Parameterised Quantum Policies for Reinforcement Learning

Shiven Tripathi
Department of Electrical Engineering
IIT Kanpur
shiven@iitk.ac.in

Somya Lohani
Department of Computer Science
IIT Kanpur
somyaloh@iitk.ac.in

1 Background

1.1 Machine Learning

Machine learning aims to solve the problem of finding patterns based on large sets of data inputs which might be labelled or through other information signals which are used to train a model to solve a task like classification, generation etc. Supervised learning uses data samples for which ground truth label values are available, whereas, in unsupervised learning, unlabelled data is leveraged. Recently a lot of progress in ML has been achieved by leveraging deep neural networks, which can be trained to perform well on complex tasks by learning informative latent representations over input samples. A growing problem of such networks is the vast amount of computing which is required in their training and inference.

1.2 Reinforcement Learning

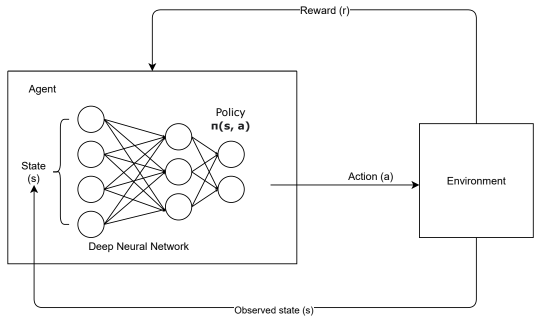

Figure 1: Architecture of Deep Reinforcement Learning

A standard formulation for RL is based on Markov decision processes with a defined state space, action space defined for each state, state transition probabilities corresponding to the action taken on a state, a reward function corresponding to taking action on a state and an objective function which signals whether the desired goal of the actions on the MDP has been achieved. RL algorithms used to solve these MDPs can be categorised into value-based or policy-based, on the basis of which function the optimisation or learning is done. The goal of any RL algorithm is to learn a policy which is mapping to actions for each state, which correlates most strongly to the completion of the attainment of the MDP. Deep neural network-based models have shown tremendous progress recently, helping achieve superhuman capability in games like chess and Go, which have state spaces exceeding the number of atoms in the universe.

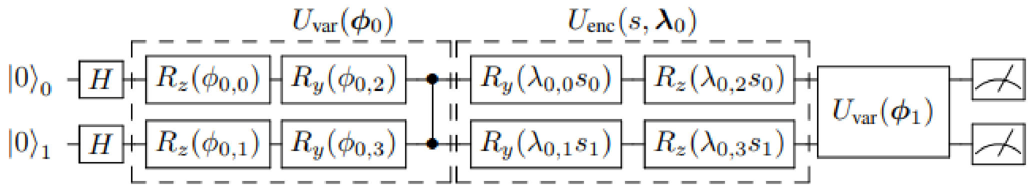

Figure 2: PQC architecture for $n = 2$ qubits and depth $D_{enc} = 1$ (from Jerbi et. al)

1.3 Parameterised Quantum Circuits

Near-term quantum computers consist of few (less than a hundred) qubits which behave noisily, resulting in large gate circuits being infeasible. To solve this hardware problem, hybrid quantum computing is evolving, which leverages some classical processing steps along with quantum computation. Recent attempts to combine quantum computing principles with machine learning have resulted in the development of parameterised quantum circuits. These consist of quantum algorithms which depend on certain free parameters, which can be tuned to optimise specific loss or cost metrics associated with the output function that the circuit computes. The key intuition behind this approach is that by designing certain sub-parts of a problem classically, the need for quantum resources is drastically reduced and can be leveraged solely for the classically intractable part of the problem.

2 Contributions

The use of policies based on PQCs is shown to solve classical RL tasks. By building environments, the solution to which requires solving the discrete logarithm problem, the quantum advantage is demonstrated in RL. Previous work has explored PQCs as value function approximators rather than direct policy functions. Also, this work solves the MDP for a classical state, showing more applicability to practical problems. The extreme amount of computational resources which classical RL algorithm leverage to solve even standard benchmark tasks makes it one of the most promising fields to show a quantum advantage. This work shows a quantum RL formulation using PQCs, which is able to handle standard RL benchmarks well, along with theoretically proven and empirically verified tasks which would be very tough for classical agents.

3 Methods

3.1 Circuit Learning

This is the standard learning algorithm used for parameterised quantum circuits which are trained to learn an objective. We use this learning rule in addition to the REINFORCE learning rule (omitted here) used in reinforcement learning. This combines the quantum and the classical learning parts of the quantum reinforcement procedure (see Figure 2).

1. Encode input data $\{x_i\}$ into some quantum state $|\psi_{in}(x_i)\rangle$ by applying a unitary input gate $U(x_i)$ to initialized qubits $|0\rangle$
2. Apply a θ-parameterised unitary $U(\theta)$ to the input state and generate an output state $|\psi_{out}(x_i, \theta)\rangle = U(\theta)\,|\psi_{\text{in}}(x_i)\rangle$
3. Measure the expectation values of some chosen observables. Specifically, we use a subset of Pauli operators $\{B_j\} \subset \{I, X, Y, Z\}^{\otimes N}$. Using some output function F, output $y_i = y(x_i, \theta)$ is defined to be $y(x_i, \theta) \equiv F(\{\langle B_j(x_i, \theta)\rangle\})$.
4. Minimize the cost function $L(f(x_i), y(x_i, \theta))$ of the teacher $f(x_i)$ and the output y_i, by tuning the circuit parameters θ iteratively.
5. Evaluate the performance by checking the cost function with respect to a data set that is taken independently from the training one.

3.1.1 Parameter Shift Rules

The algorithm to get partial derivatives using perturbed instances of parameters in parameterised quantum circuits are called parameter shift rules and were first introduced by Mitarai et al. (2018).

The underlying intuition is that the same circuit is used to compute a function (quantum) and the gradient of the function (quantum).

For any quantum circuit which has been created as a combination of unitaries, the end function computed can be written as a product of unitaries. For all unitary gates, we can write them as $U_j(\gamma_j) = \exp(i\gamma_j H_j)$, where H_j is a Hermitian operator and γ_j is a parameter specifying the gate operation. We shall now see the derivation for a singly parameterized gate, considering the partial derivative with respect to the parameter θ_i, where this parameter is associated with the gate $U_i(\theta_i)$. To simplify our calculations, we only consider the input and this parameterised gate, as mentioned above.

$$\begin{aligned} f(x;\theta_i) &= \langle 0|U_0^\dagger(x)U_i^\dagger(\theta_i)\hat{B}U_i(\theta_i)U_0(x)|0\rangle \\ &= \langle x|U_i^\dagger(\theta_i)\hat{B}U_i(\theta_i)|x\rangle. \end{aligned}$$

We rewrite $U_i^\dagger(\theta_i)\hat{B}U_i(\theta_i) = \mathcal{M}_{\theta_i}(\hat{B})$. Calculating the partial derivative with respect to the chosen parameter:

$$\nabla_{\theta_i} f(x;\theta_i) = \langle x|\nabla_{\theta_i}\mathcal{M}_{\theta_i}(\hat{B})|x\rangle \in \mathbb{R}.$$

From the result of Mitarai et al., we know that the gradient of the substituted transformation, $\mathcal{M}_{\theta_i}$, can be written as a linear combination of the same transformation but with different parameters. These different parameters are the same perturbed values we had in consideration earlier and formed our parameter shift rules.

$$\nabla_{\theta_i}\mathcal{M}_{\theta_i}(\hat{B}) = c[\mathcal{M}_{\theta_i+s}(\hat{B}) - \mathcal{M}_{\theta_i-s}(\hat{B})]$$

3.2 Parameterised Quantum Policies

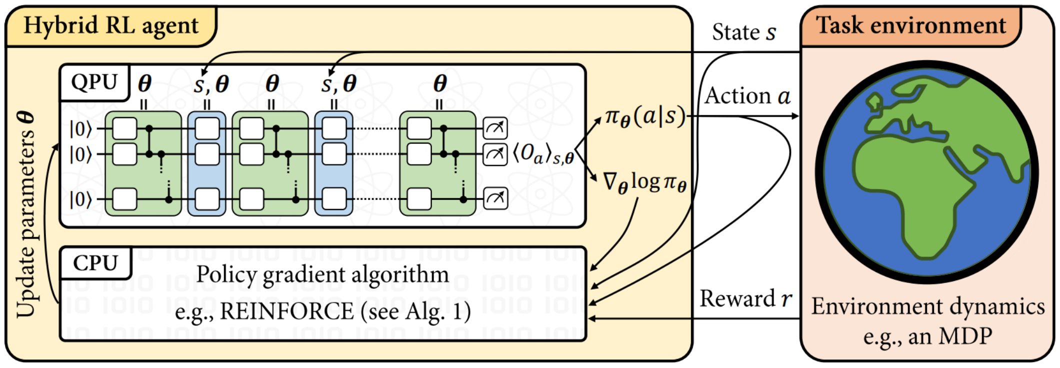

Figure 3: Pipeline for RL of parameterised Quantum Policies (from Jerbi et al.)

The main step towards learning parameterised quantum policies for our reinforcement learning task is using a parameterised quantum circuit, which is composed of unitaries ($U(s,\theta)$) acting on a fixed n qubit state. These PQC are composed of alternating blocks D_{enc} and U_{enc} blocks which essentially comprise of single-qubit rotations and entangling C-Z gates. Based on how we use the final quantum states, we define two different kinds of policies as outlined below.

3.2.1 RAW-PQC Policy

For an action space of our environment, we can partition the Hilbert space represented by our quantum state of n qubits into disjoint subspaces, each corresponding to an action. Further, a projection is associated with each of these subspaces, the measurement which defines our policy:

$$\pi_\theta(a \mid s) = \langle P_a\rangle_{s,\theta}$$

3.2.2 SOFTMAX-PQC Policy

A drawback of the above policy formulation is that the policy can't be tweaked using more information about the environment as external supervision to improve learning. We can fix this by adding additional learnable parameters to the RAW PQC Policy, converting it to a SOFTMAX-PQC Policy as follows:

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

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

3.3 PQC Classifier

The PQC classifier used to solve environments which have embedded supervised learning tasks requiring a solution to a DLP is designed 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 $V(\theta)$ parametrised by a vector θ. The resulting quantum state is then used to measure the expectation value $\langle O \rangle_{x,\theta}$ of an observable O, to be defined. Using the feature encoding unitary of Liu et al., features of the following form are created:

$$|\phi(x)\rangle = \frac{1}{\sqrt{2^k}} \sum_{i=0}^{2^k-1} \left| x \cdot g^i \right\rangle$$

for $k = n - t\log(n)$, where t is a constant defined under noise considerations. Intuitively observe that this is equivalent to a superposition of an exponentiated image of an interval of integers in log space. From this, it immediately follows that learning any labelling function for these integers under such a function (more detail later in SL-DLP) is creating decision boundaries which divide hyperplanes in log space.

By noting that every labelling functions $f_s \in \mathcal{C}$ to be learned is separating two equally sized intervals of $\log\left(\mathbb{Z}_p^*\right)$, 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 to separate hyperplanes that are normal to quantum states of the form:

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

Under the assumptions that we can measure overlap ($|\langle \phi(x) \mid \phi_{s'} \rangle|^2$) with our boundary part of the hyperplane, we can construct a classifier of the form:

$$h_{s'}(x) = 1, if\, |\langle \phi(x) \mid \phi_{s'} \rangle|^2 / \Delta \geq 1/2; -1, otherwise$$

For input points x such that $\log_g(x)$ is away from some separating regions around s and $s + \frac{p-3}{2}$, observe that the inner product $|\langle \phi(x) \mid \phi_s \rangle|^2$ is either $\Delta = \frac{2^{k+1}}{p-1}$ or 0 , whenever x is labeled $+1$ or -1 by f_s, respectively, which justifies the construction of our classifier. It can be easily verified that this classifier has an accuracy of $1 - \Delta$ whenever $s' = s$.

Since we need to use the observation signal somehow to train our circuit to learn the right parameters, we design the composition of our parametric part $V(\theta)$ and measurement O such that they result in expectation values of the form $\langle O \rangle_{x,\theta} = |\langle \phi(x) \mid \phi_{s'} \rangle|^2$. Observing that from the following equality, we can get our measurement and parametric part for the circuit: $|\phi_{s'}\rangle = \hat{V}(s') |0\rangle$, the following equality holds:

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

Hence, an obvious choice of variational circuit is $V(\theta) = \hat{V}(s')$, combined with a measurement operator $O = |0^{\otimes n}\rangle \langle 0^{\otimes n}|$.

4 Results

4.1 Benchmark Environments

On cart pole and frozen lake standard benchmark environments provided by OpenAI, solutions are provided by the PQC agent. We find that satisfactory performance can be achieved even with a low number of qubits, further validating the hypothesis that near-term quantum approaches can also be leveraged for practical solutions. This performance is achieved with interaction with the classical environment, and quantum processing is only used for policy determination.

4.2 Quantum Advantage

Discrete logarithm problem is classically intractable but has efficient solutions using quantum approaches. By designing environments which are modifications of standard benchmarks but have supervised learning based on solutions of DLP, it can be theoretically proven to be very tough for classical systems. It is demonstrated that using a quantum parameterised circuit, such problems can yield solutions for such an environment. For example, for the cliff walk environment adding a temporal structure, an agent, if it assigns the correct temporal value, moves to the intended fixed state and receives a reward. Otherwise, the episode terminates.
We start by designing a function which has DLP embedded as follows. For 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 $\{g^y, y \in \mathbb{Z}_{p-1}\} = \mathbb{Z}_p^*$).

$$f_s(x) = \{+1, if \log_g x \in \left[s, s + \frac{p-3}{2}\right], -1, otherwise.$$

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.

4.2.1 SL-DLP

We construct an RL environment where the above supervised learning task is trivially embedded into the environment. For a given class, the states are data points, and the action is a guess on the label of the data point, with the reward being related to whether the agent gets it right.

Proof of Classical Hardness: For the above classification task, the performance measured in terms of testing accuracy is:

$$Acc_f\left(f^*\right) = \frac{1}{|S|} \sum_s Pr\left[f(s) = f^*(s)\right]$$

For the MDP associated with this classification task and length-1 episodes of interaction, the value function of any policy $\pi(a \mid s)$ is given by

$$\begin{aligned} V_\pi\left(s_0\right) &= \frac{1}{|S|} \sum_{s_0} \left(\pi\left(f^*\left(s_0\right) \mid s_0\right) - \pi\left(-f^*\left(s_0\right) \mid s_0\right)\right) \\ &= \frac{1}{|S|} \sum_{s_0} 2\pi\left(f^*\left(s_0\right) \mid s_0\right) - 1 \\ &= 2Acc_\pi\left(f^*\right) - 1, \end{aligned}$$

which is trivially related to the testing accuracy of this policy on the classification task. Note that we also have $V_{rand}\left(s_0\right) = 0$ and $V_{opt}\left(s_0\right) = 1$.

Since for a learner to perform well in such an RL environment would directly imply performing well at the classification task, we can conclude that this task is classically hard.

Proof of Quantum Learnability: 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 action $+1$, from which the label $f^*\left(s_0\right)$ can be inferred), and use Theorem 2 of Liu et al. to train a classifier that has test accuracy at least 0.99 with probability at least $2/3$ (this process can be repeated $\mathcal{O}\left(\log\left(\delta^{-1}\right)\right)$ times to increase this probability to $1-\delta$ via majority voting). This classifier has a value function $V_\pi\left(s_0\right) \geq 0.98$. Such a value function is enough for us to say that a quantum learner agent can solve the environment.

4.2.2 Cliffwalk DLP

While the previous environment readily demonstrates quantum advantage, it is still a purely random environment in which anyway the performance of a classical agent is limited. Our next environment, which is a variant of the cliffwalk environment, has some deterministic parts. In cliffwalk environments, all states are ordered in a chain, and some actions lead to immediate death and episode termination, while others lead to the next state in the chain. It is demonstrated using numerical results a quantum separation when an agent has to solve this environment along with an additional constraint of assigning the correct solution to DLP in picking state transitions.

5 Limitations

The results are shown by implementing a quantum processing unit on a simulator. It is highly likely that some of the learning of any such model would be hindered in a real quantum system which would be noisy. The discrete logarithm problem embedded environments largely were just theoretical constructs useful to demonstrate some quantum advantage, but it would remain unclear if this progress directly leads to the practical application of quantum RL.

6 Discussion

Somewhat due to the lack of informativeness of the training reward signals, or the sheer large state space, we find that classical RL algorithms tend to be very resource intensive. A step towards the demonstration of quantum advantage in RL through certain constructed task environments could potentially reduce the resource demands while enabling solutions to even more complex problems, which can be formulated as an MDP.

7 References

Jerbi, S., Gyurik, C., Marshall, S. C., Briegel, H. J., Dunjko, V. (2021). Parametrized Quantum Policies for Reinforcement Learning, Advances in Neural Information Processing Systems

K. Mitarai and M. Negoro and M. Kitagawa and K. Fujii (2018).Quantum circuit learning, Physical Review A, American Physical Society (APS)

Yunchao Liu and Srinivasan Arunachalam, and Kristan Temme (2021). A rigorous and robust quantum speed-up in supervised machine learning, Nature Physics, Springer Science and Business Media