Computational speedups and learning separations in quantum machine learning

Gyurik, C.

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.

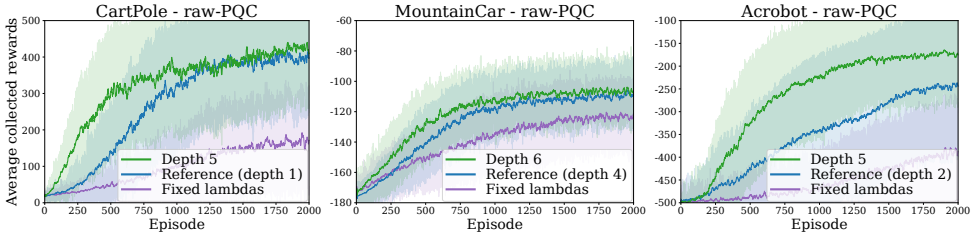


Figure C.1: **Influence of the model architecture for RAW-PQC agents.** The blue curves in each plot correspond to the learning curves from Fig. 5.2 and are taken as a reference.

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., $\mathcal{O}(n)$.

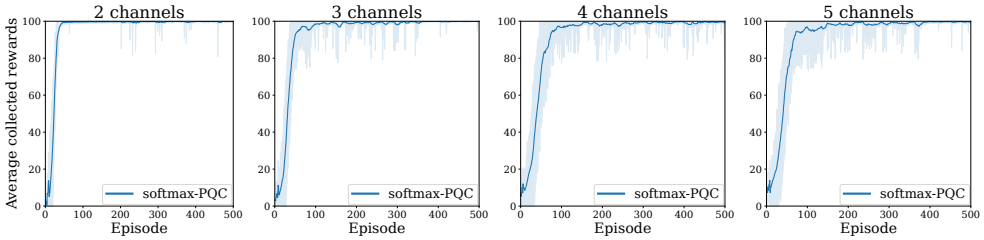
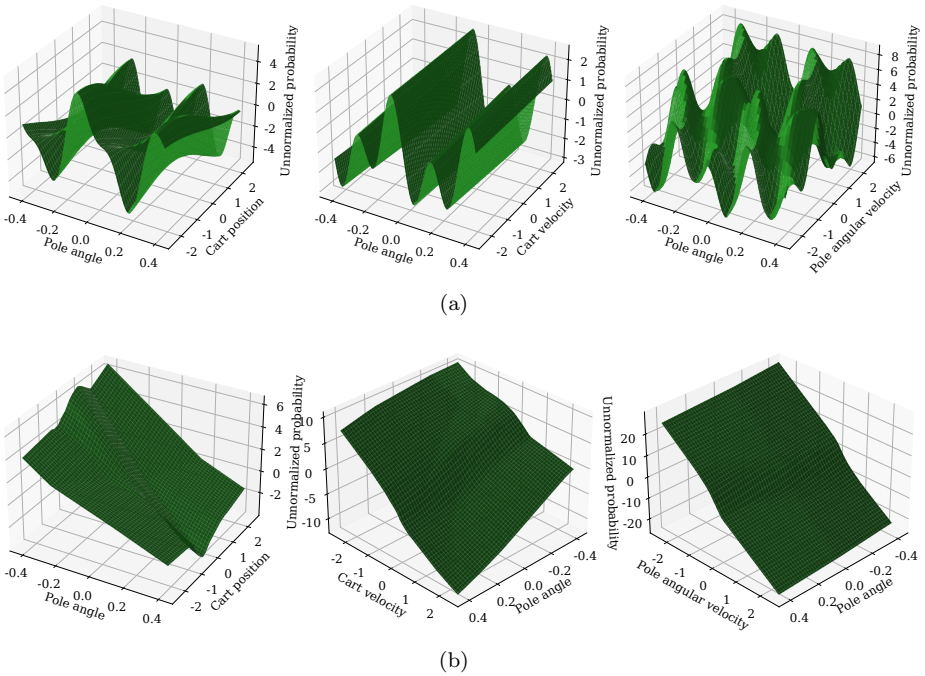


Figure C.2: **Performance of our SOFTMAX-PQC agents on the CognitiveRadio environment proposed in Ref. [58].** Average performance of 20 agents for system sizes (and number of qubits) $n = 2$ to 5.



(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.

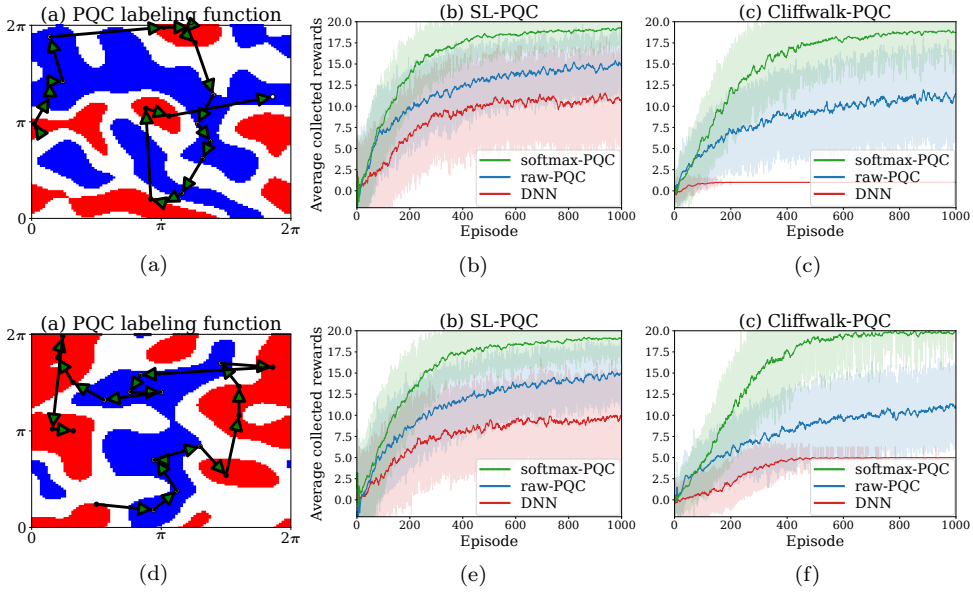


Figure C.4: **Different random initializations of PQC-generated environments and their associated learning curves.** See Fig. 5.4 for details. The additional learning curves (20 agents per curve) of randomly-initialized RAW-PQC agents highlight the hardness of these environments for PQC policies drawn from the same family as the environment-generating PQCs.

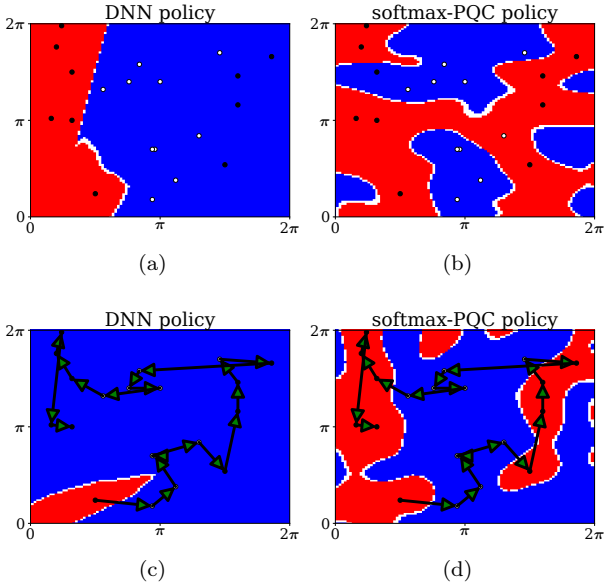


Figure C.5: **Prototypical policies learned by SOFTMAX-PQC agents and DNN agents in PQC-generated environments.** All policies are associated to the labeling function of Fig. C.4.d. Policies (a) and (b) are learned in the SL-PQC environment while policies (c) and (d) are learned in the Cliffwalk-PQC 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).