NExt ApplicationS of Quantum Computing

D5.2: Specification of QRL algorithm for inventory management

Document Properties

Contract Number	951821
Contractual Deadline	M25 (30/09/2022)
Dissemination Level	Public (PU)
Nature	Report
Editors	Vedran Dunjko, ULEI Simon Marshall, ULEI
Authors	Vedran Dunjko, ULEI
Reviewers	Andrés Gomez, CESGA Emmanuel Jeandel , LORIA
Date	7/10/2022
Keywords	Quantum algorithms, quantum reinforcement learning, reinforcement learning
Status	Final version
Release	1.1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951821

History of Changes

Release	Date	Author, Organisation	Description of Changes
0.1	31/07/2022	Simon Marshall, ULEI	NEASQC format
0.2	25/08/2022	Vedran Dunjko, ULEI	Executive summary added. Description of change of targeted application added. Section on applications to inventory management added.
1.0	20/09/2022	Vedran Dunjko, ULEI	Final version including the corrections of reviewers.
1.1	07/10/2022	Vedran Dunjko, ULEI	Dissemination status re-classified as public

Table of Contents

1. Executive Summary

The main objective of this deliverable is the theoretical development of quantum machine learning machinery for reinforcement learning, which is near-term-device friendly, yet sufficiently general to be applicable to the task of inventory management. This deliverable documents this achievement.

With the advent of real-world quantum computing, the idea that parametrized quantum computations can be used as hypothesis families in a quantum-classical machine learning system is gaining increasing traction. Such hybrid quantum-classical systems, which use a classical loop over a parameterized quantum circuit, have already shown the potential to tackle real-world tasks in supervised and generative learning, and recent works have established their provable advantages in special artificial tasks. Yet, in the case of reinforcement learning, which is arguably most challenging and where learning boosts would be extremely valuable, no proposal has been successful in solving even standard benchmarking tasks, nor in showing a theoretical learning advantage over classical algorithms. In this work, we achieve both. We propose a hybrid quantum-classical reinforcement learning model using very few qubits, which we show can be effectively trained to solve several standard benchmarking environments. Moreover, we demonstrate, and formally prove, the ability of parametrized quantum circuits to solve certain learning tasks that are intractable to classical models, including current state-of-art deep neural networks, under the widely-believed classical hardness of the discrete logarithm problem.

Our approach is general, and has shown applicable to both discrete and continuous state domains, with discrete actions, and is thus sufficiently general for application to the problem of inventory management. Furthermore, it is built around parametrized circuits, and is thus within the accepted paradigm for near-term-friendly proposals.

2. Introduction

Hybrid classical-quantum machine learning models constitute one of the most promising applications of near-term quantum computers (Bharti et al., 2021; Preskill, 2018). In these models, parametrized and data-dependent quantum computations define a hypothesis family for a given learning task, and a classical optimization algorithm is used to train them. For instance, parametrized quantum circuits (PQCs) (Benedetti et al., 2019) have already proven successful in classification (Farhi & Neven, 2018; Havlicek et al., 2019; Peters et al., 2021; Schuld et al., 2020; Schuld & Killoran, 2019), generative modeling (J.-G. Liu & Wang, 2018; Zhu et al., 2019) and clustering (Otterbach et al., 2017) problems. Moreover, recent results have shown proofs of their learning advantages in artificially constructed tasks (Havlicek et al., 2019; Huang et al., 2021) , some of which are based on widely believed complexity-theoretic (Havlicek et al., 2019) assumptions (Du et al., 2020; Huang et al., 2021; Y. Liu et al., 2021; Sweke et al., 2021). All these results, however, only consider supervised and generative learning settings.

Arguably, the largest impact quantum computing can have is by providing enhancements to the hardest learning problems. From this perspective, reinforcement learning (RL) stands out as a field that can greatly benefit from a powerful hypothesis family. This is showcased by the boost in learning performance that deep neural networks (DNNs) have provided to RL (Mnih et al., 2015), which enabled systems like AlphaGo (Silver et al., 2017), among other achievements (Berner et al., 2019; Mirowski et al., 2018). Nonetheless, the true potential of near-term quantum approaches in RL remains very little explored. The few existing works (Chen et al., 2020; Jerbi et al., 2021; Lockwood & Si, 2020; Wu et al., 2020) have failed so far at solving classical benchmarking tasks using PQCs and left open the question of their ability to provide a learning advantage.

2.1. Contributions

In this work, we demonstrate the potential of policies based on PQCs in solving classical RL environments. To do this, we first propose new model constructions, describe their learning algorithms, and show numerically the influence of design choices on their learning performance. In our numerical investigation, we consider benchmarking environments from OpenAI Gym (Brockman et al., 2016), for which good and simple DNN policies are known, and in which we demonstrate that PQC policies can achieve comparable performance. Second, inspired by the classification task of Havlíček *et al.* (Havlicek et al., 2019), conjectured to be classically hard by the authors, we construct analogous RL environments where we show an empirical learning advantage of our PQC policies over standard DNN policies used in deep RL. In the same direction, we construct RL environments with a provable gap in performance between a family of PQC policies and any efficient classical learner. These environments essentially build upon the work of Liu *et al.* (Y. Liu et al., 2021) by embedding into a learning setting the discrete logarithm problem (DLP), which is the problem solved by Shor's celebrated quantum algorithm (Shor, 1999) but widely believed to be classically hard to solve (Blum & Micali, 1984).

We note that an accompanying tutorial (Jerbi et al., 2021), implemented as part of the quantum machine learning library TensorFlow Quantum (Broughton et al., 2020), provides the code required to reproduce our numerical results and explore different settings. It also implements the Q-learning approach for PQC-based RL of Skolik *et al.* (Skolik et al., 2021)

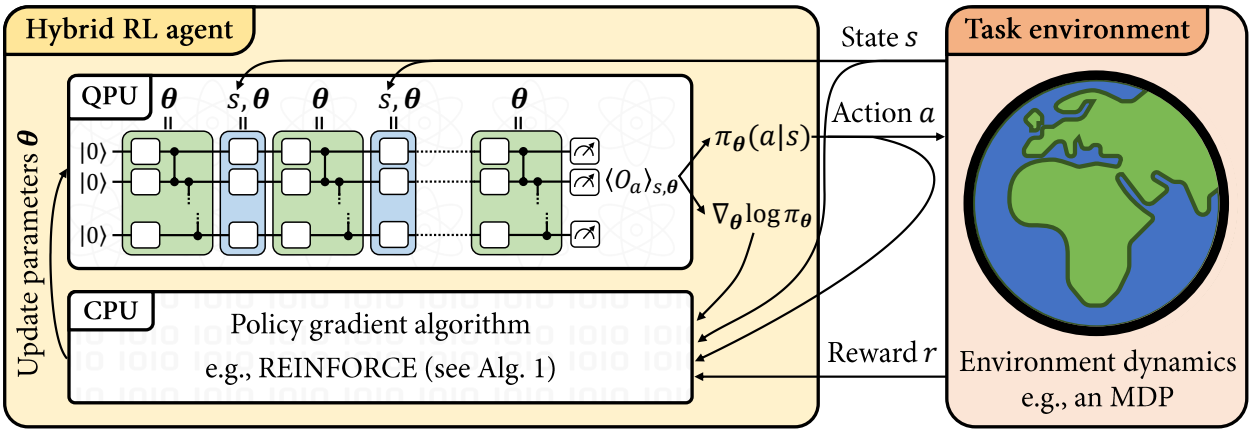

***Figure 1**: **Training parametrized quantum policies for reinforcement learning.** We consider a quantum-enhanced RL scenario where a hybrid quantum-classical agent learns by interacting with a classical environment. For each state s it perceives, the agent samples its next action a from its policy $\pi_{\boldsymbol{\theta}}(a|s)$ and perceives feedback on its behavior in the form of a reward r. For our hybrid agents, the policy $\pi_{\boldsymbol{\theta}}$ is specified by a PQC (see Def. 1) evaluated (along with the gradient $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}$) on a quantum processing unit (QPU). The training of this policy is performed by a classical learning algorithm, such as the* REINFORCE *algorithm (see Alg. 1), which uses sample interactions and policy gradients to update the policy parameters $\boldsymbol{\theta}$.*

2.2. Related work

Recently, a few works have been exploring hybrid quantum approaches for RL. Among these, Refs. (Chen et al., 2020; Lockwood & Si, 2020) also trained PQC-based agents in classical RL environments. However, these take a value-based approach to RL, meaning that they use PQCs as value-function approximators instead of direct policies. The learning agents in these works are also tested on OpenAI Gym environments (namely, a modified FrozenLake and CartPole), but do not achieve sufficiently good performance to be solving them, according to the Gym specifications. Ref. (Skolik et al., 2021) shows that, using some of our design choices for PQCs in RL (i.e., data re-uploading circuits (Pérez-Salinas et al., 2020) with trainable observable weights and input scaling parameters), one can also solve these environments using a value-based approach. An actor-critic approach to quantum RL (QRL) was introduced in Ref. (Wu et al., 2020), using both a PQC actor (or policy) and a PQC critic (or value-function approximator). In contrast to our work, these are trained in quantum environments (e.g., quantum-control environments), that provide a quantum state to the agent, which acts back with a continuous classical action. These aspects make it a very different learning setting to ours. Ref. (Jerbi et al., 2021) also describes a hybrid quantum-classical algorithm for value-based RL. The function-approximation models on which this algorithm is applied are however not PQCs but energy-based neural networks (e.g., deep and quantum Boltzmann machines). Finally, our work provides an alternative approach to take advantage of quantum effects in designing QRL agents compared to earlier approaches (Crawford et al., 2018; Dong et al., 2008; Dunjko et al., 2016; Neukart et al., 2018; Paparo et al., 2014), which are mainly based on (variations of) Grover's search algorithm (Grover, 1996) or quantum annealers (Johnson et al., 2011) to speed up sampling routines.

3. Parameterised quantum policies: definitions and learning algorithm

In this section, we give a detailed construction of our parametrized quantum policies and describe their associated training algorithms. We start however with a short introduction to the basic concepts of quantum computation, introduced in more detail in (De Wolf, 2019; Nielsen & Chuang, 2000).

3.1. Quantum computation: a primer

A quantum system composed of n qubits is represented by a 2^n-dimensional complex Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. Its quantum state is described by a vector $|\psi\rangle \in \mathcal{H}$ of unit norm $\langle\psi|\psi\rangle = 1$, where we adopt the bra-ket notation to describe vectors $|\psi\rangle$, their conjugate transpose $\langle\psi|$ and inner-products $\langle\psi|\psi'\rangle$ in $\mathcal{H}$. Single-qubit computational basis states are given by $|0\rangle = (1,0)^T, |1\rangle = (0,1)^T$, and their tensor products describe general computational basis states, e.g., $|10\rangle = |1\rangle \otimes |0\rangle = (0,0,1,0)^T$.

A quantum gate is a unitary operation U acting on $\mathcal{H}$. When a gate U acts non-trivially only on a subset $S \subseteq [n]$ of qubits, we identify it to the operation $U \otimes \mathbb{1}_{[n]\setminus S}$, where $\mathbb{1}_x$ denotes the identity acting on x. In this work, we are mainly interested in the single-qubit Pauli gates Z, Y and their associated rotations R_z, R_y:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, R_z(\theta) = \exp\left(-i\frac{\theta}{2}Z\right), \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, R_y(\theta) = \exp\left(-i\frac{\theta}{2}Y\right), \tag{3.1}$$

for rotation angles $\theta \in \mathbb{R}$, and the 2-qubit Ctrl-Z gate $\bullet\!\!-\!\!\bullet = \mathrm{diag}(1,1,1,-1)$.

A projective measurement is described by a Hermitian operator O called an observable. Its spectral decomposition $O = \sum_m \alpha_m P_m$ in terms of eigenvalues α_m and orthogonal projections P_m defines the outcomes of this measurement, according to the Born rule: a measured state $|\psi\rangle$ gives the outcome α_m and gets projected onto the state $P_m|\psi\rangle / \sqrt{p(m)}$ with probability $p(m) = \langle\psi| P_m |\psi\rangle = \langle P_m \rangle_\psi$. The expectation value of the observable O with respect to $|\psi\rangle$ is $\mathbb{E}_\psi[O] = \sum_m p(m)\alpha_m = \langle O \rangle_\psi$.

3.2. The RAW-PQC and SOFTMAX-PQC policies

At the core of our parametrized quantum policies is a PQC defined by a unitary $U(s, \boldsymbol{\theta})$ that acts on a fixed n-qubit state (e.g., $|0\rangle^{\otimes n}$). This unitary encodes an input state $s \in \mathbb{R}^d$ of the classical environment, and is parametrized by a trainable vector $\boldsymbol{\theta}$. Although different choices of PQCs are possible, throughout our numerical experiments (e.g., Sec. 5.2), we consider so-called hardware-efficient PQCs (Kandala et al., 2017) with an alternating-layered architecture (Pérez-Salinas et al., 2020; Schuld et al., 2021). This architecture is depicted in Fig. 2 and essentially consists in an alternation of D_{enc} encoding unitaries U_{enc} (composed of single-qubit rotations R_z, R_y) and $D_{\text{enc}}+1$ variational unitaries U_{var} (composed of single-qubit rotations R_z, R_y and entangling Ctrl-Z gates $\bullet\!\!-\!\!\bullet$).

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

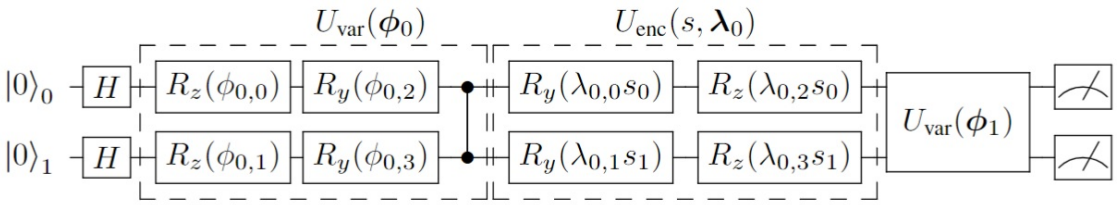

***Figure 2*: PQC architecture for $n = 2$ qubits and depth $D_{\text{enc}} = 1$.** *This architecture is composed of alternating layers of encoding unitaries $U_{\text{enc}}(s, \boldsymbol{\lambda}_i)$ taking as input a state vector $s = (s_0, \ldots, s_{d-1})$ and scaling parameters $\boldsymbol{\lambda}_i$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

Definition 1 (RAW- and SOFTMAX-PQC)**.** *Given a PQC acting on n qubits, taking as input a state $s \in \mathbb{R}^d$, rotation angles $\boldsymbol{\phi} \in [0, 2\pi]^{|\boldsymbol{\phi}|}$ and scaling parameters $\boldsymbol{\lambda} \in \mathbb{R}^{|\boldsymbol{\lambda}|}$, such that its corresponding unitary $U(s, \boldsymbol{\phi}, \boldsymbol{\lambda})$ produces the quantum state $|\psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}}\rangle = U(s, \boldsymbol{\phi}, \boldsymbol{\lambda}) |0^{\otimes n}\rangle$, we define its associated* RAW-PQC *policy as:*

$$\pi_{\boldsymbol{\theta}}(a|s) = \langle P_a \rangle_{s,\boldsymbol{\theta}} \tag{3.2}$$

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

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{\beta \langle O_a \rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'} e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}}} \tag{3.3}$$

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

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

In our PQC construction, we include trainable *scaling parameters* $\boldsymbol{\lambda}$, used in every encoding gate to rescale its input components. This modification to the standard data encoding in PQCs comes in light of recent considerations on the structure of PQC functions (Schuld et al., 2019). These additional parameters allow to represent functions with a wider and richer spectrum of frequencies, and hence provide shallow PQCs with more expressive power.

3.3. Learning algorithm

In order to analyze the properties of our PQC policies without the interference of other learning mechanisms (Weng, 2018), we train these policies using the basic Monte Carlo policy gradient algorithm REINFORCE (Sutton, Barto, et al., 1998; Williams, 1992) (see Alg. 1). This algorithm consists in evaluating Monte Carlo estimates of the value function $V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right]$, $\gamma \in [0, 1]$, using batches of interactions with the environment, and updating the policy parameters $\boldsymbol{\theta}$ via a gradient ascent on $V_{\pi_{\boldsymbol{\theta}}}(s_0)$. T denotes the horizon of interest, i.e., the number of steps in the future we wish to consider. The resulting updates (see line 8 of Alg. 1) involve the gradient of the log-policy $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$, which we therefore need to compute for our policies. We describe this computation in the following lemma.

Lemma 1. *Given a* SOFTMAX-PQC *policy $\pi_{\boldsymbol{\theta}}$(a—s), the gradient of its logarithm is given by:*

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) = \beta \Big(\nabla_{\boldsymbol{\theta}} \langle O_a \rangle_{s,\boldsymbol{\theta}} - \sum\nolimits_{a'} \pi_{\boldsymbol{\theta}}(a'|s) \nabla_{\boldsymbol{\theta}} \langle O_{a'} \rangle_{s,\boldsymbol{\theta}} \Big). \tag{3.4}$$

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

Input: a PQC policy $\pi_{\boldsymbol{\theta}}$ from Def. 1; a value-function approximator $\widetilde{V}_{\boldsymbol{\omega}}$

Initialize parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$;
while *True* **do**
  Generate N episodes $\{(s_0, a_0, r_1, \ldots, s_{H-1}, a_{H-1}, r_H)\}_i$ following $\pi_{\boldsymbol{\theta}}$;
  for *episode* i *in batch* **do**
    Compute the returns $G_{i,t} \leftarrow \sum_{t'=1}^{H-t} \gamma^{t'} r_{t+t'}^{(i)}$;
    Compute the gradients $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)}|s_t^{(i)})$ using Lemma 1;
  Fit $\{\widetilde{V}_{\boldsymbol{\omega}}(s_t^{(i)})\}_{i,t}$ to the returns $\{G_{i,t}\}_{i,t}$;
  Compute $\Delta\boldsymbol{\theta} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)}|s_t^{(i)}) \left(G_{i,t} - \widetilde{V}_{\boldsymbol{\omega}}(s_t^{(i)})\right)$;
  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}$;

Partial derivatives with respect to observable weights are trivially given by $\partial_{w_{a,i}}\langle O_a\rangle_{s,\boldsymbol{\theta}} = \langle\psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}}|H_{a,i}|\psi_{s,\boldsymbol{\phi},\boldsymbol{\lambda}}\rangle$ *(see Def. 1), while derivatives with respect to rotation angles* $\partial_{\phi_i}\langle O_a\rangle_{s,\boldsymbol{\theta}}$ *and scaling parameters*[1] $\partial_{\lambda_i}\langle O_a\rangle_{s,\boldsymbol{\theta}}$ *can be estimated via the parameter-shift rule (Mitarai et al., 2018; Schuld et al., 2019):*

$$\partial_i \langle O_a\rangle_{s,\boldsymbol{\theta}} = \frac{1}{2}\big(\langle O_a\rangle_{s,\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e_i}} - \langle O_a\rangle_{s,\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e_i}}\big), \tag{3.5}$$

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

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) = \nabla_{\boldsymbol{\theta}} \langle P_a\rangle_{s,\boldsymbol{\theta}} \,/\, \langle P_a\rangle_{s,\boldsymbol{\theta}} \tag{3.6}$$

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

In some of our environments, we additionally rely on a linear value-function baseline to reduce the variance of the Monte Carlo estimates (Greensmith et al., 2004). We choose it to be identical to that of Ref. (Duan et al., 2016).

3.4. Efficient policy sampling and policy-gradient evaluation

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

By design, sampling from our RAW-PQC policies can be done with a single execution (and measurement) of the PQC: the projective measurement corresponding to the observable $O = \sum_a aP_a$ naturally samples a basis state associated to action a with probability $\langle P_a\rangle_{s,\boldsymbol{\theta}}$. However, as Eq. (3.6) indicates, in order to train these policies using REINFORCE, one is nonetheless required to estimate the expectation values $\langle P_a\rangle_{s,\boldsymbol{\theta}}$, along with the gradients $\nabla_{\boldsymbol{\theta}}\langle P_a\rangle_{s,\boldsymbol{\theta}}$. Fortunately, these quantities can be estimated efficiently up to some additive error ε, using only $\mathcal{O}(\varepsilon^{-2})$ repeated executions and measurements on a quantum computer.

In the case of our SOFTMAX-PQC policies, it is less clear whether similar noisy estimates $\widetilde{\langle O_a\rangle}_{s,\boldsymbol{\theta}}$ of the expectation values $\langle O_a\rangle_{s,\boldsymbol{\theta}}$ are sufficient to evaluate policies of the form of Eq. (3.3). We show however that, using these noisy estimates, we can compute a policy $\widetilde{\pi}_{\boldsymbol{\theta}}$ that produces samples close to that of the true policy $\pi_{\boldsymbol{\theta}}$. We state our result formally in the following lemma, proven in Appendix A.2.

Lemma 2. *For a* SOFTMAX-PQC *policy* $\pi_{\boldsymbol{\theta}}(a—s)$ *defined by a unitary* $U(s,\boldsymbol{\theta})$ *and observables* O_a*, call* $\widetilde{\langle O_a\rangle}_{s,\boldsymbol{\theta}}$ *approximations of the true expectation values* $\langle O_a\rangle_{s,\boldsymbol{\theta}}$ *with at most* ε *additive error. Then the approximate policy* $\widetilde{\pi}_{\boldsymbol{\theta}}(a|s) = \mathrm{softmax}_\beta(\widetilde{\langle O_a\rangle}_{s,\boldsymbol{\theta}})$ *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* $\pi_{\boldsymbol{\theta}}$*.*

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

We also obtain a similar result for the log-policy gradient of SOFTMAX-PQCs (see Lemma 1), that we show can be efficiently estimated to additive error in ℓ_∞-norm (see Appendix A.2 for a proof).

4. Performance comparison in benchmaking environments

In the previous section, we have introduced our quantum policies and described several of our design choices. We defined the RAW-PQC and SOFTMAX-PQC models and introduced two original features for PQCs: trainable observables at their output and trainable scaling parameters for their input. In this section, we evaluate the influence of these design choices on learning performance through numerical simulations. We consider three classical benchmarking environments from the OpenAI Gym library (Brockman et al., 2016): CartPole, MountainCar and Acrobot. All three have continuous state spaces and discrete action spaces (see Appendix A.4 for their specifications). Moreover, simple NN-policies, as well as simple closed-form policies, are known to perform very well in these environments (OpenAI, 2020), which makes them an excellent test-bed to benchmark PQC policies.

4.1. RAW-PQC v.s. SOFTMAX-PQC

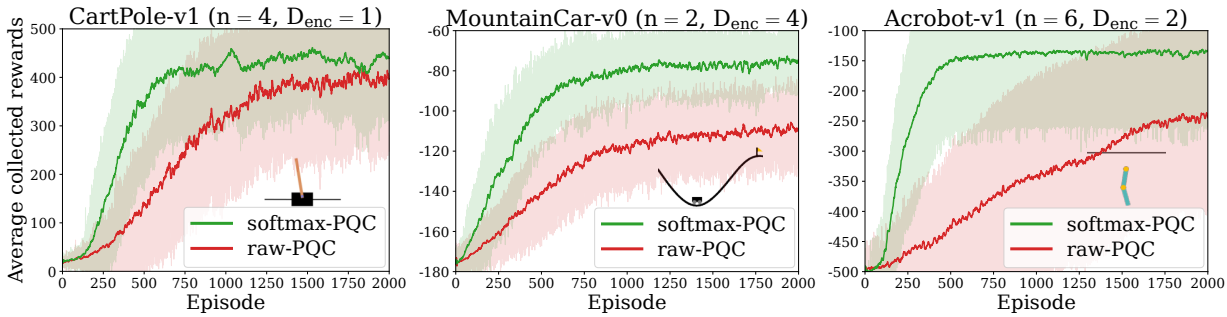

***Figure 3**: **Numerical evidence of the advantage of SOFTMAX-PQC over RAW-PQC in benchmarking environments.** The learning curves (20 agents per curve) of randomly-initialized SOFTMAX-PQC agents (green curves) and RAW-PQC agents (red curves) in OpenAI Gym environments: CartPole-v1, MountainCar-v0, and Acrobot-v1. Each curve is temporally averaged with a time window of 10 episodes (complete "games" until the agent fails). All agents have been trained using the REINFORCE algorithm (see Alg. 1), with value-function baselines for the MountainCar and Acrobot environments.*

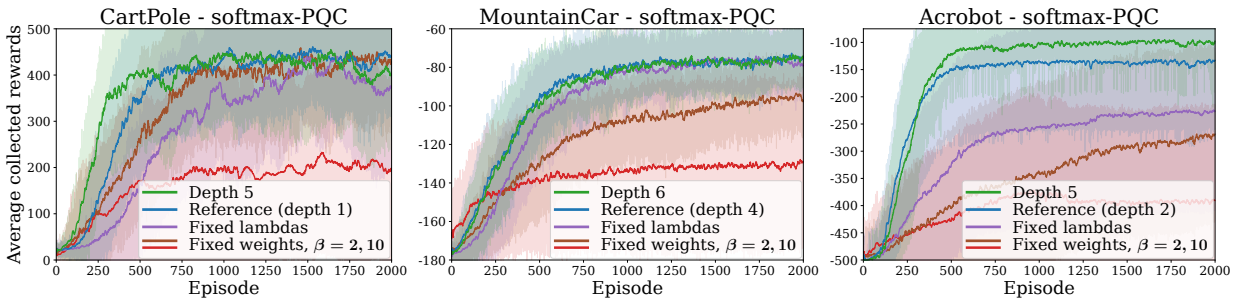

***Figure 4**: **Influence of the model architecture for SOFTMAX-PQC agents.** The blue curves in each plot correspond to the learning curves from Fig. 3 and are taken as a reference. Other curves highlight the influence of individual hyperparameters. For RAW-PQC agents, see Appendix A.5.*

In our first set of experiments, presented in Fig. 3, we evaluate the general performance of our proposed policies. The full specification of all the hyperparameters is given in the appendix. The aim of these experiments is twofold: first, to showcase that quantum policies based on shallow PQCs and acting on very few qubits can be trained to good performance in our selected environments; second, to test the advantage of SOFTMAX-PQC policies over RAW-PQC policies that we conjectured in the Sec. 3.2. To assess these claims, we take a similar approach for each of our benchmarking environments, in which we evaluate the average learning performance of 20 RAW-PQC and 20 SOFTMAX-PQC agents. Apart from the PQC depth, the shared hyperparameters of these two models were jointly picked as to give the best overall performance

for both; the hyperparameters specific to each model were optimized independently. As for the PQC depth D_{enc}, the latter was chosen as the minimum depth for which near-optimal performance was observed for either model. The simulation results confirm both our hypotheses: quantum policies can achieve good performance on the three benchmarking tasks that we consider, and we can see a clear separation between the performance of SOFTMAX-PQC and RAW-PQC agents.

4.2. Influence of architectural choices

The results of the previous subsection however do not indicate whether other design choices we have made in Sec. 3.2 had an influence on the performance of our quantum agents. To address this, we run a second set of experiments, presented in Fig. 4. In these simulations, we evaluate the average performance of our SOFTMAX-PQC agents after modifying one of three design choices: we either increment the depth of the PQC (until no significant increase in performance is observed), fix the input-scaling parameters $\boldsymbol{\lambda}$ to $\mathbf{1}$, or fix the observable weights $\boldsymbol{w}$ to $\mathbf{1}$. By comparing the performance of these agents with that of the agents from Fig. 3, we can make the following observations:

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

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

5. Quantum advantage of PQC agents in RL environments

The proof-of-concept experiments of the previous section show that our PQC agents can learn in basic classical environments, where they achieve comparable performance to standard DNN policies. This observation naturally raises the question of whether there exist RL environments where PQC policies can provide a learning advantage over standard classical policies. In this section, we answer this question in the affirmative by constructing: a) environments with a provable separation in learning performance between quantum and any classical (polynomial-time) learners, and b) environments where our PQC policies of Sec. 4 show an empirical learning advantage over standard DNN policies.

5.1. Quantum advantage of PQC policies over any classical learner

In this subsection, we construct RL environments with theoretical guarantees of separation between quantum and classical learning agents. These constructions are predominantly based on the recent work of Liu *et al.* (Y. Liu et al., 2021), which defines a classification task out of the discrete logarithm problem (DLP), i.e., the problem solved in the seminal work of Shor (Shor, 1999). In broad strokes, this task can be viewed as an encryption of an easy-to-learn problem. For an "un-encrypted" version, one defines a labeling f_s of integers between 0 and $p-2$ (for a large prime p), where the integers are labeled positively if and only if they lie in the segment $[s, s+(p-3)/2]$ (mod $p-1$). Since this labeling is linearly separable, the concept class $\{f_s\}_s$ is then easy to learn. To make it hard, the input integers x (now between 1 and $p-1$) are first encrypted using modular exponentiation, i.e., the secure operation performed in the Diffie–Hellman key exchange protocol. In the encrypted problem, the logarithm of the input integer $\log_g(x)$ (for a generator g of $\mathbb{Z}_p^*$, see Appendix A.6) hence determines the label of x. Without the ability to decrypt by solving DLP, which is widely believed to be classically intractable, the numbers appear randomly labeled. Moreover, Liu *et al.* show that achieving non-trivial labeling accuracy $1/2 + 1/\text{poly}(n)$ (for $n = \log(p)$, i.e., slightly better than random guessing) with a classical polynomial-time algorithm using $\text{poly}(n)$ examples would lead to an efficient classical algorithm that solves DLP (Y. Liu et al., 2021). In contrast, the same authors construct a family of quantum learners based on Shor's algorithm, that can achieve a labeling accuracy larger than 0.99 with high probability.

5.1.1. SL-DLP

Our objective is to show that analogous separations between classical and quantum learners can be established for RL environments, in terms of their attainable value functions. We start by pointing out that supervised learning (SL) tasks (and so the classification problem of Liu *et al.*) can be trivially embedded into RL environments (Dunjko et al., 2017): for a given concept f_s, the states x are datapoints, an action a is an agent's guess on the label of x, an immediate reward specifies if it was correct (i.e., $f_s(x) = a$), and subsequent states are chosen uniformly at random. In such settings, the value function is trivially related to the testing accuracy of the SL problem, yielding a direct reduction of the separation result of Liu *et al.* (Y. Liu et al., 2021) to an RL setting. We call this family of environments SL-DLP.

5.1.2. Cliffwalk-DLP

In the above described SL-DLP construction, we made the environment fully random in order to simulate the process of obtaining i.i.d. samples in an SL setting. It is an interesting question whether similar results can be obtained for environments that are less random, and endowed with temporal structure, which is characteristic of RL. In our second family of environments (Cliffwalk-DLP), we supplement the SL-DLP construction with next-state transitions inspired by the textbook "cliff walking" environment of Sutton & Barto (Sutton, Barto, et al., 1998): all states are ordered in a chain and some actions of the agent can lead to immediate episode termination. We keep however stochasticity in the environment by allowing next states to be uniformly sampled, with a certain probability δ (common in RL to ensure that an agent is not simply memorizing a correct sequence of actions). This allows us to show that, as long as sufficient randomness is provided, we still have a simple classical-quantum separation.

5.1.3. Deterministic-DLP

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

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

Theorem 1. *There exist families of reinforcement learning environments which are: i) fully random (i.e., subsequent states are independent from the previous state and action); ii) partially random (i.e., the previous moves determine subsequent states, except with a probability δ at least 0.86 where they are chosen uniformly at random), and iii) fully deterministic; such that there exists a separation in the value functions achievable by a given quantum polynomial-time agent and any classical polynomial-time agent. Specifically, the value of the initial state for the quantum agent $V_q(s_0)$ is $\varepsilon-$close to the optimal value function (for a chosen ε, and with probability above 2/3). Further, if there exists a classical efficient learning agent that achieves a value $V_c(s_0)$ better than $V_{\text{rand}}(s_0)+\varepsilon'$ (for a chosen ε', and with probability above 0.845), then there exists a classical efficient algorithm to solve DLP. Finally, we have $V_q(s_0)-V_c(s_0)$ larger than some constant, which depends on the details of the environment.*

The remaining point we need to address here is that the learning agents of Liu *et al.* do not rely on PQCs but rather support vector machines (SVMs) based on quantum kernels (Havlicek et al., 2019; Schuld & Killoran, 2019). Nonetheless, using a connection between these quantum SVMs and PQCs (Schuld & Killoran, 2019), we construct PQC policies which are as powerful in solving the DLP environments as the agents of Liu *et al.* (even under similar noise considerations). We state our result in the following informal theorem, that we re-state formally, along with the details of our construction in Appendices A.10 and A.11.

Theorem 2 (informal version)**.** *Using a training set of size polynomial in $n=\log(p)$ and a number of (noisy) quantum circuit evaluations also polynomial in n, we can train a PQC classifier on the DLP task of Liu* et al. *of size n that achieves a testing accuracy arbitrarily close to optimal, with high probability. This PQC classifier can in turn be used to construct close-to-optimal quantum agents in our DLP environments, as prescribed by Theorem 1.*

The above cannot be achieved using just a classical computer in polynomial time.

5.2. Quantum advantage of PQC policies over DNN policies

While the DLP environments establish a proof of the learning advantage PQC policies can have in theory, these environments remain extremely contrived and artificial. They are based on algebraic properties that agents must explicitly decrypt in order to perform well. Instead, we would like to consider environments that are less tailored to a specific decryption function, which would allow more general agents to learn. To do this, we take inspiration from the work of Havlíček *et al.* (Havlicek et al., 2019), who, in order to test their PQC classifiers, define a learning task generated by similar quantum circuits.

5.3. PQC-generated environments

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

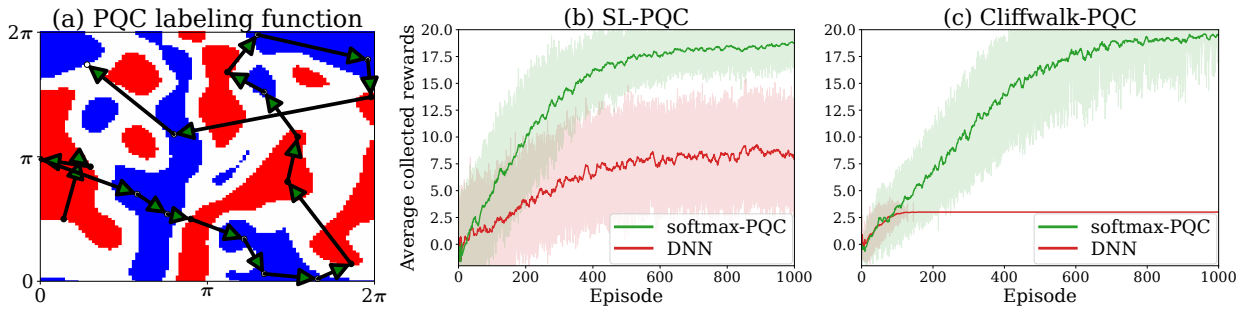

Figure 5: Numerical evidence of the advantage of PQC policies over DNN policies in PQC-generated environments. *(a) Labeling function and training data used for both RL environments. The data labels (red for $+1$ label and blue for -1 label) are generated using a* RAW-PQC *of depth $D_{enc} = 4$ with a margin $\Delta = 0.3$ (white areas). The training samples are uniformly sampled from the blue and red regions, and arrows indicate the rewarded path of the cliffwalk environment. (b) and (c) The learning curves (20 agents per curve) of randomly-initialized* SOFTMAX-PQC *agents and DNN agents in RL environments where input states are (b) uniformly sampled from the dataset and (c) follow cliffwalk dynamics. Each curve is temporally averaged with a time window of 10 episodes.*

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

5.4. Performance comparison

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

Despite this hyperparametrization, we find (see Fig. 5, and Fig. 9 in Appendix A.5 for different environment instances) that the performance of DNN policies on these tasks remains limited compared to that of SOFTMAX-PQCs, that learn close-to-optimal policies on both tasks. Moreover, we observe that the separation in performance gets boosted by the cliffwalk temporal structure. This is likely do to the increased complexity of this task, as, in order to move farther in the cliffwalk, the policy family should allow learning new labels without "forgetting" the labels of earlier states. In these particular case studies, the SOFTMAX-PQC policies exhibited sufficient flexibility in this sense, whereas the DNNs we considered did not (see Appendix A.5 for a visualization of these policies). Note that these results do not reflect the difficulty of our tasks at the sizes we consider (a look-up table would perform optimally) but rather highlight the inefficacy of these DNNs at learning PQC functions.

6. Applicability to *inventory management* and other industrial problems

The main objective of task T5.1 in this phase of the project was to develop quantum machine learning machinery for reinforcement learning which: i) may harbour a capacity for quantum advantage, ii) is compatible with current approaches to work under near-term quantum computing restrictions and iii) is general enough to be in later stages applied to instances of inventory management and other related industrial problems in simulations and, if possible, on real devices.

With this deliverable we achieve this task. We have provided a rigorous proof for the capacity for quantum advantage, the overall paradimg is within the variational circuit framework, and the method is policy gradient based, meaning it can work with continuous and discrete input domains.

With regards to the specific problem of inventory management, it constitutes a sequential decision problem where the actions of an agent control the re-stocking of an inventory based on previous supply-and-demand experiences. It is a critical problem in modern business, widely acknowledged as hard (as it also involves dealing with temporal correlations and uncertainties), and may thus benefit from advanced reinforcement learning solutions. Prior work in the classical domain has paved the way for the application of a RL algorithm. For example, references (Boute et al., 2022; Geevers, 2020; Gijsbrechts et al., 2021) constitute just a few of the available studies documenting and benchmarking how RL machinery of the type developed here can be applied to the problem. In the remainder of this project we will make further steps to implement these methods and apply them to (perhaps simplified) industrially-relevant instances of inventory management problems.

7. Conclusions

In this work, we have investigated the design of quantum RL agents based on PQCs. We proposed several constructions and showed the impact of certain design choices on learning performance. In particular, we introduced the SOFTMAX-PQC model, where a softmax policy is computed from expectation values of a PQC with both trainable observable weights and input scaling parameters. These added features to standard PQCs used in ML (e.g., as quantum classifiers) enhance both the expressivity and flexibility of PQC policies, which allows them to achieve a learning performance on benchmarking environments comparable to that of standard DNNs. We additionally demonstrated the existence of task environments, constructed out of PQCs, that are very natural for PQC agents, but on which DNN agents have a poor performance. To strengthen this result, we constructed several RL environments, each with a different degree of degeneracy (i.e., closeness to a supervised learning task), where we showed a rigorous separation between a class of PQC agents and any classical learner, based on the widely-believed classical hardness of the discrete logarithm problem. We believe that our results constitute strides toward a practical quantum advantage in RL using near-term quantum devices, and form the basis to address the problem inventory management, which is the objective of our use case.

List of Figures

List of Tables

beep

Bibliography

Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, *4*(4), 043001.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., et al. (2021). Noisy intermediate-scale quantum (nisq) algorithms. *arXiv preprint arXiv:2101.08448*.

Blum, M., & Micali, S. (1984). How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, *13*(4), 850–864.

Boute, R. N., Gijsbrechts, J., van Jaarsveld, W., & Vanvuchelen, N. (2022). Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research*, *298*(2), 401–412. https://doi.org/https://doi.org/10.1016/j.ejor.2021.07.016

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., Massey, P., Niu, M. Y., Halavati, R., Peters, E., et al. (2020). Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*.

Chen, S. Y.-C., Yang, C.-H. H., Qi, J., Chen, P.-Y., Ma, X., & Goan, H.-S. (2020). Variational quantum circuits for deep reinforcement learning. *IEEE Access*, *8*, 141007–141024.

Crawford, D., Levit, A., Ghadermarzy, N., Oberoi, J. S., & Ronagh, P. (2018). Reinforcement learning using quantum boltzmann machines. *Quantum Information & Computation*, *18*(1-2), 51–74.

De Wolf, R. (2019). Quantum computing: Lecture notes. *arXiv preprint arXiv:1907.09415*.

Dong, D., Chen, C., Li, H., & Tarn, T.-J. (2008). Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(5), 1207–1220.

Du, Y., Hsieh, M.-H., Liu, T., & Tao, D. (2020). Expressive power of parametrized quantum circuits. *Physical Review Research*, *2*(3), 033125.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. *International conference on machine learning*, 1329–1338.

Dunjko, V., Liu, Y.-K., Wu, X., & Taylor, J. M. (2017). Exponential improvements for quantum-accessible reinforcement learning. *arXiv preprint arXiv:1710.11160*.

Dunjko, V., Taylor, J. M., & Briegel, H. J. (2016). Quantum-enhanced machine learning. *Physical review letters*, *117*(13), 130501.

Farhi, E., & Neven, H. (2018). Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*.

Geevers, K. (2020). Deep reinforcement learning in inventory management. http://essay.utwente.nl/85432/

Gijsbrechts, J., Boute, R. N., Van Mieghem, J. A., & Zhang, D. (2021). Can deep reinforcement learning improve inventory management? performance on dual sourcing, lost sales and multi-echelon problems. *Manufacturing Service Operations Management*, *298*(2), 401–412. https://doi.org/http://dx.doi.org/10.2139/ssrn.3302881

Google. (2018). Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum circuits.

Goto, T., Tran, Q. H., & Nakajima, K. (2021). Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces. *Physical Review Letters*, *127*(9), 090506.

Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, *5*(Nov), 1471–1530.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 212–219.

Havlicek, V., Corcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, *567*(7747), 209–212.

Huang, H.-Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., & McClean, J. R. (2021). Power of data in quantum machine learning. *Nature communications*, *12*(1), 1–9.

Jerbi, S., Trenkwalder, L. M., Poulsen Nautrup, H., Briegel, H. J., & Dunjko, V. (2021). Quantum enhancements for deep reinforcement learning in large spaces. *PRX Quantum*, *2*, 010328.

Jerbi et al. (2021). Parametrized quantum circuits for reinforcement learning. https://www.tensorflow.org/quantum/tutorials/quantum_reinforcement_learning

Johnson, M. W., Amin, M. H., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A. J., Johansson, J., Bunyk, P., et al. (2011). Quantum annealing with manufactured spins. *Nature*, *473*(7346), 194–198.

Kandala, A., Mezzacapo, A., Temme, K., Takita, M., Brink, M., Chow, J. M., & Gambetta, J. M. (2017). Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, *549*(7671), 242–246.

Liu, J.-G., & Wang, L. (2018). Differentiable learning of quantum circuit born machines. *Physical Review A*, *98*(6), 062324.

Liu, Y., Arunachalam, S., & Temme, K. (2021). A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, *17*(9), 1013–1017.

Lockwood, O., & Si, M. (2020). Reinforcement learning with quantum variational circuit. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *16*(1), 245–251.

Mirowski, P., Grimes, M., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., Simonyan, K., Zisserman, A., Hadsell, R., et al. (2018). Learning to navigate in cities without a map. *Advances in Neural Information Processing Systems*, *31*, 2419–2430.

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Physical Review A*, *98*(3), 032309.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Neukart, F., Von Dollen, D., Seidel, C., & Compostella, G. (2018). Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces. *Frontiers in Physics*, *5*, 71.

Nielsen, M. A., & Chuang, I. L. (2000). *Quantum computation and quantum information*. Cambridge University Press.

OpenAI. (2020). Leaderboard of openai gym environments.

Otterbach, J., Manenti, R., Alidoust, N., Bestwick, A., Block, M., Bloom, B., Caldwell, S., Didier, N., Fried, E. S., Hong, S., et al. (2017). Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*.

Paparo, G. D., Dunjko, V., Makmal, A., Martin-Delgado, M. A., & Briegel, H. J. (2014). Quantum speedup for active learning agents. *Physical Review X*, *4*(3), 031002.

Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., & Latorre, J. I. (2020). Data re-uploading for a universal quantum classifier. *Quantum*, *4*, 226.

Pérez-Salinas, A., López-Núñez, D., Garcıéa-Sáez, A., Forn-Dıéaz, P., & Latorre, J. I. (2021). One qubit as a universal approximant. *Physical Review A*, *104*(1), 012405.

Peters, E., Caldeira, J., Ho, A., Leichenauer, S., Mohseni, M., Neven, H., Spentzouris, P., Strain, D., & Perdue, G. N. (2021). Machine learning of high dimensional data on a noisy quantum processor. *arXiv preprint arXiv:2101.09581*.

Preskill, J. (2018). Quantum computing in the nisq era and beyond. *Quantum*, *2*, 79.

Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. *Physical Review A*, *99*(3), 032331.

Schuld, M., Bocharov, A., Svore, K. M., & Wiebe, N. (2020). Circuit-centric quantum classifiers. *Physical Review A*, *101*(3), 032308.

Schuld, M., & Killoran, N. (2019). Quantum machine learning in feature hilbert spaces. *Physical review letters*, *122*(4), 040504.

Schuld, M., Sweke, R., & Meyer, J. J. (2021). Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, *103*(3), 032430.

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, *41*(2), 303–332.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354.

Skolik, A., Jerbi, S., & Dunjko, V. (2021). Quantum agents in the gym: A variational quantum algorithm for deep q-learning. *arXiv preprint arXiv:2103.15084*.

Sutton, R. S., Barto, A. G. et al. (1998). *Reinforcement learning: An introduction*.

Suzuki, Y., Kawase, Y., Masumura, Y., Hiraga, Y., Nakadai, M., Chen, J., Nakanishi, K. M., Mitarai, K., Imai, R., Tamiya, S., et al. (2020). Qulacs: A fast and versatile quantum circuit simulator for research purpose. *arXiv preprint arXiv:2011.13524*.

Sweke, R., Seifert, J.-P., Hangleiter, D., & Eisert, J. (2021). On the quantum versus classical learnability of discrete distributions. *Quantum*, *5*, 417.

Weng, L. (2018). Policy gradient algorithms.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229–256.

Wu, S., Jin, S., Wen, D., & Wang, X. (2020). Quantum reinforcement learning in continuous action space. *arXiv preprint arXiv:2012.10711*.

Zhu, D., Linke, N. M., Benedetti, M., Landsman, K. A., Nguyen, N. H., Alderete, C. H., Perdomo-Ortiz, A., Korda, N., Garfoot, A., Brecque, C., et al. (2019). Training of quantum circuits on a hybrid quantum computer. *Science advances*, *5*(10), eaaw9918.

A. Appendix

Outline The Supplementary Material is organized as follows. In Appendix A.1, we derive the expression of the log-policy gradient for SOFTMAX-PQCs presented in Lemma 1. In Appendix A.2, we prove Lemmas 2 and 3 on the efficient policy sampling and the efficient estimation of the log-policy gradient for SOFTMAX-PQC policies. In Appendix A.3, we clarify the role of the trainable observables in our definition of SOFTMAX-PQC policies. In Appendix A.4, we give a specification of the environments considered in our numerical simulations, as well the hyperparameters we used to train all RL agents. In Appendix A.5, we present additional plots and numerical simulations that help our understanding and visualization of PQC polices. In Appendix A.6, we give a succinct description of the DLP classification task of Liu *et al.* In Appendices A.7 to A.9, we prove our main Theorem 1 on learning separations in DLP environments. In appendix A.10, we construct PQC agents with provable guarantees of solving the DLP environments, stated and proven in Theorem 3.

A.1. Derivation of the log-policy gradient

For a SOFTMAX-PQC defined in Def. 1, we have:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) &= \nabla_{\boldsymbol{\theta}} \log e^{\beta \langle O_a \rangle_{s,\boldsymbol{\theta}}} - \nabla_{\boldsymbol{\theta}} \log \sum_{a'} e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}} \\ &= \beta \nabla_{\boldsymbol{\theta}} \langle O_a \rangle_{s,\boldsymbol{\theta}} - \sum_{a'} \frac{e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}} \beta \nabla_{\boldsymbol{\theta}} \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}}{\sum_{a''} e^{\beta \langle O_{a''} \rangle_{s,\boldsymbol{\theta}}}} \\ &= \beta \left(\nabla_{\boldsymbol{\theta}} \langle O_a \rangle_{s,\boldsymbol{\theta}} - \sum_{a'} \pi_{\boldsymbol{\theta}}(a'|s) \nabla_{\boldsymbol{\theta}} \langle O_{a'} \rangle_{s,\boldsymbol{\theta}} \right).\end{aligned}$$

A.2. Efficient implementation of SOFTMAX-PQC policies

A.2.1. Efficient approximate policy sampling

In this section we prove Lemma 2, restated below:

Lemma 2. *For a* SOFTMAX-PQC *policy $\pi_{\boldsymbol{\theta}}(a—s)$ 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 $\widetilde{\pi}_{\boldsymbol{\theta}}(a|s) = \text{softmax}_\beta(\langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}})$ has total variation distance $\mathcal{O}(\beta\varepsilon)$ to $\pi_{\boldsymbol{\theta}} = \text{softmax}_\beta(\langle O_a \rangle_{s,\boldsymbol{\theta}})$. 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 $\left\{ \langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}} \right\}_{1 \le a \le |A|}$, obtained all to additive error ε, i.e.,

$$\left| \langle \widetilde{O_a} \rangle_{s,\boldsymbol{\theta}} - \langle O_a \rangle_{s,\boldsymbol{\theta}} \right| \le \varepsilon, \quad \forall a$$

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:

$$\begin{aligned} \frac{e^{-\beta\varepsilon}e^{\beta\langle O_a\rangle_{s,\boldsymbol{\theta}}}}{e^{\beta\varepsilon}\sum_{a'}e^{\beta\langle O_{a'}\rangle_{s,\boldsymbol{\theta}}}} \leq & \frac{e^{\beta\langle \widetilde{O_a}\rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'}e^{\beta\langle \widetilde{O_{a'}}\rangle_{s,\boldsymbol{\theta}}}} \leq \frac{e^{\beta\varepsilon}e^{\beta\langle O_a\rangle_{s,\boldsymbol{\theta}}}}{e^{-\beta\varepsilon}\sum_{a'}e^{\beta\langle O_{a'}\rangle_{s,\boldsymbol{\theta}}}} \\ \Leftrightarrow \quad e^{-2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s) \leq & \quad \widetilde{\pi}_{\boldsymbol{\theta}}(a|s) \quad \leq e^{2\beta\varepsilon}\pi_{\boldsymbol{\theta}}(a|s). \end{aligned} \tag{A.1}$$

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 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. □

A.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 3. *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. (A.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:

$$\begin{aligned} e^{-2\beta\varepsilon'}\pi_{\boldsymbol{\theta}}(a|s)\left(\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} - \varepsilon\right) \leq \widetilde{\pi}_{\boldsymbol{\theta}}(a|s)\partial_i\langle\widetilde{O_a}\rangle_{s,\boldsymbol{\theta}} \leq e^{2\beta\varepsilon'}\pi_{\boldsymbol{\theta}}(a|s)\left(\partial_i\langle O_a\rangle_{s,\boldsymbol{\theta}} + \varepsilon\right) \\ \Rightarrow \quad e^{-2\beta\varepsilon'}\left(\sum_a x_{a,i} - \varepsilon\right) \leq \quad \sum_a \widetilde{x}_{a,i} \quad \leq e^{2\beta\varepsilon'}\left(\sum_a x_{a,i} + \varepsilon\right) \end{aligned}$$

where we summed the first inequalities over all a. Hence:

$$\begin{aligned} \left|\sum_a x_{a,i} - \sum_a \widetilde{x}_{a,i}\right| &\leq \left|e^{2\beta\varepsilon'}\left(\sum_a x_{a,i} + \varepsilon\right) - e^{-2\beta\varepsilon'}\left(\sum_a x_{a,i} - \varepsilon\right)\right| \\ &\leq \left|(e^{2\beta\varepsilon'} + e^{-2\beta\varepsilon'})\varepsilon + (e^{2\beta\varepsilon'} - e^{-2\beta\varepsilon'})\sum_a x_{a,i}\right| \\ &\leq \left|2\cosh(2\beta\varepsilon')\varepsilon + 2\sinh(2\beta\varepsilon')\sum_a x_{a,i}\right| \\ &\underset{\beta\varepsilon'\to 0^+}{=} \left|\varepsilon + 4\beta\varepsilon'\sum_a x_{a,i} + \mathcal{O}\left((\beta\varepsilon')^2\varepsilon\right) + \mathcal{O}\left((\beta\varepsilon')^3\right)\right|. \end{aligned} \tag{A.2}$$

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. (3.5)) formulation of $\partial_i \langle O_a\rangle$ for derivatives with respect to 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. (A.2), 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:

$$|\partial_i \log \pi_{\boldsymbol{\theta}}(a|s) - \partial_i \log \widetilde{\pi_{\boldsymbol{\theta}}}(a|s)| \underset{\beta\varepsilon\to 0^+}{\leq} 3\beta\varepsilon + \mathcal{O}(\beta\varepsilon^3) \quad \forall i$$ □

A.3. The role of trainable observables in SOFTMAX-PQC policies

In Sec. 3.2, 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.

A.3.1. Training the eigenbasis and the eigenvalues of an observable

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 $V_{\text{var}}^\dagger(\boldsymbol{\phi}_L)O_a V_{\text{var}}(\boldsymbol{\phi}_L)$.

A.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 (Goto et al., 2021)), 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 (Pérez-Salinas et al., 2021).

A.4. Environments specifications and hyperpameters

In Table 1, we present a specification of the environments we consider in our numerical simulations. These are standard benchmarking environments from the OpenAI Gym library (Brockman et al., 2016), described in Ref. (OpenAI, 2020), PQC-generated environments that we define in Sec. 5.2, and the CognitiveRadio environment of Ref. (Chen et al., 2020) that we discuss in Appendix A.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 1: Environments specifications. *The reward function of Mountaincar-v0 has been modified compared to the standard specification of OpenAI Gym (Brockman et al., 2016), similarly to Ref. (Duan et al., 2016).*

In Tables 2 and 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 (Google, 2018) in Python and simulated using a Qulacs backend (Suzuki et al., 2020) in C++. For the tutorial (Jerbi et al., 2021), the TensorFlow Quantum library (Broughton et al., 2020) 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.

A.5. Deferred plots and shape of policies learned by PQCs v.s. DNNs

A.5.1. Influence of architectural choices on RAW-PQC agents

In Fig. 6, we run a similar experiment to that of Sec. 4.2 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

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 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.*

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.

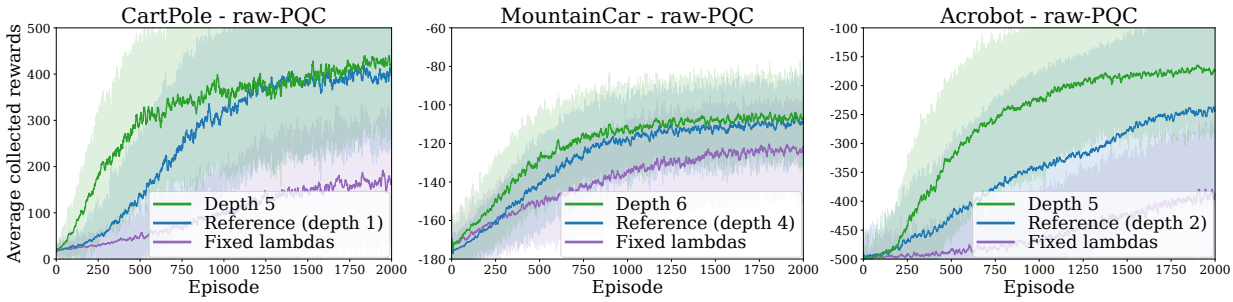

***Figure 6*: Influence of the model architecture for RAW-PQC agents.** *The blue curves in each plot correspond to the learning curves from Fig. 3 and are taken as a reference.*

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	$[\boldsymbol{w}_i \cdot (Z_0, \dots, Z_5)^T]_{1\leq i\leq 3}$	$\{90, 180\}$	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 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. 4.2, 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 2.*

A.5.2. Shape of the policies learned by PQCs v.s. DNNs

In CartPole-v1 The results of the Sec. 4 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. (3.3)). Typical instances of these functions are depicted in Figure 8. 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.* (Schuld et al., 2019) 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. 9 shows the analog results to Fig. 5 in the main text but with two different random initializations of the environment-generating PQC. Both confirm our observations. In Fig. 10, 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.

A.5.3. Additional numerical simulation on the CognitiveRadio environment

In a related work on value-based RL with PQCs, the authors of Ref. (Chen et al., 2020) 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. (Chen et al., 2020) 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. (Chen et al., 2020) in a policy-gradient setting, we test the performance of our SOFTMAX-PQC agents on this environment. We find numerically (see Fig. 7) that these achieve a very similar performance to the PQC agents of Ref. (Chen et al., 2020) on the same system sizes they consider ($n = 2$ to 5), using PQCs with the same scaling of number of parameters, i.e., $\mathcal{O}(n)$.

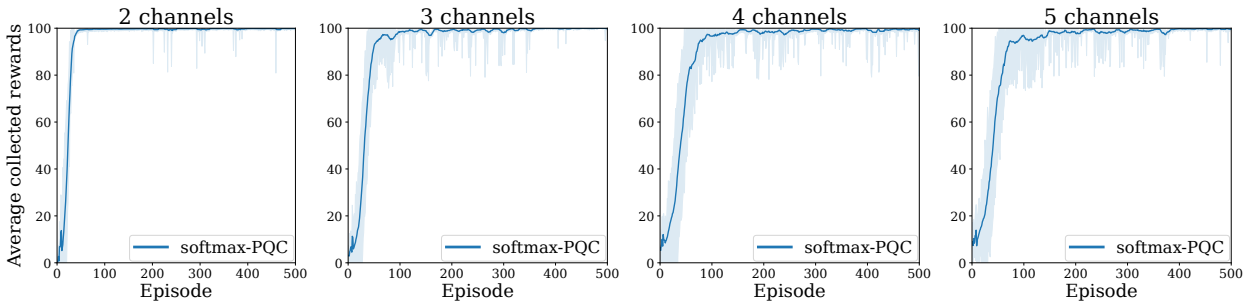

Figure 7: Performance of our SOFTMAX-PQC agents on the CognitiveRadio environment proposed in Ref. (Chen et al., 2020). *Average performance of 20 agents for system sizes (and number of qubits) $n = 2$ to 5.*

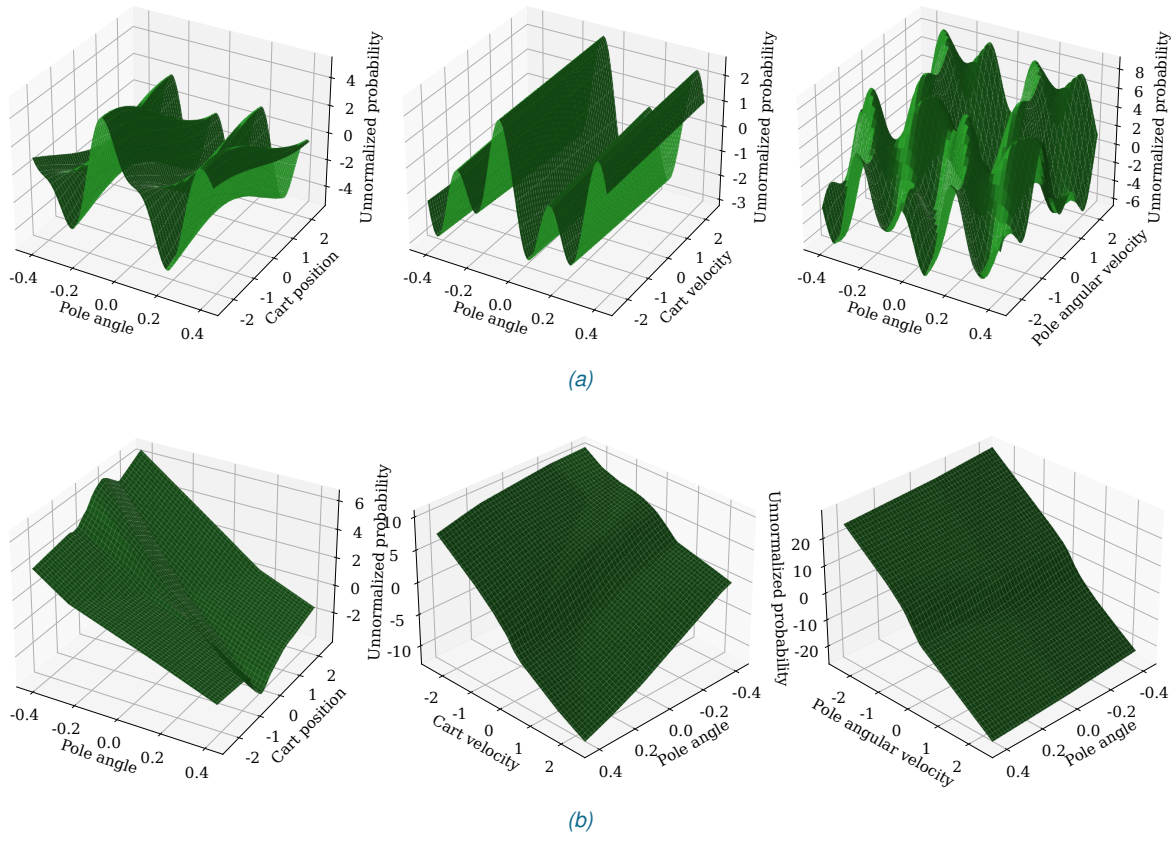

***Figure 8**: **Prototypical unnormalized policies learned by SOFTMAX-PQC agents and DNN agents in CartPole.** Due to the 4 dimensions of the state space in CartPole, we represent the unnormalized policies learned by (a) SOFTMAX-PQC agents and (b) DNN agents on 3 subspaces of the state space by fixing 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.*

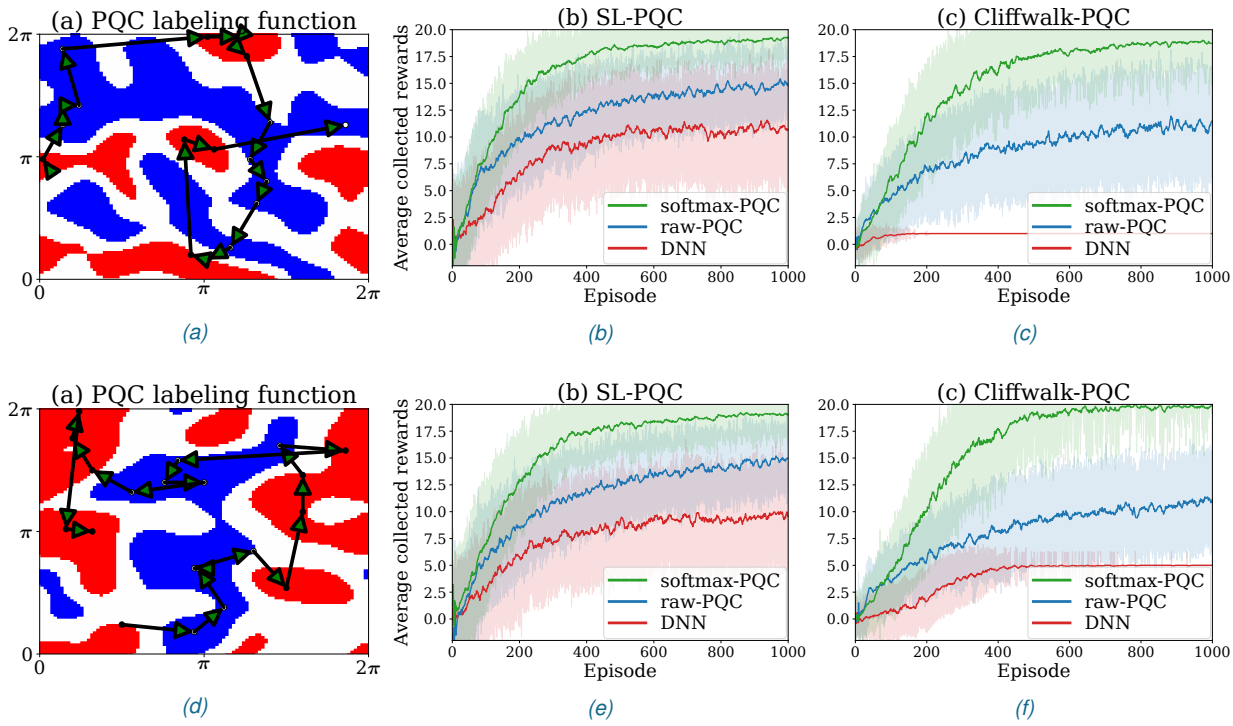

***Figure 9**: **Different random initializations of PQC-generated environments and their associated learning curves.** See Fig. 5 for details. The additional learning curves (20 agents per curve) of randomly-initialized RAW-PQC agents highlight the hardness of these environments for PQC policies drawn from the same family as the environment-generating PQCs.*

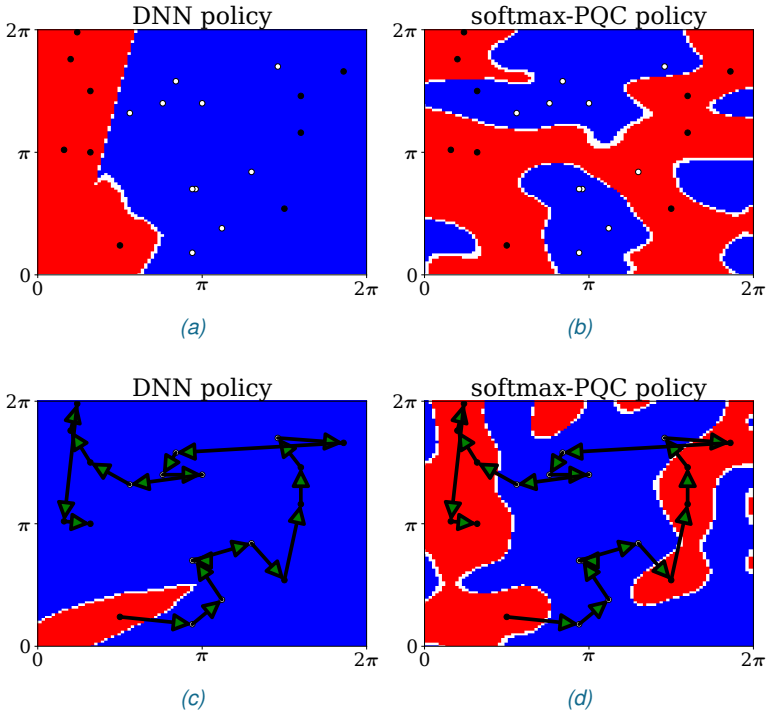

***Figure 10**: **Prototypical policies learned by SOFTMAX-PQC agents and DNN agents in PQC-generated environments.** All policies are associated to the labeling function of Fig. 9.d. Policies (a) and (b) are learned in the SL-PQC environment while policies (c) and (d) are learned in the Cliffwalk-PQC environment.*

A.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.* (Y. Liu et al., 2021) 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:

$$f_s(x) = \begin{cases} +1, & \text{if } \log_g x \in [s, s + \frac{p-3}{2}], \\ -1, & \text{otherwise.} \end{cases} \tag{A.3}$$

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. (Y. Liu et al., 2021) for a good visualization of all these objects.
The performance of a classifier f is measured in terms of its testing accuracy

$$\mathsf{Acc}_f(f_s) = \mathsf{Pr}_{x \sim \mathcal{X}}[f(x) = f_s(x)].$$

A.7. Proof of Theorem 1

In the following, we provide constructions of *a)* fully random, *b)* partially random and *c)* fully deterministic environments satisfying the properties of Theorem 1. 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 A.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 with respect to a labeling (or ground truth) function f^* is measured in terms of a testing accuracy

$$\mathsf{Acc}_f(f^*) = \frac{1}{|S|} \sum_s \mathsf{Pr}[f(s) = f^*(s)]. \tag{A.4}$$

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

$$\begin{aligned} V_\pi(s_0) &= \frac{1}{|S|} \sum_{s_0} (\pi(f^*(s_0)|s_0) - \pi(-f^*(s_0)|s_0)) \\ &= \frac{1}{|S|} \sum_{s_0} 2\pi(f^*(s_0)|s_0) - 1 \\ &= 2\mathsf{Acc}_\pi(f^*) - 1, \end{aligned}$$

which is trivially related to the testing accuracy of this policy on the classification task. Note that we also have $V_{\mathsf{rand}}(s_0) = 0$ and $V_{\mathsf{opt}}(s_0) = 1$.
Since these observations hold irrespectively of the labeling function f^*, we can show the following result:

Lemma 4 (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 A.6), for which classical hardness and quantum learnability holds. More specifically, the performance of any classical learner is upper bounded by $1/\mathsf{poly}(n)$, 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 $\mathsf{Acc}_\pi(f^*) \leq 1/2 + 1/\mathsf{poly}(n)$, and hence its value function would be $V_\pi(s_0) \leq 1/\mathsf{poly}(n)$.

For quantum learnability, we define an agent that first collects 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 $\mathcal{O}\left(\log(\delta^{-1})\right)$ 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 T, the only change is that the value function of any policy, and hence the bounds we achieve, get multiplied by a factor of $\frac{1-\gamma^T}{1-\gamma}$ for a discount factor $\gamma < 1$ and by a factor T for $\gamma = 1$.

Partially random: the Cliffwalk-DLP environment. One major criticism to the result of Lemma 4 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. (Dunjko et al., 2017), 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 (Sutton, Barto, et al., 1998). 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 5 (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 A.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 $T_S = (x_0, \ldots, x_{k-1})$ 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, we can already note that $V_\pi(x_0) = \pi(f^*(x_k)|T_S, x_k)$, that is, the probability of the policy correctly labeling the testing state x_k after having experienced the training states T_S. 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:

[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}.

Lemma 6 (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 A.9 for clarity.

By combining our three lemmas, and taking the weakest separation claim for the cases *ii)* and *iii)*, we get Theorem 1. 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. 5.2. 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.

A.8. Proof of Lemma 5

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.

A.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:

$$V_\pi(i) = \pi(i)\gamma\left((1-\delta)V_\pi(i+1) + \delta\frac{1}{p-1}\sum_{j=1}^{p-1}V_\pi(j)\right) - (1-\pi(i)) \tag{A.5}$$

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:

$$\frac{1}{p-1}\sum_{i=1}^{p-1} V_\pi(i) \leq \frac{1}{p-1}\sum_{i=1}^{p-1}\left(\pi(i)\frac{\gamma\delta}{p-1}\sum_{j=1}^{p-1} V_\pi(j) - (1-\pi(i))\right)$$
$$= \left(\frac{1}{p-1}\sum_{i=1}^{p-1}\pi(i)\right)\left(\frac{\gamma\delta}{p-1}\sum_{j=1}^{p-1} V_\pi(j) + 1\right) - 1$$

We note that the first factor is exactly the accuracy of the policy π on the classification task of Liu *et al.*:

$$\mathsf{Acc}_\pi(f^*) = \frac{1}{p-1}\sum_{i=1}^{p-1}\pi(i).$$

We hence have:

$$\frac{1}{p-1}\sum_{i=1}^{p-1} V_\pi(i) \leq \mathsf{Acc}_\pi(f^*)\left(\gamma\delta\frac{1}{p-1}\sum_{j=1}^{p-1} V_\pi(j) + 1\right) - 1$$

which is equivalent to:

$$\frac{1}{p-1}\sum_{i=1}^{p-1} V_\pi(i) \leq \frac{\mathsf{Acc}_\pi(f^*) - 1}{1 - \mathsf{Acc}_\pi(f^*)\gamma\delta}$$

when $\mathsf{Acc}_\pi(f^*)\gamma\delta < 1$.
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,

$$V_\pi(s_0) \leq \frac{\mathsf{Acc}_\pi(f^*) - 1}{1 - \mathsf{Acc}_\pi(f^*)\gamma\delta} \tag{A.6}$$

A.8.2. Lower bound on the value function

Again, by noting in Eq. (A.5) 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:

$$\begin{aligned} V_\pi(s_0) &= \frac{1}{p-1}\sum_{i=1}^{p-1} V_\pi(i) \\ &\geq \gamma\left(\frac{1-\delta}{p-1}\sum_{i=1}^{p-1} V_\pi(i+1) + \frac{\delta}{p-1}\sum_{j=1}^{p-1} V_\pi(j)\right) + \frac{1}{p-1}\sum_{i=1}^{p-1}\pi(i) - 1 \\ &= \gamma\left((1-\delta)V_\pi(s_0) + \delta V_\pi(s_0)\right) + \mathsf{Acc}_\pi(f^*) - 1 \\ &= \gamma V_\pi(s_0) + \mathsf{Acc}_\pi(f^*) - 1 \end{aligned}$$

by using the circular boundary conditions of the cliffwalk in the third line.
This inequality is equivalent to:

$$V_\pi(s_0) \geq \frac{\mathsf{Acc}_\pi(f^*) - 1}{1 - \gamma} \tag{A.7}$$

when $\gamma < 1$.

A.8.3. Bounds for classical hardness 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 5.

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):

$$\begin{aligned} V_{\text{rand}}(s_0) &= \frac{\gamma}{2}\left(\frac{1-\delta}{p-1}\sum_{i=1}^{p-1} V_{\text{rand}}(i+1) + \frac{\delta}{p-1}\sum_{j=1}^{p-1} V_{\text{rand}}(j)\right) - \frac{1}{2} \\ &= \frac{\gamma}{2} V_{\text{rand}}(s_0) - \frac{1}{2} = -\frac{1}{2-\gamma} \end{aligned}$$

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 $V_{\text{opt}} = 0$.

A.8.3.1. 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:

$$g(x,\delta,\gamma) = \frac{x-1}{1-\delta\gamma x} + \frac{1}{2-\gamma} \tag{A.8}$$

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 with respect to 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 with respect to 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 with respect to 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. (A.6), we hence have classical hardness $\forall\ (\delta, \gamma) \in [\delta_0, 1] \times [0, \gamma_1]$.

A.8.3.2. 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. (A.7):

$$V_\pi(s_0) \geq -0.1 = V_{\text{opt}} - 0.1$$

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.

A.9. Proof of Lemma 6

A.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 $\text{Pr}\left[f_c(x_k) = f^*(x_k)\right] \geq 1 - \varepsilon$.

Define $P(T) = \text{Pr}[\text{T} = T]$ and $P(x_k) = \text{Pr}[\text{x}_\text{k} = x_k]$ 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:

$$\begin{aligned}
\mathbb{E}_{P(T)}\left[\text{Acc}_{f_c}(T)\right] &= \sum_T P(T) \sum_{x_k} P(x_k) \text{Pr}\left[f_c(x_k) = f^*(x_k)|T, x_k\right] \\
&= \sum_{T, x_k} P(T, x_k) \text{Pr}\left[f_c(x_k) = f^*(x_k)|T, x_k\right] \\
&\geq \sum_{T, x_k} P(T, x_k) \text{Pr}\left[\text{success}|T, x_k\right] \times \text{Pr}\left[f_c(x_k) = f^*(x_k)|T, x_k, \text{success}\right] \\
&\geq (1-\delta)(1-\varepsilon)
\end{aligned}$$

since $\text{Pr}\left[f_c(x_k) = f^*(x_k)|T, x_k\right] \geq 1 - \varepsilon$ for instances in S_δ and $\sum_{T, x_k} P(T, x_k) \text{Pr}\left[\text{success}|T, x_k\right] \geq 1 - \delta$ by definition.
In the following, we set $1 - \varepsilon = 0.999$ and $1 - \delta \geq 0.845$ (the reason for this becomes apparent below), such

that:

$$\mathbb{E}_{P(T)}\left[\mathsf{Acc}_{f_c}(T)\right] \geq 0.844155 > \frac{5}{6} + \frac{1}{96} \tag{A.9}$$

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 $\mathsf{Acc}_{f_c}(T) \geq \frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$ 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 $\mathcal{T}_{\mathsf{succ}}$ the subset of all instances of training states $\mathcal{T} = \{T\}$ for which $\mathsf{Acc}_{f_c}(T) \geq \frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$. We prove by contradiction that $|\mathcal{T}_{\mathsf{succ}}| \geq \frac{2|\mathcal{T}|}{3}$:
Assume $|\mathcal{T}_{\mathsf{succ}}| < \frac{2|\mathcal{T}|}{3}$, then

$$\begin{aligned}
\mathbb{E}_{P(T)}\left[\mathsf{Acc}_{f_c}(T)\right] &= \sum_T P(T)\mathsf{Acc}_{f_c}(T) \\
&= \frac{1}{|\mathcal{T}|}\left(\sum_{T\in\mathcal{T}_{\mathsf{succ}}} \mathsf{Acc}_{f_c}(T) + \sum_{T\notin\mathcal{T}_{\mathsf{succ}}} \mathsf{Acc}_{f_c}(T)\right) \\
&< \frac{|\mathcal{T}_{\mathsf{succ}}|}{|\mathcal{T}|} \times 1 + \frac{|\mathcal{T}| - |\mathcal{T}_{\mathsf{succ}}|}{|\mathcal{T}|}\left(\frac{1}{2} + \frac{1}{\mathsf{poly}(n)}\right) \\
&< \frac{5}{6} + \frac{1}{3\mathsf{poly}(n)} < 0.844155
\end{aligned}$$

for large enough n, in contradiction with Eq. (A.9).

Hence, with probability at least $2/3$ over the choice of training states and the randomness of the learning algorithm, our constructed classifier has accuracy $\mathsf{Acc}_{f_c}(T) \geq \frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$. By using Theorem 8, Remark 1 of Liu *et al.*, this is sufficient to construct an efficient classical algorithm that solves DLP.

A.9.2. Proof of quantum learnability

Using the learning algorithm of Liu *et al.*, we can construct a quantum classifier that achieves accuracy $\mathsf{Acc}_q(T) \geq 0.99$ with probability at least $2/3$ over the randomness of the learning algorithm and the choice of training states T, of length $|T| = \mathsf{poly}(n)$. Now define instead training states T of length $|T| = M\mathsf{poly}(n)$, for $M = \mathcal{O}\left(\log(\delta'^{-1})\right)$ (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 $\mathsf{Acc}_{f_q}(T) \geq 0.99$ 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 $\mathsf{Acc}_{f_q}(T) \geq 0.99$, we have:

$$\begin{aligned}
&\sum_{T,x_k} P(T, x_k)\mathsf{Pr}\left[V_q(x_0) = 1|T, x_k\right] \\
&\qquad \geq \sum_T P(T)\sum_{x_k} P(x_k)\mathsf{Pr}\left[\mathsf{success}|T\right] \times \mathsf{Pr}\left[V_q(x_0) = 1|T, x_k, \mathsf{success}\right] \\
&\qquad = \sum_T P(T)\mathsf{Pr}\left[\mathsf{success}|T\right]\sum_{x_k} P(x_k) \times \mathsf{Pr}\left[f_q(x_k) = f^*(x_k)|T, x_k, \mathsf{success}\right] \\
&\qquad = \sum_T P(T)\mathsf{Pr}\left[\mathsf{success}|T\right]\mathsf{Acc}_{f_q}(T) \\
&\qquad \geq (1 - \delta') \times 0.99
\end{aligned}$$

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.

A.10. Construction of a 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.* (Y. Liu et al., 2021) (see Appendix A.6), and can hence also be used as a learning model in the DLP environments defined in Sec. 5.1.

A.10.1. Implicit v.s. 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. (Schuld & Killoran, 2019): 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

$$K(x_i, x_j) = |\langle\phi(x_i)|\phi(x_j)\rangle|^2 \tag{A.10}$$

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 $|\langle \boldsymbol{w}|\phi(x)\rangle|^2 + b$ gives the label of x.
In the explicit model, the classifier is instead obtained by training a parametrized $|\boldsymbol{w_\theta}\rangle$. 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 $|\langle \boldsymbol{w_\theta}|\phi(x)\rangle|^2$.

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 A.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.

A.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

$$|\phi(x)\rangle = \frac{1}{\sqrt{2^k}} \sum_{i=0}^{2^k-1} |x \cdot g^i\rangle \tag{A.11}$$

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 (Y. Liu et al., 2021).

By noting that every labeling functions $f_s \in \mathcal{C}$ to be learned (see Eq. (A.3)) 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:

$$|\phi_{s'}\rangle = \frac{1}{\sqrt{(p-1)/2}} \sum_{i=0}^{(p-3)/2} \left|g^{s'+i}\right\rangle. \tag{A.12}$$

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 $|\langle\phi(x)|\phi_s\rangle|^2$ 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 $|\langle\phi(x)|\phi_{s'}\rangle|^2$ can be measured:

$$h_{s'}(x) = \begin{cases} 1, & \text{if } |\langle\phi(x)|\phi_{s'}\rangle|^2/\Delta \geq 1/2, \\ -1, & \text{otherwise} \end{cases} \tag{A.13}$$

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 $\langle O\rangle_{x,\boldsymbol{\theta}} = |\langle\phi(x)|\phi_{s'}\rangle|^2$. To do this, we note that, for $|\phi_{s'}\rangle = \hat{V}(s')\,|0\rangle$, the following equality holds:

$$\begin{aligned} |\langle\phi(x)|\phi_{s'}\rangle|^2 &= \left|\langle 0^{\otimes n}|\,\hat{V}(s')^\dagger U(x_i)\,|0^{\otimes n}\rangle\right|^2 \\ &= \text{Tr}\left[|0^{\otimes n}\rangle\langle 0^{\otimes n}|\,\rho(x,s')\right] \end{aligned}$$

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 $|\langle\phi(x)|\phi_{s'}\rangle|^2$ 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. (A.13)).

A.10.3. Noisy classifier

In practice, there will be noise associated with the estimation of the inner products $|\langle\phi(x)|\phi_{s'}\rangle|^2$, 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 $|\langle\phi(x_i)|\phi_{s'}\rangle|^2 + e_{is'}$. This random variable satisfies the following equations:

$$\begin{cases} e_{is'} \in [-\Delta, \Delta] \\ \mathbb{E}[e_{is'}] = 0 \\ \text{Var}[e_{is'}] \leq 1/R \end{cases}$$

where R is the number of circuit evaluations used to estimate the inner product. We hence end up with a noisy classifier:

$$\widetilde{h}_{s'}(x_i) = \begin{cases} 1, & \text{if } \left(|\langle\phi(x_i)|\phi_{s'}\rangle|^2 + e_{is'}\right)/\Delta \geq 1/2, \\ -1, & \text{otherwise} \end{cases}$$

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:

$$\text{Pr}\left(|e_{is'}| \geq \frac{\Delta}{2}\right) \leq \frac{4}{\Delta^2 R} \tag{A.14}$$

since $\mathbb{E}[e_{is'}] = 0$ and $\text{Var}[e_{is'}] \leq 1/R$.

[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.

A.11. Proof of trainability of our PQC agent in the SL-DLP environment

In this Appendix, we describe an optimization algorithm to train the variational parameter $g^{s'}$ of the PQC classifier we defined in Appendix A.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 3. *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 $t \geq \max\left\{3\log_n(8/\delta), \log_n(16/\varepsilon)\right\}$ in the definition of Δ, and a number of circuit 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 3, 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 7), and second, that this algorithm also leads to s' close to the optimal s in log-space with high probability (Lemma 8).

Lemma 7. *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. (A.14). 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:

$$\begin{aligned}\Pr\left(\operatorname*{argmin}_{x\in X} \widetilde{\mathcal{L}}(\log_g(x)) = \operatorname*{argmin}_{x\in X} \mathcal{L}(\log_g(x))\right) &\geq 1 - \Pr\left(\bigcup_{\substack{i,j\\i\neq j}} x_i \notin I_{\log_g(x_j)} \cup \bigcup_{i,s'} |e_{i,s'}| \geq \frac{\Delta}{2}\right)\\ &\geq 1 - \delta/2\end{aligned}$$

□

Lemma 8. *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 7 and Lemma 8 can be used to prove:

Corollary 1. *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 1 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 3.