the proof to Appendix D.4.

Theorem 26. *Consider a family of concept classes* $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *and distributions* $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ *such that*

Quantum learnability:

(a) Every $c_n \in \mathcal{C}_n$ *can be evaluated quantumly in time* $\mathcal{O}(\text{poly}(n))$.

(b) There exists a polynomial p *such that for every* $n \in \mathbb{N}$ *we have*

$$|\mathcal{C}_n| \leq p(n).$$

Classical non-learnability:

(c) There exists a family $\{c_n\}_{n\in\mathbb{N}}$*, where* $c_n \in \mathcal{C}_n$*, such that*

$$(\{c_n\}_{n\in\mathbb{N}}, \{\mathcal{D}_n\}_{n\in\mathbb{N}}) \notin \mathsf{HeurP/poly}.$$

Then, $L = (\{\mathcal{C}_n\}_{n\in\mathbb{N}}, \{\mathcal{D}_n\}_{n\in\mathbb{N}})$ *exhibits a* CC/QQ *learning separation.*

At face value, it may not be clear whether there exist concept classes that satisfy both conditions (a) and (c), since condition (a) puts the concepts in BQP and it may not be clear how large HeurP/poly is relative to BQP. Notably, it is known that if the discrete logarithm is not in BPP, then it is also not in HeurP under certain distributions. Additionally, it is widely believed that a polynomial amount of advice does not significantly improve the computational complexity of solving the discrete logarithm problem. Hence, it is plausible to imagine the existence of problems $L \in \mathsf{BQP}$ for which there is a distribution $\mathcal{D}$ such that $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$. Moreover, it is interesting to observe that if there exists a single $L \in \mathsf{BQP}$ that is not in HeurP/poly under some distribution, then for every BQP-complete problem there exists a distribution under which it is not in HeurP/poly. We summarize this in the lemma below, and we defer the proof to Appendix D.4.1.

Lemma 27. *If there exists a* $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$ *with* $L \in \mathsf{BQP}$*, then for every* $L' \in \mathsf{BQP\text{-}complete}$[2] *there exists a family of distributions* $\mathcal{D}' = \{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ *such that* $(L', \mathcal{D}') \notin \mathsf{HeurP/poly}$.

In summary, to obtain a learning separation for data generated by a "genuine quantum process", it is sufficient to have a single problem $L \in \mathsf{BQP}$ that lies outside $\mathsf{HeurP/poly}$ under some distribution. An example of such a problem is the discrete logarithm. However, the resulting distribution under which the BQP-complete problem lies outside of $\mathsf{HeurP/poly}$ is artificial as it comes from explicitly encoding the discrete logarithm into the learning problem through the reduction to the BQP-complete problem. Besides the discrete logarithm, little is known about the heuristic hardness of problems in BQP (especially those that are considered "genuinely quantum"). Therefore, the question arises as to what additional properties are required for a BQP-complete problem to lie outside $\mathsf{HeurP/poly}$ under some distribution. We show that a worst-case to average-case reduction combined with the assumption that $\mathsf{BQP} \not\subseteq \mathsf{P/poly}$ is sufficient for this purpose. While the question of $\mathsf{BQP} \not\subseteq \mathsf{P/poly}$ remains open, we proceed under this assumption based on its implications for cryptography. Specifically, if $\mathsf{BQP} \subseteq \mathsf{P/poly}$, then problems like the discrete logarithm would be in $\mathsf{P/poly}$, which would break cryptographic systems assumed to be secure[3]. Under the assumption that $\mathsf{BQP} \not\subseteq \mathsf{P/poly}$, the only missing piece is that our problem $L \in \mathsf{BQP}$ that lies outside $\mathsf{P/poly}$ is random self-reducible with respect to some distribution (i.e., it admits a worst-case to average case reduction as discussed in Section 2.5.2). We summarize these findings in the lemma below, and we defer the proof to Appendix D.4.2.

Lemma 28. *If* $L \notin \mathsf{P/poly}$ *and* L *is polynomially random self-reducible with respect to some distribution* $\mathcal{D}$*, then* $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$.

By combining Lemma 27 with Lemma 28, we obtain a set of assumptions that result in provable learning separations for data that could be generated by a genuine quantum process, as stated in Theorem 26 (see also Section 6.2.1). These assumptions include the existence of a problem $L \in \mathsf{BQP}$ that is not in $\mathsf{P/poly}$ which is polynomially random self-reducible with respect to some distribution.

Corollary 29. *If there exists an* $L \in \mathsf{BQP}$ *such that* $L \notin \mathsf{P/poly}$ *and it is random self-reducible, then every* BQP*-complete problem gives rise to a* $\mathsf{CC/QQ}$ *separation.*

Although establishing such learning separations is not straightforward, the criteria listed in Lemma 27, Lemma 28 and Corollary 29 suggest some challenges that when addressed lead to provable learning separations. In particular, they highlight the need for further investigation into the heuristic hardness of problems in BQP from the perspective of quantum machine learning.

[2] With respect to many-to-one reductions (as is the case for, e.g., quantum linear system solving [101]).

[3] In cryptography it is common to assume non-uniform adversaries (i.e., with computational resources of $\mathsf{P/poly}$), and even in this case most public-key cryptosystems such as RSA and Diffie-Hellman are still assumed to be safe).

Generative modeling To the authors it is not entirely clear precisely how strong the assumption that $\mathsf{BQP} \not\subseteq \mathsf{P/poly}$ is. It is worth noting though that for sampling problems arguably more iron-clad assumptions, such as the non-collapse of the polynomial hierarchy, could potentially lead to analogous conclusions. In particular, one possibility is to use quantum supremacy arguments [10, 42] to establish learning separations in generative modeling, where the task is to learn a distribution instead of a binary function. If the distribution to be learned is in $\mathsf{SampBQP}$ (i.e., sampling problems solvable by a polynomial-time quantum algorithm), then for classical non-learnability, the corresponding requirement is that not all $\mathsf{SampBQP}$ problems are in $\mathsf{SampBPP/poly}$ (i.e., sampling problems solvable by a polynomial-time classical algorithm with polynomial-sized advice)[4]. This is analogous to the supervised learning case, but for sampling problems we might have further evidence this is unlikely. Specifically, as sketched by Aaronson in [9], if $\mathsf{SampBQP} \subseteq \mathsf{SampBPP/poly}$, then this could cause the polynomial hierarchy to collapse. In other words, one could arguably use these arguments to show that a family of distributions is not classically learnable, under the assumption that the polynomial hierarchy does not collapse.

6.2.1 Learning separations from physical systems

Many quantum many-body problems are either BQP-complete or QMA-complete when appropriately formalized, making them suitable for defining concepts that are not classically learnable (recall that this also implies a learning separation, since quantum learnability can be ensured by considering polynomially-sized concept classes). To be more precise, recall that any problem in BQP that does not lie in $\mathsf{HeurP/poly}$ with respect to some distribution can be used to construct a distribution under which a hard quantum many-body problem defines a learning problem that is not classically learnable (as shown by Theorem 26 and Lemma 27). However, the induced distribution under which the physical system is not classically learnable is artificial, as it is induced by a particular choice of reduction, and there is no evidence that these induced distributions are relevant in practice.

Examples of physical systems For concreteness, let us discuss some examples. There are many physical systems that are in some sense universal for quantum computing, such as the Bose-Hubbard model [62], the antiferromagnetic Heisenberg and antiferromagnetic XY model [159], the Fermi-Hubbard model [150], supersymmetric systems [49], interacting bosons [197], and interacting fermions [125]. In particular, each of these physical systems defines a family of Hamiltonians and, for several of these Hamiltonian families, time-evolution is BQP-complete when appropriately formalized [96, 63]. That is, for several of these universal Hamiltonian families $H(\beta)$, where β denote the Hamiltonian parameters, we can define BQP-complete concepts

$$c_H(\beta, t) = \operatorname{sign}\left(\left|\langle 0^n|e^{iH(\beta)t} Z_1 e^{-iH(\beta)t}|0^n\rangle\right|^2 - \frac{1}{2}\right),$$

[4]For a formal definition of these complexity classes we refer to [10].

where Z_1 denotes the Pauli-Z operator on the first qubit and identity elsewhere. Additionally, one could also use BQP-complete problems in high energy physics, such as scattering in scalar quantum field theory [112].

As another example, we note that for any of the universal Hamiltonian families the problem of finding the ground state energy is QMA-complete. That is, for any universal Hamiltonian family $H(\beta)$, where β denote the Hamiltonian parameters, we can define QMA-complete concepts

$$c_H(\beta) = \text{sign}\left(\text{Tr}\left[H(\beta)\left|\psi_H(\beta)\right\rangle\right] - \frac{1}{2}\right),$$

where $|\psi_H(\beta)\rangle$ denotes the ground state of $H(\beta)$. Naturally, one worries that these concepts are too hard to evaluate on a quantum computer, but there are a few workarounds. Firstly, sometimes there is a natural special case of the problem that is BQP-complete (e.g., the subset of Hamiltonians obtained through a circuit-to-Hamiltonian mapping). Moreover, more generically it holds that any problem that is QMA-complete has a restriction that is BQP-complete (i.e., take any BQP-complete problem and consider the image of this problem under a many-to-one reduction). Finally, one could use recent results on the guided local Hamiltonian problem to relax the QMA-complete problems and obtain a BQP-complete problem [196, 50, 82].

In short, by exploiting a reduction from a problem that is in BQP which under a given distribution lies outside HeurP/poly onto a chosen BQP-complete problem (as in Lemma 27), any physical system that is in some sense universal for quantum computing can be used to construct a learning separation. Nonetheless, since the reduction is implicitely used to construct the distribution under which the physical system becomes not classically learnable, the distributions will be artificial and there is no reason to believe these have any relevance in practice.

6.3 Connections to other works on (quantum) learning tasks

In this section we discuss other topics of relevance. First, in Section 6.3.1, we discuss the implications and limitations of the milestone work of Huang et al. [107] on establishing learning separations from physical systems. Next, in Section 6.3.2, we discuss how having access to data radically enhances what can be efficiently evaluated by discussing the example of evaluating parameterized quantum circuits. Afterwards, in Section 6.3.3, we discuss how two physically-motivated problems (i.e., Hamiltonian learning, and identifying order parameters for phases of matter) fit in the PAC learning setting where the learner is constrained to output hypotheses from a fixed hypothesis class.

6.3.1 Provably efficient machine learning with classical shadows

In the milestone work of Huang et al. [107], the authors design classical machine learning methods (in part built around the *classical shadow* paradigm) that can efficiently learn quantum many-body problems. One of the problems studied in [107] is that of *predicting ground states of Hamiltonian*. More precisely, for a family of Hamiltonians $H(x)$ with ground states $\rho_H(x)$, one wants to predict the expectation value of some observable O when measured on $\rho_H(x)$. That is, one wants to efficiently learn to evaluate the function

$$f_{H,O}(x) = \mathrm{Tr}\left[\rho_H(x)O\right]. \tag{6.8}$$

One of the main things that [107] show is that given a polynomial number of data points, one is able to efficiently evaluate the functions in Eq. (6.8) with a constant expected error under certain criteria. Recall that in Section 6.2.1 we argued that concepts based on physical systems can be used as a source of learning separations. Since these concepts are of a similar form as the functions described in Eq. (6.8), one might wonder how the results of Huang et al. relate.

Let us take a closer look at the requirements of the methods described in [107]. Firstly, the Hamiltonians $H(x)$ must all be geometrically-local, and the observable O must be a sum of polynomially many local observables $O = \sum_{i=1}^{L} O_i$ such that $\sum_{i=1}^{L} ||O_i||$ is bounded by a constant. Additionally, the Hamiltonians $H(x)$ must all have a constant spectral gap (i.e., the difference between the smallest and the next smallest eigenvalue) and they must depend smoothly on x (or more precisely, the average gradient of the function in Eq. (6.8) must be bounded by a constant). One might wonder what will happen if we relax the above requirements, while simultaneously maintaining the fact that a quantum computer would still be able to evaluate the function in Eq. (6.8) (and hence build a learning separation around it based on Theorem 26).

Two possible relaxations of the requirements are the absence of a constant spectral gap (while maintaining an inverse polynomial spectral gap) and a reduced smoothness dependency of the Hamiltonian family on x (i.e., compared to what is required for the methods of [107]). It turns out that if one relaxes these requirements, then under cryptographic assumptions the methods proposed by Huang et al. are no longer capable of evaluating the function in Eq. (6.8) with constant expected error. More precisely, any classical machine learning method that would still be able to evaluate the function in Eq. (6.8) up to constant expected error under the relaxed assumptions would be able to solve DLP in P/poly, which contradicts certain cryptographic assumptions. We provide a formal statement of this in the following theorem, the proof of which is deferred to Appendix D.5.

Theorem 30. *Suppose there exists a polynomial-time randomized classical algorithm $\mathcal{A}$ with the following property: for every geometrically-local family of n-qubit Hamiltonians $H(x)$ there exist a dataset $\mathcal{T}_H \in \{0,1\}^{\mathrm{poly}(n)}$ such that for every sum $O = \sum_{i=1}^{L} O_i$ of $L \in \mathcal{O}(\mathrm{poly}(n))$ many local observables with $\sum_{i=1}^{L} ||O_i|| \leq B$ for*

some constant B, the function

$$\overline{f}_{H,O}(x) = \mathcal{A}(x, O, \mathcal{T}_H)$$

satisfies

$$\mathbb{E}_{x\sim[-1,1]^m}\left[\, |\overline{f}_{H,O}(x) - f_{H,O}(x)| \,\right] < \frac{1}{6},$$

where $f_{H,O}(x) = \mathrm{Tr}\left[\rho_H(x)O\right]$ and $\rho_H(x)$ denotes the ground state of $H(x)$. Then, $\mathsf{DLP} \in \mathsf{P/poly}$.

In conclusion, Theorem 30 shows that any method similar to that of [107] cannot learn to predict ground state properties of certain physical systems discussed in Section 6.2.1. Moreover, there are a few subtle differences between the setup of [107] and the one discussed in this thesis. Firstly, the classical shadow paradigm uses data that is different from the PAC learning setting (i.e., the data does not correspond to evaluations of the function it aims to predict). This distinction in setup makes the approach of [107] more versatile, as their data can be utilized to evaluate multiple different observables (moreover, their methods also work in the PAC setting). Secondly, the functions $f_{H,O}$ in Eq. (6.8) are real-valued, which differs from our setting where we investigate functions that map onto a discrete label space. It is possible to address this difference by applying a threshold function to $f_{H,O}$ after it is learned. However, this thresholding introduces a mismatch in the types of data, as it would involve using real-valued data to learn a function with discrete values (which is clearly different from the PAC setting).

6.3.2 Power of data

In [109] the authors show how having access to data radically enhances what can be efficiently evaluated. In this section we connect the ideas from their work to the formalism we introduce in this thesis. Specifically, we will discuss a family of functions inspired by [109] that from their description alone cannot be efficiently evaluated classically, yet access to a few examples (i.e., evaluations of the function) allows them to be efficiently evaluated classically. This highlights an important difference between complexity-theoretic separations and learning separations, since in the latter one has to deal with the learner having access to data when proving classical non-learnability.

Consider a polynomial-depth parameterized quantum circuit $U(\theta, \vec{\phi})$ – with two types of parameters $\theta \in \mathbb{R}$ parameterizing a single gate and $\vec{\phi} \in \mathbb{R}^\ell$ parameterizing multiple other gates – that is universal in the sense that for every polynomial-depth circuit V there exists parameters $\vec{\phi}^* \in \mathbb{R}^\ell$ such that

$$U(0, \vec{\phi}^*)\,|0^n\rangle = V\,|0^n\rangle\,.$$

Moreover, assume the gates in U are of the form $\exp\left(-\frac{i\theta}{2}A\right)$, with $A^2 = I$ (e.g., Z- or X-rotations). By measuring the output of the circuit we define a family of single parameter functions given by

$$f_{\vec{\phi}}(\theta) = \langle 0^n|\, U(\theta, \vec{\phi})^\dagger M U(\theta, \vec{\phi})\,|0^n\rangle\,.$$

Following an argument similar to [109], due to the universality of the parameterized quantum circuit no efficient randomized classical algorithm can take as input a $\vec{\phi} \in \mathbb{R}^\ell$ and compute the function $f_{\vec{\phi}}$ on a given point $\theta \in \mathbb{R}$ up to constant error in time $\mathcal{O}(\text{poly}(n))$, unless $\mathsf{BPP} = \mathsf{BQP}$. Intuitively, one might thus think that the concept class $\{f_{\vec{\phi}} \mid \vec{\phi} \in \mathbb{R}^\ell\}$ exhibits a separation between classical and quantum learners. However, it turns out that the examples given to a classical learner radically enhance what it can efficiently evaluate. In particular, given a few of evaluations of $f_{\vec{\phi}}$ for some fixed but arbitrary $\vec{\phi} \in \mathbb{R}^\ell$, a classical learner is suddenly able to efficiently evaluate the function. To see this, note that by [144] one can write the functions as

$$f_{\vec{\phi}}(\theta) = \alpha \cos(\theta - \beta) + \gamma, \quad \text{for } \alpha, \beta, \gamma \in \mathbb{R},$$

where the coefficients α, β and γ are all independent of θ (but they do depend on $\vec{\phi}$). From this we can see that any three distinct examples $\left\{\left(\theta_i, f_{\vec{\phi}}(\theta_i)\right)\right\}_{i=1}^3$ uniquely determine $f_{\vec{\phi}}(\theta)$ and one can simply fit α, β and γ to these three examples to learn how to evaluate $f_{\vec{\phi}}$ on unseen points. We would like to point that the BQP-hard problem in question is not evaluating $f_{\vec{\phi}}$ for a fixed $\vec{\phi} \in \mathbb{R}^\ell$, but rather evaluating $f_{\vec{\phi}}$ when $\vec{\phi} \in \mathbb{R}^\ell$ is part of the input. This approach can be generalized to settings with more than one free parameter θ, by using the fact that expectation values of parameterized quantum circuits can be written as a Fourier series [171]. Specifically, when the number of frequencies appearing in the Fourier series is polynomial, then a polynomial number of examples suffices to fit the Fourier series and learn how to evaluate the expectation value of the quantum circuits for an arbitrary choice of parameters.

As discussed in Section 6.1, one way to deal with the fact that data can radically enhance what can be efficiently evaluated is to ensure that the data itself is efficiently generatable. However, for the concepts discussed above, the examples are such that only a quantum computer can generate them efficiently. In other words, these functions exemplify how hard to generate data can radically enhance what a classical learner can efficiently evaluate. As discussed in Section 6.1, another way to deal with the fact that data can radically enhance what can be efficiently evaluated is to ensure that the concepts lie outside of $\mathsf{HeurP/poly}$. However, for the case discussed above, every $f_{\vec{\phi}}$ corresponds to a function in $\mathsf{HeurP/poly}$, since the coefficients α, β and γ suffice as the advice. Finally, we note that for certain circuits one could have exponentially many terms in the Fourier series [54, 53], in which case it is unclear how to classically learn them.

6.3.3 Physically-motivated PAC learning settings with fixed hypothesis classes

Throughout this thesis we mainly focused on the setting where the learner is allowed to output arbibtrary hypotheses (barring that they have to be tractable as discussed in Appendix D.1.1). However, we want to highlight that setting where the learner is constrained to only be able to output hypothesis from a fixed hypothesis class is also relevant from a practical perspective. In particular, in this section discuss how two well-studied problems (i.e., Hamiltonian learning, and identifying order parameters

for phases of matter) fit in this setting. Recall that in this setting, it is allowed and reasonable for the hypothesis class to be classically- or quantumly- intractable.

Hamiltonian learning In Hamiltonian learning one is given measurement data from a quantum experiment, and the goal is to recover the Hamiltonian that best matches the data. Throughout the literature, various different types of measurement data have been considered. For example, it could be measurement data from ground states, (non-zero temperature) thermal sates, or time-evolved states. In our case, the data will be measurement data from time-evolved states and we formulate Hamiltonian learning in terms of a hypothesis class as follows. First, we fix a (polynomially-sized) set of Hermitian operators $\{H_\ell\}_{\ell=1}^{L}$. Next, we consider a family of Hamiltonians $\{H_\beta\}_{\beta\in\mathbb{R}^L}$, where

$$H_\beta = \sum_{\ell=1}^{L} \beta_\ell H_\ell. \tag{6.9}$$

Finally, we define the hypothesis class $\mathcal{H}^{\mathrm{HL}} = \{h_\beta\}_{\beta\in\mathbb{R}^L}$, with concepts defined as

$$h_\beta(z,t) = \mathrm{sign}\big(\mathrm{Tr}\big[U^\dagger(t)\rho_z U(t) O_z\big]\big), \quad U(t) = e^{itH_\beta}. \tag{6.10}$$

Here z describes the experimental setup, specifying the starting state (that will evolve under H_β for time t) and the observable measured at the end. A natural specification of the concepts that a learner could output are the parameters β. In particular, in Hamiltonian learning we are only concerned with identifying which concept generated the data (i.e., what is the specification of the underlying Hamiltonian), as opposed to finding a hypothesis that closely matches the data. In other words, the problem of Hamiltonian learning can naturally be formulated as PAC learning setting where the learner is constrained to only be able to output hypotheses described in Eq. (6.10).

With respect to learning separations, one might think that the above setting is a good candidate to exhibit a $\mathsf{C}_{\mathcal{H}^{\mathrm{HL}}}/\mathsf{Q}_{\mathcal{H}^{\mathrm{HL}}}$ separation, since the hypotheses are classically intractable and quantumly efficient to evaluate (assuming $\mathsf{BPP} \neq \mathsf{BQP}$). Moreover, according to the folklore, quantum learners are most likely to have its advantages for data that is "quantum-generated", which certainly seems to be the case here. However, recall that in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class the task is not to evaluate, but rather to identify the concept generating the examples. Therefore, the arguments we used throughout this thesis do not directly apply. In fact, it turns out that classical learners can efficiently identify the parameters of the Hamiltonian generating the data in many natural settings [20, 98, 108], eliminating the possibility of a $\mathsf{C}_{\mathcal{H}^{\mathrm{HL}}}/\mathsf{Q}_{\mathcal{H}^{\mathrm{HL}}}$ separation.

Order parameters and phases of matter When studying phases of matter one might want to identify what physical properties characterize the phase. One can formulate this problem as finding a specification of the correct hypothesis selected from a hypothesis class consisting of possible *order parameters*. In particular, we fix the hypotheses $\mathcal{H}^{\mathrm{order}} = \{h_\alpha\}$ to be of a very special form, which compute certain

expectation values of ground states given a specification of a Hamiltonian. That is, we formally define the hypotheses as

$$h_\alpha(\beta) = \text{sign}\left(\text{Tr}\left[O_\alpha \rho_\beta\right]\right), \tag{6.11}$$

where ρ_β denotes the ground state of some Hamiltonian specified by β (e.g., using the parameterization in Eq. (6.9)), and α specifies an observable O_α drawn from a set of observables that are deemed potential candidates for the order parameter that characterize the phase. In this setting, one might not necessarily want to evaluate the hypotheses, as they might require one to prepare the ground state, which is generally intractable (even for a quantum computer). However, one might still want to identify the observable O_α that correctly characterizes the phase of the physical system specified by β (i.e., the corresponding *order parameter*). In other words, the problem of identifying order parameters naturally fits in the PAC learning setting where the learner is constrained to only be able to output hypotheses described in Eq. (6.11).

As in the case of Hamiltonian learning, one might think that the above concepts are good candidates to exhibit a $\mathsf{C}_{\mathcal{H}^{\text{order}}}/\mathsf{Q}_{\mathcal{H}^{\text{order}}}$ separation, since the hypotheses are classically intractable and quantumly efficient to evaluate (assuming $\mathsf{BPP} \neq \mathsf{BQP}$). In fact, according to the folklore, quantum learners are most likely to have advantages for data that is "quantum-generated", which certainly seems to also be the case here. However, as already mentioned, in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class the goal is only to identify the correct hypothesis, and it is therefore not enough to just have concepts that are classically intractable. We remark that the methods of [107] also apply to phase classification, but they are more aimed at the PAC learning setting where the learner can output arbitrary hypotheses (i.e., the main goal is to predict the phase of a given physical system). In particular, their methods do not directly allow one to obtain a physically-meaningful description of the order-parameter, which is the main goal in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class (which is related to the popular theme of "explainability" in machine learning).

In conclusion, while there has been progress in studying separations in the setting where the learner is constrained to output hypotheses from a fixed hypothesis class, there is still much to be discovered. Note that if the hypothesis class is BQP-complete in the sense that it can perform arbitrary quantum computation, then a collapse similar to Lemma 3 happens and no separations are possible. All in all, we have yet to find an example of a learning setting where the data is generated by a genuine quantum process and where it is necessary to use a quantum algorithm to efficiently identify the process generating the data.

Chapter 7

Conclusion

In this chapter, we present the conclusions of the thesis. In Section 7.1, we recall the problem statement and research questions and we provide them with answers. Next, in Section 7.2, we discuss the limitations of our work. Finally, in Section 7.3, we discuss some promising directions for future work.

7.1 Research overview

In this thesis, we studied the speedups provided by several QML proposals over their classical counterparts, and how to get the best possible performance out of these proposals. Specifically, the problem statement of this thesis was:

Problem statement. *Can we substantiate the capacity of various QML proposals to (superpolynomially) outperform their classical counterparts, and what methods can we devise to attain their best possible performance?*

The above problem statement was split into 4 research question. Below, we restate these research questions and subsequently provide them with answers.

Chapter 3 investigated the potential of a class of problems arising from the quantum algorithm for topological data analysis [130] to become genuinely useful applications of quantum computers with a superpolynomial quantum speedup, with the first research question in mind:

Research question 1. *Can the linear-algebraic QML algorithms for Betti numbers maintain their speculated superpolynomial quantum speedups, even with the development of better classical algorithms?*

We showed that this algorithm along with a number of new algorithms provided by us (with applications in numerical linear algebra, machine learning and complex network analysis) solve problems that are classically intractable under widely-believed complexity-theoretic assumptions by showing that they are as hard as simulating the one clean qubit model. Specifically, our results showed that the methods of the quantum algorithm for topological data analysis withstand the sweeping dequantization

results of Tang et al. [186, 60]. To analyze whether it is possible to further strengthen the argument for quantum advantage (or, to actually find an efficient classical algorithm) for the narrow TDA problem, we investigated state-of-the-art classical algorithms and we highlighted the theoretical hurdles that, at least currently, stymie such classical approaches. Regarding near-term implementations, we identified that implementing sparse access to the input matrix is a major bottleneck in terms of the required number of qubits, we proposed multiple methods to circumvent this bottleneck via classical precompilation strategies, and we investigated the required resources to challenge the best known classical methods.

Chapter 4 studied how to implement structural risk minimization for quantum machine learning models based on parameterized quantum circuits, to answer the second research question:

Research question 2. *Can we identify hyperparameters within novel quantum learning models based on parameterized quantum circuits that impact both complexity measures and performance on training data, as is crucial for the successful implementation of structural risk minimization?*

In particular, in Theorem 12 and Theorem 14 we characterized the VC-dimension and fat-shattering dimension of these quantum models and identified hyperparameters – such as the rank and Frobenius norms of the observables – that influence these complexity measures. Moreover, in Proposition 15 and Proposition 17 we showed that these hyperparameters also influence the performance that these quantum models can have on training data. Finally, we showed how our findings can be used to construct new quantum machine learning models with favourable performance guarantees based on the principle of structural risk minimization.

Chapter 5 investigated quantum reinforcement learning agents based on parameterized quantum circuits motivated by our third research question:

Research question 3. *How can new quantum machine learning models based on parameterized quantum circuits be effectively leveraged within the realm of reinforcement learning? Specifically, can these quantum approaches demonstrate the potential to be on par with classical models in standard benchmarking tasks and outperform them in novel specific scenarios?*

We proposed several constructions and showed the impact of certain design choices on learning performance. In particular, we introduced the SOFTMAX-PQC model, where a softmax policy is computed from expectation values of a parameterized quantum circuit with both trainable observable weights and input scaling parameters. These added features to standard parameterized quantum circuits used in machine learning (e.g., as quantum classifiers) enhance both the expressivity and flexibility of parameterized quantum circuit policies, which allows them to achieve a learning performance on benchmarking environments comparable to that of standard deep neural networks. We additionally demonstrated the existence of task environments, constructed out of parameterized quantum circuit, that are very natural for parameterized quantum circuit agents, but on which deep neural network agents have a poor

performance. To strengthen this result, we constructed several reinforcement learning environments, each with a different degree of degeneracy (i.e., closeness to a supervised learning task), where we showed a rigorous separation between a class of parameterized quantum circuit agents and any classical learner, based on the widely-believed classical hardness of the discrete logarithm problem. We believe that our results constitute strides toward a practical quantum advantage in reinforcement learning using (near-term) quantum devices.

Chapter 6 focused on the identification of learning problems within the probably approximately correct (PAC) learning framework, where quantum learners exhibit exponential advantages over classical learners, as motivated by our fourth research question:

Research question 4. *How can we identify learning problems that exhibit a provable exponential speedup for quantum learning algorithms compared to their classical counterparts, and can we confirm the validity of the folklore that quantum machine learning excels when handling quantum-generated data?*

Firstly, we delved into the intricacies of precisely defining what it means for a quantum learner to exhibit an exponential advantage over its classical counterpart. Subsequently, we studied prior instances of learning separations [126, 173], pinpointing the exact source of classical hardness and the quantum edge. Lastly, we examined the folklore that quantum machine learning excels most in scenarios involving quantum-generated data. In doing so, we established a framework through which any BQP-complete problem can lead to a learning separation, thereby substantiating the quantum advantage across numerous domains in physics.

7.2 Limitations

In this section, we highlight certain limitations in the outcomes of this thesis. Firstly, concerning our findings in Chapter 3, it is important to note that our discussion of the noise-robustness of the quantum algorithm for topological data analysis lacks experiments to confirm or reject our statements due to our limited access to sufficiently large quantum hardware. Secondly, in relation to our results in Chapter 4, our analysis does not consider the structure of the feature map, which has the potential to enhance the effectiveness of structural risk minimization. Moreover, in reference to our outcomes in Chapter 5, our ability to benchmark reinforcement learning models was limited to toy problems, as we lacked access to hardware capable of handling real-world problem sizes. Lastly, with respect to our findings in Chapter 6, it is worth noting that the results of Theorem 26, in most instances, rely on contrived data distributions that may not be representative of real-world scenarios.

7.3 Future work

The findings presented in this thesis open up exciting avenues for future research, offering several promising directions to explore. In this section, we highlight a few of

these potential interesting opportunities for future work.

Regarding the topological data analysis results in Chapter 3, there are several interesting open questions. First, it remains open whether ABNE, as outlined in Section 1.1, is indeed DQC1-hard. In particular, it remans open whether LLSD retains its DQC1-hardness when restricted to combinatorial Laplacians. Exploring quantum algorithms' capabilities for poroblems in topological data analysis beyond computing Betti numbers, such as handling other aspects of barcodes [83], presents another avenue of research. Lastly, the potential utility of quantum algorithms in computing eigenvalues and eigenvectors of combinatorial Laplacians for complex network analysis, as hinted in [134], merits further investigation.

Concerning the results presented in Chapter 4, which delve into structural risk minimization for quantum machine learning models relying on parameterized quantum circuits, several interesting research directions emerge. Firstly, it is worth exploring alternative complexity measures beyond VC-dimension and fat-shattering dimension. Such an exploration can potentially uncover additional sets of hyperparameters relevant to the structural risk minimization tradeoff. Additionally, there is room for investigating how the phenomenon of overparameterization, as extensively studied in the context of neural networks [18], extends to quantum machine learning models that employ parameterized quantum circuits. This investigation can provide valuable insights into the generalization performance of these quantum models, shedding light on their behavior in comparison to classical counterparts.

In light of the results discussed in Chapter 5, which pertain to reinforcement learning with quantum machine learning models based on parameterized quantum circuits, several interesting avenues for future research come to the forefront. Firstly, an exciting direction would involve exploring the potential of our novel quantum machine learning models when combined with state-of-the-art policy gradient methods or actor-critic methods like DDPG [123], PPO [172], or A3C [138]. Such investigations can unveil synergies between classical reinforcement learning techniques and quantum enhancements, potentially leading to superior performance in complex learning tasks. Furthermore, it would be worthwhile to delve deeper into the capabilities of our novel quantum machine learning models as we scale up the number of available qubits.

The results of Chapter 6, which delve into the exponential separations between quantum and classical learning separations, prompt us to consider several interesting directions for future inquiry. Firstly, an interesting direction of future research would involve extending our investigations beyond the PAC learning framework. Delving into alternative learning frameworks, such as the Angluins learning framework [19], can broaden our understanding of quantum versus classical learning separations. Moreover, the outcomes of our work underscore the significance of probing the heuristic hardness of BQP-complete problems. Such investigations could give rise to novel learning separations within specific physics-inspired scenarios, thus contributing to a deeper understanding of the quantum advantage in practical applications.

Bibliography

[8] Scott Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463, 2007.

[9] Scott Aaronson, Jan 2013. https://cstheory.stackexchange.com/questions/15066/consequences-of-bqp-subseteq-p-poly.

[10] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011.

[11] Jayadev Acharya, Ibrahim Issa, Nirmal V Shende, and Aaron B Wagner. Measuring quantum entropy. *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019.

[12] Jayadev Acharya, Alon Orlitsky, Ananda Theertha Suresh, and Himanshu Tyagi. Estimating rényi entropy of discrete distributions. *IEEE Transactions on Information Theory*, 63:38–56, 2016.

[13] Michał Adamaszek. Extremal problems related to Betti numbers of flag complexes. *Discrete Applied Mathematics*, 173:8–15, 2014.

[14] Michał Adamaszek and Juraj Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. *Computational Geometry*, 57:8–18, 2016.

[15] Leonard Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science (SFCS 1978)*, pages 75–83. IEEE Computer Society, 1978.

[16] Hamed Ahmadi and Pawel Wocjan. On the quantum complexity of evaluating the Tutte polynomial. *Journal of Knot Theory and its Ramifications*, 19:727–737, 2010.

[17] Werner Alexi, Benny Chor, Oded Goldreich, and Claus P Schnorr. Rsa and rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17:194–209, 1988.

[18] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019.

[19] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

[20] Anurag Anshu, Srinivasan Arunachalam, Tomotaka Kuwahara, and Mehdi Soleimanifar. Sample-efficient learning of quantum many-body systems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 685–691. IEEE, 2020.

[21] Martin Anthony and Peter L Bartlett. Function learning from interpolation. *Combinatorics, Probability and Computing*, 9, 2000.

[22] Simon Apers, Sayantan Sen, and Dániel Szabó. A (simple) classical algorithm for estimating betti numbers. *arXiv:2211.09618*, 2022.

[23] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[24] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[25] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 4, 2006.

[26] Srinivasan Arunachalam and Ronald de Wolf. Guest column: A survey of quantum learning theory. *ACM SIGACT News*, 48, 2017.

[27] Ryan Babbush, Jarrod McClean, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *Physical review X Quantum*, 2021.

[28] Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44, 1998.

[29] Peter L Bartlett and Philip M Long. Prediction, learning, uniform convergence, and scale-sensitive dimensions. *Journal of Computer and System Sciences*, 56, 1998.

[30] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin Chan. Quantum algorithms for quantum chemistry and quantum materials science. *Chemical Reviews*, 120:12685–12717, 2020.

[31] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4, 2019.

[32] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. *Proceedings of 56th Annual Symposium on Foundations of Computer Science*, 2015.

[33] Dominic W Berry, Yuan Su, Casper Gyurik, Robbie King, Joao Basso, Alexander Del Toro Barba, Abhishek Rajput, Nathan Wiebe, Vedran Dunjko, and Ryan Babbush. Quantifying quantum advantage in topological data analysis. *arXiv preprint arXiv:2209.13581*, 2022.

[34] Jacob Biamonte, Mauro Faccin, and Manlio De Domenico. Complex networks from classical to quantum. *Communications Physics*, 2:1–10, 2019.

[35] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549, 2017.

[36] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, 13, 1984.

[37] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Theoretical Computer Science*, 2006.

[38] Xavi Bonet-Monroig, Ramiro Sagastizabal, M Singh, and TE O'Brien. Low-cost error mitigation by symmetry verification. *Physical Review A*, 2018.

[39] Adam D Bookatz. Qma-complete problems. *Quantum Information & Computation*, 14:361–383, 2014.

[40] Fernando GSL Brandão. *Entanglement theory and the quantum simulation of many-body physics*. PhD thesis, University of London, 2008.

[41] Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467, 2011.

[42] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical review letters*, 117, 2016.

[43] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[44] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[45] Brielin Brown, Steven T Flammia, and Norbert Schuch. Computational difficulty of computing the density of states. *Physical review letters*, 2011.

[46] Kaifeng Bu, Dax Enshan Koh, Lu Li, Qingxian Luo, and Yaobo Zhang. On the statistical complexity of quantum circuits. *arXiv preprint*, 2021.

[47] Kaifeng Bu, Dax Enshan Koh, Lu Li, Qingxian Luo, and Yaobo Zhang. Rademacher complexity of noisy quantum circuits. *arXiv preprint*, 2021.

[48] Chris Cade and P Marcos Crichigno. Complexity of supersymmetric systems and the cohomology problem. *arXiv preprint*, 2021.

[49] Chris Cade and P Marcos Crichigno. Complexity of supersymmetric systems and the cohomology problem. *arXiv 2107.00011*, 2021.

[50] Chris Cade, Marten Folkertsma, and Jordi Weggemans. Complexity of the guided local hamiltonian problem: improved parameters and extension to excited states. *arXiv 2207.10097*, 2022.

[51] Chris Cade and Ashley Montanaro. The quantum complexity of computing Schatten p-norms. *13th Conference on the Theory of Quantum Computation, Communication and Cryptography*, 2018.

[52] Matthias C Caro and Ishaun Datta. Pseudo-dimension of quantum circuits. *Quantum Machine Intelligence*, 2, 2020.

[53] Matthias C Caro, Elies Gil-Fuster, Johannes Jakob Meyer, Jens Eisert, and Ryan Sweke. Encoding-dependent generalization bounds for parametrized quantum circuits. *Quantum*, 5, 2021.

[54] Berta Casas and Alba Cervera-Lierta. Multi-dimensional fourier series with quantum circuits. *arXiv* `2302.03389`, 2023.

[55] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation. *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 2019.

[56] Jianer Chen, Xiuzhen Huang, Iyad A Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72:1346–1367, 2006.

[57] Lijie Chen and Roei Tell. Guest column: New ways of studying the BPP=P conjecture. *ACM SIGACT News*, 54:44–69, 2023.

[58] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, 8:141007–141024, 2020.

[59] Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *Journal of the ACM (JACM)*, 60:1–25, 2013.

[60] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. *Proceedings of the 52nd Annual ACM SIGACT symposium on theory of computing*, 2020.

[61] Andrew M Childs, David Gosset, and Zak Webb. The Bose-Hubbard model is QMA-complete. *International Colloquium on Automata, Languages, and Programming*, 2014.

[62] Andrew M Childs, David Gosset, and Zak Webb. The bose-hubbard model is QMA-complete. In *International Colloquium on Automata, Languages, and Programming*. Springer, 2014.

[63] Andrew Macgregor Childs. *Quantum information processing in continuous time.* PhD thesis, Massachusetts Institute of Technology, 2004.

[64] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

[65] David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. Approximating the spectrum of a graph. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[66] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[67] Marcos Crichigno and Tamara Kohler. Clique homology is qma1-hard. *arXiv preprint arXiv:2209.11793*, 2022.

[68] Manlio De Domenico and Jacob Biamonte. Spectral entropies as information-theoretic tools for complex network comparison. *Physical Review X*, 6, 2016.

[69] Ronald de Wolf. Quantum computing: Lecture notes. *arXiv:1907.09415*, 2019.

[70] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23:674–692, 2016.

[71] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[72] Vedran Dunjko, Yi-Kai Liu, Xingyao Wu, and Jacob M Taylor. Exponential improvements for quantum-accessible reinforcement learning. *arXiv preprint arXiv:1710.11160*, 2017.

[73] Vedran Dunjko and Peter Wittek. A non-review of quantum machine learning: trends and explorations. *Quantum*, 4:32, 2020.

[74] Alicja Dutkiewicz, Barbara M Terhal, and Thomas E O'Brien. Heisenberg-limited quantum phase estimation of multiple eigenvalues with a single control qubit. *arXiv preprint*, 2021.

[75] Art Duval, Caroline Klivans, and Jeremy Martin. Simplicial matrix-tree theorems. *Transactions of the American Mathematical Society*, 361:6073–6114, 2009.

[76] Beno Eckmann. Harmonische Funktionen und Randwertaufgaben in einem Komplex. *Commentarii Mathematici Helvetici*, 17:240–255, 1944.

[77] Talya Eden, Dana Ron, and Will Rosenbaum. Almost Optimal Bounds for Sublinear-Time Sampling of k-Cliques in Bounded Arboricity Graphs. *49th International Colloquium on Automata, Languages, and Programming – ICALP*, 2022.

[78] Suguru Endo, Simon C Benjamin, and Ying Li. Practical quantum error mitigation for near-future applications. *Physical Review X*, 2018.

[79] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993.

[80] Xinlong Feng and Zhinan Zhang. The rank of a random matrix. *Applied mathematics and computation*, 185:689–694, 2007.

[81] Joel Friedman. Computing Betti numbers via combinatorial Laplacians. *Algorithmica*, 21:331–346, 1998.

[82] Sevag Gharibian, Ryu Hayakawa, François Le Gall, and Tomoyuki Morimae. Improved hardness results for the guided local hamiltonian problem. *arXiv 2207.10250*, 2022.

[83] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008.

[84] András Gilyén and Tongyang Li. Distributional property testing in a quantum world. *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, 2020.

[85] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

[86] Timothy E Goldberg. Combinatorial laplacians of simplicial complexes. Master's thesis, Bard College, 2002.

[87] Shafi Goldwasser, Silvio Micali, and Po Tong. Why and how to establish a private code on a public network. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 134–144. IEEE, 1982.

[88] Google. Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum circuits. URL: github.com/quantumlib/Cirq, 2018.

[89] Takahiro Goto, Quoc Hoan Tran, and Kohei Nakajima. Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces. *Physical Review Letters*, 127(9):090506, 2021.

[90] Kiya W Govek, Venkata S Yamajala, and Pablo G Camara. Clustering-independent analysis of genomic data using spectral simplicial theory. *PLoS computational biology*, 2019.

[91] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[92] Anna Gundert and May Szedláky. Higher dimensional discrete Cheeger inequalities. *Proceedings of the 13th annual symposium on Computational Geometry*, 2014.

[93] Sam Gunn and Niels Kornerup. Review of a quantum algorithm for Betti numbers. *arXiv preprint*, 2019.

[94] Casper Gyurik and Vedran Dunjko. On establishing learning separations between classical and quantum machine learning with classical data. *arXiv 2208.06339*, 2022.

[95] Casper Gyurik, Vedran Dunjko, et al. Structural risk minimization for quantum linear classifiers. *Quantum*, 7:893, 2023.

[96] Jeongwan Haah, Matthew B Hastings, Robin Kothari, and Guang Hao Low. Quantum algorithm for simulating real time evolution of lattice hamiltonians. *SIAM Journal on Computing*, 2021.

[97] Jeongwan Haah, Robin Kothari, and Ewin Tang. Optimal learning of quantum hamiltonians from high-temperature gibbs states. *arXiv 2108.04842*, 2021.

[98] Jeongwan Haah, Robin Kothari, and Ewin Tang. Optimal learning of quantum hamiltonians from high-temperature gibbs states. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 135–146. IEEE, 2022.

[99] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53:217–288, 2011.

[100] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103:150502, 2009.

[101] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103, 2009.

[102] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567:209–212, 2019.

[103] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567, 2019.

[104] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[105] Danijela Horak and Jürgen Jost. Spectra of combinatorial Laplace operators on simplicial complexes. *Advances in Mathematics*, 244:303–336, 2013.

[106] He-Liang Huang, Xi-Lin Wang, Peter P Rohde, Yi-Han Luo, You-Wei Zhao, Chang Liu, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Demonstration of topological data analysis on a quantum processor. *Optica*, 5:193–198, 2018.

[107] Hsin-Yuan Huang, Richard Kueng, Giacomo Torlai, Victor V Albert, and John Preskill. Provably efficient machine learning for quantum many-body problems. *Science*, 377, 2022.

[108] Hsin-Yuan Huang, Yu Tong, Di Fang, and Yuan Su. Learning many-body hamiltonians with heisenberg-limited scaling. *arXiv 2210.03030*, 2022.

[109] HY Huang, M Broughton, M Mohseni, R Babbush, S Boixo, H Neven, and JR McClean. Power of data in quantum machine learning (2020). *Nature Communications*, 2021.

[110] Sofiene Jerbi, Lea M. Trenkwalder, Hendrik Poulsen Nautrup, Hans J. Briegel, and Vedran Dunjko. Quantum enhancements for deep reinforcement learning in large spaces. *PRX Quantum*, 2:010328, Feb 2021.

[111] Ian T Jolliffe. *Principal components in regression analysis*, pages 129–155. Springer, 1986.

[112] Stephen P Jordan, Hari Krovi, Keith SM Lee, and John Preskill. Bqp-completeness of scattering in scalar quantum field theory. *Quantum*, 2:44, 2018.

[113] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[114] Tosio Kato. *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media, 2013.

[115] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 1994.

[116] Michael Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[117] Michael J Kearns and Robert E Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48, 1994.

[118] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, 2017.

[119] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101:022316, 2020.

[120] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *quant-ph/9511026*, 1995.

[121] Alexei Yu Kitaev, Alexander Shen, Mikhail N Vyalyi, and Mikhail N Vyalyi. *Classical and quantum computation*. American Mathematical Society, 2002.

[122] Emanuel Knill and Raymond Laflamme. Power of one bit of quantum information. *Physical Review Letters*, 1998.

[123] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[124] Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM review*, 58:34–65, 2016.

[125] Yi-Kai Liu, Matthias Christandl, and Frank Verstraete. Quantum computational complexity of the N-representability problem: Qma complete. *Physical review letters*, 98:110503, 2007.

[126] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 2021.

[127] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.

[128] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.

[129] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996.

[130] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7:1–7, 2016.

[131] Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuit. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 245–251, 2020.

[132] László Lovász et al. Very large graphs. *Current Developments in Mathematics*, 2008:67–128, 2009.

[133] Guang Hao Low and Isaac L Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical review letters*, 118:010501, 2017.

[134] Slobodan Maletić and Milan Rajković. Combinatorial Laplacian and entropy of simplicial complexes associated with complex networks. *The European Physical Journal Special Topics*, 212:77–97, 2012.

[135] Sam McArdle, András Gilyén, and Mario Berta. A streamlined quantum algorithm for topological data analysis with exponentially fewer qubits. *arXiv:2209.12887*, 2022.

[136] Sam McArdle, Xiao Yuan, and Simon Benjamin. Error-mitigated digital quantum simulation. *Physical review letters*, 2019.

[137] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[138] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[139] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[140] J.W. Moon and Moser L. On a problem of turan. *Publ. Math. Inst. Hung. Acad. Sci.*, 1962.

[141] Tomoyuki Morimae. Hardness of classically sampling the one-clean-qubit model with constant total variation distance error. *Physical Review A*, 2017.

[142] Tomoyuki Morimae, Keisuke Fujii, and Joseph F Fitzsimons. Hardness of classically simulating the one-clean-qubit model. *Physical review letters*, 2014.

[143] Sayan Mukherjee and John Steenbergen. Random walks on simplicial complexes and harmonics. *Random structures & algorithms*, 49:379–405, 2016.

[144] Ken Nakanishi, Keisuke Fujii, and Synge Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *Physical Review Research*, 2, 2020.

[145] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2011.

[146] Thomas O'Brien, LevC Ioffe, Yuan Su, David Fushman, Hartmut Neven, Ryan Babbush, and Vadim Smelyanskiy. Quantum computation of molecular structure using data from challenging-to-classically-simulate nuclear magnetic resonance experiments. *arXiv* `2109.02163`, 2021.

[147] Thomas E O'Brien, Stefano Polla, Nicholas C Rubin, William J Huggins, Sam McArdle, Sergio Boixo, Jarrod R McClean, and Ryan Babbush. Error mitigation via verified phase estimation. *Physical review X Quantum*, 2021.

[148] Thomas E. O'Brien, Brian Tarasinski, and Barbara Terhal. Quantum phase estimation of multiple eigenvalues for small-scale (noisy) experiments. *New Journal of Physics*, 2019.

[149] Bryan O'Gorman, Sandy Irani, James Whitfield, and Bill Fefferman. Electronic structure in a fixed basis is qma-complete. *Physical review X Quantum*, 2021.

[150] Bryan O'Gorman, Sandy Irani, James Whitfield, and Bill Fefferman. Electronic structure in a fixed basis is qma-complete. *arXiv* `2103.08215`, 2021.

[151] OpenAI. Leaderboard of openai gym environments. URL: github.com/openai/gym/wiki, 2020.

[152] Braxton Osting, Sourabh Palande, and Bei Wang. Spectral sparsification of simplicial complexes for clustering and label propagation. *Journal of Computational Geometry*, 2017.

[153] Jae-Eun Park, Brian Quanz, Steve Wood, Heather Higgins, and Ray Harishankar. Practical application improvement to quantum svm: theory to practice. *arXiv preprint*, 2020.

[154] Ori Parzanchevski and Ron Rosenthal. Simplicial complexes: spectrum, homology and random walks. *Random Structures & Algorithms*, 50:225–261, 2017.

[155] Filippo Passerini and Simone Severini. Quantifying complexity in networks: the von Neumann entropy. *International Journal of Agent Technologies and Systems (IJATS)*, 1:58–67, 2009.

[156] Jordi Pérez-Guijarro, Alba Pagès-Zamora, and Javier R Fonollosa. Relation between quantum advantage in supervised learning and quantum computational advantage. *arXiv 2304.06687*, 2023.

[157] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[158] Adrián Pérez-Salinas, David López-Núñez, Artur García-Sáez, Pol Forn-Díaz, and José I Latorre. One qubit as a universal approximant. *Physical Review A*, 104(1):012405, 2021.

[159] Stephen Piddock and Ashley Montanaro. The complexity of antiferromagnetic interactions and 2d lattices. *Quantum Information & Computation*, 17:636–672, 2017.

[160] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2, 2018.

[161] TensorFlow Quantum. Parametrized quantum circuits for reinforcement learning. https://tinyurl.com/ycy58267, 2021.

[162] Christian Reiher. The clique density theorem. *Annals of Mathematics*, 2016.

[163] Mathys Rennela, Sebastiaan Brand, Alfons Laarman, and Vedran Dunjko. Hybrid divide-and-conquer approach for tree search algorithms. *Quantum*, 7:959, 2023.

[164] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[165] Maria Schuld. Supervised quantum machine learning models are kernel methods. *arXiv preprint*, 2021.

[166] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[167] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101:032308, 2020.

[168] Maria Schuld and Nathan Killoran. Quantum machine learning in feature Hilbert spaces. *Physical review letters*, 122, 2019.

[169] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

[170] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, 2021.

[171] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103, 2021.

[172] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[173] Rocco Servedio and Steven J Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 2004.

[174] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[175] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 1998.

[176] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41, 1999.

[177] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[178] Peter W Shor and Stephen P Jordan. Estimating Jones polynomials is a complete problem for one clean qubit. *Quantum Information & Computation*, 8:681–714, 2008.

[179] David Simmons, Justin Coon, and Animesh Datta. The quantum Theil index: characterizing graph centralization using von Neumann entropy. *Journal of Complex Networks*, 6:859–876, 2018.

[180] Rolando D Somma. Quantum eigenvalue estimation via time series analysis. *New Journal of Physics*, 2019.

[181] Sathyawageeswar Subramanian and Min-Hsiu Hsieh. Quantum algorithm for estimating renyi entropies of quantum states. *Physical review A*, 2021.

[182] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[183] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[184] Yasunari Suzuki, Yoshiaki Kawase, Yuya Masumura, Yuria Hiraga, Masahiro Nakadai, Jiabao Chen, Ken M Nakanishi, Kosuke Mitarai, Ryosuke Imai, Shiro Tamiya, et al. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *arXiv preprint arXiv:2011.13524*, 2020.

[185] Ryan Sweke, Jean-Pierre Seifert, Dominik Hangleiter, and Jens Eisert. On the quantum versus classical learnability of discrete distributions. *Quantum*, 5, 2021.

[186] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

[187] Kristan Temme, Sergey Bravyi, and Jay M Gambetta. Error mitigation for short-depth quantum circuits. *Physical review letters*, 2017.

[188] Barbara M Terhal and David P DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games. *Quantum Information & Computation*, 4, 2004.

[189] Shashanka Ubaru, Ismail Yunus Akhalwaya, Mark S Squillante, Kenneth L Clarkson, and Lior Horesh. Quantum topological data analysis with linear depth and exponential speedup. *arXiv preprint*, 2021.

[190] Shashanka Ubaru, Yousef Saad, and Abd-Krim Seghouane. Fast estimation of approximate matrix ranks using spectral densities. *Neural computation*, 29:1317–1351, 2017.

[191] Johan Ugander, Lars Backstrom, and Jon Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. *Proceedings of the 22nd international conference on World Wide Web*, 2013.

[192] Gregory Valiant and Paul Valiant. Estimating the unseen: an n/log(n)-sample estimator for entropy and support size, shown optimal via new clts. *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011.

[193] Dyon van Vreumingen. Quantum feature space learning: characterisation and possible advantages. Master's thesis, Leiden University, 8 2020. `https://studenttheses.universiteitleiden.nl/handle/1887/2734545`.

[194] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*. Springer, 2015.

[195] Rui Wang, Duc Duy Nguyen, and Guo-Wei Wei. Persistent spectral graph. *International journal for numerical methods in biomedical engineering*, 36:e3376, 2020.

[196] Jordi Weggemans, Marten Folkertsma, and Chris Cade. Guidable local hamiltonian problems with implications to heuristic ansatze state preparation and the quantum pcp conjecture. *arXiv 2302.11578*, 2023.

[197] Tzu-Chieh Wei, Michele Mosca, and Ashwin Nayak. Interacting boson problems can be qma hard. *Physical review letters*, 104, 2010.

[198] Lilian Weng. Policy gradient algorithms. URL: lilianweng.github.io/lil-log, 2018.

[199] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[200] Pawel Wocjan and Shengyu Zhang. Several natural BQP-complete problems. *arXiv preprint*, 2006.

[201] Michael M Wolf. Mathematical foundations of supervised learning. `https://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MA4801_2020S/ML_notes_main.pdf`, 2020.

[202] Shaojun Wu, Shan Jin, Dingding Wen, and Xiaoting Wang. Quantum reinforcement learning in continuous action space. *arXiv preprint arXiv:2012.10711*, 2020.

Summary

In this thesis, the contribution of quantum computers to machine learning has been explored, falling under the domain of Quantum Machine Learning. This domain promises novel perspectives and methods for addressing complex issues in machine learning by leveraging the unique capabilities of quantum computers. Quantum computers differ fundamentally from classical computers in their use of quantum mechanics, resulting in unique computational abilities. Unlike classical bits, which can be either 0 or 1, quantum computers employ qubits that can exist as both 0 and 1 simultaneously due to a phenomena called "superposition". Moreover, "entanglement" enables quantum computers to bring qubits into a mutually dependent state, allowing for complex parallel computations. Within the research domain of quantum machine learning, this thesis has explored various proposals regarding how quantum computers can enhance certain components of machine learning.

The first proposal examined in this thesis is the applications of quantum computers in topological data analysis. Topological data analysis is an innovative approach that extracts robust properties from datasets by understanding their inherent "shape". The focus was specifically on quantum algorithms for linear algebra, aiming to determine if they could offer superpolynomial speedups compared to classical methods. The results of this thesis demonstrated that existing quantum algorithms, along with algorithms developed in this thesis (with applications in numerical linear algebra, machine learning, and complex network analysis), solve problems that are classically deemed intractable according to widespread assumptions in complexity theory. Specifically, these results showed that the speedup provided by quantum algorithm methods for topological data analysis is resilient against the development of faster classical algorithms. These findings shed light on the potential power of quantum computers in addressing complex problems in topological data analysis, machine learning, and network analysis.

Another aspect of this thesis is the investigation of structural risk minimization in the context of quantum machine learning models. Structural Risk Minimization (SRM) is a principle in machine learning that seeks a balance between model complexity and performance on new data. It involves selecting a model from a given family by striking a balance between training error (how well the model fits the training data) and a complexity term (penalizing overly complex models). The focus on this thesis was on understanding the impact of certain design choices in machine learning models based on parameterized quantum circuits. In essence, a parameterized quantum circuit can be seen as a quantum variant of a neural network, manipulating a set of

qubits depending on parameters and seeking the right parameters for the problem at hand. The research in this thesis explored whether important settings within new quantum machine learning models based on parameterized quantum circuits can be identified, influencing both complexity measures and training error, which is crucial for the successful implementation of SRM. In particular, this thesis demonstrated how to construct new quantum machine learning models with favorable performance guarantees based on the SRM principle. These insights contribute to optimizing quantum machine learning models, enhancing their performance.

The subsequent topic explored in this thesis was how quantum computers can improve reinforcement learning. Reinforcement learning revolves around learning through interaction to achieve a specific goal, typically modeled by the interaction between an "agent" (the learner) and an "environment". The agent takes continuous actions, and after each action, the environment responds by providing the agent with a "reward". The goal of the agent is to maximize these rewards over time. In this thesis, quantum models based on parameterized quantum circuits were introduced within reinforcement learning. Notably, these models demonstrated comparable performance to traditional classical models (such as deep neural networks), while showing superior performance in certain scenarios. These results suggest that quantum models can be a powerful tool for solving complex problems in reinforcement learning.

The final part of the research in this thesis focused on identifying learning tasks within computational learning theory for which quantum learning algorithms have exponential advantages over classical algorithms. Computational learning theory is a mathematical framework introduced in the 1990s with the aim of providing formal arguments about why and how machine learning can be successful in practice. This thesis delved deep into the details to precisely define what it means for a quantum learner to have an exponential advantage over its classical counterparts. It then explored previous cases of exponential advantages, identifying the exact source of classical complexity and the advantage of quantum models. Finally, it investigated the general belief that quantum machine learning performs best in scenarios with data generated by quantum processes. This involved establishing a framework in which any problem with data generated by quantum processes can lead to an exponential quantum advantage. This opens doors to the application of quantum computing in specific scenarios where classical algorithms fall short.

In summary, this thesis provides an exploration of quantum machine learning, applying the unique capabilities of quantum computers to diverse domains within machine learning. The proposals researched ranged from the application of quantum algorithms in topological data analysis, understanding the influence of design choices on structural risk minimization and the introduction of quantum models in reinforcement learning to examinations of learning tasks in computational learning theory where quantum learning algorithms can offer exponential advantages. As a whole, this thesis contributes to the understanding of the promising role of quantum computers in addressing complex problems within machine learning.

Samenvatting

In dit proefschrift is onderzocht hoe quantumcomputers kunnen bijdragen aan machine learning, hetgeen valt binnen het vakgebied Quantum Machine Learning. Dit vakgebied belooft nieuwe perspectieven en methoden voor het oplossen van complexe problemen in machine learning door gebruik te maken van de unieke mogelijkheden van quantumcomputers. Quantumcomputers verschillen fundamenteel van klassieke computers door hun gebruik van kwantummechanica, wat leidt tot unieke rekenmogelijkheden. In tegenstelling tot klassieke bits, die ofwel 0 of 1 kunnen zijn, maken quantumcomputers gebruik van qubits die dankzij "superpositie" zowel 0 als 1 tegelijk kunnen zijn. Bovendien stellen "verstrengelingen" quantumcomputers in staat om qubits in een onderling afhankelijke toestand te brengen, wat complexe parallelle berekeningen mogelijk maakt. Binnen het onderzoeksgebied quantum machine learning heeft dit proefschrift verschillende voorstellen onderzocht met betrekking tot manieren waarop quantumcomputers delen van machine learning kunnen verbeteren.

Het eerste voorstel dat dit proefschrift heeft onderzocht, was het verkennen van de toepassingen van quantum computers in topologische data-analyse. Topologische data-analyse is een innovatieve benadering om robuuste eigenschappen uit datasets te halen door de inherente "vorm" ervan te begrijpen. Binnen dit kader heeft dit proefschrift zich specifiek gericht op quantumalgoritmen voor lineaire algebra, met als doel te begrijpen of ze superpolynomiale versnellingen kunnen bieden ten opzichte van klassieke methoden. De resultaten in dit proefschrift hebben aangetoond dat bestaande quantumalgoritmes voor topologische data-analyse, samen met een aantal nieuwe algoritmes die in dit proefschrift zijn ontwikkeld (met toepassingen in numerieke lineaire algebra, machine learning en complexe netwerkanalyse), problemen oplossen die klassiek gezien niet efficiënt oplosbaar zijn volgens wijdverspreide aannames in de complexiteitstheorie. Concreet hebben deze resultaten laten zien dat de versnelling die de methoden van het quantumalgoritme voor topologische data-analyse bieden, bestand is tegen het ontwikkelen van eventuele snellere klassieke algoritmes. Deze resultaten geven inzicht in de potentiële kracht van quantumcomputers bij het oplossen van complexe problemen in topologische data-analyse, machine learning en netwerkanalyse.

Een ander aspect van dit proefschrift is het onderzoek naar structural risk minimization in het kader van quantum machine learning-modellen. Structural Risk Minimization (SRM) is een principe in machine learning dat de balans zoekt tussen modelcomplexiteit en prestaties op nieuwe data. Het houdt in dat je een model kiest uit een bepaalde familie van modellen, waarbij je een evenwicht zoekt tussen

trainingsfout (hoe goed het model past bij de trainingsdata) en een complexiteitsterm (die té complexe modellen bestraft). Hierbij lag de focus op het begrijpen van de invloed van bepaalde ontwerpkeuzes in machine learning modellen gebaseerd op geparametriseerde quantumcircuits. Kortweg, een geparametriseerd quantumcircuit kun je zien als een quantumvariant van een neuraal netwerk, waarbij men een aantal qubits manipuleert op een manier die afhangt van enkele parameters, en waarbij je op zoek gaat naar de juiste parameters voor het probleem dat je probeert op te lossen. Het onderzoek in dit proefschrift vroeg zich af of we belangrijke instellingen binnen nieuwe quantum-machine learning-modellen gebaseerd op parameteriseerbare quantumcircuits kunnen identificeren, die zowel complexiteitstermen als de trainingsfout beïnvloeden, wat cruciaal is voor het succesvol implementeren van SRM. In het bijzonder heeft dit proefschrift gedemonstreerd hoe we nieuwe quantummachineleermodellen kunnen construeren met gunstige prestatiegaranties op basis van het principe van SRM. Deze inzichten kunnen bijdragen aan het optimaliseren van quantum machine learning-modellen, waardoor ze beter presteren.

Het volgende onderwerp dat dit proefschrift onderzocht, was hoe quantumcomputers kunnen helpen bij "reinforcement learning". Reinforcement learning draait om leren door interactie om een bepaald doel te bereiken. Dit wordt meestal gemodelleerd door de interactie tussen een "agent" (de leerling) en een "omgeving". De agent neemt voortdurend acties, en na elke actie reageert de omgeving door de agent een "beloning" te geven. Het doel van de agent is om deze beloningen in de loop van de tijd te maximaliseren. Binnen de reinforcement learning zijn er in dit proefschrift quantummodellen geïntroduceerd die gebaseerd zijn op geparametriseerde quantumcircuits. Opmerkelijk was dat deze modellen vergelijkbare prestaties konden leveren als traditionele klassieke model (zoals diepe neurale netwerken), terwijl ze in bepaalde scenario's superieure prestaties vertoonden. Deze ontdekking suggereert dat quantummodellen een krachtig instrument kunnen zijn bij het oplossen van complexe problemen in reinforcement learning.

Het laatste deel van het onderzoek in dit proefschrift betrof de identificatie van leertaken binnen de "computational learning theory" waarvoor quantumleeralgoritmen exponentiële voordelen hebben ten opzichte van klassieke algoritmen. Computational learning theory is een wiskundig framework geïntroduceerd in de jaren '90 met het doel formele argumenten te verschaffen over waarom en hoe machine learning in de praktijk succesvol kan zijn. Eerst dook dit proefschrift diep in de details om precies te definiëren wat het betekent dat een quantum-leerling een exponentieel voordeel heeft ten opzichte van zijn klassieke tegenhanger. Daarna onderzocht het eerdere gevallen van exponentiele voordelen, waarbij we de precieze bron van klassieke complexiteit en het voordeel van quantum modellen identificeerden. Ten slotte onderzocht het de algemene opvatting dat quantum machine learning het beste presteert in scenario's met data gegenereerd door quantum-processen. Hierbij hebben we een kader opgesteld waarin elk probleem met data gegenereerd door quantum-processen kan leiden tot een exponentieel quantumvoordeel. Dit opent deuren naar de toepassing van quantum computing in specifieke scenario's waarin klassieke algoritmen tekortschieten.

Samenvattend biedt dit proefschrift een verkenning van Quantum Machine Learning, waarbij de unieke capaciteiten van quantumcomputers worden toegepast op uiteenlopende domeinen binnen machine learning. Van de toepassing van quantu-

malgoritmen in topologische data-analyse tot het begrijpen van de invloed van ontwerpkeuzes op structural risk minimization, en de introductie van quantummodellen in reinforcement learning. Het onderzoek sluit af met een blik op leertaken in de computational learning theory, waar quantumleeralgoritmen exponentiële voordelen kunnen bieden. In zijn geheel draagt dit proefschrift bij aan het begrip van de veelbelovende rol van quantumcomputers in het oplossen van complexe problemen binnen machine learning.

Curriculum Vitae

Casper was born on the 7th of September 1994 in Elst. He obtained a BSc in computer science and mathematics from the University of Amsterdam in 2015, and an MSc in mathematics from the same institution in 2018. Under the supervision of Vedran Dunjko he completed his PhD at Leiden University. Presently, he still works at Leiden University, where he is a post-doctoral researcher in the applied Quantum algorithms (aQa) group.

Appendix A

Towards quantum advantage via topological data analysis

A.1 LLSD is DQC1-hard

Following the definition of [178], for any problem $L \in \mathsf{DQC1}$ and every $x \in L$, there exists a quantum circuit U of depth $T \in \mathcal{O}(\mathrm{poly}(|x|))$ that operates on $n \in \mathcal{O}(\mathrm{poly}(|x|))$ qubits such that

- $x \in L_{yes} \implies p_0 \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(|x|)}$,
- $x \in L_{no} \implies p_0 \leq \frac{1}{2} - \frac{1}{\mathrm{poly}(|x|)}$,

where $p_0 = \mathrm{Tr}\left[(|0\rangle\langle 0| \otimes I) U \rho U^\dagger\right]$ and $\rho = |0\rangle\langle 0| \otimes I/2^{n-1}$. From this it can be gathered that if we can estimate p_0 to within $1/\mathrm{poly}(|x|)$ additive precision, then we can solve L.

For a positive semidefinite matrix $H \in \mathbb{C}^{2^n \times 2^n}$ and a threshold $b \in \mathbb{R}_{\geq 0}$, we define the *normalized subtrace* of H up to b as

$$\overline{\mathrm{Tr}_b}(H) = \frac{1}{2^n} \sum_{0 \leq \lambda_k \leq b} \lambda_k,$$

where $\lambda_0 \leq \cdots \leq \lambda_{2^n-1}$ denote the eigenvalues of H. The following result by Brandão shows that if we can estimate the normalized subtrace $\overline{\mathrm{Tr}_b}$ of log-local Hamiltonians up to additive inverse polynomial precision, then we can solve any problem in $\mathsf{DQC1}$. In other words, estimating $\overline{\mathrm{Tr}_b}$ of log-local Hamiltonians up to additive inverse polynomial precision is $\mathsf{DQC1}$-hard.

Proposition 31 (Brandão [40])**.** *Given as input a description of an n-qubit quantum circuit U of depth $T \in \mathcal{O}(poly(n))$ together with a polynomial $r(n)$, one can efficiently construct a log-local Hamiltonian $H \in \mathbb{C}^{T2^n \times T2^n}$ and a threshold $b \in \mathcal{O}(poly(n))$ such*

that

$$\left|\overline{\mathrm{Tr}_b}(H) - p_0\right| \leq \frac{1}{r(n)}, \tag{A.1}$$

where $p_0 = \mathrm{Tr}\left[(|0\rangle\langle 0| \otimes I)U\rho U^\dagger\right]$ *and* $\rho = |0\rangle\langle 0| \otimes I/2^{n-1}$*. Moreover,* H *also satisfies:*

(iii) H *is positive semidefinite.*

(iv) There exists a $\delta \in \Omega\left(1/poly(n)\right)$ *such that* H *has no eigenvalues in the interval* $[b, b+\delta]$*.*

Remark(s). *The Hamiltonian in the above proposition is obtained by applying Kitaev's circuit-to-Hamiltonian construction directly to the circuit* U*. However, instead of adding penalty terms* $\sum_{i=1}^{n} |1\rangle\langle 1|_i \otimes |0\rangle\langle 0|_{\text{clock}}$ *to constrain the initial state of all qubits* $i = 1, \ldots, n$*, we only add a single penalty term* $|1\rangle\langle 1|_1 \otimes |0\rangle\langle 0|_{\text{clock}}$ *that constraints the first qubit to* $|0\rangle$ *(i.e., a clean qubit) and leaves the rest unconstrained to emulate the maximally mixed state.*

We will show that we can efficiently estimate the normalized subtrace $\overline{\mathrm{Tr}_b}$ in Equation A.1 to within additive inverse polynomial precision using an oracle for LLSD. To be precise, we show that we can estimate this normalized subtrace to within additive inverse polynomial precision using a polynomial amount of nonadaptive queries to an oracle for LLSD (whose input is restricted to log-local Hamiltonians), together with polynomial-time classical preprocessing of the inputs and postprocessing of the outputs. In other words, we provide a polynomial-time truth-table reduction from the problem of estimating $\overline{\mathrm{Tr}_b}$ to LLSD. We gather this in Lemma 32, which together with Proposition 31 shows that LLSD with the input restricted to log-local Hamiltonians is DQC1-hard under polynomial-time truth-table reductions.

Lemma 32. *Given as input* $H \in \mathbb{C}^{T2^n \times T2^n}$ *and* $b \in \mathcal{O}\left(poly(n)\right)$ *as described in Proposition 31, together with a polynomial* $q(n)$*, one can compute a quantity* Λ *that satisfies*

$$|\Lambda - \overline{\mathrm{Tr}_b}(H)| \leq \frac{1}{q(n)},$$

using a polynomial number of queries to an oracle for LLSD*, together with polynomial-time classical preprocessing of the inputs and postprocessing of the outputs.*

Proof. Define $\Delta = (3q(n))^{-1}$, $M = b/\Delta$, $\epsilon = (6Mbq(n))^{-1}$ and let $\delta < \Delta/3$ be such that H has no eigenvalues in the interval $[b, b+\delta]$. Also, define the thresholds $x_j = (j+1)\Delta$, for $j = 0, \ldots, M-1$. Next, denote by $\hat{\chi}_j$ the outcome of LLSD with threshold $b = x_j$ and precision parameters δ, ϵ as defined above. That is, $\hat{\chi}_j$ is an estimate of $\hat{y}_j$ to within additive accuracy ϵ, where

$$\hat{y}_j = N_H(0, x_j) + \hat{\gamma}_j, \text{ with } 0 \leq \hat{\gamma}_j \leq N_H(x_j, x_j + \delta).$$

Subsequently, define $\chi_0 = \hat{\chi}_0$, $y_0 = \hat{y}_0$ and

$$y_j = \hat{y}_j - \hat{y}_{j-1}, \tag{A.2}$$
$$\chi_j = \hat{\chi}_j - \hat{\chi}_{j-1}, \tag{A.3}$$

for $1 \leq j \leq M-1$. Finally, define the estimate

$$\Lambda = \sum_{j=0}^{M-1} \chi_j x_j. \tag{A.4}$$

We will show that Λ is indeed an estimate of $\overline{\mathrm{Tr}_b}(H)$ to within additive precision $\pm 1/q(n)$. To do so, we define $\gamma_0 = \hat{\gamma}_0$ and $\gamma_j = \hat{\gamma}_j - \hat{\gamma}_{j-1}$ for $1 \leq j \leq M-1$, and we define and expand

$$\Gamma = \sum_{j=0}^{M-1} y_j x_j = \sum_{j=0}^{M-1} \left(N_H(x_{j-1}, x_j) + \gamma_j\right) x_j = \underbrace{\sum_{j=0}^{M-1} N_H(x_{j-1}, x_j) x_j}_{\mathcal{B} :=} + \underbrace{\sum_{j=0}^{M-1} \gamma_j x_j}_{\mathcal{E}_{bin} :=}.$$

We start by upper-bounding the magnitude of the $\mathcal{E}_{bin}$ term. To do so, we rewrite

$$\begin{aligned}
\mathcal{E}_{bin} &= \sum_{j=0}^{M-1} \gamma_j x_j = \hat{\gamma}_0 x_0 + \sum_{j=1}^{M-1} (\hat{\gamma}_j - \hat{\gamma}_{j-1}) x_j \\
&= \sum_{j=0}^{M-1} \hat{\gamma}_j x_j - \sum_{j=1}^{M-1} \hat{\gamma}_{j-1} x_j \\
&= \sum_{j=0}^{M-1} \hat{\gamma}_j x_j - \sum_{j=1}^{M-1} \hat{\gamma}_{j-1} (x_{j-1} + \Delta) \\
&= \sum_{j=0}^{M-1} \hat{\gamma}_j x_j - \sum_{j=1}^{M-1} \hat{\gamma}_{j-1} x_{j-1} - \Delta \sum_{j=1}^{M-1} x_{j-1} \\
&= \underbrace{\hat{\gamma}_{M-1}}_{=0} x_{M-1} - \Delta \underbrace{\sum_{j=1}^{M-1} \hat{\gamma}_{j-1}}_{\leq 1},
\end{aligned}$$

and we conclude that $|\mathcal{E}_{bin}| \leq \Delta$. Next, we upper-bound the absolute difference of $\mathcal{B}$ and $\overline{\mathrm{Tr}_b}(H)$.

$$\left|\mathcal{B} - \overline{\mathrm{Tr}_b}(H)\right| = \left| \sum_{j=0}^{M-1} N_H(x_{j-1}, x_j) x_j - \overline{\mathrm{Tr}_b}(H) \right| \leq \sum_{j=0}^{M-1} \Delta \cdot N_H(x_{j-1}, x_j) \leq \Delta.$$

Finally, we upper-bound the absolute difference between Λ and Γ.

$$|\Lambda - \Gamma| = \left| \sum_{j=0}^{M-1} (\chi_j - y_j) x_j \right| \leq \left| \sum_{j=0}^{M-1} 2\epsilon x_j \right| \leq M \cdot 2\epsilon \cdot b = \frac{1}{3q(n)}$$

Combining all of the above we find that

$$\begin{aligned} |\Lambda - \overline{\mathrm{Tr}_b}(H)| &\leq |\Lambda - \Gamma| + |\Gamma - \overline{\mathrm{Tr}_b}(H)| \\ &\leq |\Lambda - \Gamma| + |\mathcal{B} - \overline{\mathrm{Tr}_b}(H)| + |\mathcal{E}| \\ &\leq \frac{1}{3q(n)} + \Delta + \Delta = \frac{1}{q(n)}. \end{aligned}$$

□

A.2 Quantum algorithms for SUES and LLSD

In this section we give a quantum algorithm for SUES and a quantum algorithm for LLSD. Moreover, if the input is a log-local Hamiltonian, then the quantum algorithms we give in this section turn out to be a DQC1 algorithm in the case of LLSD, and a $\mathsf{DQC1}_{\log n}$ algorithm in the case of SUES. That is, if the input is a log-local Hamiltonian, then these algorithms can be implemented in the one clean qubit model, where in the case of SUES we need to measure logarithmically many qubits (as opposed to just one), in order to read out the entire encoding of the eigenvalue.

By scaling the input $H' = H/\Lambda$, where $\Lambda \in \mathcal{O}(\mathrm{poly}(n))$ is an upper bound on the largest eigenvalue of H, we can assume without loss of generality that $||H|| < 1$. Moreover, we will use that allowing up to $\mathcal{O}(\log(n))$ clean qubits does not change the class DQC1 [178]. That is, the class of problems that can be solved in polynomial time using the one clean qubit model of computation is the same as the class of problems that can be solved in polynomial time using the k-clean qubit model of computation, for $k \in \mathcal{O}(\log n)$. We use this result since the quantum algorithms we describe need additional ancilla qubits, which have to be initialized in the all-zeros state and hence be 'clean'.

A.2.1 Quantum algorithm for SUES

In this section we describe a quantum algorithm for SUES, which when the input is restricted to log-local Hamiltonians turns out to be a $\mathsf{DQC1}_{\log n}$ algorithm. That is, if the input is a log-local Hamiltonian, then this algorithm can be implemented using the one clean qubit model of computation where we are allowed to measure logarithmically many of the qubits at the end, in order to read out the encoding of the eigenvalue.

The quantum algorithm for SUES implements an approximation of the unitary e^{iH} using Hamiltonian simulation, to which it applies quantum phase estimation with the eigenvector register starting out in the maximally mixed state. In the remainder of this section we will show that we can control the errors such that quantum phase

estimation applied to the approximation of e^{iH} outputs the corresponding eigenvalue of H up to precision $\delta \in \Omega\left(1/\text{poly}(n)\right)$, with error probability $\mu \in \Omega\left(1/\text{poly}(n)\right)$. Because the maximally mixed state is in a given eigenstate with uniform probabilities over all eigenstates, this shows that this quantum algorithm is able to output a sample from a (δ, μ)-approximation of the uniform distribution over the eigenvalues of H.

Errors can arise in two places, namely due to the imprecisions of the unitary implemented by the Hamiltonian simulation and due to the imprecisions of estimating eigenvalues using quantum phase estimation. First, we discuss the errors of the Hamiltonian simulation step. Given sparse access to H, we can implement a unitary V such that

$$||V - e^{iH}|| < \gamma, \tag{A.5}$$

in time $\mathcal{O}\left(\text{poly}\left(n, \log(1/\gamma)\right)\right)$ [133]. The algorithms for Hamiltonian simulation of matrices specified by an oracle unfortunately require more than $\mathcal{O}\left(\log n\right)$ ancilla qubits, which implies that they can not be implemented using the one clean qubit model. On the other hand, if H is a log-local Hamiltonian, then Hamiltonian simulation techniques based on the Trotter-Suzuki formula can implement a unitary V that satisfies Equation A.5 in time $\mathcal{O}\left(\text{poly}(n, 1/\gamma)\right)$ [129], while only using a constant number of ancilla qubits [51]. Therefore, if H is a log-local Hamiltonian, then using the one clean qubit model we can implement a unitary V that satisfies Equation A.5 in time $\mathcal{O}\left(\text{poly}(n, 1/\gamma)\right)$.

Denote by λ_j and ζ_j the output of the quantum phase estimation routine (where for now we assume that it works perfectly, i.e., introduces no error) when run using e^{iH} and V, respectively. Then, by Equation A.5 we have

$$|e^{i\lambda_j} - e^{i\zeta_j}| \leq \gamma,$$

where we assume that $|\lambda_j - \zeta_j| \leq \pi$ by adding multiples of 2π to λ_j if necessary. With some algebra [51], we can show that this implies that

$$|\lambda_j - \zeta_j| \leq \pi\gamma/2.$$

Choosing the accuracy of the Hamiltonian simulation to be $\gamma = \delta/\pi \in \Omega\left(1/\text{poly}(n)\right)$, we get that

$$|\lambda_j - \zeta_j| < \delta/2. \tag{A.6}$$

Next, we will consider the errors that arise from using the quantum phase estimation routine to estimate the eigenvalues ζ_j of the unitary V. The quantum phase estimation routine requires a register of t ancilla qubits (also called the eigenvalue register), onto which the eigenvalue will be loaded. If we take

$$t = \log(2/\delta) + \lceil \log(2 + 1/2\mu) \rceil \in \mathcal{O}\left(\log n\right)$$

qubits in the eigenvalue register, then quantum phase estimation outputs an estimate $\overline{\zeta_j}$ that satisfies

$$|\overline{\zeta_j} - \zeta_j| \leq \delta/2,$$

with probability at least $(1-\mu)$ [145]. In particular, with probability at least $(1-\mu)$ this estimate satisfies

$$|\overline{\zeta_j} - \lambda_j| \leq |\overline{\zeta_j} - \zeta| + |\zeta_j - \lambda_j| \leq \delta.$$

This requires $\widetilde{\mathcal{O}}(2^t) = \widetilde{\mathcal{O}}(\text{poly}(n))$ applications of the unitary V, each of which can be implemented in $\mathcal{O}(\text{poly}(n))$ time as discussed above. In addition, this quantum phase estimation step requires only $\mathcal{O}(\log n)$ ancilla qubits, making it possible to be implemented using the one clean qubit model.

In conclusion, both the Hamiltonian simulation and the quantum phase estimation can be implemented up to the required precision in time $\mathcal{O}(\text{poly}(n))$. Moreover, if H is a log-local Hamiltonian, then this can be done using the one clean qubit model. Finally, to read out the encoding of the eigenvalue, we need to measure the $t \in \mathcal{O}(\log(n))$ qubits in the eigenvalue register, resulting in a $\mathsf{DQC1}_{\log n}$ algorithm for SUES if the input is a log-local Hamiltonian.

A.2.2 Quantum algorithm for LLSD

In this section, we will describe two quantum algorithms for LLSD, both of which turn into DQC1 algorithms when the input is restricted to log-local Hamiltonians. That is, if the input is a log-local Hamiltonian, then these algorithms can be implemented using the one clean qubit model of computation.

Counting eigenvalues below the threshold

A straightforward approach is to solving LLSD is to repeatedly sample from the output of SUES and then compute the fraction of samples that lie below the given threshold. The downside of this is that it requires one to measure the entire eigenvalue register consisting of logarithmically many qubits, which is prohibitive as we are only allowed to measure a single qubit in the one clean qubit model. This can be circumvented by simply adding an extra clean qubit and flipping this qubit conditioned on the state in the eigenvalue register being smaller than the given threshold. This extra qubit will be flipped with probability close to the low-lying spectral density, allowing us to obtain a solution to LLSD by only measuring this single qubit. Moreover, if H is a log-local Hamiltonian, then this 'fully quantum' algorithm can be implemented using the one clean qubit model, as it requires only a few more additional clean qubits on top of those required for the quantum algorithm for SUES discussed in Section A.2.1.

Note that the outcome probabilities of this 'fully quantum' algorithm are identical to those obtained by measuring the entire eigenvalue register, followed by classical counting of the number of samples below the given threshold. Consequently, the same error analysis applies in both cases. In the rest of this section we will discuss the error analysis of classically counting the number of samples below the given threshold.

Let $m = \epsilon^{-2} \in \mathcal{O}(\text{poly}(n))$ and draw for $j = 1, \ldots, m$ a sample $\overline{\lambda}_{k_j}$ from SUES

with $\delta/2$ as the precision parameter. Next, compute

$$\chi_j = \begin{cases} 1 \text{ if } \overline{\lambda}_{k_j} \in (a - \delta/2, b + \delta/2), \\ 0 \text{ otherwise.} \end{cases}$$

For now we assume that all samples $\overline{\lambda}_{k_j}$ were *correctly sampled*, i.e., each k_j is drawn uniformly at random from the set $\{0, \ldots, 2^n - 1\}$ and $|\lambda_{k_j} - \overline{\lambda}_{k_j}| \leq \delta/2$, where λ_{k_j} denotes the eigenvalue of which $\overline{\lambda}_{k_j}$ is an estimate. We now show that under this assumption the quantity

$$\chi := \frac{1}{m} \sum_{j=1}^{m} \chi_j \tag{A.7}$$

is, with high probability, a correct solution to LLSD. By the Chernoff-Hoeffding inequality χ is, with high probability, an estimate to within additive precision ϵ of

$$y := \Pr_{\overline{\lambda} \sim \text{SUES}} \left[\overline{\lambda} \in (a - \delta/2, b + \delta/2) \right],$$

where the probability is taken over the $\overline{\lambda}$ being correctly sampled from SUES. Because we assume that the $\overline{\lambda}$ are correctly samples from SUES, we know that they satisfy $|\lambda - \overline{\lambda}| \leq \delta/2$, where λ denotes the eigenvalue of which $\overline{\lambda}$ is an estimate. This implies that

(v) $y \leq \Pr_{\lambda \sim_U \{\lambda_j\}_{j=1}^{2^n}} \left[\lambda \in (a - \delta, b + \delta) \right] = N_H(a - \delta, b + \delta)$,

(vi) $y \geq \Pr_{\lambda \sim_U \{\lambda_j\}_{j=1}^{2^n}} \left[\lambda \in (a, b) \right] = N_H(a, b)$,

where the probabilities are taken over the λ being sampled uniformly from the set of all eigenvalues of H. Combining this with the Chernoff-Hoeffding inequality, we find that χ is, with high probability, an estimate of y up to additive precision ϵ, where y satisfies

$$N_H(a, b) \leq y \leq N_H(a - \delta, b + \delta).$$

That is, if all $\overline{\lambda}_{k_j}$ were sampled correctly from SUES, then χ is with high probability a correct solution to LLSD.

Finally, we consider the probability that all samples $\overline{\lambda}_{k_j}$ were indeed sampled correctly. By the union bound this probability is at least $1 - m\mu$, where μ denotes the sampling error probability of SUES. Because $m \in \mathcal{O}(\text{poly}(n))$, we can choose $\mu \in \Omega\left(1/\text{poly}(\epsilon^{-2}, n)\right) = \Omega\left(1/\text{poly}(n)\right)$ such that all our samples are sampled correctly with probability close to 1. Therefore, we conclude that the χ defined in Equation A.7 is a correct solution to LLSD, with probability close to 1. Moreover, χ can be obtained from a polynomial number of samples from SUES, and can therefore be computed in time $\mathcal{O}(\text{poly}(n))$.

Using trace estimation of eigenvalue transform

In our paper, we use a result of Cade & Montanaro [51] to argue that the complexity of estimating the spectral entropy of a Hermitian matrix is closely related to DQC1. In their work, Cade & Montanaro describe a DQC1 algorithm can estimate traces of general functions of Hermitian matrices (i.e., beyond spectral entropies). This algorithm could also be used to extract other interesting properties encoded in the spectrum of the combinatorial Laplacian. To illustrate this and connect even further to this line of work, we provide an alternative algorithm for LLSD based on this algorithm. The main result we will utilize is the following Lemma.

Lemma 33 (Cade & Montanaro [51])**.** *For a log-local Hamiltonian $H \in \mathbb{C}^{2^n \times 2^n}$, and any log-space polynomial-time computable function $f : I \to [-1, 1]$ (where I contains the spectrum of H) that is Lipschitz continuous with constant K (i.e., $|f(x) - f(y)| \leq K|x - y|$ for all $x, y \in I$), there exists a* DQC1 *algorithm to estimate $\mathrm{Tr}(f(H))/2^n = \sum_j f(\lambda_j)/2^n$ up to additive accuracy $\epsilon(K + 1)$, where λ_j denote the eigenvalues of H, and $\epsilon \in \Omega(1/poly(n))$.*

It is clear that if the function f is the step-function with threshold $b + \delta/2$ given by

$$f(x) = \begin{cases} 1 \text{ if } x \leq b + \delta/2, \\ 0 \text{ otherwise,} \end{cases}$$

then the quantity estimated by the algorithm of Lemma 33 is a correct solution to LLSD. However, as this function is not Lipschitz continuous, we will use a smooth approximation based on the following lemma.

Lemma 34 (Smooth approximation of the sign function)**.** *Let $\delta > 0$, $\epsilon \in (0, 1)$ and $\gamma = \frac{\delta\sqrt{2\epsilon - \epsilon^2}}{1-\epsilon}$. Then, the function $g_\gamma(x) = \frac{x}{\sqrt{x^2+\gamma^2}}$ satisfies*

(vii) for all $x \in [-2, 2]$: $-1 \leq g_\gamma(x) \leq 1$,

(viii) for all $x \in [-2, 2] \backslash (-\delta, \delta)$: $|g_\gamma(x) - sgn(x)| \leq \epsilon$, and

(ix) $\sup_{x\in[-2,2]} |g'_\gamma(x)| \leq \frac{1}{\gamma}$.

Proof. *(i)* It is clear that for all $x \in [-2, 2]$ we have: $-1 \leq g_\gamma(-2) \leq g_\gamma(x) \leq g_\gamma(2) \leq 1$.

(ii) Let $x \in (\delta, 2]$, then

$$|g_\gamma(x) - \mathrm{sgn}(x)| = |g_\gamma(x) - 1| \leq |g_\gamma(\delta) - 1| = \epsilon.$$

For $x \in [-2, -\delta)$ we note that

$$|g_\gamma(x) - \mathrm{sgn}(x)| = |g_\gamma(x) + 1| \leq |g_\gamma(-\delta) + 1| = | - (g_\gamma(\delta) - 1)| = \epsilon.$$

(iii) It is clear that: $\sup_{x\in[-2,2]} |g'_\gamma(x)| = |g'_\gamma(0)| = \frac{1}{\gamma}$.

□

Let $\gamma = \frac{(\delta/2)\sqrt{2\epsilon-\epsilon^2}}{1-\epsilon}$ and define $g = g_\gamma$ as in Lemma 34. We define our smooth approximation of the step-function by

$$\hat{f}(x) = \frac{g(-x+b')+1}{2},$$

where $b' = b + \delta/2$. By Lemma 34 we know that $\hat{f}$ is Lipschitz continuous on $[0,1]$ with constant $1/\gamma \in \mathcal{O}(\text{poly}(n))$, and that it satisfies

- for all $x \in [0,1] : 0 \leq \hat{f}(x) \leq 1$, and
- for all $x \in [0,1] \backslash (b, b+\delta) : |\hat{f}(x) - f(x)| \leq \epsilon/2$.

Subsequently, we define our estimation objective

$$y = \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j \in [0,b]} f(\lambda_j) + \sum_{j\,:\,\lambda_j \in [b,b+\delta]} \hat{f}(\lambda_j)\right),$$

and we note that y indeed satisfies $N_H(0,b) \leq y \leq N_H(0,b+\delta)$, since

$$\begin{aligned} y &= \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j \in [0,b]} f(\lambda_j) + \sum_{j\,:\,\lambda_j \in [b,b+\delta]} \hat{f}(\lambda_j)\right) \\ &= N_H(0,b) + \underbrace{\frac{1}{2^n} \sum_{j\,:\,\lambda_j \in [b,b+\delta]} \underbrace{\hat{f}(\lambda_j)}_{\in[0,1]}}_{\in [0, N_H(b,b+\delta)]}. \end{aligned}$$

Now our goal is to use Lemma 33 to obtain an ϵ-approximation of y. To this end, we first define

$$\Lambda = \frac{1}{2^n}\sum_{j=1}^{2^n} \hat{f}(\lambda_j),$$

and we upper-bound the absolute difference between y and Λ as follows

$$\begin{aligned} \left|\Lambda - y\right| &\leq \frac{1}{2^n}\left|\sum_{j\,:\,\lambda_j \in [0,b]} \left(\hat{f}(\lambda_j) - f(\lambda_j)\right) + \sum_{j\,:\,\lambda_j \in [b+\delta,1]} \hat{f}(\lambda_j)\right| \\ &\leq \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j \in [0,b]} \left|\hat{f}(\lambda_j) - f(\lambda_j)\right| + \sum_{j\,:\,\lambda_j \in [b+\delta,1]} \left|\hat{f}(\lambda_j)\right|\right) \\ &\leq \frac{1}{2^n}\left(\sum_{j\,:\,\lambda_j \in [0,b]} \epsilon/2 + \sum_{j\,:\,\lambda_j \in [b+\delta,1]} \epsilon/2\right) \leq \epsilon/2. \end{aligned}$$

Finally, let χ be the output of the algorithm of Lemma 33 applied to our function

$\hat{f}$ with precision parameter $\hat{\epsilon} = \epsilon/(2(K+1)) \in \Omega(1/\text{poly}(n))$. In particular, χ satisfies $|\chi - \Lambda| \leq \hat{\epsilon}(K+1) = \epsilon/2$. We conclude that χ is a correct solution to LLSD since

$$|\chi - y| \leq |\chi - \Lambda| + |\Lambda - y| \leq \epsilon/2 + \epsilon/2 = \epsilon,$$

and y indeed satisfies $N_H(0, b) \leq y \leq N_H(0, b + \delta)$ as discussed earlier.

A.3 Betti number and spectral gap calculations

The purpose of this section is to prove Propositions 8 and 9.

Definition 22. *Given two simplicial complexes X and Y, define their* join $X * Y$ *to be the simplicial complex consisting of faces $\sigma \otimes \tau := \sigma \cup \tau$ for all $\sigma \in X$, $\tau \in Y$.*

Observe that $K(m, k) = K(m, k-1) * K(m, 1)$.

In this section, we will work with *reduced* homology. This is identical to regular homology, except that we have an extra 1-dimensional space C_{-1} and an extra boundary map $\partial_0 : C_0 \to C_{-1}$ which maps every vertex (0-simplex) to the unique basis vector of C_{-1}. This has the effect that the reduced homology H_0 is equal to the number of connected components *minus one*, rather than simply the number of connected components. The rest of the homology groups H_k for $k > 0$ are unchanged.

The homology of the join is given by the well-known Kunneth formula.

Lemma 35. (Kunneth formula)

$$\widetilde{H}_k(X * Y) = \bigoplus_{i+j=k-1} \widetilde{H}_i(X) \otimes \widetilde{H}_j(Y) \tag{A.8}$$

$$\implies \widetilde{\beta}_k(X * Y) = \sum_{i+j=k-1} \widetilde{\beta}_i(X)\widetilde{\beta}_j(Y) \tag{A.9}$$

We would also like to relate the Laplacian of $X * Y$ to the Laplacians of X and Y.

Lemma 36. *Let $\sigma \in X$ be an i-simplex and $\tau \in Y$ a j-simplex with $i + j = k - 1$. Then*

$$\Delta_k^{X*Y}(\sigma \otimes \tau) = (\Delta_i^X \sigma) \otimes \tau + \sigma \otimes (\Delta_j^Y \tau) \tag{A.10}$$

Proof. Let's work in the *graded algebra* $C_{-1} \oplus C_0 \oplus C_1 \oplus \ldots$. We have

$$\begin{aligned}
\Delta &= \partial^\dagger \partial + \partial \partial^\dagger \\
\partial(\sigma \otimes \tau) &= (\partial\sigma) \otimes \tau + (-1)^{|\sigma|}\sigma \otimes (\partial\tau) \\
\partial^\dagger(\sigma \otimes \tau) &= (\partial^\dagger\sigma) \otimes \tau + (-1)^{|\sigma|}\sigma \otimes (\partial^\dagger\tau) \\
\implies \Delta(\sigma \otimes \tau) &= (\Delta\sigma) \otimes \tau + \sigma \otimes (\Delta\tau)
\end{aligned}$$

□

Corollary 37. *Let* $\operatorname{spec} \Delta$ *denote the set of eigenvalues of* Δ.

$$\operatorname{spec} \Delta_k^{X*Y} = \bigcup_{i+j=k-1} \operatorname{spec} \Delta_i^X + \operatorname{spec} \Delta_j^Y \tag{A.11}$$

Here the plus notation for sets means $A + B = \{a + b : a \in A,\ b \in B\}$.

Proof. Use Lemma 36 and let $\sigma \in C_i^X$ and $\tau \in C_j^Y$ be eigenchains of Δ_i^X and Δ_j^Y respectively. □

Proposition 38. (Restatement of Proposition 8.)
The $(k-1)^{th}$ *Betti number of the clique complex of* $K(m,k)$ *is*

$$\beta_{k-1} = (m-1)^k \tag{A.12}$$

Proof. $K(m,k) = K(m,k-1) * K(m,1)$ and the Betti numbers of $K(m,1)$ are $(m-1, 0, 0, \dots)$. Thus by induction using the Kunneth formula, we have $\beta_{k-1} = (m-1)^k$. □

Proposition 39. (Restatement of Proposition 9.)
The combinatorial Laplacian $\Delta_{k-1}^G = (\partial_{k-1}^G)^\dagger \partial_{k-1}^G + \partial_k^G (\partial_k^G)^\dagger$ *of the clique complex of* $K(m,k)$ *has spectral gap*

$$\lambda_{\min} = m \tag{A.13}$$

Proof. Again $K(m,k) = K(m,k-1) * K(m,1)$. The spectrum of the $\Delta_0^{K(m,1)}$ is 0 with multiplicity $m-1$, and m with multiplicity 1. Thus by induction using Corollary 37, the spectrum of $\Delta_{k-1}^{K(m,k)}$ is (ignoring multiplicities) $\{0, m, 2m, \dots, km\}$. This gives $\lambda_{\min} = m$. □

A.4 SWES is DQC1-hard

In this section, we will show that SWES is DQC1-hard. We will do so by showing that we estimate the DQC1-hard normalized subtrace $\overline{\mathrm{Tr}}_b(H)$ from Proposition 31 up to additive polynomial precision $\epsilon \in \Omega(1/\mathrm{poly})$ using a polynomial number of queries to an oracle for SWES, together with polynomial-time classical preprocessing of the input and postprocessing of the output.

First, by considering how H is constructed in [40], we note that $\mathrm{Tr}(H)$ is known and that $\mathrm{Tr}(H)/2^n \in \mathcal{O}(\mathrm{poly}(n))$. Next, we define $\hat{\epsilon} = (\epsilon/(\mathrm{Tr}(H)/2^n))$ and $m = 1/\hat{\epsilon}^2$. Subsequently, let $\overline{\lambda}_{k_1}, \dots, \overline{\lambda}_{k_m}$ denote samples drawn from SWES with estimation precision $\delta/2$, where δ is such that H has no eigenvalues in $[b, b+\delta]$. For now we assume that all samples were *correctly sampled*, i.e., $|\overline{\lambda}_{k_j} - \lambda_{k_j}| \leq \delta/2$, where λ_{k_j} denotes the eigenvalue of which $\overline{\lambda}_{k_j}$ is an estimate. Afterwards, we estimate the normalized subtrace $\overline{\mathrm{Tr}}_b(H)$ by computing the ratio of samples that is below $b + \delta/2$

$$\chi = \frac{1}{m} \sum_{j\,:\,\overline{\lambda}_{k_j} \leq b+\delta/2} 1$$

By the Chernoff-Hoeffding inequality (together with the fact that H has no eigenvalues in $[b, b+\delta]$), this ratio χ is, with high probability, an estimate of

$$\Lambda = \sum_{j\,:\,\lambda_j \leq b} \lambda_j / \mathrm{Tr}(H),$$

up to additive precision $\hat{\epsilon}$. Therefore, $(\mathrm{Tr}(H)/2^n) \cdot \chi$ is, with high probability, an ϵ estimate of $(\mathrm{Tr}(H)/2^n) \cdot \Lambda = \overline{\mathrm{Tr}}_b(H)$.

Finally, we consider the probability that all samples $\overline{\lambda}_{k_j}$ were indeed sampled correctly. By the union bound this probability is $M \cdot \mu$, where μ denotes the sampling error probability of SWES. Because $m \in \mathcal{O}(\mathrm{poly}(n))$, we can choose $\mu \in \Omega(1/m) = \mathcal{O}(\mathrm{poly}(n))$ such that all our samples are sampled correctly with probability close to 1.

Appendix B

Structural risk minimization for quantum linear classifiers

B.1 Proofs of Section 4.1

B.1.1 Proofs of Proposition 12 and Lemma 13

Proposition 12. *Let $\mathbb{O} \subseteq \mathrm{Herm}\big(\mathbb{C}^{2^n}\big)$ be a family of n-qubit observables with $r = \dim\big(\sum_{\mathcal{O}\in\mathbb{O}} \mathrm{Im}\,\mathcal{O}\big)$[1]. Then, the VC dimension of*

$$\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}} = \Big\{ c(x) = \mathrm{sign}\big(\mathrm{Tr}\left[\mathcal{O}\rho_\Phi(x)\right] - d\big) \;\Big|\; \mathcal{O} \in \mathbb{O},\, d \in \mathbb{R} \Big\} \tag{4.1}$$

satisfies

$$\mathrm{VC}\big(\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}\big) \leq \dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1 \leq r^2 + 1. \tag{4.2}$$

Proof. Define $V = \sum_{\mathcal{O}\in\mathbb{O}} \mathrm{Im}\mathcal{O} \subset \mathbb{C}^{2^n}$ and let P_V denote the orthogonal projector onto V. Let $\Phi : \mathcal{X} \to \mathrm{Herm}\big(\mathbb{C}^{2^n}\big)$ denote the feature map of $\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}$ and define $\Phi' = P_V \Phi P_V$. Note that $\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}(\Phi') = \mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}(\Phi)$. It is known that the VC dimension of linear classifiers on $\mathbb{R}^\ell$ is $\ell + 1$, and it is clear that $\mathrm{Herm}\big(V\big) \simeq \mathrm{Herm}\big(\mathbb{C}^r\big) \simeq \mathbb{R}^{r^2}$. Also, note that $\mathrm{Span}\big(\mathbb{O}\big)$ is a subspace of $\mathrm{Herm}\big(V\big)$. We therefore conclude that

$$\begin{aligned}
\mathrm{VC}\big(\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}(\Phi)\big) &= \mathrm{VC}\big(\mathcal{C}^{\mathbb{O}}_{\mathrm{qlin}}(\Phi')\big) \\
&\leq \mathrm{VC}\big(\text{linear classifiers on } \mathrm{Span}\big(\mathbb{O}\big)\big) \\
&= \dim\big(\mathrm{Span}\big(\mathbb{O}\big)\big) + 1 \\
&\leq \mathrm{VC}\big(\text{linear classifiers on } \mathrm{Herm}\big(V\big) \simeq \mathbb{R}^{r^2}\big) = r^2 + 1.
\end{aligned}$$

[1] Here $\sum$ denotes the sum of vector spaces and $\mathrm{Im}\,\mathcal{O}$ denotes the image (or column space) of the operator $\mathcal{O}$.

□

Lemma 13. *The vector spaces defined in Eq.* (4.3) *and Eq.* (4.4) *satisfy*[2]

$$\dim(H) \leq \dim(V) \leq \dim(H)^2.$$

Proof. First, we note that V is contained in the space of Hermitian operators on H. Since the dimension of the space of Hermitian operators on H is equal to $\dim(H)^2$, this implies that

$$\dim(V) \leq \dim(H)^2.$$

Next, we fix a basis of H which we denote $\{|\psi_k\rangle\}_{k=1}^{\dim(H)}$, where we each $|\psi_k\rangle$ is of the form $|\psi_i(\theta)\rangle$ for some $i \in \{1, \ldots, L\}$ and $\theta \in \mathbb{R}^m$. To show that $\dim(V) \geq \dim(H)$, we will show that the operators $\{|\psi_k\rangle \langle\psi_k|\}_{k=1}^{\dim(H)} \subset V$ are linearly independent. We do so by contradiction, i.e., we assume they are not linearly independent and show that this leads to a contradiction. That is, we assume that there exists a $k' \in \{1, \ldots, \dim(H)\}$ and $\{\alpha_k\}_{k \neq k'} \subset \mathbb{R}$ such that

$$|\psi'_k\rangle \langle\psi'_k| = \sum_{k \neq k'} \alpha_k |\psi_k\rangle \langle\psi_k| .$$

This implies that

$$\begin{aligned} |\psi'_k\rangle &= \big(|\psi'_k\rangle \langle\psi'_k| \big) |\psi'_k\rangle \\ &= \Big(\sum_{k \neq k'} \alpha_k |\psi_k\rangle \langle\psi_k| \Big) |\psi'_k\rangle \\ &= \sum_{k \neq k'} \left(\alpha_k \langle\psi_k \mid \psi'_k\rangle\right) |\psi_k\rangle , \end{aligned}$$

which shows that $\{|\psi_k\rangle\}_{k=1}^{\dim(H)}$ are not linearly independent. This clearly contradicts the assumption that $\{|\psi_k\rangle\}_{k=1}^{\dim(H)}$ is basis of H. We therefore conclude that the operators $\{|\psi_k\rangle \langle\psi_k|\}_{k=1}^{\dim(H)} \subset V$ are linearly independent, which shows that $\dim(V) \geq \dim(H)$. □

B.1.2 Relationship Proposition 12 and ranks of observables

In this section we discuss one possible way to relate the quantity r in Proposition 12 with the ranks of the observables by considering the overlaps of the images of the observables. Specifically, consider a family of observables $\{\mathcal{O}_i\}_{i=1}^n$, where each ob-

[2]Note that there exists ansatzes for which the inequalities are strict, i.e., $\dim(H) < \dim(V) < \dim(H)^2$ (e.g., see the first example discussed in Section 4.3).

servable is of rank R[3]. Next, define the quantities

$$I_i = \dim\left(\operatorname{Im}\mathcal{O}_i \cap [\operatorname{Im}\mathcal{O}_{i+1} + \cdots + \operatorname{Im}\mathcal{O}_n]\right) \tag{B.1}$$

and

$$O_i = R - I_i. \tag{B.2}$$

Note that O_i measures the extent to which the image of the observable $\mathcal{O}_i$ overlaps with the images of the observables $\mathcal{O}_{i+1}, \ldots, \mathcal{O}_n$. Specifically, O_i is equal to zero if the images are fully overlapping, and it is equal to R if there is no overlap at all. Now Lemma 40 below provides a way to relate the quantity r in Proposition 12 with the ranks of the observables R and the overlaps of the images O_i. Note that we consider the case where the family of observables is finite, whereas in the case of explicit quantum linear classifiers this family is infinite. However, since all images live in a finite dimensional space, summing only finitely many images is already sufficient. More precisely, for any family of n-qubit observables $\mathbb{O}$ (possibly infinitely large) there exists a $\mathbb{O}' \subseteq \mathbb{O}$ with $|\mathbb{O}'| \leq 2^n$ and

$$\sum_{\mathcal{O}' \in \mathbb{O}'} \operatorname{Im}\mathcal{O}' = \sum_{\mathcal{O} \in \mathbb{O}} \operatorname{Im}\mathcal{O}.$$

In Lemma 40 below we can thus w.l.o.g. consider the case where the family of observables is finite.

Lemma 40. *Consider a family of observables $\mathbb{O} = \{\mathcal{O}_i\}_{i \in I}$, where each observable is of rank R. Then, for r defined in Proposition 12 and $\{O_i\}_{i \in I}$ defined in Eq.* (B.2)*, we have that*

$$r = R + \sum_{i=1}^{n-1} O_i$$

Proof. The proof is basically a repeated application of the formula

$$\dim\left(\operatorname{Im}\mathcal{O}_1 + \operatorname{Im}\mathcal{O}_2\right) = \dim\left(\operatorname{Im}\mathcal{O}_1\right) + \dim\left(\operatorname{Im}\mathcal{O}_2\right) - \dim\left(\operatorname{Im}\mathcal{O}_1 \cap \operatorname{Im}\mathcal{O}_2\right).$$

[3]The results in this section hold more generally for families with varying ranks, though for simplicity (and to more closely relate it to Proposition 15) we assume all observables have some fixed rank R (from which it should be clear how to adapt it to the case where the observables can have different ranks).

Specifically, by repeatedly applying the above formula we find that

$$\begin{aligned} r = \dim\left(\sum_{i=1}^{n} \operatorname{Im}\mathcal{O}_i\right) &= \dim\left(\operatorname{Im}\mathcal{O}_1\right) + \dim\left(\sum_{i=2}^{n} \operatorname{Im}\mathcal{O}_i\right) \\ &\quad - \dim\left(\operatorname{Im}\mathcal{O}_1 \cap \sum_{i=2}^{n} \operatorname{Im}\mathcal{O}_i\right) \\ &= \dim\left(\operatorname{Im}\mathcal{O}_1\right) + \dim\left(\operatorname{Im}\mathcal{O}_2\right) + \dim\left(\sum_{i=3}^{n} \operatorname{Im}\mathcal{O}_i\right) \\ &\quad - \dim\left(\operatorname{Im}\mathcal{O}_1 \cap \sum_{i=2}^{n} \operatorname{Im}\mathcal{O}_i\right) \\ &\quad - \dim\left(\operatorname{Im}\mathcal{O}_2 \cap \sum_{i=3}^{n} \operatorname{Im}\mathcal{O}_i\right) \\ &= nR - (I_1 + \cdots + I_{n-1}) \\ &= R - \sum_{i=1}^{n-1}(R - I_i) = R - \sum_{i=1}^{n-1} O_i \end{aligned}$$

□

B.1.3 Proof of Proposition 14

Proposition 14. *Let $\mathbb{O} \subseteq \operatorname{Herm}\left(\mathbb{C}^{2^n}\right)$ be a family of n-qubit observables with $\eta = \max_{\mathcal{O}\in\mathbb{O}} \|\mathcal{O}\|_F$. Then, the fat-shattering dimension of*

$$\mathcal{F}_{\text{qlin}}^{\mathbb{O}} = \left\{ f_{\mathcal{O},d}(x) = \operatorname{Tr}\left[\mathcal{O}\rho_\Phi(x)\right] - d \;\middle|\; \mathcal{O} \in \mathbb{O},\ d \in \mathbb{R} \right\} \tag{4.5}$$

is upper bounded by

$$\operatorname{fat}_{\mathcal{F}_{\text{qlin}}^{\mathbb{O}}}(\gamma) \leq O\left(\frac{\eta^2}{\gamma^2}\right). \tag{4.6}$$

Proof. Due to the close relationship to standard linear classifiers, we can utilize previously obtained results in that context. In particular, for our approach we use the following proposition.

Proposition 41 (Fat-shattering dimension of linear functions [175])**.** *Consider the family of real-valued functions on the ball of radius R inside $\mathbb{R}^N$ given by*

$$\mathcal{F}_{\text{lin}} = \left\{ f_{w,d}(x) = \langle w, x\rangle - d \;\middle|\; w \in \mathbb{R}^N \text{ with } ||w|| = 1,\ d \in \mathbb{R} \text{ with } |d| \leq R \right\}.$$

The fat-shattering dimension of $\mathcal{F}_{\text{lin}}$ can be bounded by

$$\operatorname{fat}_{\mathcal{F}_{\text{lin}}}(\gamma) \leq \min\{9R^2/\gamma^2, N+1\} + 1.$$

The context in the above proposition is closely related, yet slightly different than that of quantum linear classifiers. Firstly, n-qubit density matrices lie within the ball of radius $R = 1$ inside $\mathrm{Herm}\big(\mathbb{C}^{2^n}\big)$ equipped with the Frobenius norm. However, as in our case the hyperplanes arise from the family of observables $\mathbb{O}$, whose Frobenius norms are upper bounded by η, we cannot directly apply the above proposition. We therefore adapt the above proposition by exchanging the role of R with the upper bound on the norms of the observables in $\mathbb{O}$, resulting in the following lemma.

Lemma 42. *Consider the family of real-valued functions on the ball of radius $R = 1$ inside $\mathbb{R}^N$ given by*

$$\mathcal{F}_{\mathrm{lin}}^{\leq\eta} = \Big\{ f_{w,d}(x) = \langle w, x\rangle - d \;\Big|\; w \in \mathbb{R}^N \text{ with } ||w|| \leq \eta,\ d \in \mathbb{R} \text{ with } |d| \leq \eta \Big\}.$$

The fat shattering dimension of $\mathcal{F}_{\mathrm{lin}}^{\leq\eta}$ can be upper bounded by

$$\mathrm{fat}_{\mathcal{F}_{\mathrm{lin}}^{\leq\eta}}(\gamma) \leq \min\{9\eta^2/\gamma^2, N+1\} + 1.$$

Proof. Let us first determine the fat-shattering dimension of the family of linear functions with norm precisely equal to η on points that lie within the ball of radius $R = 1$, i.e.,

$$\mathcal{F}_{\mathrm{lin}}^{=\eta} = \Big\{ f_{w,d}(x) = \langle w, x\rangle - d \;\Big|\; w \in \mathbb{R}^N \text{ with } ||w|| = \eta,\ d \in \mathbb{R} \text{ with } |d| \leq \eta \Big\}.$$

Suppose $\mathcal{F}_{\mathrm{lin}}^{=\eta}$ can γ-shatter a set of points $\{x_1, \ldots, x_k\}$ that lie within the ball of radius $R = 1$. Because $\langle w, x_i\rangle = \langle w/\eta, \eta x_i\rangle$, we find that $\mathcal{F}_{\mathrm{lin}}^{=1}$ can γ-shatter the set of points $\eta x_1, \ldots, \eta x_k$ that lie within the ball of radius $R = \eta$. By Proposition 41 we have $k \leq \min\{9\eta^2/\gamma^2, N+1\} + 1$. Thus, the fat-shattering dimension of $\mathcal{F}_{\mathrm{lin}}^{=\eta}$ on points within the ball of radius $R = 1$ is upper bounded by

$$\mathrm{fat}_{\mathcal{F}_{\mathrm{lin}}^{=\eta}}(\gamma) \leq \min\{9\eta^2/\gamma^2, N+1\} + 1.$$

To conclude the desired results, note that this bound is monotonically increasing in η, and thus allowing hyperplanes with with norm $\|w\| < \eta$ will not increase the fat-shattering dimension.

□

From the above lemma we can immediately infer an upper bound on the fat-shattering dimension of quantum linear classifiers by identifying that as vector spaces $\mathrm{Herm}\big(\mathbb{C}^{2^n}\big) \simeq \mathbb{R}^{4^n}$.

□

Sample complexity in the PAC-learning framework

Besides being related to generalization performance, the fat-shattering dimension is also related to the so-called *sample complexity* in the probably approximately correct (PAC) learning framework [117]. The sample complexity captures the amount classi-

fier queries required to find another classifier that with high probability agrees with the former classifier on unseen examples.

By plugging the upper bound of Proposition 14 into previously established theorems on the sample complexity of families of classifiers [21, 29], we derive the following corollary, which can be viewed as a dual of the result of [8].

Corollary 43. *Let $\mathbb{O} \subseteq \mathrm{Herm}\left(\mathbb{C}^{2^n}\right)$ be a family of observables with $\eta = \max_{\mathcal{O}\in\mathbb{O}} \|\mathcal{O}\|_F$ and consider the family of real-valued functions $\mathcal{F}_{\mathrm{qlin}}^{\mathbb{O}}$ defined in Eq. (4.5). Fix an element $F \in \mathcal{F}_{\mathrm{qlin}}^{\mathbb{O}}$ as well as parameters $\epsilon, \nu, \gamma > 0$ with $\gamma\epsilon \geq 7\nu$. Suppose we draw m examples $\mathcal{D} = \{\rho_1, \ldots, \rho_m\}$ independently according to a distribution P, and then choose any function $H \in \mathcal{F}_{\mathrm{qlin}}^{\mathbb{O}}$ such that $|H(\rho_i) - F(\rho_i)| \leq \nu$ for all $\rho_i \in \mathcal{D}$. Then, with probability at least $1-\delta$ over P, we have that*

$$\Pr_{\rho\sim P}\big(|H(\rho) - F(\rho)| > \gamma\big) \leq \epsilon,$$

provided that

$$m \in \Omega\bigg(\frac{1}{\gamma^2\epsilon^2}\Big(\frac{\eta^2}{\gamma^2\epsilon^2}\log^2\frac{1}{\gamma\epsilon} + \log\frac{1}{\delta}\Big)\bigg).$$

Proof. Follows directly from plugging the uppper bound of Proposition 14 into Corollary 2.4 of [8].

□

B.2 Proofs of propositions Section 4.2

B.2.1 Proof of Proposition 15

Proposition 15. *Let $\mathcal{C}_{\mathrm{qlin}}^{(r)}$ denote the family of quantum linear classifiers corresponding to observables of exactly rank r, that is,*

$$\mathcal{C}_{\mathrm{qlin}}^{(r)} = \Big\{c(\rho) = \mathrm{sign}\big(\mathrm{Tr}\left[\mathcal{O}\rho\right] - d\big) \;\Big|\; \mathcal{O} \in \mathrm{Herm}\big(\mathbb{C}^{2^n}\big),\, \mathrm{rank}\big(\mathcal{O}\big) = r,\, d \in \mathbb{R}\Big\} \quad (4.7)$$

Then, the following statements hold:

(x) For every finite set of examples $\mathcal{D}$ that is correctly classified by a quantum linear classifier $c \in \mathcal{C}_{\mathrm{qlin}}^{(k)}$ with $0 < k < 2^n$, there exists a quantum linear classifier $c \in \mathcal{C}_{\mathrm{qlin}}^{(r)}$ with $r > k$ that also correctly classifies $\mathcal{D}$.

(xi) There exists a finite set of examples that can be correctly classified by a classifier $c \in \mathcal{C}_{\mathrm{qlin}}^{(r)}$, but which no classifier $c' \in \mathcal{C}_{\mathrm{qlin}}^{(k)}$ with $k < r$ can classify correctly.

Proof. (i): Suppose $c_{\mathcal{O},b} \in \mathcal{C}_{\mathrm{qlin}}^{(k)}$ correctly classifies $\mathcal{D}$. Next, we define

$$\delta = \min_{x\in\mathcal{D}_-} \big|\mathrm{Tr}\left[\mathcal{O}\rho_x\right] - d\big|,$$

where $\mathcal{D}_-$ is the subset of examples with label -1, and note that since $\mathcal{D}$ is correctly classified we have $\delta > 0$. Fix the basis we work in to be the eigenbasis of $\mathcal{O}$ ordered in such a way that

$$\mathcal{O} = \mathrm{diag}(\lambda_1, \ldots, \lambda_k, 0, \ldots, 0)$$

and define

$$P = \frac{1}{r-k}\mathrm{diag}(\underbrace{0, \ldots, 0}_{k \text{ times}}, \underbrace{1, \ldots, 1}_{r-k \text{ times}}, \underbrace{0, \ldots, 0}_{2^n - r \text{ times}}).$$

For every $0 < \epsilon < \delta$ we have that $\mathcal{O}' = \mathcal{O} + \epsilon P$ has $\mathrm{rank}(\mathcal{O}') = r$. What remains to be shown is that $c_{\mathcal{O}',b} \in \mathcal{C}^{(r)}_{\mathrm{qlin}}$ correctly classifies $\mathcal{D}$. To do so, first let $x \in \mathcal{D}_+$ (i.e., labeled $+1$) and note that

$$\mathrm{Tr}\left[\mathcal{O}'\rho_x\right] - b = \underbrace{\left(\mathrm{Tr}\left[\mathcal{O}\rho_x\right] - b\right)}_{\geq 0} + \underbrace{\epsilon \mathrm{Tr}\left[P\rho_x\right]}_{\geq 0} \geq 0,$$

which shows that indeed $c_{\mathcal{O}',b}(x) = +1$. Next, let $x \in \mathcal{D}_-$ (i.e., labeled -1) and note that

$$\mathrm{Tr}\left[\mathcal{O}'\rho_x\right] - b = \underbrace{\left(\mathrm{Tr}\left[\mathcal{O}\rho_x\right] - b\right)}_{\leq -\delta} + \underbrace{\epsilon \mathrm{Tr}\left[P\rho_{x^+}\right]}_{< \delta} < 0,$$

which shows that indeed $c_{\mathcal{O}',b}(x) = -1$.

(ii):

We will describe a protocol that queries a classifier $c_{\mathcal{O},b}$ and based on its outcomes checks whether $\mathcal{O}$ is approximately equal to a fixed target observable $\mathcal{T}$ of rank r. We will show that if the queries to $c_{\mathcal{O},b}$ are labeled in a way that agrees with the target classifier that uses the observable $\mathcal{T}$, then the spectrum of $\mathcal{O}$ has to be point-wise within distance ϵ of the spectrum of $\mathcal{T}$. In particular, this will show that the rank of $\mathcal{O}$ has to be at least r if we make ϵ small enough. Consequently, if the rank of $\mathcal{O}$ is less than r, then at least one query made during the protocol has to be labeled differently by $c_{\mathcal{O},b}$ than the target classifier. In the end, the queries made to the classifier during the protocol will therefore constitute the set of examples described in the theorem.

Let us start with some definition. For a classifier $c_{\mathcal{O},b}(\rho) = \mathrm{sgn}\big(\mathrm{Tr}\big[\mathcal{O}\rho\big] - b\big)$ we define its effective observable $\mathcal{O}_{\mathrm{eff}} = \mathcal{O} - bI$ which we express in the computational basis as $\mathcal{O}_{\mathrm{eff}} = (O_{ij})$. Next, we define our target classifier to be $c_{\mathcal{T},-1}$ where the observable $\mathcal{T}$ is given by

$$\mathcal{T} = -r\left|0\right\rangle\left\langle 0\right| + \sum_{i=1}^{r-1} i\left|i\right\rangle\left\langle i\right|,$$

and we define its effective observable $\mathcal{T}_{\mathrm{eff}} = \mathcal{T} + I$ which we express in the computational basis as $\mathcal{T}_{\mathrm{eff}} = (T_{ij})$. Rescaling $\mathcal{O}_{\mathrm{eff}}$ with a positive scalar does not change the output of the corresponding classifier. Therefore, to make the protocol well-defined, we define $\mathcal{O}_{\mathrm{eff}}$ to be the unique effective observable whose first diagonal element is scaled to be equal to $O_{00} = -(r+1)$.

Our approach is as follows. First, we query $c_{\mathcal{O},b}$ in such a way that if the outcomes

agree with with the target classifier $c_{\mathcal{T},-1}$, then the absolute values of the off-diagonal entries in the first row and column of $\mathcal{O}_{\text{eff}}$ must be close to zero (i.e., approximately equal to those of $\mathcal{T}_{\text{eff}}$). Afterwards, we again query $c_{\mathcal{O},b}$ but now in such a way that if the outcomes agree with the target classifier $c_{\mathcal{T},-1}$, then the diagonal elements of $\mathcal{O}_{\text{eff}}$ must be approximately equal to those of $\mathcal{T}_{\text{eff}}$. In the end, we query $c_{\mathcal{O},b}$ one final time but this time in such a way that if the outcomes agree with the target classifier $c_{\mathcal{T},-1}$, then the absolute values of the remaining off-diagonal elements of $\mathcal{O}_{\text{eff}}$ must be close to zero (i.e., again approximately equal to those of $\mathcal{T}_{\text{eff}}$). Finally, we use Gershgorin's circle theorem to show that the spectrum of $\mathcal{O}_{\text{eff}}$ has to be point-wise close to the spectrum of $\mathcal{T}_{\text{eff}}$. We remark that this procedure could be generalized to a more complete tomography approach, where one uses queries to the classifier $c_{\mathcal{O},b}$ in order to reconstruct the entire spectrum of $\mathcal{O}_{\text{eff}}$.

First, we query the quantum states $|i\rangle$ for $i = 0, \ldots, 2^n - 1$. Without loss of generality, we can assume that the classifiers $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$ agree on the label, i.e.,

$$c_{\mathcal{O},b}\big(\,|0\rangle\langle 0|\,\big) = -1, \text{ and } c_{\mathcal{O},b}\big(\,|i\rangle\langle i|\,\big) = +1 \text{ for } i = 1, \ldots, 2^n - 1, \tag{B.3}$$

as otherwise a set of examples containing just these states would already separate $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$.

In order to show that the absolute value of the off-diagonal elements of the first row and column of $\mathcal{O}_{\text{eff}}$ must be close to zero and that the diagonal elements of $\mathcal{O}_{\text{eff}}$ must be close to those of $\mathcal{T}_{\text{eff}}$, we consider the quantum states given by

$$|\gamma_\theta(\alpha)\rangle = \sqrt{1-\alpha}\,|0\rangle + e^{i\theta}\sqrt{\alpha}\,|j\rangle\,, \quad \text{with } \alpha \in [0,1] \text{ and } \theta \in [0, 2\pi). \tag{B.4}$$

Its expectation value with respect to $\mathcal{O}_{\text{eff}}$ is given by

$$\langle\gamma_\theta(\alpha)|\,\mathcal{O}_{\text{eff}}\,|\gamma_\theta(\alpha)\rangle = (1-\alpha)O_{00} + \alpha O_{jj} + \sqrt{\alpha(1-\alpha)}C_\theta, \tag{B.5}$$

where $C_\theta := \text{Re}\big(e^{i\theta}O_{0j}\big)$, and its expectation value with respect to $\mathcal{T}_{\text{eff}}$ is given by

$$\langle\gamma_\theta(\alpha)|\,\mathcal{T}_{\text{eff}}\,|\gamma_\theta(\alpha)\rangle = (1-\alpha)T_{00} + \alpha T_{jj}. \tag{B.6}$$

Crucially, by Equation (B.3) we know that the label of $|\gamma_\theta(\alpha)\rangle$ goes from -1 to $+1$ as α goes $0 \to 1$. Note that the expectation value of $|\gamma_\theta(\alpha)\rangle$ with respect to $\mathcal{T}_{\text{eff}}$ is independent from the phase θ.

To determine that $\big|O_{0j}\big|$ is smaller than $\delta > 0$, we query the classifier $c_{\mathcal{O},b}$ on the states $|\gamma_{\hat{\theta}}(\hat{\alpha})\rangle$ for all $\hat{\theta}$ in a ζ-mesh of $[0, 2\pi)$ and for all $\hat{\alpha}$ in a ξ-mesh of $[0, 1]$ and we suppose they are labeled the same as the target classifier $c_{\mathcal{T},-1}$ would label them. Using these queries we can find estimates $\hat{\alpha}^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\hat{\theta})$ that are ξ-close to the unique $\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\theta) = \alpha'$ that satisfies

$$\langle\gamma_\theta(\alpha')|\,\mathcal{O}_{\text{eff}}\,|\gamma_\theta(\alpha')\rangle = 0, \tag{B.7}$$

by finding the smallest $\hat{\alpha}$ where the label has gone from -1 to $+1$. We refer to the α' satisfying Equation (B.7) as the *crossing point at phase* θ. Because the label assigned by $c_{\mathcal{T},-1}$ does not depend on the phase θ, and since all states $|\gamma_{\hat{\theta}}(\hat{\alpha})\rangle$ were assigned

the same label by $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$, we find that the crossing point estimate $\hat{\alpha}^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\hat{\theta})$ is the same for all $\hat{\theta}$. In particular, this implies that the actual crossing points $\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\hat{\theta})$ have to be within ξ-distance of each other for all $\hat{\theta}$.

Before we continue, we first show that if $c_{\mathcal{O},b}$ assigns the same labels as $c_{\mathcal{T},-1}$, then O_{jj} is bounded above by a quantity that only depends on n. Fix $\tilde{\theta}$ to be any point inside the ζ-mesh such that $C_{\tilde{\theta}} \leq 0$, and define the function $E(\alpha) = (1-\alpha)O_{00} + \alpha O_{jj} + \sqrt{(1-\alpha)\alpha}C_{\tilde{\theta}}$. By our choice of $\mathcal{T}$, we have that $\alpha^{\mathcal{T}}_{\text{cross}} \in (\frac{r+1}{2r+1}, \frac{r+1}{r+3})$. Therefore, if $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$ agree on the entire ξ-mesh for a small enough ξ, then it must hold that $\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\tilde{\theta}) \in (\frac{1}{2}, \frac{2^n+1}{2^n+2})$. By the mean value theorem there exists an $\alpha' \in (\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\tilde{\theta}), \frac{2^n+1}{2^n+2})$ such that

$$E'(\alpha') = \frac{E(\frac{2^n+1}{2^n+2}) - E(\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\tilde{\theta}))}{\frac{2^n+1}{2^n+2} - \alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\tilde{\theta})}. \tag{B.8}$$

After some rewriting, we can indeed conclude from the above equation that O_{jj} is bounded above by a quantity that only depends on n.

Next, write $O_{0j} = \left|O_{0j}\right|e^{i\phi}$ with $\phi \in [0, 2\pi)$, let $\hat{\theta}_{\text{abs}}$ denote the point in the ζ-mesh of $[0, 2\pi)$ that is closest to $2\pi - \phi$, and let $\hat{\theta}_0$ denote the point in the ζ-mesh of $[0, 2\pi)$ that is closest to $\pi/2 - \phi$ modulo 2π. By our previous discussion we know that $\left|\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\hat{\theta}_{\text{abs}}) - \alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\hat{\theta}_0)\right| < \xi$, which together with the previously established bound on O_{jj} implies that

$$\left|C_{\hat{\theta}_{\text{abs}}} - C_{\hat{\theta}_0}\right| < f(\xi), \tag{B.9}$$

where f is a continuous function (independent from $c_{\mathcal{O},b}$) with $f(\xi) \to 0$ as $\xi \to 0$. Moreover, using the inequality $\cos(\zeta) \geq 1 - \lambda\zeta$, where $\lambda \approx 0.7246$ is a solution of $\lambda\big(\pi - \arcsin(\lambda)\big) = 1 + \sqrt{1-\lambda^2}$, together with the inequality $\cos(\pi/2 - \zeta) \leq \zeta$, we can derive that

$$\begin{aligned}
\left|C_{\hat{\theta}_{\text{abs}}} - C_{\hat{\theta}_0}\right| &= \Big|\left|O_{0j}\right|\cos\big(\hat{\theta}_{\text{abs}} + \phi\big) - \left|O_{0j}\right|\cos\big(\hat{\theta}_0 + \phi\big)\Big| \\
&\geq \left|O_{0j}\right| \cdot \Big|\cos\big(\zeta\big) - \cos\big(\pi/2 - \zeta\big)\Big| \\
&\geq \left|O_{0j}\right| \cdot \Big|1 - \big(\lambda + 1\big)\zeta\Big|.
\end{aligned} \tag{B.10}$$

Finally, by combining Equation (B.9) with Equation (B.10) we can conclude that

$$\left|O_{0j}\right| < \frac{f(\xi)}{1 - (\lambda+1)\zeta},$$

which for ξ and ζ small enough shows that $\left|O_{0j}\right| < \delta$ for any chosen precision $\delta > 0$ (i.e., the fineness of both meshes ξ and ζ will depend on the choice of δ).

To determine that O_{jj} is within distance $\delta' > 0$ of T_{jj} we again query the classifier $c_{\mathcal{O},b}$ but this time on the states $|\gamma_0(\hat{\alpha})\rangle$ for all $\hat{\alpha}$ in a ξ'-mesh of $[0, 1]$ and we suppose they are labeled the same as the target classifier $c_{\mathcal{T},-1}$ would. Using these queries

we can find estimates $\hat{\alpha}^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0)$, $\hat{\alpha}^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0)$ that are ξ'-close to the corresponding actual crossing point. As we assumed that all queries are labeled the same by $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$, the crossing point estimate $\hat{\alpha}^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0)$ has to be equal to the crossing point estimate $\hat{\alpha}^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0)$. In particular, this implies that the actual crossing points $\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0)$ and $\alpha^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0)$ have to be within ξ'-distance of each other. Next, define $g(\alpha, C)$ to be the unique coefficient $O \in \mathbb{R}_{\geq 0}$ that satisfies

$$(1-\alpha)O_{00} + \alpha O + \sqrt{\alpha(1-\alpha)}C = 0.$$

It is clear that g is a continuous function in α and C that is independent from $c_{\mathcal{O},b}$, and that $T_{jj} = g(\alpha^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0), 0)$ and $O_{jj} = g(\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0), C_0)$. Finally, we let $\delta > 0$ and $\xi' > 0$ be small enough such that if $\left|\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0) - \alpha^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0)\right| < \xi'$ and $\left|C_0\right| < \delta$, then

$$\left|O_{jj} - T_{jj}\right| = \left|g(\alpha^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0), 0) - g(\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0), C_0)\right| < \delta'.$$

In conclusion, to determine that O_{jj} is within distance $\delta' > 0$ of T_{jj} we first do the required queries to determine that $\left|C_0\right| = \left|O_{0j}\right| < \delta$, after which we do the required queries to determine that $\left|\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(0) - \alpha^{\mathcal{T}_{\text{eff}}}_{\text{cross}}(0)\right| < \xi'$, which together indeed implies that O_{jj} is within distance $\delta' > 0$ of T_{jj}.

In order to show that the absolute value of the remaining off-diagonal elements of $\mathcal{O}_{\text{eff}}$ must be close to zero (i.e., close to those of $\mathcal{T}_{\text{eff}}$) we consider the quantum states given by

$$|\mu_\theta(\alpha)\rangle = \frac{\sqrt{1-\alpha}}{\sqrt{2}}\big(\,|0\rangle + |i\rangle\,\big) + e^{i\theta}\sqrt{\alpha}\,|j\rangle\,, \qquad \text{with } \alpha \in [0,1] \text{ and } \theta \in [0, 2\pi). \tag{B.11}$$

Its expectation value with respect to $\mathcal{O}_{\text{eff}}$ is given by

$$\langle\mu_\theta(\alpha)|\,\mathcal{O}_{\text{eff}}\,|\mu_\theta(\alpha)\rangle = \big(1-\alpha\big)\big(O_{00} + O_{ii} + \text{Re}(O_{0i})\big) + \alpha O_{jj} \tag{B.12}$$

$$+ \sqrt{2\alpha(1-\alpha)}C_\theta, \tag{B.13}$$

where $C_\theta := \text{Re}\big(e^{i\theta}(O_{0j} + O_{ij})\big)$, and its expectation value with respect to $\mathcal{T}_{\text{eff}}$ is given by

$$\langle\mu_\theta(\alpha)|\,\mathcal{T}_{\text{eff}}\,|\mu_\theta(\alpha)\rangle = \big(1-\alpha\big)\big(T_{00} + T_{ii}\big) + \alpha T_{jj}. \tag{B.14}$$

Crucially, by our choice of $\mathcal{T}$ we know that the label of $|\mu_\theta(\alpha)\rangle$ goes from -1 to $+1$ as α goes $0 \to 1$. Note that the expectation value of $|\mu_\theta(\alpha)\rangle$ with respect to $\mathcal{T}_{\text{eff}}$ is independent from the phase θ.

To determine that $\left|O_{ij}\right|$ is smaller than $\delta'' > 0$ for $i, j \geq 1$ and $i \neq j$, we query the classifier $c_{\mathcal{O},b}$ on the states $|\gamma_{\hat{\theta}}(\hat{\alpha})\rangle$ for all $\hat{\theta}$ in a ζ''-mesh of $[0, 2\pi)$ and for all $\hat{\alpha}$ in a ξ''-mesh of $[0,1]$ and we suppose they are labeled the same as the target classifier $c_{\mathcal{T},-1}$ would. Using these queries we can find estimates $\hat{\alpha}^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\hat{\theta})$ that are ξ-close to the unique $\alpha^{\mathcal{O}_{\text{eff}}}_{\text{cross}}(\theta) = \alpha'$ that satisfies

$$\langle\mu_\theta(\alpha')|\,\mathcal{O}_{\text{eff}}\,|\mu_\theta(\alpha')\rangle = 0, \tag{B.15}$$

by finding the smallest $\hat{\alpha}$ where the label has gone from -1 to $+1$. Because the label assigned by $c_{\mathcal{T},-1}$ does not depend on the phase θ, and since all states $|\mu_{\hat{\theta}}(\hat{\alpha})\rangle$ were assigned the same label by $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$, we find that the crossing point estimate $\hat{\alpha}_{\text{cross}}^{\mathcal{O}_{\text{eff}}}(\hat{\theta})$ is the same for all $\hat{\theta}$. In particular, this implies that the actual crossing points $\alpha_{\text{cross}}^{\mathcal{O}_{\text{eff}}}(\hat{\theta})$ have to be within ξ''-distance of each other for all $\hat{\theta}$. Subsequently, write $O_{0j} + O_{ij} = \left|O_{0j} + O_{ij}\right|e^{i\phi}$ with $\phi \in [0, 2\pi)$, let $\hat{\theta}_{\text{abs}}$ denote the point in the ζ''-mesh of $[0, 2\pi)$ that is closest to $2\pi - \phi$, and let $\hat{\theta}_0$ denote the point in the ζ''-mesh of $[0, 2\pi)$ that is closest to $\pi/2 - \phi$ modulo 2π. By our previous discussion we know that $\left|\alpha_{\text{cross}}^{\mathcal{O}_{\text{eff}}}(\hat{\theta}_{\text{abs}}) - \alpha_{\text{cross}}^{\mathcal{O}_{\text{eff}}}(\hat{\theta}_0)\right| < \xi''$, which implies

$$\left|C_{\hat{\theta}_{\text{abs}}} - C_{\hat{\theta}_0}\right| < h(\xi''), \tag{B.16}$$

where h is a continuous function (independent from $c_{\mathcal{O},b}$ and $c_{\mathcal{T},-1}$) with $h(\xi'') \to 0$ as $\xi'' \to 0$. Moreover, using the inequality $\cos(\zeta'') \geq 1 - \lambda\zeta''$, where $\lambda \approx 0.7246$ is a solution of $\lambda\big(\pi - \arcsin(\lambda)\big) = 1 + \sqrt{1-\lambda^2}$, together with the inequality $\cos(\pi/2 - \zeta'') \leq \zeta''$, we can derive that

$$\begin{aligned}
\left|C_{\hat{\theta}_{\text{abs}}} - C_{\hat{\theta}_0}\right| &= \Big|\left|O_{0j} + O_{ij}\right|\cos\big(\hat{\theta}_{\text{abs}} + \phi\big) - \left|O_{0j} + O_{ij}\right|\cos\big(\hat{\theta}_0 + \phi\big)\Big| \\
&\geq \left|O_{0j} + O_{ij}\right| \cdot \Big|\cos\big(\zeta''\big) - \cos\big(\pi/2 - \zeta''\big)\Big| \\
&\geq \left|O_{0j} + O_{ij}\right| \cdot \Big|1 - \big(\lambda + 1\big)\zeta''\Big|.
\end{aligned} \tag{B.17}$$

Finally, by combining Equation (B.16) with Equation (B.17) we can conclude that

$$\left|O_{0j} + O_{ij}\right| < \frac{h(\xi'')}{1 - (\lambda+1)\zeta''},$$

which for ξ'' and ζ'' small enough shows that $\left|O_{0j} + O_{ij}\right| < \delta''/2$ (i.e., the fineness of both meshes ξ'' and ζ'' will depend on the choice of δ''). In conclusion, to determine that $\left|O_{ij}\right|$ is smaller than $\delta'' > 0$ we first do the required queries to determine that $\left|O_{0j}\right| < \delta''/2$, after which we do the required queries to determine that $\left|O_{0j} + O_{ij}\right| < \delta''/2$, which together indeed implies that $\left|O_{ij}\right| < \delta''$.

All in all, we have described a (finite) set of states such that if the label assigned by $c_{\mathcal{O},b}$ agrees with the label assigned by $c_{\mathcal{T},-1}$, then the absolute value of the off-diagonal elements of the first row of $\mathcal{O}_{\text{eff}}$ have to be smaller than δ, the diagonal elements of $\mathcal{O}_{\text{eff}}$ have to be within δ'-distance of those of $\mathcal{T}_{\text{eff}}$, and the remaining off diagonal elements of $\mathcal{O}_{\text{eff}}$ have to be smaller than δ''. Finally, we choose $\delta, \delta', \delta'' = 1/2^{n+1}$ and use the above protocol to establish that for $1 \leq i \leq r-1$ the Gershgorin discs D_i of $\mathcal{O}_{\text{eff}}$ (i.e., with center O_{ii} and radius $\sum_j |O_{ij}|$) have to be contained in the disks $\tilde{D}_i$ with center $i+1$ and radius $1/2$. Moreover, we establish that the Gershgorin disc D_0 has to be contained in the disks $\tilde{D}_0$ with center $-r+1$ and radius $1/2$. Since the disks $\tilde{D}_i$ as disjoint, so are the Gershgorin discs D_i, which implies that $\mathcal{O}_{\text{eff}}$ must have at least r distinct eigenvalues, and thus that $\text{rank}\big(\mathcal{O}\big) \geq r$. Consequently, if $\text{rank}\big(\mathcal{O}\big) < r$, then $c_{\mathcal{O},b}$ must disagree with $c_{\mathcal{T},-1}$ on the label of at least one of the

states queried during the protocol.

□

B.2.2 Proof of Proposition 16

Proposition 16. *Let $\mathcal{C}_{\mathrm{lin}}(\Phi)$ denote the family of linear classifiers that is equipped with a feature map Φ. Also, let $\mathcal{C}_{\mathrm{qlin}}^{(\leq r)}(\Phi')$ denote the family of quantum linear classifiers that uses observables of rank at most r and which is equipped with a quantum feature map Φ'. Then, the following statements hold:*

(xii) For every feature map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^N$ with $\sup_{x\in\mathbb{R}^\ell} ||\Phi(x)|| = M < \infty$, there exists a feature map $\Phi' : \mathbb{R}^\ell \to \mathbb{R}^{N+1}$ such that $||\Phi'(x)|| = 1$ for all $x \in \mathbb{R}^\ell$ and the families of linear classifiers satisfy $\mathcal{C}_{\mathrm{lin}}(\Phi) \subseteq \mathcal{C}_{\mathrm{lin}}(\Phi')$.

(xiii) For every feature map $\Phi : \mathbb{R}^\ell \to \mathbb{R}^N$ with $||\Phi(x)|| = 1$ for all $x \in \mathbb{R}^\ell$, there exists a quantum feature map $\Phi' : \mathbb{R}^\ell \to \mathrm{Herm}\left(\mathbb{C}^{2^n}\right)$ that uses $n = \lceil \log N + 1 \rceil + 1$ qubits such that the families of linear classifiers satisfy $\mathcal{C}_{\mathrm{lin}}(\Phi) \subseteq \mathcal{C}_{\mathrm{qlin}}^{(\leq 1)}(\Phi')$.

(xiv) For every quantum feature map $\Phi : \mathbb{R}^\ell \to \mathrm{Herm}\left(\mathbb{C}^{2^n}\right)$, there exists a classical feature map $\Phi' : \mathbb{R}^\ell \to \mathbb{R}^{4^n}$ such that the families of linear classifiers satisfy $\mathcal{C}_{\mathrm{qlin}}(\Phi) = \mathcal{C}_{\mathrm{lin}}(\Phi')$.

Proof. (i): First, we define the feature map $\Phi' : \mathbb{R}^\ell \to \mathbb{R}^{N+1}$ which maps

$$x \mapsto \frac{\Phi(x)}{M} + \sqrt{1 - \frac{||\Phi(x)||^2}{M^2}} e_{N+1},$$

where e_{N+1} denotes the $(N+1)$-th standard basis vector. Note that this feature map indeed satisfies that $||\Phi'(x)|| = 1$ for all $x \in \mathbb{R}^\ell$. Next, for any classifier $c_{w,b} \in \mathcal{C}_{\mathrm{qlin}}(\Phi)$ we define $w' = w$ and $b' = b/M$ and we note that for any $x \in \mathbb{R}^\ell$ we have

$$\begin{aligned} c_{w',b'}(\Phi'(x)) &= \mathrm{sign}\big(\langle w', \Phi'(x)\rangle - b'\big) \\ &= \mathrm{sign}\big(M^{-1}\big[\langle w, \Phi(x)\rangle - b\big]\big) \\ &= \mathrm{sign}\big(\langle w, \Phi(x)\rangle - b\big) = c_{w,b}(\Phi(x)). \end{aligned}$$

(ii): First, we define the feature map $\widetilde{\Phi} : \mathbb{R}^\ell \to \mathbb{R}^{N+1}$ which maps

$$x \mapsto \Phi(x) + e_{N+1},$$

where e_{N+1} denotes the $(N+1)$-th standard basis vector. Next, for any classifier $c_{w,b} \in \mathcal{C}_{\mathrm{lin}}(\Phi)$ we define $\tilde{w} = w - be_{N+1}$ and we note that for all $x \in \mathbb{R}^\ell$ we have

$$c_{\tilde{w},0}(\widetilde{\Phi}(x)) = \mathrm{sign}\big(\langle \widetilde{\Phi}(x), \widetilde{w}\rangle\big) = \mathrm{sign}\big(\langle \Phi(x), w\rangle - b\big) = c_{w,b}(\Phi(x)).$$

Therefore, it suffices to show that we can implement any linear classifier on $\mathbb{R}^{N+1}$ with $b = 0$ as a quantum linear classifier on $n = \lceil \log N + 1 \rceil + 1$ qubits. To do so, we

define the quantum feature map $\Phi' : \mathbb{R}^\ell \to \mathrm{Herm}\big(\mathbb{C}^{2^n}\big)$ which maps

$$x \to \rho_x = \left(\frac{|\Phi(x)\rangle + |0\rangle}{\sqrt{2}}\right)\left(\frac{\langle\Phi(x)| + \langle 0|}{\sqrt{2}}\right),$$

where $|0\rangle$ is a vector that does not lie in the support of Φ (note this vectors exists since we have chosen n large enough). Finally, for any linear classifier $c_{w,0} \in \mathcal{C}_{\mathrm{lin}}(\Phi)$ on $\mathbb{R}^{N+1}$ we define $b' = ||w||^2/2$ and $\mathcal{O} = |w'\rangle\langle w'|$, where $|w'\rangle = |w\rangle + ||w||\,|0\rangle$ and we note that for all $x \in \mathbb{R}$ we have

$$\begin{aligned} c_{\mathcal{O},b'}(\Phi'(x)) &= \mathrm{sign}\big(\mathrm{Tr}\left[\mathcal{O}\rho_x\right] - b'\big) \\ &= \mathrm{sign}\big(\frac{1}{2}\Big|\,\langle w \mid \Phi(x)\rangle + ||w||\,\Big|^2 - \frac{||w||^2}{2}\big) \\ &= \mathrm{sign}\big(\langle w, \Phi(x)\rangle\big) = c_{w,0}(\Phi(x)). \end{aligned}$$

(iii): This follows directly from the fact that $\mathrm{Herm}\left(\mathbb{C}^{2^n}\right) \simeq \mathbb{R}^{4^n}$.

□

B.2.3 Proof of Proposition 17

Proposition 17. *Let $\mathcal{C}^{(\eta)}_{\mathrm{qlin}}$ denote the family of quantum linear classifiers corresponding to all n-qubit observables of Frobenius norm η, that is,*

$$\mathcal{C}^{(\eta)}_{\mathrm{qlin}} = \Big\{c(\rho) = \mathrm{sign}\big(\mathrm{Tr}\left[\mathcal{O}\rho\right] - d\big) \;\Big|\; \mathcal{O} \in \mathrm{Herm}\big(\mathbb{C}^{2^n}\big) \text{ with } ||\mathcal{O}||_F = \eta,\, d \in \mathbb{R}\Big\}. \tag{4.10}$$

Then, for every $\eta \in \mathbb{R}_{>0}$ and $0 < m \leq 2^n$ there exists a set of m examples consisting of binary labeled n-qubit pure states that satisfies the following two conditions:

(xv) There exists a classifier $c \in \mathcal{C}^{(\eta)}_{\mathrm{qlin}}$ that correctly classifies all examples with margin $\eta/\sqrt{m}$.

(xvi) No classifier $c' \in \mathcal{C}^{(\eta')}_{\mathrm{qlin}}$ with $\eta' < \eta$ can classify all examples correctly with margin $\geq \eta/\sqrt{m}$.

Proof. Define $\mathcal{D}_m = \mathcal{D}^+_m \cup \mathcal{D}^-_m$ whose positive examples (i.e., labeled $+1$) are given by

$$\mathcal{D}^+_m = \big\{\,|i\rangle\langle i| \mid i = 1, \ldots, \frac{m}{2}\big\},$$

and whose negative examples (i.e., labeled -1) are given by

$$\mathcal{D}^-_m = \big\{\,|i\rangle\langle i| \mid i = \frac{m}{2} + 1, \ldots, m\big\}.$$

To classify this set of examples we take the classifier $c_{\mathcal{O},0} \in \mathcal{C}^{(\eta)}_{\mathrm{qlin}}$ whose observable is

given by

$$\mathcal{O} = \frac{\eta}{\sqrt{m}} \left(\Big(\sum_{i=1}^{m/2} |i\rangle \langle i| \Big) + \Big(\sum_{j=\frac{m}{2}+1}^{m} |j\rangle \langle j| \Big) \right).$$

We remark that $c_{\mathcal{O},0}$ can indeed classify the set of examples $\mathcal{D}_r$ with margin $\eta/\sqrt{m}$.

Now suppose $c_{\mathcal{O}',b'} \in \mathcal{C}^{\eta'}_{\text{qlin}}$ with $\eta' < \eta$ can classify $\mathcal{D}_m$ with margin γ', that is

$$\operatorname{Tr}\big[\mathcal{O}' |i\rangle \langle i|\big] \begin{cases} \geq b' + \gamma' & \text{if } i = 1, \ldots, \frac{m}{2}, \\ \leq b' - \gamma' & \text{if } i = \frac{m}{2}+1, \ldots, m. \end{cases} \tag{B.18}$$

Define $\rho_+ = \sum_{i=1}^{m/2} |i\rangle \langle i|$ and $\rho_- = \sum_{i=\frac{m}{2}+1}^{m} |i\rangle \langle i|$ and note that Equation (B.18) implies that

$$\operatorname{Tr}\big[\mathcal{O}'\rho_+\big] \geq \frac{m}{2}b' + \frac{m}{2}\gamma'$$

and that

$$\operatorname{Tr}\big[\mathcal{O}'\rho_-\big] \leq \frac{m}{2}b' - \frac{m}{2}\gamma'$$

By combining these two inequalities we find that

$$\operatorname{Tr}\big[\mathcal{O}'(\rho_+ - \rho_-)\big] \geq \frac{m}{2}b' - \frac{m}{2}b' + \frac{m}{2}\gamma' + \frac{m}{2}\gamma' = m\gamma'. \tag{B.19}$$

Finally, by the Cauchy–Schwarz inequality we find that

$$\operatorname{Tr}\big[\mathcal{O}'(\rho_+ - \rho_-)\big] \leq \underbrace{||\mathcal{O}'||_F}_{<\eta} \cdot \underbrace{||\rho_+ - \rho_-||_F}_{=\sqrt{m}} < \eta\sqrt{m}. \tag{B.20}$$

Combining Equation (B.19) and (B.20) we find that

$$m\gamma' \leq \operatorname{Tr}\big[\mathcal{O}'(\rho_+ - \rho_-)\big] < \eta\sqrt{m}$$

from which we can conclude that $\gamma' < \eta/\sqrt{m}$.

□

Appendix C

Parametrized quantum policies for reinforcement learning

C.1 Derivation of the log-policy gradient

For a SOFTMAX-PQC defined in Def. 16, we have:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) &= \nabla_{\boldsymbol{\theta}} \log e^{\beta \langle O_a \rangle_{s,\boldsymbol{\theta}}} - \nabla_{\boldsymbol{\theta}} \log \sum_{a'} e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}} \\ &= \beta \nabla_{\boldsymbol{\theta}} \langle O_a \rangle_{s,\boldsymbol{\theta}} - \sum_{a'} \frac{e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}} \beta \nabla_{\boldsymbol{\theta}} \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}}{\sum_{a''} e^{\beta \langle O_{a''} \rangle_{s,\boldsymbol{\theta}}}} \\ &= \beta \left(\nabla_{\boldsymbol{\theta}} \langle O_a \rangle_{s,\boldsymbol{\theta}} - \sum_{a'} \pi_{\boldsymbol{\theta}}(a'|s) \nabla_{\boldsymbol{\theta}} \langle O_{a'} \rangle_{s,\boldsymbol{\theta}} \right).\end{aligned}$$

C.2 Efficient implementation of SOFTMAX-PQC policies

C.2.1 Efficient approximate policy sampling

In this section we prove Lemma 19, restated below:

Lemma 19. *For a* SOFTMAX-PQC *policy $\pi_{\boldsymbol{\theta}}$ defined by a unitary $U(s, \boldsymbol{\theta})$ and observables O_a, call $\langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}}$ approximations of the true expectation values $\langle O_a \rangle_{s,\boldsymbol{\theta}}$ with at most ε additive error. Then the approximate policy $\widetilde{\pi}_{\boldsymbol{\theta}} = \mathrm{softmax}_\beta(\langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}})$ has total variation distance $\mathcal{O}(\beta\varepsilon)$ to $\pi_{\boldsymbol{\theta}} = \mathrm{softmax}_\beta(\langle O_a \rangle_{s,\boldsymbol{\theta}})$. Since expectation values can be efficiently estimated to additive error on a quantum computer, this implies efficient approximate sampling from $\pi_{\boldsymbol{\theta}}$.*

Proof. Consider $|A|$ estimates $\left\{\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}\right\}_{1\leq a\leq|A|}$, obtained all to additive error ε, i.e.,

$$\left|\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}} - \langle O_a\rangle_{s,\boldsymbol{\theta}}\right| \leq \varepsilon, \quad \forall a$$

and used to compute an approximate policy

$$\widetilde{\pi}_{\boldsymbol{\theta}}(a|s) = \frac{e^{\beta\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'} e^{\beta\langle\widetilde{O_{a'}}\rangle_{s,\boldsymbol{\theta}}}}.$$

Due to the monoticity of the exponential, we have, for all a:

$$\begin{aligned} &\frac{e^{-\beta\varepsilon}e^{\beta\langle O_a\rangle_{s,\boldsymbol{\theta}}}}{e^{\beta\varepsilon}\sum_{a'} e^{\beta\langle O_{a'}\rangle_{s,\boldsymbol{\theta}}}} \leq \frac{e^{\beta\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'} e^{\beta\langle\widetilde{O_{a'}}\rangle_{s,\boldsymbol{\theta}}}} \leq \frac{e^{\beta\varepsilon}e^{\beta\langle O_a\rangle_{s,\boldsymbol{\theta}}}}{e^{-\beta\varepsilon}\sum_{a'} e^{\beta\langle O_{a'}\rangle_{s,\boldsymbol{\theta}}}} \\ \Leftrightarrow \quad & e^{-2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s) \leq \quad \widetilde{\pi}_{\boldsymbol{\theta}}(a|s) \quad \leq e^{2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s). \end{aligned} \tag{C.1}$$

Hence,

$$\begin{aligned} \mathrm{TV}(\pi_{\boldsymbol{\theta}}, \widetilde{\pi}_{\boldsymbol{\theta}}) &= \sum_a |\widetilde{\pi}_{\boldsymbol{\theta}}(a|s) - \pi_{\boldsymbol{\theta}}(a|s)| \\ &\leq \sum_a \left|e^{2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s) - e^{-2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s)\right| \\ &= \sum_a \left|e^{2\beta\varepsilon} - e^{-2\beta\varepsilon}\right|\pi_{\boldsymbol{\theta}}(a|s) \\ &= 2|\sinh(2\beta\varepsilon)| \underset{\beta\varepsilon\to 0^+}{=} 4\beta\varepsilon + \mathcal{O}\left((\beta\varepsilon)^3\right), \end{aligned}$$

where $\mathrm{TV}(.,.)$ denotes the total-variation distance, and we used

$$\{\widetilde{\pi}_{\boldsymbol{\theta}}(a|s), \pi_{\boldsymbol{\theta}}(a|s)\} \in [e^{-2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s), e^{2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s)]$$

in the first inequality. □

C.2.2 Efficient estimation of the log-policy gradient

Using a similar approach to the proof of the previous section, we show the following lemma:

Lemma 44. *For a* SOFTMAX-PQC *policy $\pi_{\boldsymbol{\theta}}$ defined by a unitary $U(s,\boldsymbol{\theta})$ and observables O_a, call $\partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}$ approximations of the true derivatives $\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}}$ with at most ε additive error, and $\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}$ approximations of the true expectation values $\langle O_a\rangle_{s,\boldsymbol{\theta}}$ with at most $\varepsilon' = \varepsilon(4\beta\max_a\|O_a\|)^{-1}$ additive error. Then the approximate log-policy gradient $\nabla_{\boldsymbol{\theta}}\log\widetilde{\pi_{\boldsymbol{\theta}}}(a|s) = \beta\big(\nabla_{\boldsymbol{\theta}}\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}} - \sum_{a'}\widetilde{\pi_{\boldsymbol{\theta}}}(a'|s)\nabla_{\boldsymbol{\theta}}\langle\widetilde{O_{a'}}\rangle_{s,\boldsymbol{\theta}}\big)$ has distance $\mathcal{O}(\beta\varepsilon)$ to $\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a|s)$ in ℓ_∞-norm.*

Proof. Call $x_{a,i} = \pi_{\boldsymbol{\theta}}(a|s)\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}}$ and $\widetilde{x}_{a,i} = \widetilde{\pi}_{\boldsymbol{\theta}}(a|s)\partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}$, such that:

$$\partial_i \log\widetilde{\pi_{\boldsymbol{\theta}}}(a|s) = \beta\Big(\partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}} - \sum\nolimits_{a'}\widetilde{x}_{a',i}\Big).$$

and similarly for $\partial_i \log\pi_{\boldsymbol{\theta}}(a|s)$.
Using Eq. (C.1) and that $|\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} - \partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}| \leq \varepsilon, \forall a, i$, we have:

$$e^{-2\beta\varepsilon'}\pi_{\boldsymbol{\theta}}(a|s)\left(\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} - \varepsilon\right) \leq \widetilde{\pi}_{\boldsymbol{\theta}}(a|s)\partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}} \tag{C.2}$$

$$\leq e^{2\beta\varepsilon'}\pi_{\boldsymbol{\theta}}(a|s)\left(\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} + \varepsilon\right) \tag{C.3}$$

which implies that

$$e^{-2\beta\varepsilon'}\left(\sum_a x_{a,i} - \varepsilon\right) \leq \sum_a \widetilde{x}_{a,i} \leq e^{2\beta\varepsilon'}\left(\sum_a x_{a,i} + \varepsilon\right) \tag{C.4}$$

where we summed the first inequalities over all a. Hence:

$$\begin{aligned}
\left|\sum_a x_{a,i} - \sum_a \widetilde{x}_{a,i}\right| &\leq \left|e^{2\beta\varepsilon'}\left(\sum_a x_{a,i} + \varepsilon\right) - e^{-2\beta\varepsilon'}\left(\sum_a x_{a,i} - \varepsilon\right)\right| \\
&\leq \left|(e^{2\beta\varepsilon'} + e^{-2\beta\varepsilon'})\varepsilon + (e^{2\beta\varepsilon'} - e^{-2\beta\varepsilon'})\sum_a x_{a,i}\right| \\
&\leq \left|2\cosh(2\beta\varepsilon')\varepsilon + 2\sinh(2\beta\varepsilon')\sum_a x_{a,i}\right| \\
&\underset{\beta\varepsilon'\to 0^+}{=} \left|\varepsilon + 4\beta\varepsilon'\sum_a x_{a,i} + \mathcal{O}\left((\beta\varepsilon')^2\varepsilon\right) + \mathcal{O}\left((\beta\varepsilon')^3\right)\right|.
\end{aligned} \tag{C.5}$$

We also have

$$\left|\sum_a x_{a,i}\right| = \left|\sum_a \pi_{\boldsymbol{\theta}}(a|s)\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}}\right| \leq \max_{a,i}|\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}}| \leq \max_a \|O_a\|$$

where the last inequality derives from the parameter-shift rule (Eq. (5.4)) formulation of $\partial_i\langle O_a\rangle$ for derivatives w.r.t. rotation angles of the PQC and the fact that $\partial_i\langle O_a\rangle$ are simply expectation values $\langle H_{a,i}\rangle$ with $\|H_{a,i}\| \leq \|O_a\|$ for observable weights. Applying the triangular inequality on the right side of Eq. (C.5), we hence have:

$$\left|\sum_a x_{a,i} - \sum_a \widetilde{x}_{a,i}\right| \underset{\beta\varepsilon'\to 0^+}{\leq} \varepsilon + 4\beta\varepsilon'\max_a\|O_a\| + \mathcal{O}\left((\beta\varepsilon')^2\varepsilon\right) + \mathcal{O}\left((\beta\varepsilon')^3\right).$$

For $\varepsilon' = \varepsilon(4\beta\max_a\|O_a\|)^{-1}$ and using $|\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} - \partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}| \leq \varepsilon, \forall a, i$, we finally have:

$$|\partial_i \log\pi_{\boldsymbol{\theta}}(a|s) - \partial_i \log\widetilde{\pi_{\boldsymbol{\theta}}}(a|s)| \underset{\beta\varepsilon\to 0^+}{\leq} 3\beta\varepsilon + \mathcal{O}(\beta\varepsilon^3) \quad \forall i$$ □

C.3 Role of trainable observables in SOFTMAX-PQC

In Sec. 5.1.1, we presented a general definition of the SOFTMAX-PQC observables $O_a = \sum_i w_{a,i} H_{a,i}$ in terms of an arbitrary weighted sum of Hermitian matrices $H_{a,i}$. In this appendix, we clarify the role of such a decomposition.

C.3.1 Training the eigenbasis and the eigenvalues

Consider a projective measurement defined by an observable $O = \sum_m \alpha_m P_m$, to be performed on a quantum state of the form $V(\boldsymbol{\theta})\,|\psi\rangle$, where $V(\boldsymbol{\theta})$ denotes a (variational) unitary. Equivalently, one could also measure the observable $V^\dagger(\boldsymbol{\theta})OV(\boldsymbol{\theta})$ on the state $|\psi\rangle$. Indeed, these two measurements have the same probabilities $p(m) = \langle\psi|\, V^\dagger(\boldsymbol{\theta})P_m V(\boldsymbol{\theta})\,|\psi\rangle$ of measuring any outcome α_m. Note also that the possible outcomes α_m (i.e., the eigenvalues of the observable O) remain unchanged.

From this observation, it is then clear that, by defining an observable $O = \sum_m \alpha_m P_m$ using projections P_m on each computational basis state of the Hilbert space $\mathcal{H}$ and arbitrary eigenvalues $\alpha_m \in \mathbb{R}$, the addition of a *universal* variational unitary $V(\boldsymbol{\theta})$ prior to the measurement results in a family of observables $\{V^\dagger(\boldsymbol{\theta})OV(\boldsymbol{\theta})\}_{\boldsymbol{\theta},\boldsymbol{\alpha}}$ that covers all possible Hermitian observables in $\mathcal{H}$. Moreover, in this setting, the parameters that define the eigenbasis of the observables $V^\dagger(\boldsymbol{\theta})OV(\boldsymbol{\theta})$ (i.e., $\boldsymbol{\theta}$) are completely distinct from the parameters that define their eigenvalues (i.e., $\boldsymbol{\alpha}$). This is not the case for observables that are expressed as linear combinations of non-commuting matrices, for instance.

In our simulations, we consider restricted families of observables. In particular, we take the Hermitian matrices $H_{a,i}$ to be diagonal in the computational basis (e.g., tensor products of Pauli-Z matrices), which means they, as well as O_a, can be decomposed in terms of projections on the computational basis states. However, the resulting eigenvalues $\boldsymbol{\alpha}$ that we obtain from this decomposition are in our case degenerate, which means that the weights $\boldsymbol{w}_a$ underparametrize the spectrums of the observables O_a. Additionally, the last variational unitaries $V_{\text{var}}(\boldsymbol{\phi}_L)$ of our PQCs are far from universal, which restricts the accessible eigenbasis of all variational observables $V_{\text{var}}^\dagger(\boldsymbol{\phi}_L)O_a V_{\text{var}}(\boldsymbol{\phi}_L)$.

C.3.2 The power of universal observables

Equivalently to the universal family of observables $\{V^\dagger(\boldsymbol{\theta})OV(\boldsymbol{\theta})\}_{\boldsymbol{\theta},\boldsymbol{\alpha}}$ that we defined in the previous section, one can construct a family of observables $\{O_{\boldsymbol{w}} = \sum_i w_i H_i\}_{\boldsymbol{w}}$ that parametrizes all Hermitian matrices in $\mathcal{H}$ (e.g., by taking H_i to be single components of a Hermitian matrix acting on $\mathcal{H}$). Note that this family is covered by our definition of SOFTMAX-PQC observables. Now, given access to data-dependent quantum states $|\psi_s\rangle$ that are expressive enough (e.g., a binary encoding of the input s, or so-called universal quantum feature states [89]), one can approximate arbitrary functions of s using expectations values of the form $\langle\psi_s|\, O_{\boldsymbol{w}}\,|\psi_s\rangle$. This is because the observables $O_{\boldsymbol{w}}$ can encode an arbitrary quantum computation. Hence, in the case of our SOFTMAX-PQCs, one could use such observables and such encodings $|\psi_s\rangle$ of the input states s to approximate any policy $\pi(a|s)$ (using an additional softmax),

without the need for any variational gates in the PQC generating $|\psi_s\rangle$.

As we mentioned in the previous section, the observables that we consider in this work are more restricted, and moreover, the way we encode the input states s leads to non-trivial encodings $|\psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}}\rangle$ in general. This implies that the variational parameters $\boldsymbol{\phi}, \boldsymbol{\lambda}$ of our PQCs have in general a non-trivial role in learning good policies. One can even show here that these degrees of freedom are sufficient to make such PQCs universal function approximators [158].

C.4 Environments specifications and hyperpameters

In Table C.1, we present a specification of the environments we consider in our numerical simulations. These are standard benchmarking environments from the OpenAI Gym library [43], described in Ref. [151], PQC-generated environments that we define in Sec. 5.3.2, and the CognitiveRadio environment of Ref. [58] that we discuss in Appendix C.5.

Environment	State dimension	Number of actions	Horizon	Reward function	Termination conditions
CartPole-v1	4	2	500	+1 until termination	• Pole angle or cart position outside of bounds • Reaching horizon
MountainCar-v0	2	3	200	−1 + height until termination	Reaching goal or horizon
Acrobot-v1	6	3	500	−1 until termination	Reaching goal or horizon
SL-PQC	2	2	20	+1 for good action −1 for wrong action	Reaching horizon
Cliffwalk-PQC	2	2	20	+1 for good action −1 for wrong action	• Doing wrong action • Reaching horizon
CognitiveRadio	2 to 5 (discrete)	2 to 5	100	+1 for good action −1 for wrong action	Reaching horizon

Table C.1: **Environments specifications.** The reward function of Mountaincar-v0 has been modified compared to the standard specification of OpenAI Gym [43], similarly to Ref. [71].

In Tables C.2 and C.3, we list the hyperparameters used to train our agents on the various environments we consider. All agents use an ADAM optimizer. For the plots presented in this manuscript, all quantum circuits were implemented using the Cirq library [88] in Python and simulated using a Qulacs backend [184] in C++. For the tutorial [161], the TensorFlow Quantum library [44] was used.
All simulations were run on the LEO cluster (more than 3000 CPUs) of the University of Innsbruck, with an estimated total compute time (including hyperparametrization) of 20 000 CPU-hours.

C.5 Deferred plots and shape of policies PQCs vs. DNNs

C.5.1 Influence of architectural choices on RAW-PQC

In Fig. C.1, we run a similar experiment to that of Sec. 5.2.2 in the main text, but on RAW-PQC agents instead of SOFTMAX-PQC agents. We observe that both increasing the depth of the PQCs and training the scaling parameters $\boldsymbol{\lambda}$ have a similar positive influence on the learning performance, and even more pronounced than for SOFTMAX-PQC agents. Nonetheless, we also observe that, even at greater depth, the final performance, as well as the speed of convergence, of RAW-PQC agents remain limited compared to that of SOFTMAX-PQC agents.

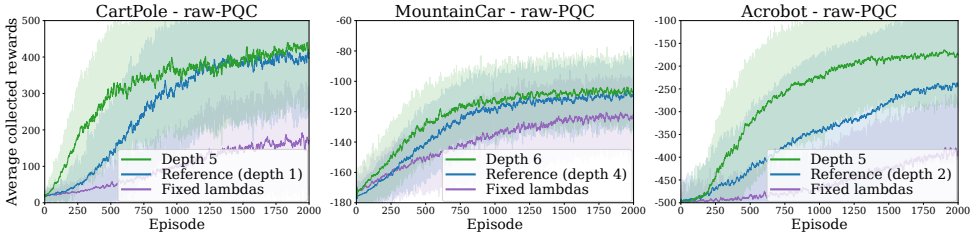


Figure C.1: **Influence of the model architecture for RAW-PQC agents.** The blue curves in each plot correspond to the learning curves from Fig. 5.2 and are taken as a reference.

Environment	Model	Learning rates	Discount γ	Final β	Batch size	Depth	Width
CartPole-v1	SOFTMAX-PQC	$[0.01, 0.1, 0.1]$	1	1	10	$\{1, 5\}$	4
	RAW-PQC	$[0.01, 0., 0.1]$	1	✗	10	$\{1, 5\}$	4
MountainCar-v0	SOFTMAX-PQC	$[0.01, 0.1, 0.01]$	1	1.5	10	$\{4, 6\}$	2
	RAW-PQC	$[0.01, 0., 0.01]$	1	✗	10	$\{4, 6\}$	2
Acrobot-v1	SOFTMAX-PQC	$[0.01, 0.1, 0.1]$	1	1	10	$\{2, 5\}$	6
	RAW-PQC	$[0.01, 0., 0.1]$	1	✗	10	$\{2, 5\}$	6
SL-PQC	SOFTMAX-PQC	$[0.01, 0.1, 0.01]$	0.9	1	10	4	2
	DNN	0.01	0.9	1	10	4	16
Cliffwalk-PQC	SOFTMAX-PQC	$[0.01, 0.1, 0.1]$	0.9	1	10	4	2
	DNN	0.01	0.9	1	10	4	16
CognitiveRadio	SOFTMAX-PQC	$[0.01, 0.1, 0.1]$	0.9	1	1	3	2 to 5

Table C.2: **Hyperparmeters 1/2.** For PQC policies, we choose 3 distinct learning rates $[\alpha_{\boldsymbol{\phi}}, \alpha_{\boldsymbol{w}}, \alpha_{\boldsymbol{\lambda}}]$ for rotation angles $\boldsymbol{\phi}$, observable weights $\boldsymbol{w}$ and scaling parameters $\boldsymbol{\lambda}$, respectively. For SOFTMAX-PQCs, we take a linear annealing schedule for the inverse temperature parameter β starting from 1 and ending up in the final β. The batch size is counted in number of episodes used to evaluate the gradient of the value function. Depth indicates the number of encoding layers D_{enc} for PQC policies, or the number of hidden layers for a DNN policy. Width corresponds to the number of qubits n on which acts a PQC (also equal to the dimension d of the environment's state space), or the number of units per hidden layer for a DNN.

Environment	Model	Entang. topology	Train entang.	Observables	Number of params.	Baseline
CartPole-v1	SOFTMAX-PQC	All-to-all	Yes	$[wZ_0Z_1Z_2Z_3, (-\dots)]$	$\{31, 119\}$	No
	RAW-PQC	All-to-all	Yes	$[Z_0Z_1Z_2Z_3, (-\dots)]$	$\{30, 118\}$	No
MountainCar-v0	SOFTMAX-PQC	One-to-one	No	$[w_0Z_0, w_1Z_0Z_1, w_2Z_1]$	$\{39, 55\}$	Yes
	RAW-PQC	One-to-one	No	$[P_{0,1}, P_2, P_3]$	$\{36, 52\}$	Yes
Acrobot-v1	SOFTMAX-PQC	Circular	Yes	$\left[\boldsymbol{w}_i \cdot (Z_0, \dots, Z_5)^T\right]_{1 \leq i \leq 3}$	$\{90, 180\}$	Yes
	RAW-PQC	Circular	Yes	$[P_{0..21}, P_{22..42}, P_{43..63}]$	$\{72, 162\}$	Yes
SL-PQC	SOFTMAX-PQC	One-to-one	No	$[wZ_0Z_1, (-\dots)]$	37	No
	DNN	✗	✗	✗	902	No
Cliffwalk-PQC	SOFTMAX-PQC	One-to-one	No	$[wZ_0Z_1, (-\dots)]$	37	No
	DNN	✗	✗	✗	902	No
CognitiveRadio	SOFTMAX-PQC	Circular	No	$[w_0Z_0, w_1Z_1, \dots, w_nZ_n]$	30 to 75	No

Table C.3: **Hyperparmeters 2/2.** We call entangling layer a layer of 2-qubit gates in the PQC. Circular and all-to-all topologies of entangling layers are equivalent for $n = 2$ qubits, so we call them one-to-one in that case. When trained, entangling layers are composed of $R_{zz} = e^{-i\theta(Z\otimes Z)/2}$ rotations, otherwise, they are composed of Ctrl-Z gates. For policies with 2 actions, the same observable, up to a sign change, is used for both actions. Z_i refers to a Pauli-Z observable acting on qubit i, while $P_{i..j}$ indicates a projection on basis states i to j. In the experiments of Sec. 5.2.2, when the weights of the SOFTMAX-PQC are kept fixed, the observables used for MountainCar-v0 and Acrobot-v1 are $[Z_0, Z_0Z_1, Z_1]$, and those used for CartPole-v1 are $[Z_0Z_1Z_2Z_3, -Z_0Z_1Z_2Z_3]$. The different number of parameters in a given row correspond to the different depths in that same row in Table C.2.

C.5.2 Shape of the policies learned by PQCs v.s. DNNs

In CartPole-v1 The results of the Sec. 5.2 demonstrate that our PQC policies can be trained to good performance in benchmarking environments. To get a feel of the solutions found by our agents, we compare the SOFTMAX-PQC policies learned on CartPole to those learned by standard DNNs (with a softmax output layer), which are known to easily learn close-to-optimal behavior on this task. More specifically, we look at the functions learned by these two models, prior to the application of the softmax normalization function (see Eq. (5.2)). Typical instances of these functions are depicted in Figure C.3. We observe that, while DNNs learn simple, close to piece-wise linear functions of their input state space, PQCs tend to naturally learn very oscillating functions that are more prone to instability. While the results of Schuld *et al.* [166] already indicated that these highly oscillating functions would be natural for PQCs, it is noteworthy to see that these are also the type of functions naturally learned in a direct-policy RL scenario. Moreover, our enhancements to standard PQC classifiers show how to make these highly oscillating functions more amenable to real-world tasks.

In PQC-generated environments Fig. C.4 shows the analog results to Fig. 5.4 in the main text but with two different random initializations of the environment-generating PQC. Both confirm our observations. In Fig. C.5, we compare the policies learned by prototypical SOFTMAX-PQC and DNN agents in these PQC-generated environments. We observe that the typical policies learned by DNNs are rather simple, with up to 2 (or 3) regions, delimited by close-to-linear boundaries, as opposed to the policies learned by SOFTMAX-PQCs, which delimit red from blue regions with wide margins. These observations highlight the inherent flexibility of SOFTMAX-PQC policies and their suitability to these PQC-generated environments, as opposed to the DNN (and RAW-PQC) policies we consider.

C.5.3 Additional simulations on CognitiveRadio

In a related work on value-based RL with PQCs, the authors of Ref. [58] introduced the CognitiveRadio environment as a benchmark to test their RL agents. In this environment, the agent is presented at each interaction step with a binary vector $(0, 0, 0, 1, 0)$ of size n that describes the occupation of n radio channels. Given this state, the agent must select one of the n channels as its communication channel, such as to avoid collision with occupied channels (a ± 1 reward reflects these collisions). The authors of Ref. [58] consider a setting where, in any given state, only one channel is occupied, and its assignment changes periodically over time steps, for an episode length of 100 steps. While this constitutes a fairly simple task environment with discrete state and action spaces, it allows to test the performance of PQC agents on a family of environments described by their system size n and make claims on the parameter complexity of the PQCs as a function of n. As to reproduce the findings of Ref. [58] in a policy-gradient setting, we test the performance of our SOFTMAX-PQC agents on this environment. We find numerically (see Fig. C.2) that these achieve a very similar performance to the PQC agents of Ref. [58] on the same system sizes they consider ($n = 2$ to 5), using PQCs with the same scaling of number of parameters, i.e., $\mathcal{O}(n)$.

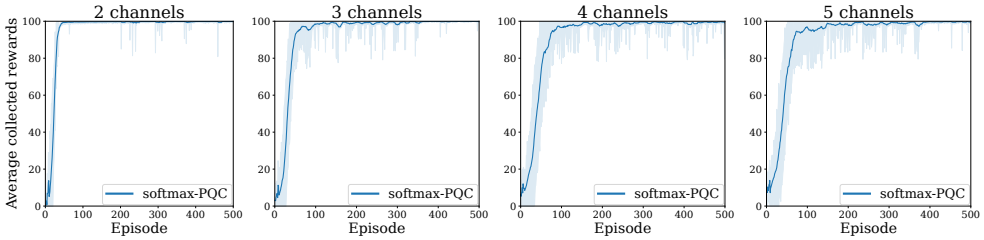
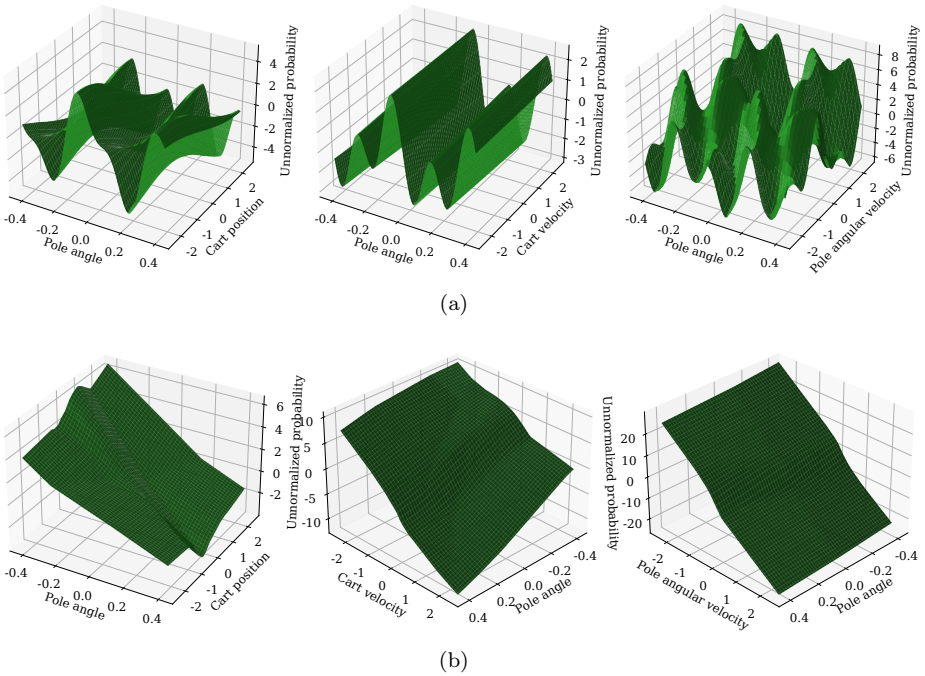


Figure C.2: **Performance of our SOFTMAX-PQC agents on the CognitiveRadio environment proposed in Ref. [58].** Average performance of 20 agents for system sizes (and number of qubits) $n = 2$ to 5.



(a)

(b)

Figure C.3: **Prototypical unnormalized policies learned by SOFTMAX-PQC agents and DNN agents in CartPole.** Due to the 4 dimensions of the state space in CartPole, we represent the unnormalized policies learned by (a) SOFTMAX-PQC agents and (b) DNN agents on 3 subspaces of the state space by fixing unrepresented dimensions to 0 in each plot. To get the probability of the agent pushing the cart to the left, one should apply the logistic function (i.e., 2-dimensional softmax) $1/(1 + exp(-z))$ to the z-axis values of each plot.

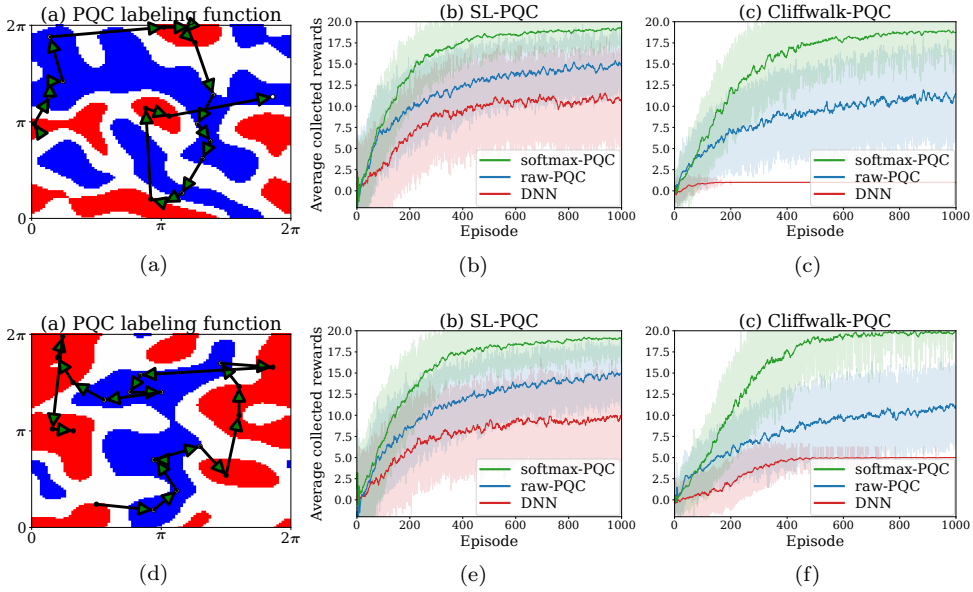


Figure C.4: **Different random initializations of PQC-generated environments and their associated learning curves.** See Fig. 5.4 for details. The additional learning curves (20 agents per curve) of randomly-initialized RAW-PQC agents highlight the hardness of these environments for PQC policies drawn from the same family as the environment-generating PQCs.

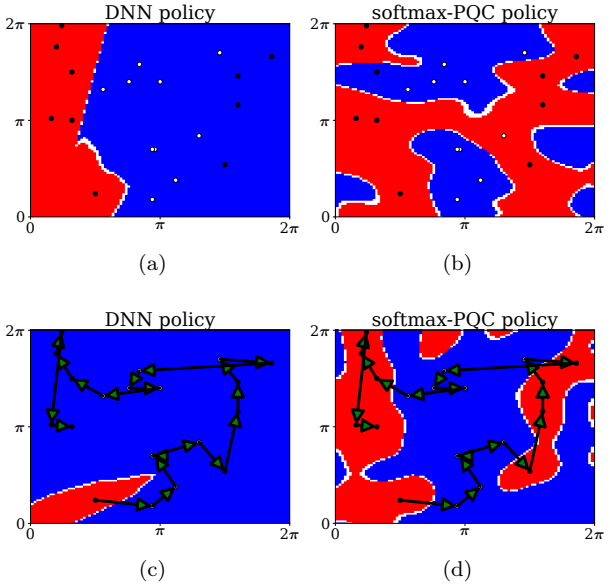


Figure C.5: **Prototypical policies learned by SOFTMAX-PQC agents and DNN agents in PQC-generated environments.** All policies are associated to the labeling function of Fig. C.4.d. Policies (a) and (b) are learned in the SL-PQC environment while policies (c) and (d) are learned in the Cliffwalk-PQC environment.

C.6 Supervised learning task of Liu *et al.*

Define p a large prime number, $n = \lceil \log_2(p-1) \rceil$, and g a generator of $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ (i.e., a $g \in \mathbb{Z}_p^*$ such that $\{g^y, y \in \mathbb{Z}_{p-1}\} = \mathbb{Z}_p^*$). The DLP consists in computing $\log_g x$ on input $x \in \mathbb{Z}_p^*$. Based on DLP, Liu *et al.* [128] define a concept class $\mathcal{C} = \{f_s\}_{s \in \mathbb{Z}_{p-1}}$ over the input space $\mathcal{X} = \mathbb{Z}_p^*$, where each labeling function of this concept class is defined as follows:

$$f_s(x) = \begin{cases} +1, & \text{if } \log_g x \in [s, s + \frac{p-3}{2}], \\ -1, & \text{otherwise.} \end{cases} \tag{C.6}$$

Each function $f_s : \mathbb{Z}_p^* \to \{-1, 1\}$ hence labels half the elements in $\mathbb{Z}_p^*$ with a label $+1$ and the other half with a label -1. We refer to Figure 1 in Ref. [128] for a good visualization of all these objects.
The performance of a classifier f is measured in terms of its testing accuracy

$$\mathrm{Acc}_f(f_s) = \Pr_{x \sim \mathcal{X}}[f(x) = f_s(x)].$$

C.7 Proof of Theorem 20

In the following, we provide constructions of *a)* fully random, *b)* partially random and *c)* fully deterministic environments satisfying the properties of Theorem 20. We consider the three families of environments separately and provide individual lemmas specifying their exact separation properties.

Fully random: the SL-DLP environment. This result is near-trivially obtained by noting that any classification problem can be easily mapped to a (degenerate) RL problem. For this, the environment will be an MDP defined as follows: its state space is the input space of the classification problem, its action space comprises all possible labels, rewards are trivially $+1$ for assigning a correct label to an input state and -1 otherwise, and the initial and next-state transition probabilities are state-independent and equal to the input distribution of the classification task. The optimal policy of this MDP is clearly the optimal classifier of the corresponding SL task. Consider now the classification task of Liu *et al.*, defined in detail in Appendix C.6: the input distribution is taken to be uniform on the state space, i.e., $P(s_t) = \frac{1}{|S|}$, and the performance of a classifier f w.r.t. a labeling (or ground truth) function f^* is measured in terms of a testing accuracy

$$\mathrm{Acc}_f(f^*) = \frac{1}{|S|} \sum_s \Pr[f(s) = f^*(s)]. \tag{C.7}$$

For the MDP associated to this classification task and length-1 episodes of interaction, the value function of any policy $\pi(a|s)$ is given by

$$\begin{aligned} V_\pi(s_0) &= \frac{1}{|S|}\sum_{s_0}\left(\pi(f^*(s_0)|s_0) - \pi(-f^*(s_0)|s_0)\right) \\ &= \frac{1}{|S|}\sum_{s_0} 2\pi(f^*(s_0)|s_0) - 1 \\ &= 2\mathrm{Acc}_\pi(f^*) - 1, \end{aligned}$$

which is trivially related to the testing accuracy of this policy on the classification task. Note that we also have $V_{\mathrm{rand}}(s_0) = 0$ and $V_{\mathrm{opt}}(s_0) = 1$.
Since these observations hold irrespectively of the labeling function f^*, we can show the following result:

Lemma 45 (Quantum advantage in SL-DLP)**.** *There exists a uniform family of SL-DLP MDPs, each derived from a labeling function f^* of the DLP concept class $\mathcal{C}$ (see Appendix C.6), for which classical hardness and quantum learnability holds. More specifically, the performance of any classical learner is upper bounded by $1/\mathrm{poly}(n)$, while that of a class of quantum agents is lower bounded by 0.98 with probability above $2/3$ (over the randomness of their interaction with the environment and noise in their implementation).*

Proof. Classical hardness is trivially obtained by contraposition: assuming no classical polynomial-time algorithm can solve DLP, then using Theorem 1 of Liu *et al.*, any classical policy would have testing accuracy $\mathrm{Acc}_\pi(f^*) \leq 1/2 + 1/\mathrm{poly}(n)$, and hence its value function would be $V_\pi(s_0) \leq 1/\mathrm{poly}(n)$.

For quantum learnability, we define an agent that first collects $\mathrm{poly}(n)$ random length-1 interactions (i.e., a random state s_0 and its associated reward for an action $+1$, from which the label $f^*(s_0)$ can be inferred), and use Theorem 2 of Liu *et al.* to train a classifier that has test accuracy at least 0.99 with probability at least $2/3$ (this process can be repeated $\mathcal{O}\left(\log\left(\delta^{-1}\right)\right)$ times to increase this probability to $1-\delta$ via majority voting). This classifier has a value function $V_\pi(s_0) \geq 0.98$. □

Note that this proof trivially generalizes to episodes of interaction with length greater than 1, when preserving the absence of temporal correlation in the states experienced by the agents. For episodes of length H, the only change is that the value function of any policy, and hence the bounds we achieve, get multiplied by a factor of $\frac{1-\gamma^H}{1-\gamma}$ for a discount factor $\gamma < 1$ and by a factor H for $\gamma = 1$.

Partially random: the Cliffwalk-DLP environment. One major criticism to the result of Lemma 45 is that it applies to a very degenerate, fully random RL environment. In the following, we show that similar results can be obtained in environments based on the same classification problem, but while imposing more temporal structure and less randomness (such constructions were introduced in Ref. [72], but for the purpose of query separations between RL and QRL). For instance, one can

consider cliffwalk-type environments, inspired by the textbook "cliff walking" environment of Sutton & Barto [183]. This class of environments differs from the previous SL-DLP environments in its state and reward structure: in any episode of interaction, experienced states follow a fixed "path" structure (that of the cliff) for correct actions, and a wrong action yields to immediate "death" (negative reward and episode termination). We slightly modify this environment to a "slippery scenario" in which, with a δ probability, any action may lead to a uniformly random position on the cliff. This additional randomness allows us to prove the following separation:

Lemma 46 (Quantum advantage in Cliffwalk-DLP)**.** *There exists a uniform family of Cliffwalk-DLP MDPs with arbitrary slipping probability $\delta \in [0.86, 1]$ and discount factor $\gamma \in [0, 0.9]$, each derived from a labeling function f^* of the DLP concept class $\mathcal{C}$, for which classical hardness and quantum learnability holds. More specifically, the performance of any classical learner is upper bounded by $V_{\text{rand}}(s_0) + 0.1$, while that of a class of quantum agents is lower bounded by $V_{\text{opt}}(s_0) - 0.1$ with probability above $2/3$ (over the randomness of their interaction with the environment and noise in their implementation). Since $V_{\text{rand}}(s_0) \leq -\frac{1}{2}$ and $V_{\text{opt}} = 0$, we always have a classical-quantum separation.*

The proof of this lemma is deferred to Appendix C.8 for clarity.

Fully deterministic: the Deterministic-DLP environment. The simplest example of a deterministic RL environment where separation can be proven is a partially observable MDP (POMDP) defined as follows: it constitutes a 1-D chain of states of length $k+2$, where k is poly(n). We refer to the first k states as "training states", and we call the last two states "test" and "limbo" states, respectively. The training states are of the form $(x, f_s(x))$, i.e., a point uniformly sampled and its label. The actions are $+1, -1$, and both lead to the same subsequent state on the chain (since the same $(x, f_s(x))$ can appear twice in the chain, this is the reason why the environment is partially observable), and no reward is given for the first k states. In the test state, the agent is only given a point x with no label. A correct action provides a reward of 1 and leads to the beginning of the chain, while an incorrect action leads to the limbo state, which self-loops for both actions and has no rewards. In other words, after poly-many examples where the agent can learn the correct labeling, it is tested on one state. Failure means it will never obtain a reward.

For each concept f_s, we define exponentially many environments obtained by random choices of the states appearing in the chain. In a given instance, call $T = (x_0, \ldots, x_{k-1})$ the training states of that instance, x_k its testing state and l its limbo state. The interaction of an agent with the environment is divided into episodes of length $k+1$, but the environment keeps memory of its state between episodes. This means that, while the first episode starts in x_0, depending on the performance of the agent, later episodes start either in x_0 or in l. For a policy π, we define the value $V_\pi(s_0)$ as the expected reward[1] of this policy in any episode of length $k+1$ with an initial state $s_0 \in \{x_0, l\}$. Since the testing state x_k is the only state to be rewarded,

[1]Note that we assume here a discount factor $\gamma = 1$, but our results would also hold for an arbitrary $\gamma > 0$, if we scale the reward of the testing state to γ^{-k}.

we can already note that $V_\pi(x_0) = \pi(f^*(x_k)|T, x_k)$, that is, the probability of the policy correctly labeling the testing state x_k after having experienced the training states T. Also, since $s_0 \in \{x_0, l\}$ and $V_\pi(l) = 0$, we have $V_\pi(x_0) \geq V_\pi(s_0)$.

With this construction, we obtain the following result:

Lemma 47 (Quantum advantage in Deterministic-DLP)**.** *There exists a uniform family of Deterministic-DLP POMDPs (exponentially many instances for a given concept f_s of the DLP classification problem) where:*
1) (classical hardness) if there exists a classical learning agent which, when placed in a randomly chosen instance of the environment, has value $V_c(s_0) \geq 1/2 + 1/\text{poly}(n)$ (that is, $1/\text{poly}(n)$ better than a random agent), with probability at least 0.845 over the choice of environment and the randomness of its learning algorithm, then there exists an efficient classical algorithm to solve DLP,
2) (quantum learnability) there exists a class of quantum agents that attains a value $V_q(s_0) = 1$ (that is, the optimal value) with probability at least 0.98 over the choice of environment and randomness of the learning algorithm.

The proof of this lemma is deferred to Appendix C.9 for clarity.

By combining our three lemmas, and taking the weakest separation claim for the cases *ii)* and *iii)*, we get Theorem 20. For the interested reader, we list the following remarks, relating to the proofs of these lemmas:

- SL-DLP and Deterministic-DLP are the two closest environments to the DLP classification task of Liu *et al.* While the value function in SL-DLP is trivially equivalent to the accuracy of the classification problem, we find the value function in Deterministic-DLP to be *weaker* than this accuracy. Namely, a high accuracy trivially leads to a high value while a high (or non-trivial) value does not necessarily lead to a high (or non-trivial) accuracy (in all these cases, the high probability over the randomness of choosing the environments and of the learning algorithms is implied). This explains why the classical hardness statement for Deterministic-DLP is weaker than in SL-DLP.

- In Cliffwalk-DLP, it is less straightforward to relate the testing accuracy of a policy to its performance on the deterministic parts of the environment, which explains why we trivially upper bound this performance by 0 on these parts. We believe however that these deterministic parts will actually make the learning task much harder, since they strongly restrict the part of the state space the agents can see. This claim is supported by our numerical experiments in Sec. 5.3.2. Also, since we showed classical hardness for fully deterministic environments, it would be simple to construct a variant of Cliffwalk-DLP where these deterministic parts would be provably hard as well.

C.8 Proof of Lemma 46

Consider a slippery cliffwalk environment defined by a labeling function f^* in the concept class $\mathcal{C}$ of Liu *et al.* This cliffwalk has $p - 1$ states ordered, w.l.o.g., in their natural order, and correct actions (the ones that do not lead to immediate "death")

$f^*(i)$ for each state $i \in \mathbb{Z}_p^*$. For simplicity of our proofs, we also consider circular boundary conditions (i.e, doing the correct action on the state $p-1$ of the cliff leads to the state 1), random slipping at each interaction step to a uniformly sampled state on the cliff with probability $\delta > 0$, an initialization of each episode in a uniformly sampled state $i \in \mathbb{Z}_p^*$, and a 0 (-1) reward for doing the correct (wrong) action in any given state.

C.8.1 Upper bound on the value function

The value function of any policy π which has probability $\pi(i)$ (we abbreviate $\pi(f^*(i)|i)$ to $\pi(i)$) of doing the correct action in state $i \in \mathbb{Z}_p^*$ is given by:

$$V_\pi(i) = \pi(i)\gamma\left((1-\delta)V_\pi(i+1) + \delta\frac{1}{p-1}\sum_{j=1}^{p-1}V_\pi(j)\right) - (1-\pi(i)) \tag{C.8}$$

Since this environment only has negative rewards, we have that $V_\pi(i) \leq 0$ for any state i and policy π, which allows us to write the following inequality:

$$V_\pi(i) \leq \pi(i)\gamma\left(\delta\frac{1}{p-1}\sum_{j=1}^{p-1}V_\pi(j)\right) - (1-\pi(i))$$

We use this inequality to bound the following term:

$$\begin{aligned}\frac{1}{p-1}\sum_{i=1}^{p-1}V_\pi(i) &\leq \frac{1}{p-1}\sum_{i=1}^{p-1}\left(\pi(i)\frac{\gamma\delta}{p-1}\sum_{j=1}^{p-1}V_\pi(j) - (1-\pi(i))\right)\\ &= \left(\frac{1}{p-1}\sum_{i=1}^{p-1}\pi(i)\right)\left(\frac{\gamma\delta}{p-1}\sum_{j=1}^{p-1}V_\pi(j)+1\right) - 1\end{aligned}$$

We note that the first factor is exactly the accuracy of the policy π on the classification task of Liu *et al.*:

$$\mathrm{Acc}_\pi(f^*) = \frac{1}{p-1}\sum_{i=1}^{p-1}\pi(i).$$

We hence have:

$$\frac{1}{p-1}\sum_{i=1}^{p-1}V_\pi(i) \leq \mathrm{Acc}_\pi(f^*)\left(\gamma\delta\frac{1}{p-1}\sum_{j=1}^{p-1}V_\pi(j)+1\right) - 1$$

which is equivalent to:

$$\frac{1}{p-1}\sum_{i=1}^{p-1}V_\pi(i) \leq \frac{\mathrm{Acc}_\pi(f^*)-1}{1-\mathrm{Acc}_\pi(f^*)\gamma\delta}$$

when $\mathrm{Acc}_\pi(f^*)\gamma\delta < 1$.
We now note that this average value function is exactly the value function evaluated on the initial state s_0 of the agent, since this state is uniformly sampled from $\mathbb{Z}_p^*$ for every episode. Hence,

$$V_\pi(s_0) \leq \frac{\mathrm{Acc}_\pi(f^*) - 1}{1 - \mathrm{Acc}_\pi(f^*)\gamma\delta} \tag{C.9}$$

C.8.2 Lower bound on the value function

Again, by noting in Eq. (C.8) that we have $V_\pi(i) \leq 0$ and $\pi(i) \leq 1$ for any policy π and state $i \in \mathbb{Z}_p^*$, we have:

$$V_\pi(i) \geq \gamma \left((1-\delta)V_\pi(i+1) + \frac{\delta}{p-1}\sum_{j=1}^{p-1} V_\pi(j) \right) - (1 - \pi(i))$$

We use this inequality to bound the value function at the initial state s_0:

$$\begin{aligned}
V_\pi(s_0) &= \frac{1}{p-1}\sum_{i=1}^{p-1} V_\pi(i) \\
&\geq \gamma \left(\frac{1-\delta}{p-1}\sum_{i=1}^{p-1} V_\pi(i+1) + \frac{\delta}{p-1}\sum_{j=1}^{p-1} V_\pi(j) \right) + \frac{1}{p-1}\sum_{i=1}^{p-1}\pi(i) - 1 \\
&= \gamma\left((1-\delta)V_\pi(s_0) + \delta V_\pi(s_0)\right) + \mathrm{Acc}_\pi(f^*) - 1 \\
&= \gamma V_\pi(s_0) + \mathrm{Acc}_\pi(f^*) - 1
\end{aligned}$$

by using the circular boundary conditions of the cliffwalk in the third line.
This inequality is equivalent to:

$$V_\pi(s_0) \geq \frac{\mathrm{Acc}_\pi(f^*) - 1}{1 - \gamma} \tag{C.10}$$

when $\gamma < 1$.

C.8.3 Bounds on classical- vs. and quantum- learnability

We use the bounds derived in the two previous sections to prove classical hardness and quantum learnability of this task environment, as stated in Lemma 46.

For this, we start by noting the following expression for the value function of a random policy (one that does random actions in all states):

$$\begin{aligned}
V_{\mathrm{rand}}(s_0) &= \frac{\gamma}{2}\left(\frac{1-\delta}{p-1}\sum_{i=1}^{p-1} V_{\mathrm{rand}}(i+1) + \frac{\delta}{p-1}\sum_{j=1}^{p-1} V_{\mathrm{rand}}(j) \right) - \frac{1}{2} \\
&= \frac{\gamma}{2} V_{\mathrm{rand}}(s_0) - \frac{1}{2} = -\frac{1}{2-\gamma}
\end{aligned}$$

again due to the circular boundary conditions of the cliffwalk and the resulting absence of termination conditions outside of "death".
As for the value function of the optimal policy, this is trivially $V_{\text{opt}} = 0$.

Proof of classical hardness

For any policy π, we define the function $g(x, \delta, \gamma) = V(x, \delta, \gamma) - V_{\text{rand}}(\gamma)$, where we adopt the short-hand notation $x = \text{Acc}_\pi(f^*)$ and call V the upper bound on the value function $V_\pi(s_0)$ of π. The expression of $g(x, \delta, \gamma)$ (for $(x, \delta, \gamma) \neq (1, 1, 1)$) is given by:

$$g(x, \delta, \gamma) = \frac{x - 1}{1 - \delta\gamma x} + \frac{1}{2 - \gamma} \tag{C.11}$$

To prove classical hardness, it is sufficient to show that $x \leq 0.51$ implies $g(x, \delta, \gamma) \leq 0.1$ for $\delta \in [\delta_0, 1]$, $\gamma \in [0, \gamma_1]$ and a $\{\delta_0, \gamma_1\}$ pair of our choosing. To see this, notice that the contraposition gives $x = \text{Acc}_\pi(f^*) > 0.51$ which is sufficient to construct an efficient algorithm that solves DLP. To achieve this result, we show the three following inequalities, $\forall\ x \leq 0.51$ and $\forall\ (\delta, \gamma) \in [\delta_0, 1] \times [0, \gamma_1]$:

$$g(x, \delta, \gamma) \overset{(i)}{\leq} g(0.51, \delta, \gamma) \overset{(ii)}{\leq} g(0.51, \delta_0, \gamma) \overset{(iii)}{\leq} g(0.51, \delta_0, \gamma_1)$$

where δ_0 and γ_1 are chosen such that $g(0.51, \delta_0, \gamma_1) \leq 0.1$.

Proof of (i). We look at the derivative of g w.r.t. x:

$$\frac{\partial g(x, \delta, \gamma)}{\partial x} = \frac{1 - \delta\gamma}{(1 - \delta\gamma x)^2} \geq 0 \quad \forall (x, \delta, \gamma) \in [0, 1]^3 \backslash (1, 1, 1)$$

and hence g is an increasing function of x, which gives our inequality. □

Proof of (ii). We look at the derivative of g w.r.t. δ:

$$\frac{\partial g(x, \delta, \gamma)}{\partial \delta} = \frac{\gamma(x - 1)x}{(1 - \delta\gamma x)^2} \leq 0 \quad \forall (x, \delta, \gamma) \in [0, 1]^3 \backslash (1, 1, 1)$$

and hence g is a decreasing function of δ, which gives our inequality. □

Proof of (iii). We look at the derivative of g w.r.t. γ:

$$\frac{\partial g(x, \delta, \gamma)}{\partial \gamma} = \frac{\delta(x - 1)x}{(1 - \delta\gamma x)^2} + \frac{1}{(2 - \gamma)^2} \quad \forall (x, \delta, \gamma) \in [0, 1]^3 \backslash (1, 1, 1)$$

We have:

$$\frac{\partial g(x, \delta, \gamma)}{\partial \gamma} \geq 0 \Leftrightarrow \left((\delta x)^2 + \delta(x^2 - x)\right)\gamma^2 - 2\delta(2x^2 - x)\gamma + 4\delta(x^2 - x) + 1 \geq 0$$

By setting $x = 0.51$ and $\delta = 0.86$, we find

$$\frac{\partial g(0.51, 0.86, \gamma)}{\partial \gamma} \geq 0 \quad \forall \gamma \in [0, 1]$$

since the roots of the second-degree polynomial above are approximately $\{-2.91, 2.14\}$ and we have $(\delta x)^2 + \delta(x-1)x \approx -0.0225 < 0$.
Hence $g(0.51, \delta_0, \gamma)$ is an increasing function of γ, which gives our inequality. □

Given that $g(0.51, 0.86, 0.9) \approx 0.0995 < 0.1$, we then get our desired result for $\delta_0 = 0.86$ and $\gamma_1 = 0.9$. Noting that $V_\pi(s_0) - V_{\text{rand}}(\gamma) \leq g(x, \delta, \gamma) \leq 0.1$ from Eq. (C.9), we hence have classical hardness $\forall\ (\delta, \gamma) \in [\delta_0, 1] \times [0, \gamma_1]$.

Proof of quantum learnability

Proving quantum learnability is more trivial, since, for $\text{Acc}_\pi(f^*) \geq 0.99$ and $\gamma \leq 0.9$, we directly have, using Eq. (C.10):

$$V_\pi(s_0) \geq -0.1 = V_{\text{opt}} - 0.1$$

To conclude this proof, we still need to show that we can obtain in this environment a policy π such that $\text{Acc}_\pi(f^*) \geq 0.99$ with high probability. For that, we use agents that first collect $\text{poly}(n)$ *distinct* samples (states s and their inferred labels $f^*(s)$) from the environment (distinct in order to avoid biasing the distribution of the dataset with the cliffwalk temporal structure). This can be done efficiently in $\text{poly}(n)$ interactions with the environment, since each episode is initialized in a random state $s_0 \in \mathbb{Z}_p^*$. We then use the learning algorithm of Liu *et al.* to train a classifier π with the desired accuracy, with high probability.

C.9 Proof of Lemma 47

C.9.1 Proof of classical hardness

Suppose that a polynomial-time classical agent achieves a value $V_c(s_0) \geq \frac{1}{2} + \frac{1}{\text{poly}(n)}$ with probability $(1 - \delta)$ over the choice of environment and the randomness of its learning algorithm. We call "success" the event $V_c(s_0) \geq \frac{1}{2} + \frac{1}{\text{poly}(n)}$ and S_δ the subset of the instances $S = \{T, x_k\}$ for which, theoretically, a run of the agent would "succeed" (this is hence a set that depends on the randomness of the agent).

Note that, on every instance in S_δ, $\pi(f^*(x_k)|T, x_k) = V_c(x_0) \geq V_c(s_0) \geq \frac{1}{2} + \frac{1}{\text{poly}(n)}$. Since this probability is bounded away from $1/2$ by an inverse polynomial, this means that we can "boost" it to a larger probability $(1 - \varepsilon)$. More specifically, out of the policy π obtained after interacting for k steps with the environment, we define a classifier f_c acting on x_k such that we sample $\mathcal{O}\left(\log(\varepsilon^{-1})\right)$-many times from $\pi(a|T, x_k)$ and label x_k by majority vote. For the instances in S_δ, the probability of correctly labeling x_k is $\Pr\left[f_c(x_k) = f^*(x_k)\right] \geq 1 - \varepsilon$.

Define $P(T) = \Pr[\mathrm{T} = T]$ and $P(x_k) = \Pr[\mathrm{x_k} = x_k]$ the probabilities of sampling certain training states T and a testing state x_k, when choosing an instance of the

environment. We now look at the following quantity:

$$\begin{aligned}
\mathbb{E}_{P(T)}\left[\mathrm{Acc}_{f_c}(T)\right] &= \sum_T P(T) \sum_{x_k} P(x_k) \Pr\left[f_c(x_k) = f^*(x_k) | T, x_k\right] \\
&= \sum_{T, x_k} P(T, x_k) \Pr\left[f_c(x_k) = f^*(x_k) | T, x_k\right] \\
&\geq \sum_{T, x_k} P(T, x_k) \Pr\big[\text{success} | T, x_k\big] \\
&\qquad \times \Pr\big[f_c(x_k) = f^*(x_k) | T, x_k, \text{success}\big] \\
&\geq (1-\delta)(1-\varepsilon)
\end{aligned}$$

since $\Pr\left[f_c(x_k) = f^*(x_k) | T, x_k\right] \geq 1 - \varepsilon$ for instances in S_δ and by definition we have

$$\sum_{T, x_k} P(T, x_k) \Pr\big[\text{success} | T, x_k\big] \geq 1 - \delta.$$

In the following, we set $1 - \varepsilon = 0.999$ and $1 - \delta \geq 0.845$ (the reason for this becomes apparent below), such that:

$$\mathbb{E}_{P(T)}\left[\mathrm{Acc}_{f_c}(T)\right] \geq 0.844155 > \frac{5}{6} + \frac{1}{96} \tag{C.12}$$

Now, consider the following learning algorithm: given a training set T, construct a Deterministic-DLP environment that uses this T and a randomly chosen x_k, and define the classifier f_c that boosts the $\pi(a|T, x_k)$ obtained by running our classical agent on this environment (as explained above). We want to show that f_c has accuracy $\mathrm{Acc}_{f_c}(T) \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ with probability at least $2/3$ over the choice of T and the randomness of its construction, which is sufficient to solve DLP classically. For that, we show a stronger statement. Call $\mathcal{T}_{\mathrm{succ}}$ the subset of all instances of training states $\mathcal{T} = \{T\}$ for which $\mathrm{Acc}_{f_c}(T) \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$. We prove by contradiction that $|\mathcal{T}_{\mathrm{succ}}| \geq \frac{2|\mathcal{T}|}{3}$:
Assume $|\mathcal{T}_{\mathrm{succ}}| < \frac{2|\mathcal{T}|}{3}$, then

$$\begin{aligned}
\mathbb{E}_{P(T)}\left[\mathrm{Acc}_{f_c}(T)\right] &= \sum_T P(T) \mathrm{Acc}_{f_c}(T) \\
&= \frac{1}{|\mathcal{T}|} \left(\sum_{T \in \mathcal{T}_{\mathrm{succ}}} \mathrm{Acc}_{f_c}(T) + \sum_{T \notin \mathcal{T}_{\mathrm{succ}}} \mathrm{Acc}_{f_c}(T) \right) \\
&< \frac{|\mathcal{T}_{\mathrm{succ}}|}{|\mathcal{T}|} \times 1 + \frac{|\mathcal{T}| - |\mathcal{T}_{\mathrm{succ}}|}{|\mathcal{T}|} \left(\frac{1}{2} + \frac{1}{\mathrm{poly}(n)} \right) \\
&< \frac{5}{6} + \frac{1}{3\mathrm{poly}(n)} < 0.844155
\end{aligned}$$

for large enough n, in contradiction with Eq. (C.12).

Hence, with probability at least $2/3$ over the choice of training states and the ran-

domness of the learning algorithm, our constructed classifier has accuracy $\mathrm{Acc}_{f_c}(T) \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$. By using Theorem 8, Remark 1 of Liu *et al.*, this is sufficient to construct an efficient classical algorithm that solves DLP.

C.9.2 Proof of quantum learnability

Using the learning algorithm of Liu *et al.*, we can construct a quantum classifier that achieves accuracy $\mathrm{Acc}_q(T) \geq 0.99$ with probability at least $2/3$ over the randomness of the learning algorithm and the choice of training states T, of length $|T| = \mathrm{poly}(n)$. Now define instead training states T of length $|T| = M\mathrm{poly}(n)$, for $M = \mathcal{O}\left(\log\left(\delta'^{-1}\right)\right)$ (hence $|T|$ is still polynomial in n), and use each of the M segments of T to train M independent quantum classifiers. Define f_q as a classifier that labels x_k using a majority vote on the labels assigned by each of these classifiers. This constructed classifier has accuracy $\mathrm{Acc}_{f_q}(T) \geq 0.99$ with now probability $1 - \delta'$ over the choice of training states T and the randomness of the learning algorithm.

We then note that, by calling "success" the event $\mathrm{Acc}_{f_q}(T) \geq 0.99$, we have:

$$\begin{aligned}
&\sum_{T,x_k} P(T,x_k)\mathrm{Pr}\big[V_q(x_0)=1|T,x_k\big] \\
&\qquad \geq \sum_T P(T)\sum_{x_k} P(x_k)\mathrm{Pr}\big[\mathrm{success}|T\big] \\
&\qquad\qquad \times \mathrm{Pr}\big[V_q(x_0)=1|T,x_k,\mathrm{success}\big] \\
&\qquad = \sum_T P(T)\mathrm{Pr}\big[\mathrm{success}|T\big]\sum_{x_k} P(x_k) \\
&\qquad\qquad \times \mathrm{Pr}\big[f_q(x_k)=f^*(x_k)|T,x_k,\mathrm{success}\big] \\
&\qquad = \sum_T P(T)\mathrm{Pr}\big[\mathrm{success}|T\big]\mathrm{Acc}_{f_q}(T) \\
&\qquad \geq (1-\delta')\times 0.99
\end{aligned}$$

which means that our constructed agent achieves a value $V_q(x_0) = 1$ (which also implies $V_q(s_0) = 1$) with probability at least $(1-\delta')\times 0.99$ over the choice of environment and the randomness of the learning algorithm. By setting $(1-\delta') = 0.98/0.99$, we get our statement.

C.10 Construction of PQC agent for the DLP environments

In the two following appendices, we construct a PQC classifier that can achieve close-to-optimal accuracy in the classification task of Liu *et al.* [128] (see Appendix C.6), and can hence also be used as a learning model in the DLP environments defined in Sec. 5.3.1.

C.10.1 Implicit vs. explicit quantum SVMs

To understand the distinction between the quantum learners of Liu *et al.* and the PQC policies we are constructing here, we remind the reader of the two models for quantum SVMs defined in Ref. [169]: the explicit and the implicit model. Both models share a feature-encoding unitary $U(x)$ that encodes data points x into feature state $|\phi(x)\rangle = U(x)\,|0^{\otimes n}\rangle$.
In the implicit model, one first evaluates the kernel values

$$K(x_i, x_j) = |\langle \phi(x_i)\rangle\, \phi(x_j)|^2 \tag{C.13}$$

for the feature states associated to every pair of data points $\{x_i, x_j\}$ in the dataset, then uses the resulting kernel matrix in a classical SVM algorithm. This algorithm returns a hyperplane classifier in feature space, defined by its normal vector $\langle \boldsymbol{w}| = \sum_i \alpha_i \langle \phi(x_i)|$ and bias b, such that the sign of $|\langle \boldsymbol{w} \mid \phi(x)\rangle|^2 + b$ gives the label of x.
In the explicit model, the classifier is instead obtained by training a parametrized $|\boldsymbol{w}_{\boldsymbol{\theta}}\rangle$. Effectively, this classifier is implemented by applying a variational unitary $V(\boldsymbol{\theta})$ on the feature states $|\phi(x)\rangle$ and measuring the resulting quantum states using a fixed observable, with expectation value $|\langle \boldsymbol{w}_{\boldsymbol{\theta}} \mid \phi(x)\rangle|^2$.

In the following sections, we describe how the implicit quantum SVMs of Liu *et al.* can be transformed into explicit models while guaranteeing that they can still represent all possible optimal policies in the DLP environments. And in Appendix C.11, we show that, even under similar noise considerations as Liu *et al.*, these optimal policies can also be found using poly(n) random data samples.

C.10.2 Description of the PQC classifier

As we just described, our classifier belongs to a family of so-called explicit quantum SVMs. It is hence described by a PQC with two parts: a feature-encoding unitary $U(x)$, which creates features $|\phi(x)\rangle = U(x)\,|0^{\otimes n}\rangle$ when applied to an all-0 state, followed by a variational circuit $V(\boldsymbol{\theta})$ parametrized by a vector $\boldsymbol{\theta}$. The resulting quantum state is then used to measure the expectation value $\langle O\rangle_{x,\boldsymbol{\theta}}$ of an observable O, to be defined. We rely on the same feature-encoding unitary $U(x)$ as the one used by Liu *et al.*, i.e., the unitary that creates feature states of the form

$$|\phi(x)\rangle = \frac{1}{\sqrt{2^k}} \sum_{i=0}^{2^k-1} |x \cdot g^i\rangle \tag{C.14}$$

for $k = n - t\log(n)$, where t is a constant defined later, under noise considerations. This feature state can be seen as the uniform superposition of the image (under exponentiation $s' \mapsto g^{s'}$) of an interval of integers $[\log_g(x), \log_g(x) + 2^k - 1]$ in log-space. Note that $U(x)$ can be implemented in $\widetilde{\mathcal{O}}(n^3)$ operations [128].

By noting that every labeling functions $f_s \in \mathcal{C}$ to be learned (see Eq. (C.6)) is delimiting two equally-sized intervals of $\log(\mathbb{Z}_p^*)$, we can restrict the decision boundaries to be learned by our classifier to be half-space dividing hyperplanes in log-space. In feature space, this is equivalent to learning separating hyperplanes that are normal

to quantum states of the form:

$$|\phi_{s'}\rangle = \frac{1}{\sqrt{(p-1)/2}} \sum_{i=0}^{(p-3)/2} |g^{s'+i}\rangle \,. \tag{C.15}$$

Noticeably, for input points x such that $\log_g(x)$ is away from some delimiting regions around s and $s + \frac{p-3}{2}$, we can notice that the inner product $|\langle\phi(x)\rangle\, \phi_s|^2$ is either $\Delta = \frac{2^{k+1}}{p-1}$ or 0, whenever x is labeled $+1$ or -1 by f_s, respectively. This hence leads to a natural classifier to be built, assuming overlaps of the form $|\langle\phi(x)\rangle\, \phi_{s'}|^2$ can be measured:

$$h_{s'}(x) = \begin{cases} 1, & \text{if } |\langle\phi(x)\rangle\, \phi_{s'}|^2/\Delta \geq 1/2, \\ -1, & \text{otherwise} \end{cases} \tag{C.16}$$

which has an (ideal) accuracy $1 - \Delta$ whenever $s' = s$.

To complete the construction of our PQC classifier, we should hence design the composition of its variational part $V(\boldsymbol{\theta})$ and measurement O such that they result in expectation values of the form $\langle O\rangle_{x,\boldsymbol{\theta}} = |\langle\phi(x)\rangle\, \phi_{s'}|^2$. To do this, we note that, for $|\phi_{s'}\rangle = \hat{V}(s')\,|0\rangle$, the following equality holds:

$$\begin{aligned} |\langle\phi(x)\rangle\, \phi_{s'}|^2 &= \left|\langle 0^{\otimes n}|\, \hat{V}(s')^\dagger U(x_i)\, |0^{\otimes n}\rangle\right|^2 \\ &= \mathrm{Tr}\left[|0^{\otimes n}\rangle\, \langle 0^{\otimes n}|\, \rho(x, s')\right] \end{aligned}$$

where $\rho(x, s') = |\psi(x, s')\rangle\, \langle\psi(x, s')|$ is the density matrix of the quantum state $|\psi(x, s')\rangle = \hat{V}(s')^\dagger U(x_i)\, |0^{\otimes n}\rangle$. Hence, an obvious choice of variational circuit is $V(\boldsymbol{\theta}) = \hat{V}(s')$, combined with a measurement operator $O = |0^{\otimes n}\rangle\, \langle 0^{\otimes n}|$. Due to the similar nature of $|\phi'_s\rangle$ and $|\phi(x)\rangle$, it is possible to use an implementation for $\hat{V}(s')$ that is similar to that of $U(x_i)$ (take $x_i = g^{s'}$ and $k \approx n/2$).[2] We also note that, for points x such that $\log_g(x)$ is $(p-1)\Delta/2$ away from the boundary regions of $h_{s'}$, the non-zero inner products $|\langle\phi(x)\rangle\, \phi_{s'}|^2$ are equal to $\Delta = \mathcal{O}(n^{-t})$. These can hence be estimated efficiently to additive error, which allows to efficiently implement our classifier $h_{s'}$ (Eq. (C.16)).

C.10.3 Noisy classifier

In practice, there will be noise associated with the estimation of the inner products $|\langle\phi(x)\rangle\, \phi_{s'}|^2$, namely due to the additive errors associated to sampling. Similarly to Liu *et al.*, we model noise by introducing a random variable $e_{is'}$ for each data point x_i and variational parameter $g^{s'}$, such that the estimated inner product is

[2] Note that we write $\hat{V}(s')$ and $U_{s'}$ to be parametrized by s' but the true variational parameter here is $g^{s'}$, since we work in input space and not in log-space.

$|\langle\phi(x_i)\rangle\,\phi_{s'}|^2 + e_{is'}$. This random variable satisfies the following equations:

$$\begin{cases} e_{is'} \in [-\Delta, \Delta] \\ \mathbb{E}[e_{is'}] = 0 \\ \mathrm{Var}[e_{is'}] \leq 1/R \end{cases}$$

where R is the number of circuit evaluations used to estimate the inner product. We hence end up with a noisy classifier:

$$\widetilde{h}_{s'}(x_i) = \begin{cases} 1, & \text{if } \left(|\langle\phi(x_i)\rangle\,\phi_{s'}|^2 + e_{is'}\right)/\Delta \geq 1/2, \\ -1, & \text{otherwise} \end{cases}$$

The noise has the effect that some points which would be correctly classified by the noiseless classifier have now a non zero probability of being misclassified. To limit the overall decrease in classification accuracy, we focus on limiting the probability of misclassifying points x_i such that $\log_g(x_i)$ is $(p-1)\Delta/2$ away from the boundary points s' and $s' + \frac{p-3}{2}$ of $g_{s'}$. We call $I_{s'}$ the subset of $\mathbb{Z}_p^*$ comprised of these points. For points in $I_{s'}$, the probability of misclassification is that of having $|e_{is'}| \geq \Delta/2$. We can use Chebyshev's inequality to bound this probability:

$$\Pr\left(|e_{is'}| \geq \frac{\Delta}{2}\right) \leq \frac{4}{\Delta^2 R} \tag{C.17}$$

since $\mathbb{E}[e_{is'}] = 0$ and $\mathrm{Var}[e_{is'}] \leq 1/R$.

C.11 Proof of trainability of PQC agent in the SL-DLP

In this Appendix, we describe an optimization algorithm to train the variational parameter $g^{s'}$ of the PQC classifier we defined in Appendix C.10. This task is non-trivial for three reasons: 1) even by restricting the separating hyperplanes accessible by our classifier, there are still $p-1$ candidates, which makes an exhaustive search for the optimal one intractable; 2) noise in the evaluation of the classifier can potentially heavily perturb its loss landscape, which can shift its global minimum and 3) decrease the testing accuracy of the noisy classifier. Nonetheless, we show that all these considerations can be taken into account for a simple optimization algorithm, such that it returns a classifier with close-to-optimal accuracy with high probability of success. More precisely, we show the following Theorem:

Theorem 48. *For a training set of size n^c such that*

$$c \geq \max\left\{\log_n(8/\delta), \log_n\left(\frac{\log(\delta/2)}{\log(1-2n^{-t})}\right)\right\}$$

for $t \geq \max\{3\log_n(8/\delta), \log_n(16/\varepsilon)\}$ in the definition of Δ, and a number of cir-

cuit evaluations per inner product $R \geq \max\left\{\frac{4n^{2(t+c)}}{\delta}, \frac{128}{\varepsilon^3}\right\}$*, then our optimization algorithm returns a noisy classifier* $\widetilde{h}_{s'}$ *with testing accuracy* $\mathrm{Acc}_{\widetilde{h}_{s'}}(f_s)$ *on the DLP classification task of* Liu *et al. such that*

$$\Pr\left(\mathrm{Acc}_{\widetilde{h}_{s'}}(f_s) \geq 1-\varepsilon\right) \geq 1-\delta.$$

The proof of this Theorem is detailed below.

Given a training set $X \subset \mathcal{X}$ polynomially large in n, i.e., $|X| = n^c$, define the training loss:

$$\mathcal{L}(s') = \frac{1}{2|X|}\sum_{x\in X}|h_{s'}(x) - f_s(x)|$$

and its noisy analog:

$$\widetilde{\mathcal{L}}(s') = \frac{1}{2|X|}\sum_{x\in X}\left|\widetilde{h}_{s'}(x) - f_s(x)\right|$$

Our optimization algorithm goes as follows: using the noisy classifier $\widetilde{h}_{s'}$, evaluate the loss function $\widetilde{\mathcal{L}}\left(\log_g(x)\right)$ for each variational parameter $g^{s'} = x \in X$, then set

$$g^{s'} = \mathrm{argmin}_{x\in X}\widetilde{\mathcal{L}}(\log_g(x)).$$

This algorithm is efficient in the size of the training set, since it only requires $|X|^2$ evaluations of $\widetilde{h}_{s'}$.

To prove Theorem 48, we show first that we can enforce $\mathrm{argmin}_{x\in X}\widetilde{\mathcal{L}}(\log_g(x)) = \mathrm{argmin}_{x\in X}\mathcal{L}(\log_g(x))$ with high probability (Lemma 49), and second, that this algorithm also leads to s' close to the optimal s in log-space with high probability (Lemma 50).

Lemma 49. *For a training set of size* n^c *such that* $c \geq \log_n(8/\delta)$*, a* $t \geq 3c$ *in the definition of* Δ*, and a number of circuit evaluations per inner product* $R \geq \frac{4n^{2(t+c)}}{\delta}$*, we have*

$$\Pr\left(\operatorname*{argmin}_{x\in X}\widetilde{\mathcal{L}}(\log_g(x)) = \operatorname*{argmin}_{x\in X}\mathcal{L}(\log_g(x))\right) \geq 1 - \frac{\delta}{2}$$

Proof. In order for the minima of the two losses to be obtained for the same $x \in X$, it is sufficient to ensure that the classifiers $h_{\log_g(x_i)}$ and $\widetilde{h}_{\log_g(x_i)}$ agree on all points x_j, for all $(x_i, x_j) \in X$. This can be enforced by having:

$$\left(\bigcap_{\substack{i,j\\ i\neq j}} x_i \in I_{\log_g(x_j)}\right) \cap \left(\bigcap_{i,s'}|e_{i,s'}| \leq \frac{\Delta}{2}\right)$$

that is, having for all classifiers $h_{\log_g(x_j)}$ that all points $x_i \in X$, $x_i \neq x_j$, are away from its boundary regions in log-space, and that the labels assigned to these points are all the same under noise.

We bound the probability of the negation of this event:

$$\Pr\left(\bigcup_{\substack{i,j\\ i\neq j}} x_i \notin I_{\log_g(x_j)} \cup \bigcup_{i,s'} |e_{i,s'}| \geq \frac{\Delta}{2}\right) \leq \Pr\left(\bigcup_{\substack{i,j\\ i\neq j}} x_i \notin I_{\log_g(x_j)}\right) + \Pr\left(\bigcup_{i,s'} |e_{i,s'}| \geq \frac{\Delta}{2}\right)$$

using the union bound.
We start by bounding the first probability, again using the union bound:

$$\begin{aligned}\Pr\left(\bigcup_{\substack{i,j\\ i\neq j}} x_i \notin I_{\log_g(x_j)}\right) &\leq \sum_{\substack{i,j\\ i\neq j}} \Pr\left(x_i \notin I_{\log_g(x_j)}\right)\\ &= \sum_{\substack{i,j\\ i\neq j}} \frac{\Delta}{2} \leq \frac{n^{2c}\Delta}{2}\end{aligned}$$

By setting $t \geq 3c$, we have $\Delta \leq 4n^{-t} \leq 4n^{-3c}$, which allows us to bound this first probability by $\delta/4$ when $c \geq \log_n(8/\delta)$.
As for the second probability above, we have

$$\begin{aligned}\Pr\left(\bigcup_{i,s'} |e_{i,s'}| \geq \frac{\Delta}{2}\right) &\leq \sum_{i,s'} \Pr\left(|e_{i,s'}| \geq \frac{\Delta}{2}\right)\\ &\leq \frac{4n^{2c}}{\Delta^2 R}\end{aligned}$$

using the union bound and Eq. (C.17). By setting $R \geq \frac{4n^{2(t+c)}}{\delta} \geq \frac{16n^{2c}}{\Delta^2\delta}$ (since $\Delta \geq 2n^{-t}$), we can bound this second probability by $\delta/4$ as well, which gives:

$$\begin{aligned}\Pr\left(\operatorname*{argmin}_{x\in X} \widetilde{\mathcal{L}}(\log_g(x)) = \operatorname*{argmin}_{x\in X} \mathcal{L}(\log_g(x))\right) &\geq 1 - \Pr\Big(\bigcup_{\substack{i,j\\ i\neq j}} x_i \notin I_{\log_g(x_j)}\\ &\qquad \cup \bigcup_{i,s'} |e_{i,s'}| \geq \frac{\Delta}{2}\Big)\\ &\geq 1 - \delta/2\end{aligned}$$

□

Lemma 50. *For a training set of size n^c such that*

$$c \geq \log_n\left(\frac{\log(\delta/2)}{\log(1-2\varepsilon)}\right),$$

then $s' = \log_g\left(\operatorname{argmin}_{x\in X}\mathcal{L}(\log_g(x))\right)$ *is within* ε *distance of the optimal* s *with probability:*

$$\Pr\left(\frac{|s'-s|}{p-1} \leq \varepsilon\right) \geq 1 - \frac{\delta}{2}$$

Proof. We achieve this result by proving:

$$\Pr\left(\frac{|s'-s|}{p-1} \geq \varepsilon\right) \leq \frac{\delta}{2}$$

This probability is precisely the probability that no $\log_g(x) \in \log_g(X)$ is within ε distance of s, i.e.,

$$\Pr\left(\bigcap_{x\in X} \log(x) \notin [s-\varepsilon(p-1), s+\varepsilon(p-1)]\right)$$

As the elements of the training set are all i.i.d., we have that this probability is equal to

$$\Pr\left(\log(x) \notin [s-\varepsilon(p-1), s+\varepsilon(p-1)]\right)^{|X|}$$

Since all the datapoints are uniformly sampled from $\mathbb{Z}_p^*$, the probability that a datapoint is in any region of size $2\varepsilon(p-1)$ is just 2ε. With the additional assumption that $|X| = n^c \geq \log_{1-2\varepsilon}(\delta/2)$ (and assuming $\varepsilon < 1/2$), we get:

$$\Pr\left(\frac{|s'-s|}{p-1} \geq \varepsilon\right) \leq (1-2\varepsilon)^{\log_{1-2\varepsilon}(\delta/2)} = \frac{\delta}{2} \qquad \square$$

Lemma 49 and Lemma 50 can be used to prove:

Corollary 51. *For a training set of size* n^c *such that*

$$c \geq \max\left\{\log_n(8/\delta), \log_n\left(\frac{\log(\delta/2)}{\log(1-2\varepsilon)}\right)\right\},$$

a $t \geq 3c$ *in the definition of* Δ*, and a number of circuit evaluations per inner product* $R \geq \frac{4n^{2(t+c)}}{\delta}$*, then our optimization algorithm returns a variational parameter* $g^{s'}$ *such that*

$$\Pr\left(\frac{|s'-s|}{p-1} \leq \varepsilon\right) \geq 1 - \delta$$

From here, we notice that, when we apply Corollary 51 for $\varepsilon' \leq \frac{\Delta}{2}$, our optimization algorithm returns an s' such that, with probability $1-\delta$, the set $I_{s'}$ is equal to I_s and is of size $(p-1)(1-2\Delta)$. In the event where $|s'-s|/(p-1) \leq \varepsilon' \leq \frac{\Delta}{2}$, we can

hence bound the accuracy of the noisy classifier:

$$\begin{aligned}
\mathrm{Acc}_{\widetilde{h}_{s'}}(f_s) &= \frac{1}{p-1}\sum_{x\in\mathcal{X}} \Pr\left(\widetilde{h}_{s'}(x) = f_s(x)\right) \\
&\geq \frac{1}{p-1}\sum_{x\in I_s} \Pr\left(\widetilde{h}_{s'}(x) = f_s(x)\right) \\
&\geq (1-2\Delta)\min_{x_i\in I_s} \Pr\left(|e_{i,s'}| \leq \frac{\Delta}{2}\right) \\
&\geq (1-2\Delta)\left(1-\frac{4}{\Delta^2 R}\right) \\
&= 1-\left(2\Delta\left(1-\frac{4}{\Delta^2 R}\right)+\frac{4}{\Delta^2 R}\right)
\end{aligned}$$

with probability $1-\delta$.
We now set $t \geq \max\{3\log_n(8/\delta), \log_n(16/\varepsilon)\}$, $\varepsilon' = n^{-t}$ and $R \geq \max\left\{\frac{4n^{2(t+c)}}{\delta}, \frac{128}{\varepsilon^3}\right\}$, such that $2\varepsilon' = 2n^{-t} \leq \Delta \leq 4n^{-t} \leq \frac{\varepsilon}{4}$, $\left(1-\frac{4}{\Delta^2 R}\right) \leq 1$ and $\frac{4}{\Delta^2 R} \leq \frac{\varepsilon}{2}$.
Using these inequalities, we get

$$\mathrm{Acc}_{\widetilde{h}_{s'}}(f_s) \geq 1-\varepsilon$$

with probability $1-\delta$, which proves Theorem 48.

Appendix D

Exponential separations between classical and quantum learners

D.1 Details regarding definitions

D.1.1 Constraining hypothesis classes to those that are efficiently evaluatable

In this section, we discuss why it makes sense to restrict the hypothesis class to be efficiently evaluatable. Specifically, we show that if we allow the learner to use hypotheses that run for superpolynomial time, then every concept class that is learnable in superpolynomial time is also learnable in polynomial time. Thus, if we do not restrict the hypotheses to be efficiently evaluatable, then the restriction that the learning algorithm has to run in polynomial time is vacuous (i.e., it imposes no extra restrictions on what can be learned). For more details we refer to [116].

Consider a concept class $\mathcal{C}$ that is learnable by a superpolynomial time learning algorithm $\mathcal{A}$ using a hypothesis class $\mathcal{H}$. To show that this concept class is learnable using a polynomial time learning algorithm, consider the hypothesis class $\mathcal{H}$ whose hypotheses are enumerated by all possible polynomially-sized sets of examples. Each hypothesis in $\mathcal{H}'$ runs the learning algorithm $\mathcal{A}$ on its corresponding set of examples, and it evaluates the hypothesis from $\mathcal{H}$ that the learning algorithm outputs based on this set of examples. Finally, consider the polynomial-time learning algorithm $\mathcal{A}'$ that queries the example oracle a polynomial number of times and outputs the specification of the hypothesis in $\mathcal{H}'$ that corresponds to the obtained set of examples. By construction, this polynomial-time learning algorithm $\mathcal{A}'$ now learns $\mathcal{C}$.

D.1.2 Proof of Lemma 3

Lemma 3. $\mathsf{CQ} = \mathsf{QQ}$.

Proof. Since any efficient classical algorithm can be simulated using an efficient quantum algorithm it is obvious that $\mathsf{CQ} \subseteq \mathsf{QQ}$. For the other inclusion, let $L = (\mathcal{C}, \mathcal{D}) \in \mathsf{QQ}$. That is, the concept classes $\mathcal{C}$ are efficiently learnable under the distributions $\mathcal{D}$ by a quantum learning algorithm $\mathcal{A}^q$ using a quantum evaluatable hypothesis class $\mathcal{H}$. To show that $L \in \mathsf{CQ}$, consider the quantum evaluatable hypothesis class $\mathcal{H}'$ whose hypotheses are enumerated by all possible polynomially-sized sets of training examples. Each hypothesis in $\mathcal{H}'$ runs the quantum learning algorithm $\mathcal{A}^q$ on its corresponding set of examples, and evaluates the hypothesis from $\mathcal{H}$ that the quantum learning algorithm $\mathcal{A}^q$ outputs based on the set of examples. Finally, consider the classical polynomial-time learning algorithm $\mathcal{A}^c$ that queries the example oracle a polynomial number of times and outputs the specification of the hypothesis in $\mathcal{H}'$ that corresponds to the obtained set of examples. By construction, this classical polynomial-time algorithm $\mathcal{A}^c$ can learn the concept classes $\mathcal{C}$ under the distributions $\mathcal{D}$ using the quantum evaluatable hypothesis class $\mathcal{H}'$. This shows that $L \in \mathsf{CQ}$.

□

D.1.3 Proof of Lemma 4

Lemma 4. $\mathsf{HeurBPP/samp} \subseteq \mathsf{HeurP/poly}$.

Proof. The proof strategy is analogous to the arguments in Section 2 of the supplementary material of [109], where they show that $\mathsf{BPP/samp} \subseteq \mathsf{P/poly}$.

Let $(L, \{\mathcal{D}_n\}_{n\in\mathbb{N}}) \in \mathsf{HeurBPP/samp}$. In particular, there exist a polynomial-time classical algorithms $\mathcal{A}$ with the following property: for every n and $\epsilon > 0$ it holds that

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathsf{Pr}\big(\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \mathcal{T}) = L(x)\big) \geq \frac{2}{3}\right] \geq 1 - \epsilon, \tag{D.1}$$

where the inner probability is over the randomization of $\mathcal{A}$ and

$$\mathcal{T} = \{(x_i, L(x_i)) \mid x_i \sim \mathcal{D}_n\}_{i=1}^{\mathrm{poly}(n)}.$$

Let $\epsilon > 0$ and partition the set of n-bit strings as follows

$$\{0,1\}^n = I^n_{\mathrm{correct}}(\epsilon) \sqcup I^n_{\mathrm{error}}(\epsilon), \tag{D.2}$$

such that for every $x \in I^n_{\mathrm{correct}}(\epsilon)$ we have

$$\mathsf{Pr}\big(\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \mathcal{T}) = L(x)\big) \geq \frac{2}{3}, \tag{D.3}$$

where the probability is taken over the internal randomization of $\mathcal{A}$ and $\mathcal{T}$. Importantly, we remark that our partition is such that

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\big[x \in I^n_{\mathrm{correct}}(\epsilon)\big] \geq 1 - \epsilon. \tag{D.4}$$

By applying the arguments of Section 2 of the supplementary material of [109] to $\mathcal{A}$ with the bitstring $0^{\lfloor 1/\epsilon \rfloor}$ fixed as input we obtain a polynomial-time classical

algorithm $\mathcal{A}'$ with the following property: for every n there exists an advice string $\alpha_{n,\epsilon} \in \{0,1\}^{\mathrm{poly(n,1/\epsilon)}1}$ such that for every $x \in I^n_{\mathrm{correct}}(\epsilon)$:

$$\mathcal{A}'(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n,\epsilon}) = L(x). \tag{D.5}$$

Intuitively, the algorithm $\mathcal{A}'(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n,\epsilon})$ runs $\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \mathcal{T})$ a certain number of times and decides its output based on a majority-vote. Moreover, $\mathcal{A}'$ does so with a particular setting of random seeds and training data $\mathcal{T}$ that makes it correct decide $L(x)$, which is collected in $\alpha_{n,\epsilon}$. Finally, from Eq. (D.4) we find that we have

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathcal{A}'(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n,\epsilon}) = L(x)\right] \geq 1-\epsilon, \tag{D.6}$$

which shows that $(L, \{\mathcal{D}_n\}_{n\in\mathbb{N}}) \in \mathsf{HeurP/poly}$.

□

D.2 Proof of Theorem 24

Theorem 24. *If the 2^c-DCRA holds, then the learning problem*

$$L_{\mathrm{modexp}} = \left(\{\mathcal{C}^{\mathrm{modexp}}_n\}_{n\in\mathbb{N}}, \{\mathcal{D}^U_n\}_{n\in\mathbb{N}}\right)$$

exhibits a CC/QC *separation, where $\mathcal{D}^U_n$ denotes the uniform distribution over $\mathbb{Z}^*_N$.*

Proof. To see why L_{modexp} is not in CC, we first note that the modular exponentiation concept class contains the cube root function f_N^{-1} discussed in Section 6.1.2. Therefore, the proof presented in [116], which shows that the cube root concept class is not in CC under the DCRA, also implies that the modular exponentiation concept class is not in CC under the DCRA. To briefly recap, recall from Section 6.1.2 that we can efficiently generate examples $(y, f_N^{-1}(y))$, for $y \in \mathbb{Z}^*_N$ uniformly at random. If we put these examples into an efficient classical learning algorithm for the modular exponentiation concept class, the learning algorithm would with high probability identify a classically efficiently evaluatable hypothesis that agrees with f_N^{-1} on a polynomial fraction of inputs. This directly violates the DCRA, which states that evaluating f_N^{-1} is classically intractable, even on a polynomial fraction of inputs (i.e., it is outside of HeurBPP).

What remains to be shown is that L_{modexp} is in QC. Suppose we are given access to an example oracle $EX(c_d, \mathcal{D}^U_n)$ which when queried returns an example $(x, x^d \bmod N)$, where x is sampled uniformly at random from $\mathbb{Z}^*_N$. To show that L_{modexp} is in QC, we will describe a $\mathcal{O}(\mathrm{poly}(n, 1/\delta))$-time quantum learning algorithm that uses queries to $EX(c_d, \mathcal{D}^U_n)$ and identifies d with probability at least $1-\delta$. Before describing the quantum learning algorithm, we will first prove the following lemma, which is used to prove that our quantum learning algorithm is correct.

Lemma 52. *Write $(p-1)(q-1) = 2^c \cdot p_1^{k_1} \cdots p_\ell^{k_\ell}$, where the p_is are distinct odd primes. Then, for any $i = 1, \ldots, \ell$ we have that with probability at least $1/2$, an*

[1]Note that the advice string also depends on ϵ, since $0^{\lfloor 1/\epsilon \rfloor}$ was fixed as input to $\mathcal{A}$.

example $(x, x^d \mod N)$ *queried from* $EX(c_d, \mathcal{D}_n^U)$ *satisfies*

$$p_i^{k_i} \mid \mathrm{ord}_N(x), \tag{D.7}$$

where $\mathrm{ord}_N(x)$ *denotes the order of* x *in* $\mathbb{Z}_N^*$.

Proof of Lemma 52. Let C_k denote the cyclic group of order k. Since $(p-1)(q-1) = 2^c a$ and $\gcd(p-1, q-1) = 2^{c'}$ as described in Definition 19, the Chinese remainder theorem tells us that

$$\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^* \simeq C_{p-1} \times C_{q-1} \simeq C_{2^{c_1}} \times C_{2^{c_2}} \times C_{p_1^{k_1}} \times \cdots \times C_{p_\ell^{k_\ell}},$$

for some c_1, c_2 such that $c_1 + c_2 = c$. For any $x \in \mathbb{Z}_N^*$ we will use

$$x = (x_0^{(1)}, x_0^{(2)}, x_1, \ldots, x_\ell)$$

to denote its corresponding element in $C_{2^{c_1}} \times C_{2^{c_2}} \times C_{p_1^{k_1}} \times \cdots \times C_{p_\ell^{k_\ell}}$. Next, note that the order of x in $\mathbb{Z}_N^*$ is the least common multiple of the orders of all $x_0^{(1)}, x_0^{(2)}, x_1, \ldots, x_\ell$ in their respective groups. What this implies is that any element $x = (x_0^{(1)}, x_0^{(2)}, x_1, \ldots, x_\ell)$ satisfies

$$p_i^{k_i} \mid \mathrm{ord}_N(x),$$

if x_i is a generator of $C_{p_i^{k_i}}$. The number of generators of $C_{p_i^{k_i}}$ is equal to $\varphi(p_i^{k_i})$ (where φ denotes Euler's totient function), and the number of elements of $x = (x_0^{(1)}, x_0^{(2)}, x_1, \ldots, x_\ell)$ such that x_i is a generator of $C_{p_i^{k_i}}$ is therefore equal to

$$2^{c_1} \cdot 2^{c_2} \cdot p_1^{k_1} \cdots p_{i-1}^{k_{i-1}} \cdot \varphi(p_i^{k_i}) \cdot p_{i+1}^{k_{i+1}} \cdots p_\ell^{k_\ell}.$$

Thus, the probability that a uniformly random $x \in \mathbb{Z}_N^*$ satisfies Eq. (D.7) is at least

$$\begin{aligned} \frac{2^{c_1} \cdot 2^{c_2} \cdot p_1^{k_1} \cdots \varphi(p_i^{k_i}) \cdots p_\ell^{k_\ell}.}{\#\mathbb{Z}_N^*} &= \frac{2^{c_1} \cdot 2^{c_2} \cdot p_1^{k_1} \cdots p_{i-1}^{k_{i-1}} \cdot \varphi(p_i^{k_i}) \cdot p_{i+1}^{k_{i+1}} \cdots p_\ell^{k_\ell}.}{(p-1)(q-1)} \\ &= \frac{\varphi(p_i^{k_i})}{p_i^{k_i}} \geq \frac{1}{2}. \end{aligned}$$

□

We now describe the quantum learning algorithm that can identify d in time $\mathcal{O}(\mathrm{poly}(n, 1/\delta))$ using queries to $EX(c_d, \mathcal{D}_n^U)$. We write $(p-1)(q-1) = 2^c \cdot p_1^{k_1} \cdots p_\ell^{k_\ell}$, where the p_is are distinct primes. The idea is to query $EX(c_d, \mathcal{D}_n^U)$ sufficiently many times such that for every $i = 1, \ldots, \ell$ we have an example $(x_i, x_i^d \mod N)$ where

$$p_i^{k_i} \mid \mathrm{ord}_N(x_i). \tag{D.8}$$

Next, we use Shor's algorithm [176] to compute $r_i = \mathrm{ord}_N(x_i)$ and $a_i = \log_{x_i}(x_i^d)$,

where $\log_a(b)$ denotes the discrete logarithm of b in the group generated by a (i.e., the smallest integer ℓ such that $a^\ell = b$). Now by elementary group theory we obtain the congruence relation

$$d \equiv a_i \mod r_i,$$

which by Eq. (D.8) implies the congruence relation

$$d \equiv a_i \mod p_i^{k_i}.$$

In other words, the examples allowed us to recover $d \mod p_i^{k_i}$ for every $i = 1, \ldots, \ell$. By the Chinese remainder theorem, all that remains is to recover $d \mod 2^c$, which we can do by brute force search since c is constant. All in all, if we query $EX(c_d, \mathcal{D}_n^U)$ sufficiently many times such that for every $i = 1, \ldots, \ell$ we have an example $(x_i, x_i^d \mod N)$ satisfying Eq. (D.8), then we can recover d.

What remains to be shown is that with probability $1 - \delta$ a total number of $\mathcal{O}(\text{poly}(n, 1/\delta))$ queries to $EX(c_d, \mathcal{D}_n^U)$ suffices to find an example $(x_i, x_i^d \mod N)$ satisfying Eq. (D.8) for every $i = 1, \ldots, \ell$. To do so, we invoke Lemma 52 and conclude from it that for any individual $i = 1, \ldots, \ell$ after $\mathcal{O}(\log(1/\delta'))$ queries with probability at least $1 - \delta'$ we found an example $(x_i, x_i^d \mod N)$ satisfying Eq. (D.8). In particular, this implies that after a total of $\mathcal{O}(\log(n, 1/\delta))$ queries we found with probability at least $1 - \delta$ examples $(x_i, x_i^d \mod N)$ satisfying Eq. (D.8) for all $i = 1, \ldots, \ell$.

□

D.2.1 Discrete cube root assumption for moduli of Definition 19

Recall that in Definition 19 we have constraint our moduli $N = pq$ to satisfy the conditions

(*a*) $\gcd(3, (p-1)(q-1)) = 1$,

(*b*) $(p-1)(q-1) = 2^c \cdot a$, where $a \in \mathbb{N}$ is odd and c is a constant,

(*c*) $\gcd(p-1, q-1) = 2^{c'}$ for some c'.

Firstly, we remark that (*a*) is a standard condition required for the function cube root function f_N^{-1} to be well-defined, and it therefore does not influence the DCRA. On the other hand, the implications that the conditions (*b*) and (*c*) have on the hardness in the DCRA are relatively unexplored. Nonetheless, there are reasons to believe that the DCRA still holds under conditions (*b*) and (*c*).

To see why conditions (*b*) and (*c*) might not influence the DCRA, we remark that the DCRA is closely-related to the security of the RSA cryptosystem. Specifically, the DCRA for a specific modulus N is equivalent to assuming that the RSA cryptosystem with public exponentiation key $e = 3$ and modulus N has an "exponential security" (i.e., deciphering a ciphertext without the private key requires time exponential in the cost of decryption). In other words, if a certain family of moduli is used in practice, or are not actively avoided, this can be considered as supporting evidence that the DCRA holds for those moduli.

In practice it is generally prefered to use so-called "cryptographically strong primes"[2] p and q when constructing the modulus $N = pq$ for the RSA cryptosystem. One of the conditions for a prime p to be a cryptographically strong prime is that $p-1$ has large prime factors. Note that if $p-1$ has large prime factors, then the largest power of 2 that divides it must be small. In other words, if p and q are cryptographically strong primes, then condition (b) holds. Moreover, if $p-1$ and $q-1$ only have large prime factors, then the probability that $p-1$ and $q-1$ share a prime factors is relatively small, and condition (c) is thus likely to hold. Finally, we note that recently factored RSA numbers[3], which is a factoring challenge of a set of cryptographically strong moduli organized by the inventors of the RSA cryptosystem, all satisfy both conditions (b) and (c). For instance, all RSA numbers that have been factored over the last five years (i.e., `RSA-250`, `RSA-240`, `RSA-768`, `RSA-232` and `RSA-230`) all have $c' \leq 2$ and $c \leq 8$ in conditions (b) and (c).

D.3 Proof of Theorem 25

Theorem 25. *$L_{\mathrm{DCRI}} = \big(\{\mathcal{C}_n^{\mathrm{DCRI}}\}_{n\in\mathbb{N}}, \{\mathcal{D}_n^U\}_{n\in\mathbb{N}}\big)$ exhibits a $\mathsf{C}_{\mathcal{H}}/\mathsf{Q}_{\mathcal{H}}$ separation, where $\mathcal{H} = \mathcal{C}^{\mathrm{DCRI}}$ and $\mathcal{D}_n^U$ denotes the uniform distribution over $\mathbb{Z}_N^*$.*

Proof. To show quantum learnability, we note that N is known and we can thus use Shor's algorithm [176] to efficiently compute $d \in \{0, \ldots, (p-1)(q-1)\}$ such that

$$(m^3)^d \equiv m \mod N, \quad \text{for all } m \in \mathbb{Z}_N^*. \tag{D.9}$$

Next, we note that from $(x, c_m(x))$ we can retrieve the kth bit of m^3, where $k = \mathrm{int}(x_1 : \cdots : x_{\lfloor \log n \rfloor})$. Since for any given $k \in [n]$ we have

$$\mathsf{Pr}_{x\sim\mathcal{D}_n^U}\left(\mathrm{int}(x_1 : \cdots : x_{\lfloor \log n \rfloor}) = k\right) = \frac{1}{n} \tag{D.10}$$

we find that $\mathcal{O}(\mathrm{poly}(n))$ examples suffices to reconstruct the full binary representation of m^3 with high probability. Finally, using d and m^3 we can compute $(m^3)^d \equiv m \mod N$.

To show classical non-learnability we show that an efficient classical learning algorithm $\mathcal{A}_{\mathrm{learn}}$ can efficiently solve the discrete cube root problem. To do so, we let $e \in \mathbb{Z}_N^*$ and our goal is to use $\mathcal{A}_{\mathrm{learn}}$ to efficiently compute $m \in \mathbb{Z}_N^*$ such that $m^3 \equiv e \mod N$. First, we generate examples

$$(x, c_m(x)) = (x, \mathrm{bin}(e, k)), \tag{D.11}$$

where $x \in \{0,1\}^n$ is sampled uniformly at random and $k = \mathrm{int}(x_1 : \cdots : x_{\lfloor \log n \rfloor})$. If we plug these examples into $\mathcal{A}_{\mathrm{learn}}$ with $\epsilon = 1/n^3$ and $\delta = 1/3$, then with high

[2] `https://en.wikipedia.org/wiki/Strong_prime`
[3] `https://en.wikipedia.org/wiki/RSA_numbers`

probability we obtain some m' such that

$$\mathsf{Pr}_{k\sim[n]}\left(\mathrm{bin}(m^3,k)\neq\mathrm{bin}((m')^3,k)\right)\leq\frac{1}{n^3}, \tag{D.12}$$

where $k\in[n]$ is sampled uniformly at random. Next, we claim that $m^3\equiv(m')^3 \mod N$. Specifically, suppose there exists some i such that $\mathrm{bin}(m^3,i)\neq\mathrm{bin}((m')^3,i)$, then this implies that

$$\mathsf{Pr}_{k\sim[n]}\left(\mathrm{bin}(m^3,k)\neq\mathrm{bin}((m')^3,k)\right)=\frac{1}{n}\sum_{k=1}^{n}\mathbb{1}\left[\mathrm{bin}(m^3,k)=\mathrm{bin}((m')^3,k)\right] \tag{D.13}$$

$$\geq\frac{1}{n}, \tag{D.14}$$

which clearly contradicts Eq. (D.12). Now since $x\mapsto x^3 \mod N$ is a bijection to and from $\mathbb{Z}_N^*$ we conclude that $m=m'$ and that we have thus solved our instances of the discrete cube root. □

D.4 Proof of Theorem 26

Theorem 26. *Consider a family of concept classes $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ and distributions $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ such that*

Quantum learnability:

(a) Every $c_n\in\mathcal{C}_n$ can be evaluated quantumly in time $\mathcal{O}(\mathrm{poly}(n))$.

(b) There exists a polynomial p such that for every $n\in\mathbb{N}$ we have

$$|\mathcal{C}_n|\leq p(n).$$

Classical non-learnability:

(c) There exists a family $\{c_n\}_{n\in\mathbb{N}}$, where $c_n\in\mathcal{C}_n$, such that

$$(\{c_n\}_{n\in\mathbb{N}},\{\mathcal{D}_n\}_{n\in\mathbb{N}})\notin\mathsf{HeurP/poly}.$$

Then, $L=(\{\mathcal{C}_n\}_{n\in\mathbb{N}},\{\mathcal{D}_n\}_{n\in\mathbb{N}})$ exhibits a $\mathsf{CC/QQ}$ learning separation.

Proof. Firstly, a quantum learner can iterate over all concepts in $\mathcal{C}_n$ and find the one that matches the examples obtained from the oracle. In other words, a quantum learner can implement empirical risk minimization through brute-force search. By Corollary 2.3 of [174] this shows that $L\in\mathsf{QQ}$.

Next, suppose $L\in\mathsf{CC}$, i.e., suppose there exists an efficient classical learning algorithm for L that uses a classically evaluatable hypothesis class. By combining the classical learning algorithm with the evaluation algorithm of the hypothesis class

we obtain a polynomial-time classical randomized algorithm $\mathcal{A}$ such that for every $c'_n \in \mathcal{C}_n$ on input $\mathcal{T} = \{(x_i, c'_n(x_i)) \mid x_i \sim \mathcal{D}_n\}_{i=1}^{\text{poly}(n)}$ and $x \in \{0,1\}^n$ we have

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathsf{Pr}\left(\mathcal{A}(x, 0^{\lfloor 1/\epsilon \rfloor}, \mathcal{T}) = c'_n(x)\right) \geq \frac{2}{3}\right] \geq 1 - \epsilon$$

If we apply $\mathcal{A}$ to the concepts $\{c_n\}_{n\in\mathbb{N}}$ we obtain $(\{c_n\}_{n\in\mathbb{N}}, \{\mathcal{D}_n\}_{n\in\mathbb{N}}) \in \mathsf{HeurBPP/samp} \subseteq \mathsf{HeurP/poly}$, which contradicts the classical non-learnability assumption. Therefore, it must hold that $L \notin \mathsf{CC}$.

□

D.4.1 Proof of Lemma 27

Lemma 27. *If there exists a $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$ with $L \in \mathsf{BQP}$, then for every $L' \in \mathsf{BQP\text{-}complete}$[4] there exists a family of distributions $\mathcal{D}' = \{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ such that $(L', \mathcal{D}') \notin \mathsf{HeurP/poly}$.*

Proof. Let $L' \in \mathsf{BQP\text{-}complete}$ and consider the many-to-one polynomial-time reduction $f : L \to L'$ such that $L(x) = L'(f(x))$. Also, consider the pushforward distributions $\mathcal{D}'_n = f(\mathcal{D}_n)$ on $\{0,1\}^{n'}$[5], i..e, the distribution induced by first sampling $x \sim \mathcal{D}_n$ and subsequently computing $f(x)$. Next, we suppose that $(L', \mathcal{D}') \in \mathsf{HeurP/poly}$. Specifically, we suppose that there exists a classical algorithm $\mathcal{A}$ and a sequence advice strings $\{\alpha_n\}_{n\in\mathbb{N}}$ as in Definition 13 such that for every $n \in \mathbb{N}$:

$$\mathsf{Pr}_{y\sim\mathcal{D}'_n}\left[\mathcal{A}(y, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n'}) = L'(y)\right] \geq 1 - \epsilon \tag{D.15}$$

By the definition of the push-forward distribution $\mathcal{D}'_n$ we have

$$\mathsf{Pr}_{y\sim\mathcal{D}'_n}\left[\mathcal{A}(y, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n'}) = L'(y)\right] = \mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathcal{A}(f(x), 0^{\lfloor 1/\epsilon \rfloor}, \alpha_{n'}) = L'(f(x))\right] \tag{D.16}$$

Finally, we define a polynomial-time classical algorithm $\mathcal{A}'$ that uses advice as follows

$$\mathcal{A}'(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_n) = \mathcal{A}(f(x), 0^{\lfloor 1/\epsilon \rfloor}, \alpha_n).$$

Then, by Eq. (D.16) we have that

$$\mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathcal{A}'(x, 0^{\lfloor 1/\epsilon \rfloor}, \alpha_n) = L(x)\right] = \mathsf{Pr}_{x\sim\mathcal{D}_n}\left[\mathcal{A}(f(x), 0^{\lfloor 1/\epsilon \rfloor}, \alpha_n) = L'(f(x))\right] \tag{D.17}$$

$$\geq 1 - \epsilon \tag{D.18}$$

which implies that $(L, \mathcal{D}) \in \mathsf{HeurP/poly}$. This contradicts the assumptions, and we therefore conclude that indeed $(L', \mathcal{D}') \notin \mathsf{HeurP/poly}$.

[4] With respect to many-to-one reductions (as is the case for, e.g., quantum linear system solving [101]).

[5] Note that f can map instances $x \in \{0,1\}^n$ to instances of size n' that are at most polynomially larger than n.

□

D.4.2 Proof of Lemma 28

Lemma 28. *If $L \notin \mathsf{P/poly}$ and L is polynomially random self-reducible with respect to some distribution $\mathcal{D}$, then $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$.*

Proof. Since L is polynomially random self-reducible (for a formal definition we refer to [79]), we know that there exists a family of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, a polynomial-time computable function f, and some integer $k_n = \mathcal{O}(\text{poly}(n))$ such that

$$\mathsf{Pr}_{y_1,\ldots,y_{k_n}\sim\mathcal{D}_n}\Big(f\big(x, L(y_1), \ldots, L(y_{k_n})\big) = L(x)\Big) \geq \frac{3}{4} \tag{D.19}$$

Suppose $(L, \mathcal{D}) \in \mathsf{HeurP/poly}$, i.e., there exists a polynomial-time classical algorithm $\mathcal{A}$ and a sequence of advice strings $\{\alpha_n\}_{n\in\mathbb{N}}$ such that

$$\mathsf{Pr}_{y\sim\mathcal{D}_n}\Big(\mathcal{A}(y, 0^{\lfloor 1/\epsilon\rfloor}, \alpha_n) = L(y)\Big) \geq 1 - \epsilon. \tag{D.20}$$

Let $\epsilon' = 1/(9k)$, then by combining Eq. (D.19) and Eq. (D.20) we get that

$$\mathsf{Pr}_{y_1,\ldots,y_{k_n}\sim\mathcal{D}_n(x)}\Big(f\big(x, \mathcal{A}(y_1, 0^{1/\epsilon'}, \alpha_n), \ldots, \mathcal{A}(y_{k_n}, 0^{1/\epsilon'}, \alpha_n)\big) = L(x)\Big) \geq \frac{2}{3} \tag{D.21}$$

In other words, if we define

$$\mathcal{A}'(x, \alpha_n) = f\big(x, \mathcal{A}(y_1, 0^{1/\epsilon'}, \alpha_n), \ldots, \mathcal{A}(y_1, 0^{1/\epsilon'}, \alpha_n)\big),$$

where $y_i \sim \mathcal{D}_n$ are sampled during the runtime of the algorithm, then we conclude that $L \in \mathsf{BPP/poly} = \mathsf{P/poly}$. This contradicts the assumption in the lemma, and we therefore conclude that $(L, \mathcal{D}) \notin \mathsf{HeurP/poly}$.

□

D.5 Proof of Theorem 30

Theorem 30. *Suppose there exists a polynomial-time randomized classical algorithm $\mathcal{A}$ with the following property: for every geometrically-local family of n-qubit Hamiltonians $H(x)$ there exist a dataset $\mathcal{T}_H \in \{0,1\}^{\text{poly}(n)}$ such that for every sum $O = \sum_{i=1}^{L} O_i$ of $L \in \mathcal{O}(\text{poly}(n))$ many local observables with $\sum_{i=1}^{L} ||O_i|| \leq B$ for some constant B, the function*

$$\overline{f}_{H,O}(x) = \mathcal{A}(x, O, \mathcal{T}_H)$$

satisfies

$$\mathbb{E}_{x\sim[-1,1]^m}\Big[\,\big|\overline{f}_{H,O}(x) - f_{H,O}(x)\big|\,\Big] < \frac{1}{6},$$

where $f_{H,O}(x) = \text{Tr}\,[\rho_H(x)O]$ and $\rho_H(x)$ denotes the ground state of $H(x)$. Then, $\mathsf{DLP} \in \mathsf{P/poly}$.

Proof. We define DLP to be the problem of computing the first bit of $\log_a x$ (i.e., the smallest positive integer ℓ such that $a^\ell \equiv x \mod p$) with respect to a generator $a \in \mathbb{Z}_p^*$ for a given $x \in \mathbb{Z}_p^*$.

First, using Shor's algorithm we can construct a polynomial-depth circuit U_{Shor} such that

$$U_{\text{Shor}} |0, x, 0^\ell\rangle = (1-\alpha) |\mathsf{DLP}(x), x, 0^\ell\rangle + \alpha |\text{garbage}\rangle , \tag{D.22}$$

for all $x \in \{0,1\}^n$ and where $\alpha = \mathcal{O}(2^{-n})$. Next, we parameterize

$$U(x) = U \cdot \left(I_0 \otimes \left[\bigotimes_{i=1}^{n} X_i(g_\gamma(x_i) \cdot 2\pi) \right] \right), \tag{D.23}$$

where X is a rotation such that $X(0)|0\rangle = |0\rangle$ and $X(2\pi)|0\rangle = |1\rangle$, and g_γ is a continuous function such that

$$g_\gamma(x_i) = \begin{cases} 0, & x_i \in [-1, -\gamma) \\ (2\gamma - x)(\gamma + x)/(4\gamma^3), & x \in (-\gamma, \gamma) \\ 1, & x_i \in (\gamma, 1] \end{cases}, \tag{D.24}$$

for some $\gamma > 0$. Finally, we add $2T$ layers of identities to $U(x)$, where T denotes the depth of U_{Shor}.

We define $H(x)$ to be the Hamiltonian family on $\mathbb{C}^{2^s} \oplus \mathbb{C}^{2^{3T}}$ with $s = n + \ell + 1$ given by

$$H(x) = H_{\text{init}} + H_{\text{clock}} + \sum_{t=1}^{3T} H_t(x), \tag{D.25}$$

with

$$H_{\text{init}} = \sum_{i=1}^{s} |0\rangle\langle 0|_i , \tag{D.26}$$

$$H_{\text{clock}} = \sum_{t=1}^{3T-1} |01\rangle\langle 01|^{\text{clock}}_{t,t+1} , \tag{D.27}$$

$$H_t(x) = \frac{1}{2}\Big(I \otimes |100\rangle\langle 100|^{\text{clock}}_{t-1,t,t+1} + I \otimes |110\rangle\langle 110|^{\text{clock}}_{t-1,t,t+1} - \tag{D.28}$$

$$U_t(x) \otimes |110\rangle\langle 100|^{\text{clock}}_{t-1,t,t+1} - U_t(x)^\dagger \otimes |100\rangle\langle 110|^{\text{clock}}_{t-1,t,t+1} \Big) \tag{D.29}$$

where $|.\rangle\langle.|_i$ acts on the ith site of $\mathbb{C}^{2^s}$, $|.\rangle\langle.|^{\text{clock}}_j$ acts on the jth site of $\mathbb{C}^{2^{3T}}$ and U_t denotes the tth layer of gates in $U(x)$. Note that $H(x)$ is 5-local for all $x \in [-1, 1]$.

The ground state of $H(x)$ is given by $\rho(x) = |\psi(x)\rangle\langle\psi(x)|$, where

$$|\psi(x)\rangle = \frac{1}{\sqrt{3T}} \sum_{t=1}^{T} (U_t \cdots U_1)(x) \, |0^s\rangle \, |1^t 0^{3T-t}\rangle \,, \tag{D.30}$$

We define $O = |0\rangle\langle 0|_0 \otimes I \otimes |1\rangle\langle 1|_T^{\text{clock}}$ and note that it is a local observable with constant norm. Now $f_{H,O}$ defined in Eq. 6.8 is such that

$$f_{H,O}(x) = \text{Tr}\left[\rho_0(x) O\right] = \frac{2}{3} p_1(x), \tag{D.31}$$

where $p_1(x)$ denotes the probability that $|\psi_{\text{out}}(x)\rangle = U(x)\,|0^s\rangle$ outputs 1 when measuring the first qubit in the computational basis. In particular, we have that

$$f_{H,O}(x) = \text{Tr}\left[\rho_0(x) O\right] = \frac{2}{3} \left((1-\alpha)\mathsf{DLP}(g_\gamma(x)) + \alpha \cdot \text{garb}\right), \tag{D.32}$$

for all $x \in [-1,1]^n$ for which there does not exists $x_i \in (-\gamma, \gamma)$, and some quantity $\text{garb} \leq 1$.

Finally, assume that we obtain $\overline{f}_{H,O}$ such that

$$\mathbb{E}_{x\sim[-1,1]^n}\left[\left|\overline{f}_{H,O} - f_{H,O}\right|\right] < \frac{1}{6}. \tag{D.33}$$

Also, suppose there exists a bitstring $y \in \{0,1\}^n$ whose corresponding corner $C_y \subset [-1,1]^n$[6] with size γ is such that

$$\left|\mathbb{E}_{x\sim C_y}\left[\overline{f}_{H,O}\right] - \mathbb{E}_{x\sim C_y}\left[f_{H,O}\right]\right| > \frac{1}{3}. \tag{D.34}$$

Then, we find that

$$\begin{aligned}
\mathbb{E}_{x\sim[-1,1]^n}\left[\left|\overline{f}_{H,O} - f_{H,O}\right|\right] &= \int_{[-1,1]^n} \left|\overline{f}_{H,O} - f_{H,O}\right| dx && \text{(D.35)}\\
&\geq \int_{C_y} \left|\overline{f}_{H,O} - f_{H,O}\right| dx && \text{(D.36)}\\
&\geq \left|\int_{C_y} \overline{f}_{H,O} - f_{H,O} dx\right| && \text{(D.37)}\\
&= \left|\left(\int_{C_y} \overline{f}_{H,O} dx\right) - \left(\int_{C_y} f_{H,O} dx\right)\right| && \text{(D.38)}\\
&= \left|\mathbb{E}_{x\sim C_y}\left[\overline{f}_{H,O}\right] - \mathbb{E}_{x\sim C_y}\left[f_{H,O}\right]\right| > \frac{1}{3}. && \text{(D.39)}
\end{aligned}$$

[6]Here $y \in \{0,1\}^n$ is mapped to $\{-1,1\}^n$ by setting all 0s to -1, and the corner C_y consists of all points $x \in [-1,1]^n$ whose ith coordinate is γ close to y_i for all $i \in [n]$.

which clearly contradicts Eq. (D.33). We therefore conclude that

$$\left|\mathbb{E}_{x\sim C_y}\left[\overline{f}_{H,O}\right] - \mathbb{E}_{x\sim C_y}\left[f_{H,O}\right]\right| < \frac{1}{3}. \tag{D.40}$$

In conclusion, for every $y \in \{0,1\}^n$ the quantity $\mathbb{E}_{x\sim C_y}[\overline{f}_{H,O}]$ is exponentially close to $\mathsf{DLP}(y)$. Finally, we can efficiently estimate $\mathbb{E}_{x\sim C_y}[\overline{f}_{H,O}]$ to within additive inverse-polynomial error, which allows us to compute $\mathsf{DLP}(y)$ in $\mathsf{BPP}/\mathsf{poly} = \mathsf{P}/\mathsf{poly}$. □

Spectral gap and smoothness Note that the Hamiltonian family constructed above indeed does not satisfy all requirements for the methods of Huang et al. [107] to work. In particular, it is known that the spectral gap of Hamiltonians obtained from Kitaev's circuit-to-Hamiltonian construction (i.e., those defined in Eq. (D.25)) have a spectral gap that is inverse polynomial in the depth of the circuit, which is our case is polynomial in the instance size n. Moreover, since we apply a function g_γ to the parameters x (which has a rapid increase between $-\gamma$ and γ), it is likely that the average gradient of the function $\mathrm{Tr}\left[O\rho_H(x)\right]$ is not bounded by a constant, but rather scales with the number of parameters m (which in our case also scales with the instance size n).