VARIATIONAL QUANTUM CIRCUITS FOR DEEP REINFORCEMENT LEARNING

by Kalmutskiy Kirill

March 2020

Reinforcement Learning

Definition

Reinforcement learning a machine learning paradigm in which an agent interacts with the environment E over a number of discrete time steps. At each time step t, the agent receives a state or observation s_t and then chooses an action at from the set of possible actions A according to its policy π and then get some reward r_t.

- $R_t = \sum_{i=t}^{T} \gamma^{s-t} r_s$ - total discounted return from time step t;
- $Q^{\pi}(s, a) = \mathbb{E}[R_t | s_t = s]$ - action-value function or Q-value function;
- $Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a)$ - the optimal action value function;

Q-learning and SARSA

Before the learning process begins, Q is initially assigned to an arbitrary fixed value. Then, at each time, the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} and then Q is updated.

- Q-learning - off-policy:
  $Q(S_t, A_t) \to Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_t, a) - Q(S_t, A_t)]$
- SARSA - on-policy:
  $Q(S_t, A_t) \to Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$

Representation

The action-value function $Q(s, a)$ can be explicitly represented by a two-dimensional table with a total number of entries $s \times a$.

Deep Reinforcement Learning

- The action-value function $Q(s, a; \theta)$ is parameterized by θ. The θ can be derived by a series of iterations from a variety of optimization methods.
- Loss function is $L(\theta) = \mathbb{E}[\big(r_t + \gamma \max_{\bar{a}} Q(\bar{s}, \bar{a}, \bar{\theta}) - Q(s, a, \theta)\big)^2]$. The target is $r_t + \gamma \max_{\bar{a}} Q(\bar{s}, \bar{a}, \bar{\theta})$ and the prediction is $Q(s, a, \theta)$, where $\bar{s}$ is the state encountered after playing action a_t at state s.

Deep Q-learning

Main ideas of Deep Q-learning: how to lower the correlation of inputs for training?

- Experience replay: sample episodes which were played earlier.
- Target Network: update $\bar{\theta}$ only after several steps of optimization.

Variational Quantum Circuits

The variational quantum circuit is one type of quantum circuits with tunable parameters which need to be optimized in an iterative manner. These parameters can be seen as the weights in artificial neural networks.

- VQC can approximate an analytical function $f(x)$;
- VQC own a better expressive power than the classical function approximators;
- VQC require fewer parameters than a conventional neural network;
- It is hard to simulate the VQC with large number of qubits via classical computers;

Variational Quantum Deep Q Learning

Target Network

For a target network, two sets of circuit parameters with the same circuit architecture were constructed. The targeted circuit is updated per 20 steps.

Experience Replay

For experience replay, the replay memory is set for the length of 80 to adapt to the testing environment **Frozen-Lake** and length of 1000 for **Cognitive Radio**, and the size of training batch is 5 for all of the environments.

Number of qubits

- Computational basis encoding;
- 4 qubits for **Frozen-Lake**;
- 2-5 qubits for **Cognitive Radio**;

Variational Quantum Deep Q Learning

Algorithm

```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function circuit $Q$ with random parameters
for episode $= 1, M$ do
    Initialise state s1
    for t $= 1, T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$
        Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $D$
        Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $D$
        Set $y_j = r_j + \gamma \max_{\overline{a}} Q(s_{j+1}, \overline{a}; \theta)$
        Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$
    end for
end for
```

Computational Basis Encoding

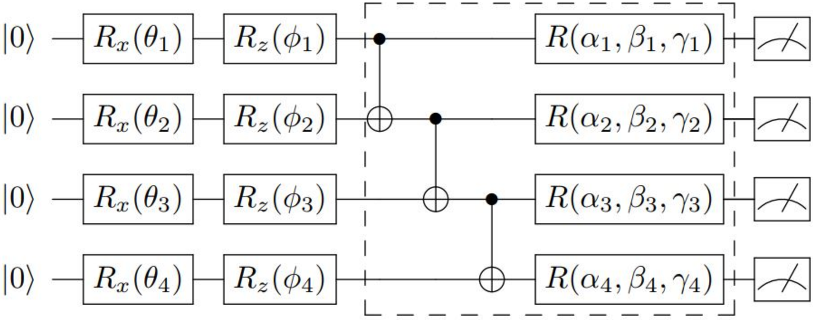
For a general n-qubit state, where $c_{q_1,\ldots,q_n} \in C$ is the amplitude of each quantum state and each $q_n \in \{0, 1\}$, it can be represented as:

$$|\psi\rangle = \sum_{(q_1,\ldots,q_n)\in\{0,1\}^n} c_{q_1,\ldots,q_n}|q_1\rangle \otimes \ldots \otimes |q_n\rangle$$

The encoding procedure is as the following: The decimal number is first converted into a binary number and then encoded into the quantum states through single qubit unitary rotation.

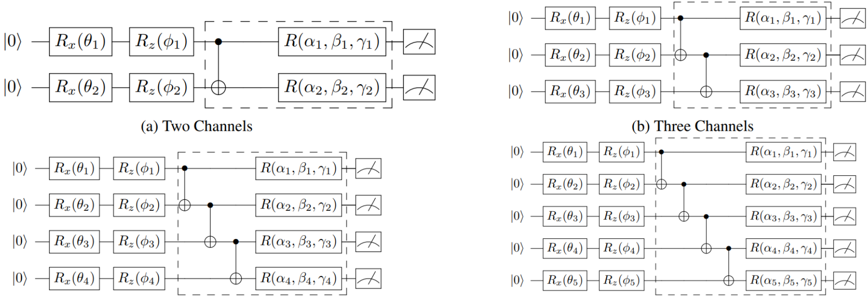
For example, the **Frozen-Lake** state observed by the agent, 12, is first converted to the binary number 1100, which will be $|1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |0\rangle$.

Generic circuit architecture



Parameters labeled α, β, γ, are the ones for iterative optimization. Note that the grouped box may repeat several times to increase the number of parameters. The number of qubits can be adjusted to fit the problem. In this work, the grouped circuit repeats two times and therefore the total number of circuit parameters subject to optimization is $4 \times 3 \times 2 = 24$. It is often to add a bias, so the total number of parameters is $24 + 4 = 28$.

Cognitive Radio


|0⟩
$R_x(\theta_1)$
$R_z(\phi_1)$
$R(\alpha_1, \beta_1, \gamma_1)$
$R_x(\theta_2)$
$R_z(\phi_2)$
$R(\alpha_2, \beta_2, \gamma_2)$
$R_x(\theta_3)$
$R_z(\phi_3)$
$R(\alpha_3, \beta_3, \gamma_3)$
$R_x(\theta_4)$
$R_z(\phi_4)$
$R(\alpha_4, \beta_4, \gamma_4)$
$R_x(\theta_5)$
$R_z(\phi_5)$
$R(\alpha_5, \beta_5, \gamma_5)$

(a) Two Channels

(b) Three Channels

(c) Four Channels

(d) Five Channels

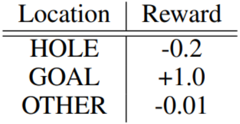
The number of channel is N, there are N time-steps in a full channel-changing cycle. The number of possible states is thus N^2. At each time-step, the agent can select one of the channel from the set of all possible channels, which is of number N.

- Classical Q-learning: N^3 params;
- NN Q-learning: $2 \times N^2 + 2 \times N$ params;
- VQC Q-learning: $N \times (3 \times 2 + 1)$ params;

Configuration

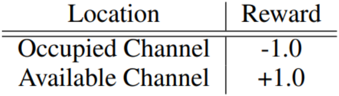
The list of reward in testing environment Frozen-Lake and Cognitive Radio

(a) Frozen-Lake



Location	Reward
HOLE	-0.2
GOAL	+1.0
OTHER	-0.01

(b) Cognitive-Radio

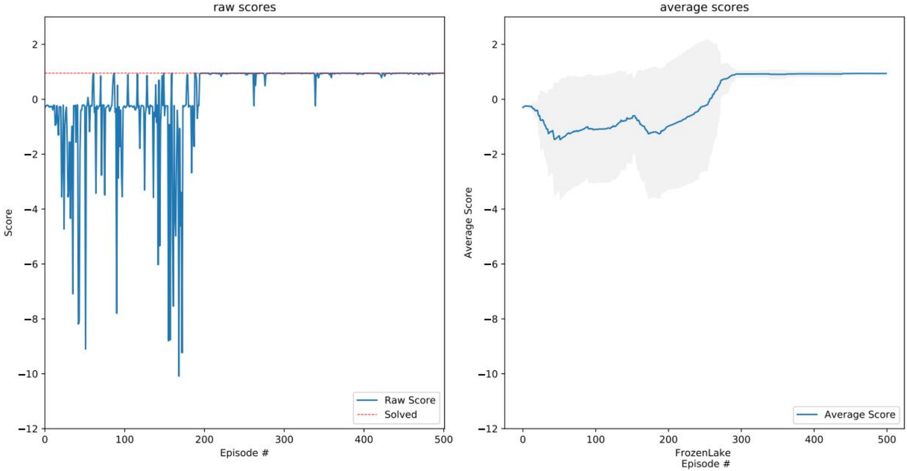


Location	Reward
Occupied Channel	-1.0
Available Channel	+1.0

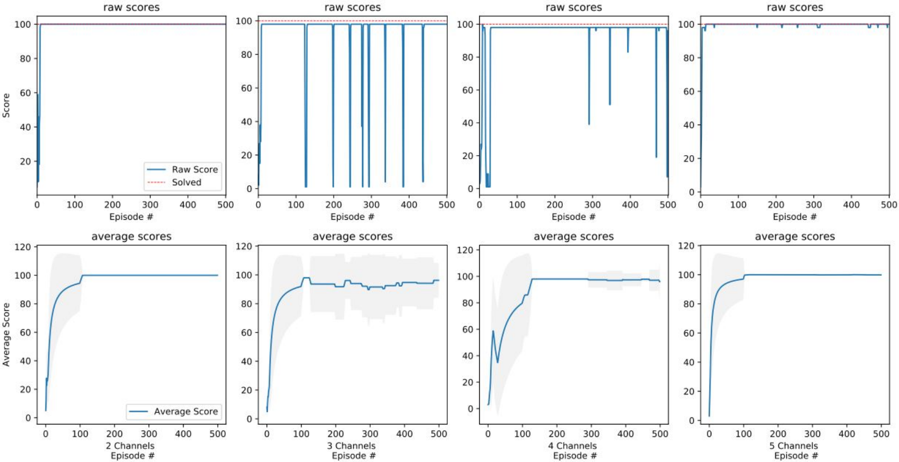
The optimization is chosen to be **RMSprop** with $learning\_rate = 0.01$, $alpha = 0.99$ and $eps = 10^{-8}$, which is used widely in deep reinforcement learning. The batch-size for the experience replay is 5.

To investigate the robustness of proposed VQC against the noise from current and possible near-term devices, the additional simulation which included the noises from the real quantum computer was performed using **Qiskit-Aer** simulation software.

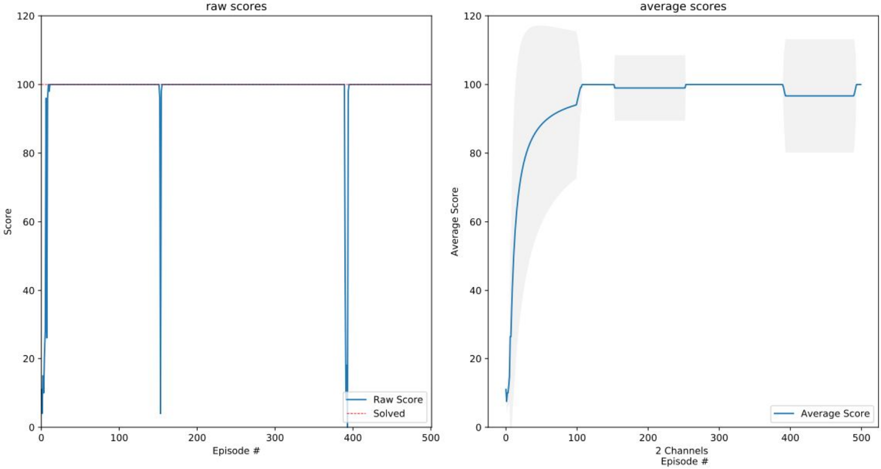
Frozen lake


raw scores
Score
2
0
−2
−4
−6
−8
−10
−12
0
100
200
300
400
500
Episode #
Raw Score
Solved
average scores
Average Score
2
0
−2
−4
−6
−8
−10
−12
0
100
200
300
400
500
FrozenLake
Episode #
Average Score

Cognitive Radio

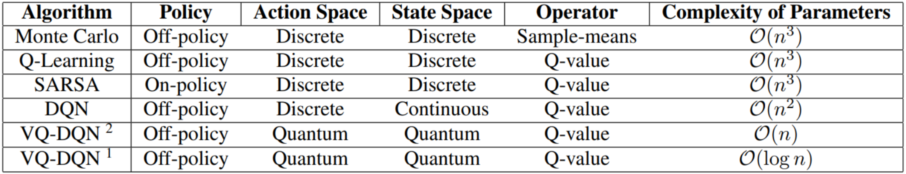


Cognitive Radio with noise



Quantum Advantage on Memory Consumption

Comparison of Classical Reinforcement Learning Algorithms with Discrete Action Space

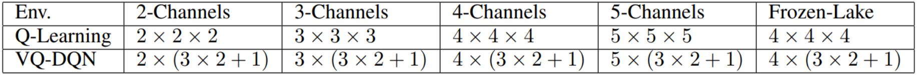


Algorithm	Policy	Action Space	State Space	Operator	Complexity of Parameters
Monte Carlo	Off-policy	Discrete	Discrete	Sample-means	$\mathcal{O}(n^3)$
Q-Learning	Off-policy	Discrete	Discrete	Q-value	$\mathcal{O}(n^3)$
SARSA	On-policy	Discrete	Discrete	Q-value	$\mathcal{O}(n^3)$
DQN	Off-policy	Discrete	Continuous	Q-value	$\mathcal{O}(n^2)$
VQ-DQN [2]	Off-policy	Quantum	Quantum	Q-value	$\mathcal{O}(n)$
VQ-DQN [1]	Off-policy	Quantum	Quantum	Q-value	$\mathcal{O}(\log n)$

[1] VQ-DQN with amplitude encoding can harvest full logarithmic less parameters compared with classical models.

[2] The number of parameters in VQ-DQN with computational basis encoding grows only linearly

Comparison of Number of Parameters in Classical Q-Learning and Quantum Deep Reinforcement Learning



Env.	2-Channels	3-Channels	4-Channels	5-Channels	Frozen-Lake
Q-Learning	$2 \times 2 \times 2$	$3 \times 3 \times 3$	$4 \times 4 \times 4$	$5 \times 5 \times 5$	$4 \times 4 \times 4$
VQ-DQN	$2 \times (3 \times 2 + 1)$	$3 \times (3 \times 2 + 1)$	$4 \times (3 \times 2 + 1)$	$5 \times (3 \times 2 + 1)$	$4 \times (3 \times 2 + 1)$