Scientific Research Publishing

Intelligent Control and Automation, 2019, 10, 63-78
http://www.scirp.org/journal/ica
ISSN Online: 2153-0661
ISSN Print: 2153-0653

Distributional Reinforcement Learning with Quantum Neural Networks

Wei Hu[1], James Hu[2]

[1]Department of Computer Science, Houghton College, Houghton, NY, USA
[2]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA
Email: wei.hu@houghton.edu

How to cite this paper: Hu, W. and Hu, J. (2019) Distributional Reinforcement Learning with Quantum Neural Networks. *Intelligent Control and Automation*, **10**, 63-78.
https://doi.org/10.4236/ica.2019.102004

Received: February 11, 2019
Accepted: April 7, 2019
Published: April 10, 2019

Abstract

Traditional reinforcement learning (RL) uses the return, also known as the expected value of cumulative random rewards, for training an agent to learn an optimal policy. However, recent research indicates that learning the distribution over returns has distinct advantages over learning their expected value as seen in different RL tasks. The shift from using the expectation of returns in traditional RL to the distribution over returns in distributional RL has provided new insights into the dynamics of RL. This paper builds on our recent work investigating the quantum approach towards RL. Our work implements the quantile regression (QR) distributional Q learning with a quantum neural network. This quantum network is evaluated in a grid world environment with a different number of quantiles, illustrating its detailed influence on the learning of the algorithm. It is also compared to the standard quantum Q learning in a Markov Decision Process (MDP) chain, which demonstrates that the quantum QR distributional Q learning can explore the environment more efficiently than the standard quantum Q learning. Efficient exploration and balancing of exploitation and exploration are major challenges in RL. Previous work has shown that more informative actions can be taken with a distributional perspective. Our findings suggest another cause for its success: the enhanced performance of distributional RL can be partially attributed to its superior ability to efficiently explore the environment.

Keywords

Continuous-Variable Quantum Computers, Quantum Reinforcement Learning, Distributional Reinforcement Learning, Quantile Regression, Distributional Q Learning, Grid World Environment, MDP Chain Environment

1. Introduction

Machine learning is teaching computer models how to learn from data. As a subfield of machine learning, reinforcement learning (RL) aims to learn sequential decision making from data [1] [2] [3]. Several significant advances in RL have been made in the recent years including the famous AlphaGo application from Google. Learning under uncertainty caused by randomness is a reality and a challenge in machine learning. In RL, the probabilistic nature of rewards, state transitions, and actions may all introduce randomness into the cumulative return. Traditional RL averages over this randomness to estimate the value function, which is then used to generate a policy. However, recent research [4] [5] [6] shows that using the full distribution over random returns can preserve multimodality in the returns and therefore make learning more effective as demonstrated by the state-of-the-art performance in a number of RL benchmarks. Given two return distributions with the same expected value, their actual value distributions can vary drastically. Therefore, using the full distribution instead of the single scalar of the expected value is more informative when deciding which action to take in RL. Related to this idea, is the introduction of replay memories in deep RL, allowing the agent to leverage previous experiences to break the correlation of training data, which uses a full batch of steps instead of one single step for training.

A distributional RL algorithm has to make two choices: 1) how to parameterize the value distributions, and 2) how to select a distance metric or loss function for the two distributions: target distribution and predicted distribution. Categorical DQN, also commonly called C51 [4], uses a categorical distribution and minimizes a cross-entropy loss function between projected Bellman update and prediction. This algorithm assigns the distribution to an a priori fixed, discrete set of possible returns.

C51 represents the value distribution as a categorical distribution over a fixed set of equidistant points and uses it to approximate the projected distributional Bellman target via minimizing their KL divergence. Furthermore, the work on C51 proves that the distributional Bellman operator is a contraction in a Wasserstein metric between probability distributions. However, C51 cannot prove that it has this contraction as it uses a cross-entropy loss function. C51 uses a predefined range of the categorical supports; therefore, it may not work well in some RL tasks.

QR distributional Q learning [5], on the other side, represents the distribution by a uniform mixture of Dirac delta functions whose locations are adjusted using quantile regression. These distributions are constructed with a discrete set of quantiles, but their supports can be tuned to minimize the Wasserstein distance between the Bellman target distribution and predicted value distribution. The output of the networks of C51 is the probability of value distribution for each action, where the support is used as fixed value. However, QR distributional Q learning outputs the support of value distribution for each action, where the probability is used as fixed value.

With the recent advances in quantum computing, machine learning has a new paradigm of computation to design and test different algorithms [7]-[24]. There are two different models of quantum computing: discrete and continuous variables [25]. IBM's quantum computers are discrete, made of qubits, while Xanadu's are continuous, made of qumodes [26]. Powered by the special properties of quantum states such as superposition, entanglement, and interference, quantum computers can process information in ways far beyond their classical counterparts. Quantum computers operate at the lowest level of physics, offering a much richer structure for computation. As a result, quantum machine learning can create entirely new learning models, impossible with classical computing.

One advantage of quantum computing is its capability to process data of high dimensions. For example, a classical computer of 64 bits can process data of size 64 bits at a time, but a quantum computer of 64 qubits can process data of size 2^{64} bits at a time, achieving an exponential increase. It is hoped that quantum computing will be useful to find new patterns in big data and solve problems that are currently intractable for classical computers. These intractable problems include quantum system simulations and molecular and atomic dynamics simulations. Finding the ground state energy of a complex molecule is a challenge for classical computers. Also, as more transistors are packed in a single CPU chip, quantum phenomenon will occur. Therefore, quantum computing will become more relevant even to classical computing in this regard.

2. Related Work

Xanadu is a company that makes photonic quantum computers, which can process information stored in quantum states of light [26]. These quantum computers can represent continuous variable using the amplitude and phase of light, which are ideal for machine learning where continuous variables are a norm.

A research team at Xanadu created light-based quantum neural networks using photonic gate circuits [27], which can solve certain problems using exponentiallyless resources than their classical counterparts. Furthermore, the networks can transform simple quantum states into more complex ones, and perform common tasks in machine learning such as regression and classification. In another work [28], these quantum networks were trained to convert laser light into states of a fixed number of photons. By adjusting the brightness and phase of the incoming light, the quantum states of one, two, or three photons can be created (**Figure 1**). Preparing a quantum state in a physics lab is a difficult task, but this technique can reduce the burden of creating an arbitrary quantum state. These quantum networks are made from a photonic quantum circuit with layers of gate structures, where the output of one layer can be the input to the next. These layers are similar to those found in classical neural networks.

In our previous work, we used these quantum neural networks to study the contextual bandit problem, and to implement Q learning and actor-critic algo-

rithms [20] [21] [22]. In this paper, we direct our attention on implementation of the QR distributional Q learning algorithm with these networks.

3. Methods

We describe the test environments, QR distributional Q learning, and QR quantum networks in this section. Traditionally, RL focuses on the mean of the return; however, distributional RL aims to model the distribution over the returns in order to gain a more complete picture of their world. The output of the QR network in distributional RL is a quantile distribution, represented by a discrete set of supports for a given set of quantiles. Although the final decision on actions is still based on the Q values, which are the expected returns. One key difference is, that in distributional RL, the goal is to get full distributions rather than their expectations.

3.1. Environments

Two environments are employed in this study to examine the performance of quantum QR distributional Q learning. A grid world is used to check the rewards that the algorithm can collect for various quantile numbers, which are key parameters of the algorithm. A MDP Chain is used to see how this new algorithm explores the environment when compared the more familiar algorithm Q learning.

3.1.1. Grid World

A grid world environment is used to evaluate the performance of quantum QR distributional Q learning, which is also used in our previous work [21] [22]. It has a size of 2 × 3 and can be configured into two modes: slippery or not slippery. In the slippery environment, a move towards the intended direction is only successful at a probability of 1/3, and can slide in two perpendicular directions at a probability of 1/3 each. Each state has a reward of zero, except at the state G that has a reward of one. H and G are the two terminal states, and each walk starts at the state S. One episode in this environment is defined as the steps by an agent from state S to either H or G (**Figure 2**).

3.1.2. MDP Chain

A MDP chain is used to investigate the ability of the quantum QR distributional Q learning algorithm to explore the environment, which is a simple deterministic chain $\{s_{-3}, s_{-2}, s_{-1}, s_0, s_1, s_2, s_3\}$ of size 7 with the start state s_0 and two terminal states s_{-3} and s_3 (**Figure 3**). The agent can move left and right. All states have zero reward, except for the left end state s_{-3} which has a reward of 0.001 and the right end state s_3 which has a reward of 1. The challenge of this environment is that it has no negative reward and going left is a suboptimal action while going right is the optimal action. Only an agent that can efficiently explore the environment can discover the subtle difference between the two positive rewards at the two ends.

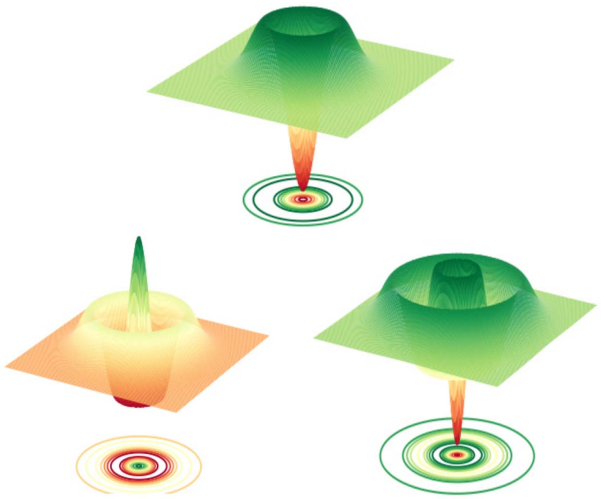


Figure 1. Winger function representation of one, two, and three photon states, displayed in sequence from top, bottom left to bottom right. The negative values of Winger function reveal the non-classical nature of these three states.

S	F	H
F	F	G

Figure 2. The grid world of size 2 × 3 used in this study. Each grid (state) is labeled with a letter which has the following meaning: Start (S), Frozen (F), Hole (H) and Goal (G). The state G has a reward of one and other states have a reward of zero. Each grid (state) is represented as an integer from 0 to 5, with the top row: 0, 1, 2 (left to right) and the bottom row: 3, 4, 5 (left to right).

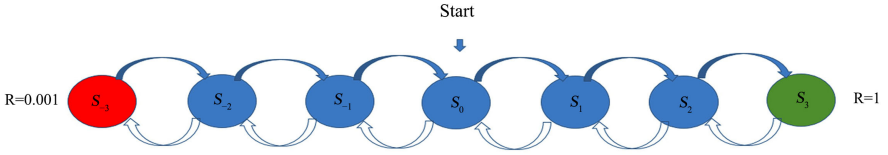


Figure 3. A MDP chain of size 7 which has both ends as terminal states with different positive rewards and s_0 as the start state.

3.2. QR Distributional *Q* Learning

3.2.1. Quantile Regression

Linear and logistic regressions are commonly used in machine learning, while quantile regression is widely employed in economics. However, because standard linear regression focuses on the conditional mean function, if other relationships among the variables are desirable, quantile regression can be useful. We might be interested in the full distribution of the data rather than just the mean of the data. We may also want to study the conditional median function, where the median is the 50th percentile, or quantile q, of a data distribution. The quantile level $\tau \in (0,1)$ splits the data into proportions τ% below and

$(1-\tau)\%$ above. Assume X is a random variable, $F_X(x)$ is its cumulative distribution function (CDF), and $F_X^{-1}(\tau)$ is the inverse, there holds the relationship: $\tau = F_X(x)$ and $x = F_X^{-1}(\tau)$. $F_X^{-1}(\tau)$ is called the quantile function with the median $F_X^{-1}(0.5)$ as a special case. The real number $x_\tau = F_X^{-1}(\tau)$ or $F_X(x_\tau) = \tau$ defines the τ th quantile of F_X or X, which means the probability that an observation is less than x_τ is τ. In linear regression, a straight line is selected by minimizing distance between the line and the data points. In contrast, quantile regression searches a line based on the selected quantile using the quantile loss function as shown in **Figure 4**. If the 70th quantile is desired, a regression line is found under the condition that 70% of the data points are below the line and 30% are above. As a result, the median of a data set minimizes the sum of absolute errors from the median, so the 50% quantile can be formulated as a solution to an optimization problem. Compared to the ordinary least squares, QR is robust to outliers in dependent variables, but is sensitive to the sparse extremes of the independent variables. Typically linear regression assumes the data follow the normal distribution; however, quantile regression does not assume a particular parametric distribution for the data.

3.2.2. QR Distributional *Q* Learning

The goal of the RL agent is to learn a policy that can gain the maximum expected return. So by definition, it is natural to work directly with these expectations. However, this approach cannot render the whole picture of the randomness as seen from the possible multimodal distribution over returns. When an agent interacts with the environment in a RL problem, the state transitions, rewards, and actions can all carry certain intrinsic randomness. Distributional RL explicitly models the future random rewards as a full distribution, allowing more accurate actions to be learned. In order to introduce QR distributional Q learning, which utilizes quantile regression to approximate the quantile function for the state-action return distribution, we need to introduce several concepts as background materials [5].

For a given policy π, the return Z^π is a random variable that represents the sum of discounted rewards.

$$Z^\pi = \sum_{t=0}^{\infty} \gamma^t R_t \tag{1}$$

where γ is the discount rate. In Equation (1), Z^π is interpreted as a distribution. Let $Z^\pi(s_0, a_0)$ be the return obtained by starting from state s_0, performing action a_0 and then following the current policy π, the well-known Q value can be obtained as follows:

$$Q^\pi(s_0, a_0) := \mathrm{E}\left[Z^\pi(s_0, a_0)\right] = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{2}$$

$$TZ^\pi(s_t, a_t) := R(s_t, a_t) + \gamma Z^\pi(s_{t+1}, a_{t+1}) \tag{3}$$

Equation (3) defines the so called target distribution or distributional Bellman target [29]. More accurately, it is a sample of the true target distribution. In the above Equation (3), both R and Z are distributions.

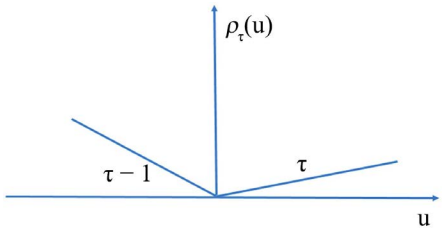


Figure 4. Quantile loss function $\rho_\tau(u) = u\left(\tau - 1_{\{u<0\}}\right)$ where $\tau \in (0,1)$. The slope is used to reflect the desired imbalance in quantile regression. When τ = 0.25, and u_pred is the predicted value of u, then the loss function ρ_τ(u-u_pred) gives a penalty of 0.25 for underestimation but three times more (0.75) for overestimation. Therefore, u_pred should stay below the median, but also cannot be too small and far away from the median, which suggests the actual value of u_pred be the τth quantile. The optimal quantile model has lowest quantile loss.

Distributional RL algorithms need to measure the distance of two distributions: the target and predicted distributions. The KL divergence is a commonly used distance between two distributions, but it is not defined when these distributions are discrete and have different supports. That is the reason for the work of C51 to use a projection to make them have same supports in order to apply KL divergence. The alternative is Wasserstein metric which is described as follows.

For $p \in [1,\infty]$, the p-Wasserstein metric W_p between distributions U and Y is defined as,

$$W_p(U,Y) = \left(\int_0^1 \left|F_Y^{-1}(\tau) - F_U^{-1}(\tau)\right|^p d\tau\right)^{1/p} \tag{4}$$

where Y is a random variable, F_Y is its cumulative distribution function, and $F_Y^{-1}(\tau)$ is the inverse. The p-Wasserstein distance is the L^p metric of the inverse of CDF, which is an extension of the Euclidean distance from point data to distribution data. When U and Y are two Dirac delta distributions located at X_1 and X_2 in R^N, the p-Wasserstein distance becomes a Euclidean of X_1 and X_2. Let $\theta = (\theta_1, \theta_2, \theta_3, \cdots, \theta_N) \in R^N$, a quantile distribution Z_θ is a uniform probability distribution supported on $\{\theta_i(s,a)\}$ which can be defined as,

$$Z_\theta(s,a) := \frac{1}{N}\sum_{i=1}^N \delta_{\theta_i(s,a)} \tag{5}$$

where δ_z is the Dirac delta function at z.

The QR distributional Q learning algorithms use $Z_\theta(s,a)$ to approximate the target distribution in Wasserstein distance W_1 through quantile regression. In other words, this approach aims to estimate the quantiles of the target distribution.

Let $\tau_i = i/N$ for each i = 1, …, N. For an arbitrary value distribution Z, we have

$$W_1(Z, Z_\theta) = \sum_{i=1}^N \int_{\tau_{i-1}}^{\tau_i} \left|F_Z^{-1}(\tau) - \theta_i\right| d\tau \tag{6}$$

The key observation used in [5] is that the minimizers of $W_1(Z, Z_\theta)$ are those $\theta = (\theta_1, \theta_2, \theta_3, \cdots, \theta_N)$ that minimize the quantile regression objective:

$$\sum_{i=1}^{N} E_{\hat{Z}\sim Z}\left[\rho_{\hat{\tau}_i}\left(\hat{Z} - \theta_i\right)\right] \tag{7}$$

which are given by $\theta_i = F_Z^{-1}(\hat{\tau}_i)$, where $\hat{\tau}_i = \frac{\tau_i + \tau_{i-1}}{2}$ and $\rho_{\hat{\tau}_i}(u) = u\left(\hat{\tau}_i - 1_{\{u<0\}}\right), \forall u \in R$. Each θ_i is the $\hat{\tau}_i$ quantile. These $\{\theta_i\}$ are the quantiles of the target distribution Z.

The gradients of quantile regression loss are independent of the magnitude of the error, which can increase gradient variance. As an improvement, Huber loss is defined as:

$$L_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & if\ |u| \le \kappa \\ \kappa\left(|u| - \frac{1}{2}\kappa\right), & otherwise \end{cases} \tag{8}$$

And the quantile Huber loss is:

$$\rho_\tau^\kappa(u) = \left|\tau - 1_{\{u<0\}}\right| L_\kappa(u) \tag{9}$$

which is a smooth version of the quantile loss defined in **Figure 4** and its gradients scale with the magnitude of the error, under some threshold κ.

Recall in Q learning, the Bellman optimality operator is defined as (notice the expectation used in the definition):

$$TQ(s,a) = E\left[R(s,a)\right] + \gamma E\left[\max_{a'} Q(s',a')\right] \tag{10}$$

So the distributional version of this operator is:

$$TZ(s,a) = R(s,a) + \gamma Z\left(s', a^*\right) \tag{11}$$

where $a^* = argmax_{a'} E_{z\sim Z(s',a')}[z]$ and s' is the next state of s. The quantile regression Q learning from [5] is presented as follows:

Algorithm 1 Quantile Regression Q Learning [5]

Require: N, κ

Input s, a, r, s', $\gamma \epsilon [0,1)$

#Compute distributional Bellman target

$Q(s',a') := \sum_j q_j \theta_j(s',a')$ where $q_j = 1/N$

$a^* \leftarrow argmax_a Q(s',a')$

$T\theta_j \leftarrow r + \gamma\theta_j(s', a^*), \forall j$

#Compute quantile regression loss

Output $\sum_{i=1}^{N} E_j\left[\rho_{\hat{\tau}_i}^k\left(T\theta_j - Q(s,a)\right)\right]$

Quantile regression finds the approximation to the inverse cumulative distribution function F_Z^{-1} with a quantile function as defined in Equation (5). The quantiles used in the algorithm imply how many times the probability distribu-

tion is divided. For example, 5-quantiles would divide the distribution into 5 intervals [0.20, 0.40, 0.60, 0.80, 1.00]. The QR distributional Q learning algorithm minimizes Wasserstein distance to the distributional Bellman target using distributional Bellman updates through quantile regression, which adjusts the support of each of the equally divided probabilities.

3.3. Quantum Neural Networks

The quantum neural networks used to implement the QR distributional Q learning algorithm are created by following the design in [27]. The photonic gatesutilized in this work are: interferometer, displacement, rotation, squeeze, and Kerr (non-Gaussian) gates (**Figure 5**). The right plot of **Figure 5** shows one layer of the network, which can be repeated to form a multi-layer structure of networks.

Typically the expected sum of future rewards is used to train an agent in RL. Distributional RL takes this idea one step further by computing the full distribution of the random returns. C51 fixes supports at equal intervals and finds the probability distribution from the output of the network. However, QR distributional Q learning sets probability values at equal intervals and gets the supports from the network output (middle plot in **Figure 5**).

The final action selection is still based on the average of the value distribution. However, holding the knowledge of the whole distribution gives more information than simply estimating one expectation. The Q learning process is to reduce the gap between the predicted value and the target value. Naturally, distributional RL process is to minimize the distance between the predicted distribution and the target distribution.

In Q learning, the output of the network is the Q value for each action, but the output of the QR network in distributional RL is the quantile distribution for each action as shown in **Figure 5**. There is one distribution for each action, computing the expected value of this distribution as its Q value, and then selecting the action with the highest Q value.

4. Results

The numerical simulations of our quantum neural networks are conducted with Strawberryfields [26].

4.1. Results on Grid World

The performance of our algorithms on the grid world is shown in **Figure 6**. In this particular environment, it seems that the quantum QR distributional Q learning algorithm works better with 3 or 6 quantiles than other options. The numerical summary of their impact on the performance of the algorithm is in **Table 1**.

4.2. Results on MDP Chain

Two competing strategies of machine learning are the exploration of new possi-

bilities and the exploitation of old certainties. Balancing exploration and exploitation is a fundamental issue in RL. The dilemma is between acting on what the agent already knows and taking risks to try something it has not experienced, which could potentially lead to better rewards than the known ones. Finding the correct balance between these two strategies is not easy as neither is consistently better than the other. Exploitation might be a good decision for achieving short term goals but exploration might result in a long term success.

Common strategies for exploration such as ε-greedy do not work well when deep exploration is required. Bayesian techniques that can explore a shallow environment efficiently cannot do well in a deep structure, as this could lead to an exponentially larger number of trials. Inspired by the work in [30], we compare the abilities of quantum QR distributional Q learning and quantum Q learning to explore a MDP chain. The results (**Figure 7**) show that quantum QR distributional Q learning does a better exploration than quantum Q learning. The learning curves reveal that quantum Q learning can only learn the suboptimal policy and more importantly, they show that quantum Q learning spends a lot of timeoscillating rather than actually exploring the environment at the beginning of each run. The optimal and suboptimal episode lengths are both 3 as the starting state is s_0. The results in this section provide another proof of knowing the value distribution instead of its expected value is beneficial.

Table 1. The average rewards and average episode lengths are based on 5 runs of the same experiments with 100 episodes, which show the detailed influence of different number of quantiles on the performance of quantum QR distributional Q learning.

Numerical summary of the influence by the number of quantiles on quantum QR distributional Q learning					
Number of Quantiles	2	3	4	5	6
Average Rewards	0.89	0.982	0.954	0.952	0.954
Average Episode Lengths	14.04	13.44	12.45	11.63	11.28

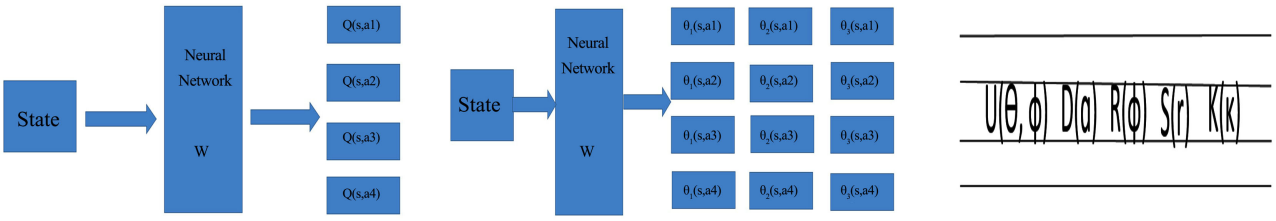
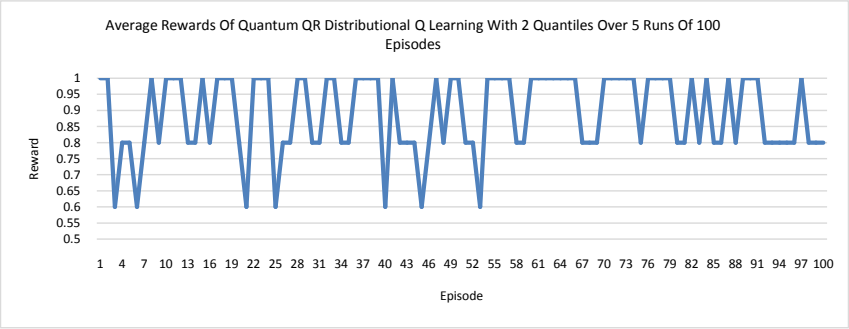
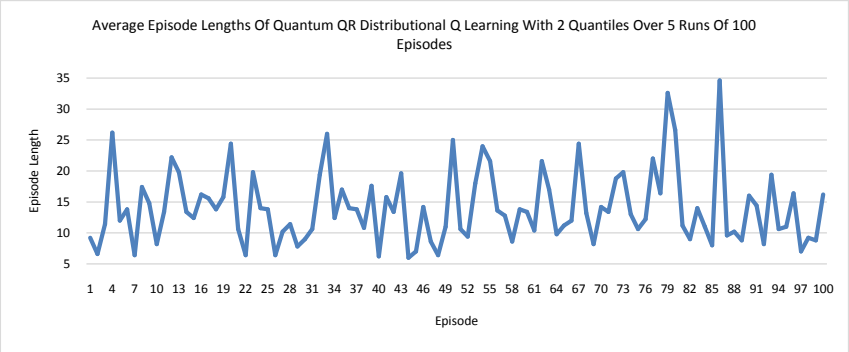
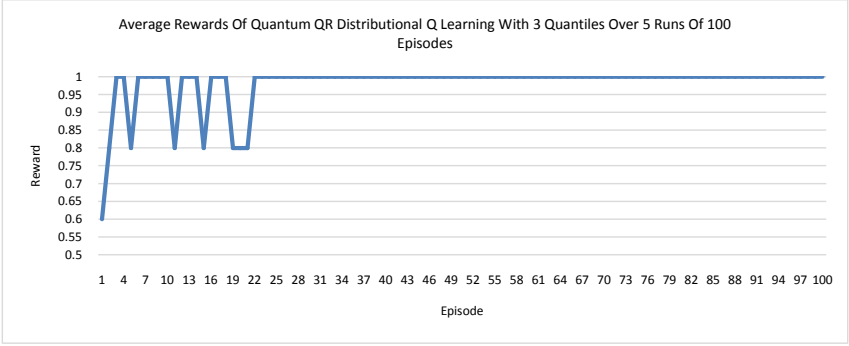
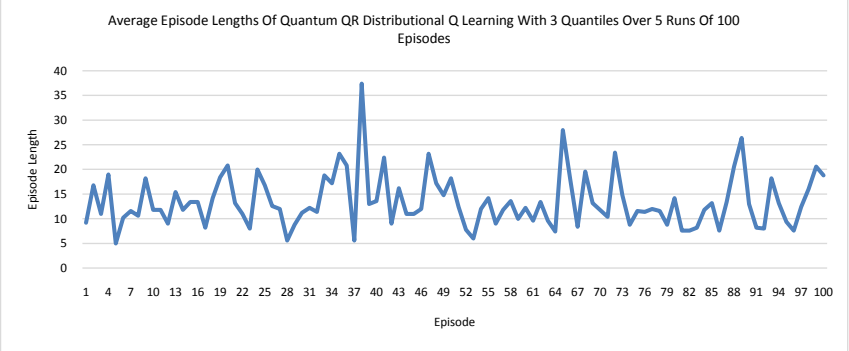


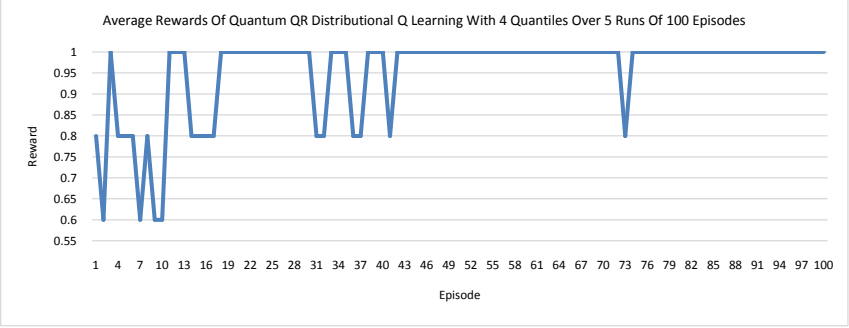
Figure 5. On the left, there is the architecture of the neural networks for Q learning to compute the Q function (assuming there are 4 actions and W is a collection of parameters for the networks), in the middle, there is the architecture of the neural networks for QR distributional Q learning to compute the distribution over returns (assuming there are 4 actions and 3 quantiles, and W is a collection of parameters for the networks), and on the right, there is the physical representation of the actual parametrized circuit structure for a CV quantum neural network made of photonic gates: interferometer, displacement, rotation, squeeze, and Kerr (non-Gaussian) gates. The output is the Fock space measurements. More details of this quantum network can be found in [20] [21] [22] [27].


Average Rewards Of Quantum QR Distributional Q Learning With 2 Quantiles Over 5 Runs Of 100 Episodes
Reward
Episode


Average Episode Lengths Of Quantum QR Distributional Q Learning With 2 Quantiles Over 5 Runs Of 100 Episodes
Episode Length
Episode


Average Rewards Of Quantum QR Distributional Q Learning With 3 Quantiles Over 5 Runs Of 100 Episodes
Reward
Episode


Average Episode Lengths Of Quantum QR Distributional Q Learning With 3 Quantiles Over 5 Runs Of 100 Episodes
Episode Length
Episode


Average Rewards Of Quantum QR Distributional Q Learning With 4 Quantiles Over 5 Runs Of 100 Episodes
Reward
Episode

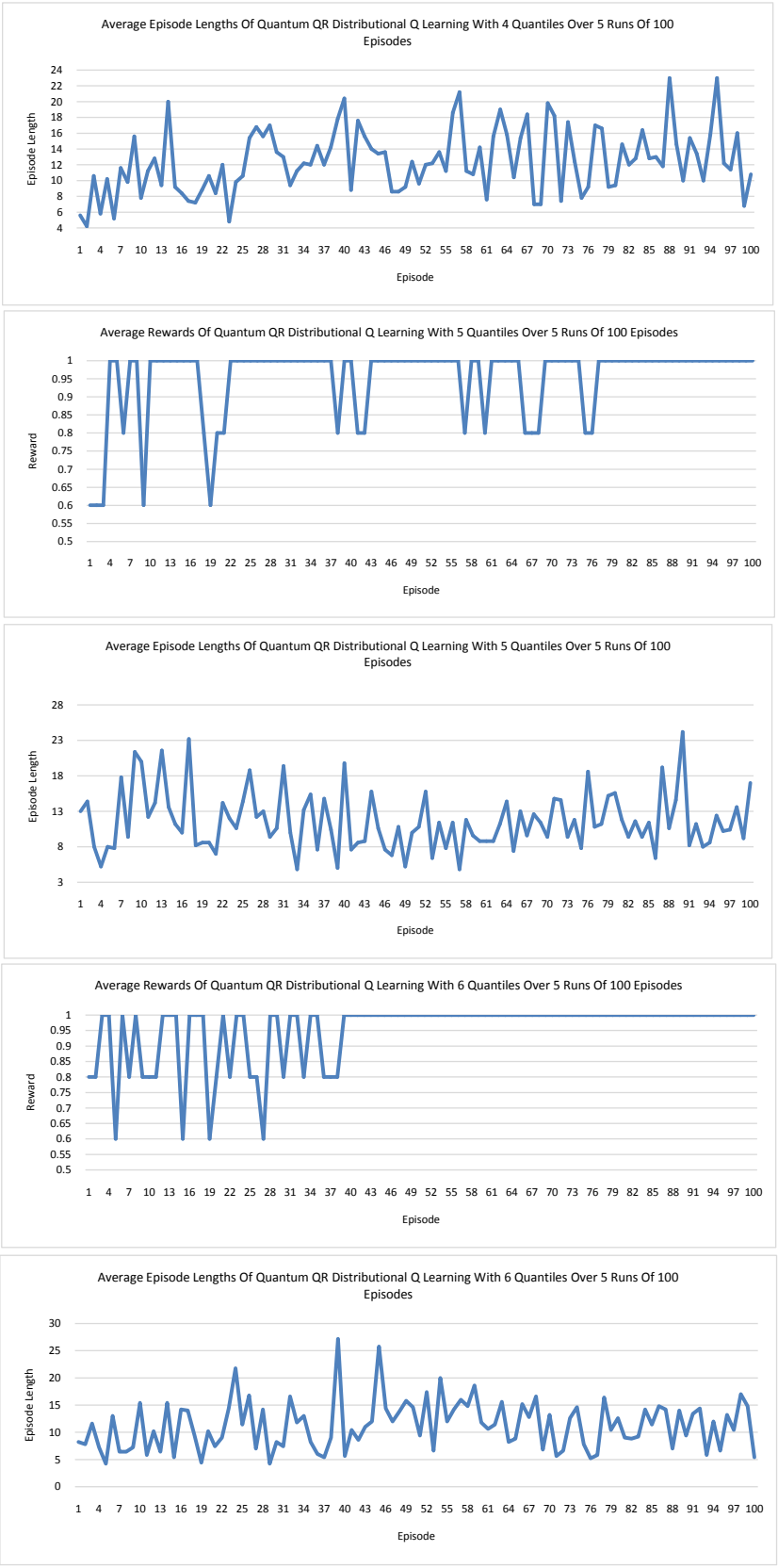


Figure 6. These plots along with **Table 1** show the detailed impact of choosing different number of quantiles on the performance of quantum QR distributional Q learning.

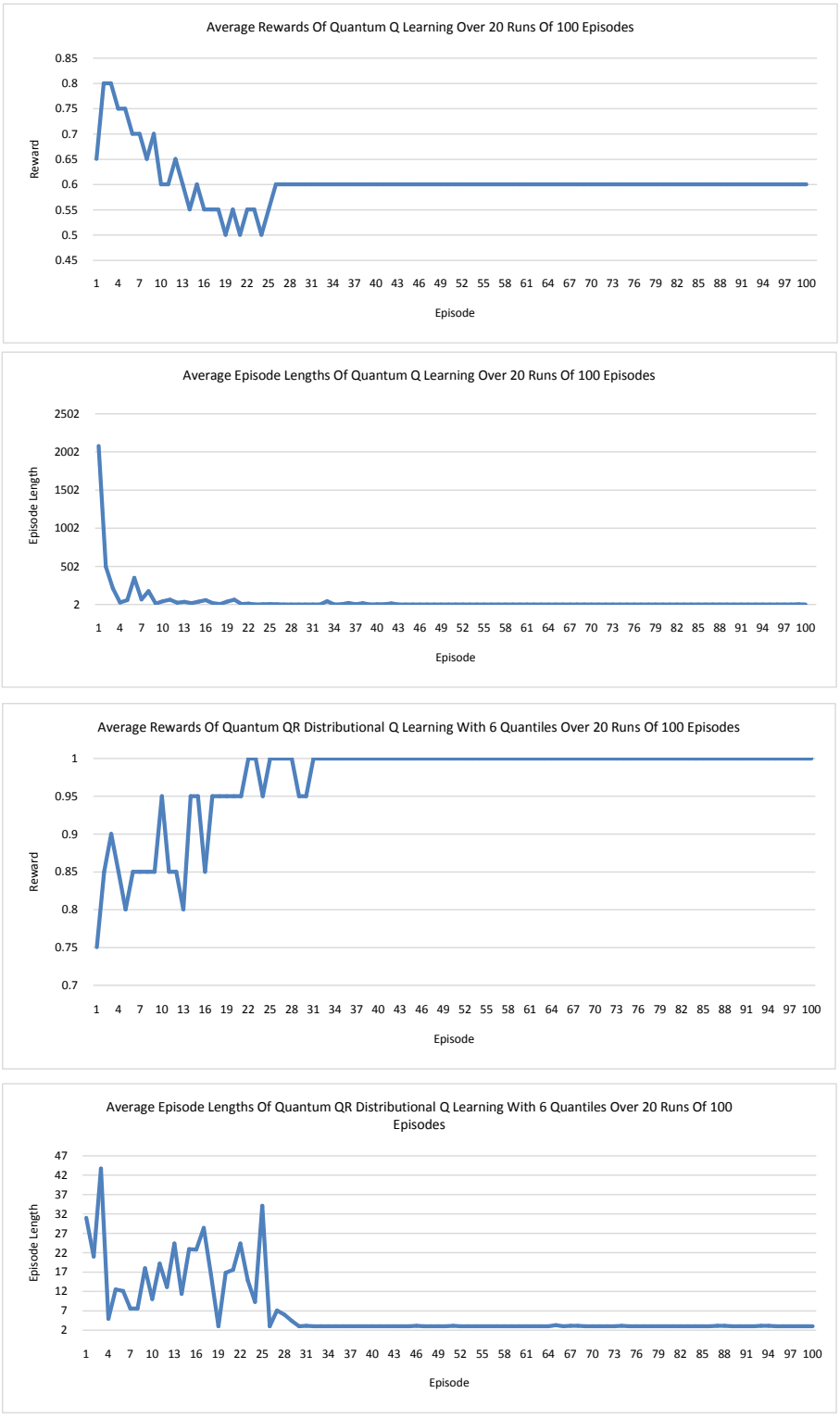


Figure 7. These plots show that quantum QR Distributional Q learning of 6 quantiles with two layers of quantum networks can find the optimal policy (going right) while the standard quantum Q learning with two layers of quantum networks can only discover the suboptimal policy (going left). At the beginning, they both do not know where to go which can be seen from their oscillating rewards, but eventually they each find a policy to converge. The shortest episode length is 3 from S_0 to both ends.

5. Conclusions

In RL, the state-action value function Q(*s*, *a*) represents the expected return for taking action *a* in state *s*. Instead of using a scalar expected return, distributional RL algorithms compute a full distribution over these returns, which makes the learning more accurate and faster than previous methods. QR distributional *Q* learning employs quantile regression to minimize the Wasserstein distance between the target distribution and the predicted distribution by adjusting their supports. The output of the QR network is a set of supports that form the core of the quantile distribution definition.

The problem this study is concerned with is that of quantum RL. Research has shown that the quantum approach to machine learning can result in improved performances. The work covered in this report examines the implementation and performance of QR distributional *Q* learning on quantum computers. Learning the full distribution over returns rather than their expectation is the main idea of distributional RL. Therefore, our aim is to evaluate the features of quantum QR distributional *Q* learning, by testing its ability to collect rewards in a grid world with a different number of quantiles and then comparing its capability of exploring a MDP chain environment with standard quantum *Q* learning. Our findings demonstrate that quantum QR distributional *Q* learning can explore the environment more efficiently than quantum *Q* learning.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Sutton, R.S. and Barto, A.G. (2018) Reinforcement Learning, an Introduction. Second Edition, a Bradford Book.

[2] Ganger, M., Duryea, E. and Hu, W. (2016) Double Sarsa and Double Expected Sarsa with Shallow and Deep Learning. *Journal of Data Analysis and Information Processing*, **4**, 159. https://doi.org/10.4236/jdaip.2016.44014

[3] Duryea, E., Ganger, M. and Hu, W. (2016) Exploring Deep Reinforcement Learning with Multi Q-Learning. *Intelligent Control and Automation*, **7**, 129. https://doi.org/10.4236/ica.2016.74012

[4] Bellemare, M.G., Dabney, W. and Munos, R. (2017) A Distributional Perspective on Reinforcement Learning.

[5] Dabney, W., Rowland, M., Bellemare, M.G. and Munos, R. (2017) Distributional Reinforcement Learning with Quantile Regression.

[6] Dabney, W., Ostrovski, G., Silver, D. and Munos, R. (2018) Implicit Quantile Networks for Distributional Reinforcement Learning.

[7] Clausen, J. and Briegel, H.J. (2018) Quantum Machine Learning with Glow for Episodic Tasks and Decision Games. *Physical Review A*, **97**, Article ID: 022303.

[8] Crawford, D., Levit, A., Ghadermarzy, N., Oberoi, J.S. and Ronagh, P. (2016) Reinforcement Learning Using Quantum Boltzmann Machines.

[9] Dunjko, V., Taylor, J.M. and Briegel, H.J. (2018) Advances in Quantum Reinforcement Learning.

[10] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N. and Lloyd, S. (2017) Quantum Machine Learning. *Nature*, **549**, 195-202. https://doi.org/10.1038/nature23474

[11] Dunjko, V. and Briegel, H.J. (2018) Machine Learning and Artificial Intelligence in the Quantum Domain: A Review of Recent Progress. *Reports on Progress in Physics*, **81**, Article ID: 074001. https://doi.org/10.1088/1361-6633/aab406

[12] Hu, W. (2018) Towards a Real Quantum Neuron. *Natural Science*, **10**, 99-109. https://doi.org/10.4236/ns.2018.103011

[13] Hu, W. (2018) Empirical Analysis of a Quantum Classifier Implemented on IBM's 5Q Quantum Computer. *Journal of Quantum Information Science*, **8**, 1-11. https://doi.org/10.4236/jqis.2018.81001

[14] Hu, W. (2018) Empirical Analysis of Decision Making of an AI Agent on IBM's 5Q Quantum Computer. *Natural Science*, **10**, 45-58. https://doi.org/10.4236/ns.2018.101004

[15] Hu, W. (2018) Comparison of Two Quantum Clustering Algorithms. *Natural Science*, **10**, 87-98. https://doi.org/10.4236/ns.2018.103010

[16] Ganger, M. and Hu, W. (2019) Quantum Multiple Q-Learning. *International Journal of Intelligence Science*, **9**, 1-22. https://doi.org/10.4236/ijis.2019.91001

[17] Duryea, E. and Hu, W. (2019) Quantum Dyna Q Learning. *International Journal of Intelligence Science*.

[18] Mitarai, K., Negoro, M., Kitagawa, M. and Fujii, K. (2018) Quantum Circuit Learning.

[19] Schuld, M. and Killoran, N. (2018) Quantum Machine Learning in Feature Hilbert Spaces.

[20] Hu, W. and Hu, J. (2019) Training a Quantum Neural Network to Solve the Contextual Multi-Armed Bandit Problem. *Natural Science*, **11**, 17-27. https://doi.org/10.4236/ns.2019.111003

[21] Hu, W. and Hu, J. (2019) Q Learning with Quantum Neural Networks. *Natural Science*, **11**, 31-39. https://doi.org/10.4236/ns.2019.111005

[22] Hu, W. and Hu, J. (2019) Reinforcement Learning with Deep Quantum Neural Networks. *Journal of Quantum Information Science*, **9**, 1-14.

[23] Farhi, E. and Neven, H. (2018) Classification with Quantum Neural Networks on Near Term Processors.

[24] Dong, D., Chen, C., Li, H. and Tarn, T.-J. (2008) Quantum Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* (*Cybernetics*), **38**, 1207-1220. https://doi.org/10.1109/TSMCB.2008.925743

[25] Serafini, A. (2017) Quantum Continuous Variables: A Primer of Theoretical Methods. CRC Press, Boca Raton.

[26] Killoran, N., Izaac, J., Quesada, N., Bergholm, V., Amy, M. and Weedbrook, C. (2018) Strawberry Fields: A Software Platform for Photonic Quantum Computing.

[27] Killoran, N., Bromley, T.R., Arrazola, J.M., Schuld, M., Quesada, N. and Lloyd, S. (2018) Continuous-Variable Quantum Neural Networks.

[28] Arrazola, J.M., Bromley, T.R., Izaac, J., Myers, C.R., Brádler, K. and Killoran, N. (2018) Machine Learning Method for State Preparation and Gate Synthesis on Photonic Quantum Computers.

[29] Bellman, R. (1957) A Markovian Decision Process. *Journal of Mathematics and Mechanics*, **6**, 679-684. https://doi.org/10.1512/iumj.1957.6.56038

[30] Moerland, T.M., Broekens, J. and Jonker, C.M. (2018) The Potential of the Return Distribution for Exploration in RL.