Quantum Circuit for Regression Learning with Backpropagation

Natt Luangsirapornchai
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
6372033021@student.chula.ac.th

Prabhas Chongstitvatana
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
prabhas.c@chula.ac.th

Abstract- This work reports the experiment on building quantum circuit for regression learning using qiskit library. The circuit is based on Watabe's work with some modification based on our interpretation. The parameters and results are stored and used in the quantum circuits. We performed the experiments on fitting the linear function and quadratic function. From the experiment, learning the linear function is more accurate than quadratic function within 120 times of training. The initial parameters effect on the number of rounds of training and the accuracy of the result are investigated.

Keywords- Quantum Computing, Quantum Machine Learning

I. INTRODUCTION

Quantum machine learning (QML) was initiated by scientists in 2013 [1]. There are many QML research such as Iris Cong's "Quantum Convolutional Neural Networks"[2], Quantum Neural Network (Fahri E. & Neven H. [3]). This paper provides the result of a modification of quantum learning circuit offered by Masaya Watabe's Team. [4] The backpropagation process is augmented according to our interpretation.

II. METHODOLOGY

There are several ways for making QML circuits. This research provides the result and evaluation from the proposed quantum circuit. The pattern of circuit can be described as:

$$|\psi_{out}\rangle = U(\theta)|\psi_{in}\rangle \quad (1)$$

which $U(\theta)$, $|\psi_{in}\rangle$ and $|\psi_{out}\rangle$ stands for weight matrix, input state, and output state accordingly.

Like other ML training, the backpropagation process is used to regulate the weight matrix of this quantum circuit. It is operated by the chain rule of a partial differential on complex valued vector space. The probability amplitude c, observation probability p, and loss function L are related to the backpropagation equation as :

$$\frac{\partial L}{\partial \theta} = 2\mathrm{Re}\left[\frac{\partial L}{\partial p_\theta^j}\frac{\partial p_\theta^j}{\partial c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}\right]. \quad (2)$$

III. PARAMETER AND CIRCUIT STRUCTURE

We propose a quantum circuit based on Watabe's work [4]. It is a parameterized quantum circuit (ansatz) in which parameters are converted into the circuit. The circuit is composed of 3 sections, Preparation, Quantum parameter network, and Measurement. The preparation circuit is composed of rotational Pauli-Y gate and Pauli-Z gate respectively. The quantum states are set up depending on the input parameters. The quantum parameter network circuit is prepared as a weighted node. This part is composed of two blocks,

978-1-6654-1197-4/21/$31.00 ©2021 IEEE

local unitary blocks and entanglement blocks. The local unitary block which is composed of rotational Pauli-Y gate and Pauli-Z gate. It changes the quantum state depending on applied parameters. The entanglement block makes the entangling state between each qubit. Controlled Z-gate is applied in this block. Both blocks are arranged in alternating order. In some research, this part is called "Quantum Variational Classification" which is used for finding an optimal hyperplane in Support Vector Machine on quantum circuit[5]. Measurement is the last part that reads the output state from the circuit. The overview of the circuit as shown in Fig 1.

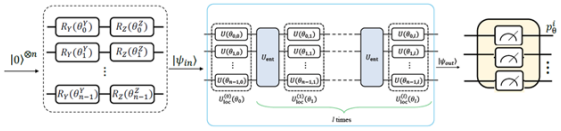


Fig.1 Overview of the Masaya Watabe's quantum circuit [4]

Masaya Watabe's research provides two types of quantum circuit task, regression and classification. In this research, we consider only a regression task. There are two types of parameters that are supplied to this circuit, one is from converted input data and another is variational parameter. In the preparation, one-dimensional input data x is converted into values in range [0, 2π) by the following equations.:

$$\theta^Z = \cos^{-1} x^2$$

$$\theta^Y = \sin^{-1} x \quad (3)$$

After that, the converted input values are fed into the rotational Pauli-Y gate and Pauli-Z gate as parameters in the preparation section. Meanwhile, the variational parameters are first randomized and fed into the quantum parameter network section. These randomized δ parameters are being calibrated in the training process. While this quantum circuit is repeatedly evaluated, the parameters are converged into the appropriate value, a classic deep learning network. When the circuit is measured and given a result, the result will be compared with the target value to find an error. The error is used to calibrate the randomized parameters in the backpropagation process. Although this circuit works similar to a neural network, it is not included because of two significant properties, no activation function and no hidden layer.

IV. ERROR IN REGRESSION

The expected value $\langle Z \rangle$ of this circuit can be described by this equation:

$$\langle Z \rangle = 1 \cdot p_{1,\theta}^{|0\rangle} + (-1) \cdot p_{1,\theta}^{|1\rangle} \quad (4)$$

which $p_{1,\theta}^{|0\rangle}$ and $p_{1,\theta}^{|1\rangle}$ represents the probability of observed 1st qubit state, $|0\rangle$ and $|1\rangle$.

For the regression task, the error δ can be expressed in the form of a partial derivative of the least square loss function which can be written along with the target value f(x) and the expected value $\langle Z \rangle$ as follows:

$$\delta = \frac{\partial L}{\partial \langle Z \rangle} = \left(2\langle Z \rangle - f(x)\right) \quad (5)$$

In another word, this error means comparing target value with the doubled expected value.In the backpropagation section, the following error is applied to make matrices to fix the variational parameters. There are 3 steps of backpropagation operating on the nodes, dot product node, notation gate node, and observation probability node.

1. Dot product node

When the input state vector X with 2nstates (n qubits) pass through the W-weight local unitary block, producing an output state vector Y; its backpropagation, in the term of partial differentiation, can be shown as Fig 2 and expressed in the form of equation as below:

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{W}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \mathbf{Y}}$$
$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{Y}} \cdot \mathbf{X}^{\mathrm{T}} \quad (6)$$

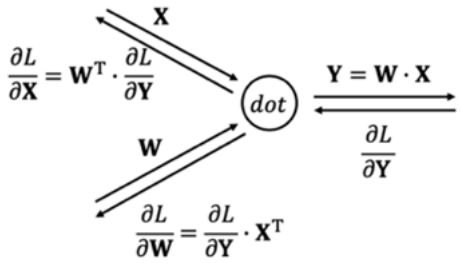


Fig.2 Dot product node operation [4]

2. Rotation gate node

When Parameter Θ is applied into an rotation gate in the local unitary, it is projected into the 2x2 matrix form. So its backpropagation form can be expressed as Fig.3 and below equation:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial u_{1,1}}\frac{\partial u_{1,1}}{\partial \theta} + \frac{\partial L}{\partial u_{1,2}}\frac{\partial u_{1,2}}{\partial \theta} + \frac{\partial L}{\partial u_{2,1}}\frac{\partial u_{2,1}}{\partial \theta} + \frac{\partial L}{\partial u_{2,2}}\frac{\partial u_{2,2}}{\partial \theta} \quad (7)$$

3. Observation Probability node

In the measurement section, the probability amplitude of the output state is converted into an observation probability via repeated evaluation. The backpropagation in this section can be expressed as Fig 4.

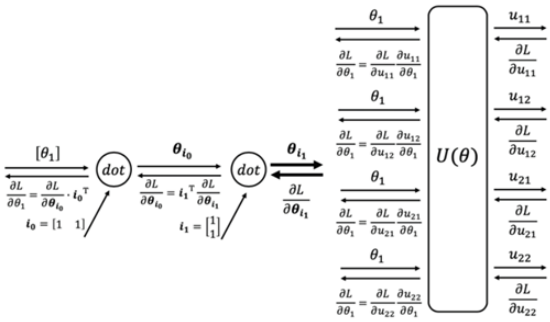


Fig.3 Rotation gate node [4]

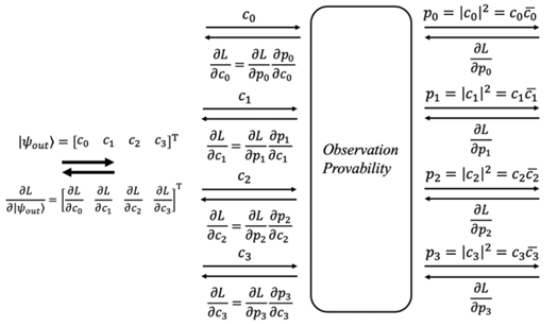


Fig.4 Observation probability node [4]

Which probability amplitude c and observation probabilities p are included. The derivative version is:

$$\frac{\partial p_j}{\partial c_j} = \frac{\partial}{\partial c_j}(c_j \bar{c_j}) = \bar{c_j} \quad (8)$$

V. ERROR CORRECTION

Some of the relation between error value and backpropagation process is not specified in [4]. This section provides the augmented equation from our interpretation.

1. Relation between $\frac{\partial L}{\partial p}$ and expected value

We transform the expected value into the $\frac{\partial L}{\partial p}$ in the observation probability node by the following equation:

$$\frac{\partial L}{\partial p} = \frac{\partial L}{\partial < Z >}\frac{\partial < Z >}{\partial p} \quad (9)$$

which $\frac{\partial <Z>}{\partial p}$ can describe as:

$$\frac{\partial < Z >}{\partial p_\theta^x} = \begin{cases} 1 & p_\theta^x \in p_\theta^{|0\rangle} \\ -1 & p_\theta^x \in p_\theta^{|1\rangle} \end{cases} \quad (10)$$

2. Considering n qubits and 1 qubit in backpropagation

We design 2 types of error form, one expected value and average of expected value from n values. Such as the quantum learning circuit operating on 3 qubits has an error $\frac{\partial L}{\partial p}$with 2^n states as Fig. 5

978-1-6654-1197-4/21/$31.00 ©2021 IEEE

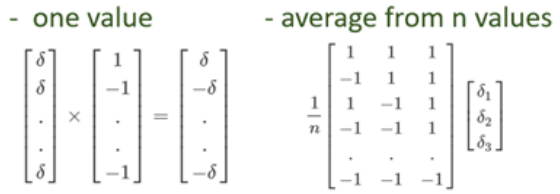


Fig. 5 The form of error $\frac{\partial L}{\partial p}$

VI. EXPERIMENT

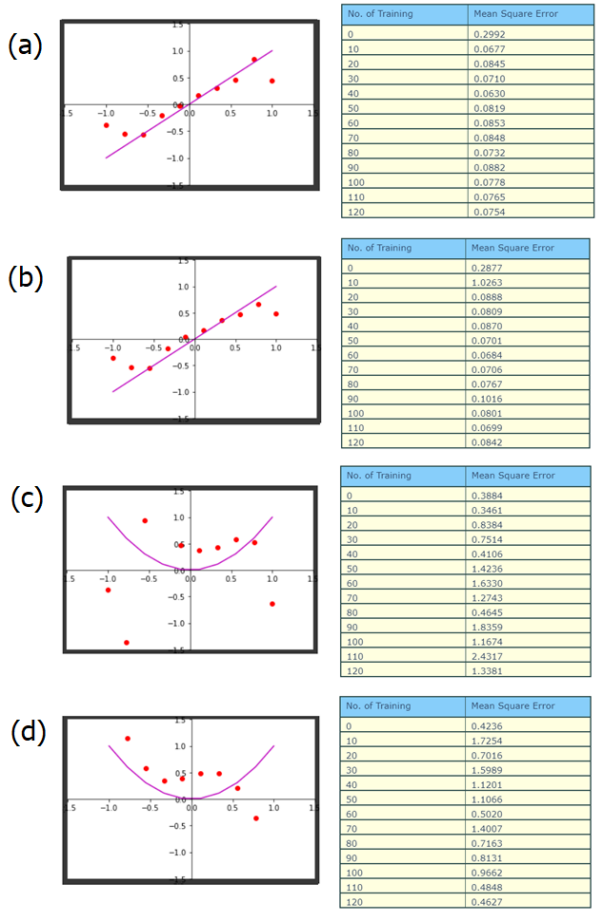
The experiment is conducted with Python, composed of IBM's library framework "QISKIT", numpy library, sympy library, matplot library, and Google Collaboratory as a text editor and simulation. Qiskit is used for building the learning circuit. Numpy and sympy are used for calculating the matrix operation and tensor product in the process of backpropagation. Matplotlib is used for plotting the result of curve fitting. The circuit uses 3 qubits and 3 layers of quantum parameter network (3 entanglement blocks and 4 local unitary blocks). We conduct the training 120 times on linear function: $f(x) = x$ and quadratic function $f(x) = x^2$ which x ∈ [-1,1] and use the 2-type errors. The training is performed around 120 times. Seeding the initiated random variables is used to observe the training affected behavior. We configure the random weight with seed(0). After that, we perform the experiment with random seed.

VII. RESULT

The result of the experiment is shown for the curve fitting problem with number of training and mean square error in Fig.6

The result shows that the circuit conducts the curve fitting to the linear function $f(x) = x$ converge in 120-time training with the Mean Square Error lower than 0.1. In contrary, for the quadratic function $f(x) = x^2$ it is not converged with the Mean Square Error (MSE) higher than 0.4. There is no difference between 2-type errors calculation.

We also show the result of linear function testing with random seed as Fig. 7. .All of them are conducted on the linear function $f(x) = x$.



No. of Training	Mean Square Error
0	0.2992
10	0.0677
20	0.0845
30	0.0710
40	0.0630
50	0.0819
60	0.0853
70	0.0848
80	0.0732
90	0.0882
100	0.0778
110	0.0765
120	0.0754

No. of Training	Mean Square Error
0	0.2877
10	1.0263
20	0.0888
30	0.0809
40	0.0870
50	0.0701
60	0.0684
70	0.0706
80	0.0767
90	0.1016
100	0.0801
110	0.0699
120	0.0842

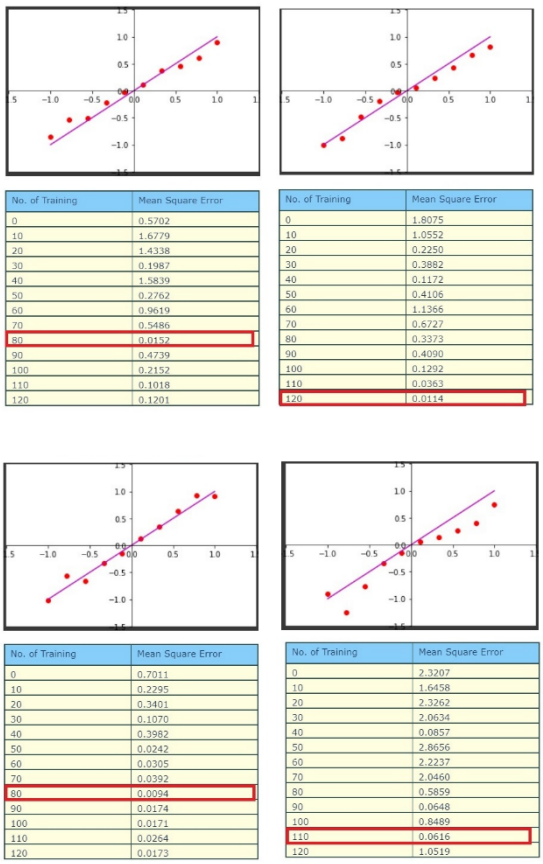
No. of Training	Mean Square Error
0	0.3884
10	0.3461
20	0.8384
30	0.7514
40	0.4106
50	1.4236
60	1.6330
70	1.2743
80	0.4645
90	1.8359
100	1.1674
110	2.4317
120	1.3381

No. of Training	Mean Square Error
0	0.4236
10	1.7254
20	0.7016
30	1.5989
40	1.1201
50	1.1066
60	0.5020
70	1.4007
80	0.7163
90	0.8131
100	0.9662
110	0.4848
120	0.4627

Fig. 6 (a) conducting $f(x) = x$ with one expected value, (b) conducting $f(x) = x$ with average of expected value from n values, (c) conducting $f(x) = x^2$ with one expected value, (d) conducting $f(x) = x^2$ with average of expected value from n values

The result shows that the circuit for the linear function $f(x) = x$ perform well. But the amount of training to get the least value of MSE in each seed pattern is obviously distinct. We can indicate that the circuit spends at least 120 times of training to get the best result for this task.

978-1-6654-1197-4/21/$31.00 ©2021 IEEE



No. of Training	Mean Square Error
0	0.5702
10	1.6779
20	1.4338
30	0.1987
40	1.5839
50	0.2762
60	0.9619
70	0.5486
80	0.0152
90	0.4739
100	0.2152
110	0.1018
120	0.1201

No. of Training	Mean Square Error
0	1.8075
10	1.0552
20	0.2250
30	0.3882
40	0.1172
50	0.4106
60	1.1366
70	0.6727
80	0.3373
90	0.4090
100	0.1292
110	0.0363
120	0.0114

No. of Training	Mean Square Error
0	0.7011
10	0.2295
20	0.3401
30	0.1070
40	0.3982
50	0.0242
60	0.0305
70	0.0392
80	0.0094
90	0.0174
100	0.0171
110	0.0264
120	0.0173

No. of Training	Mean Square Error
0	2.3207
10	1.6458
20	2.3262
30	2.0634
40	0.0857
50	2.8656
60	2.2237
70	2.0460
80	0.5859
90	0.0648
100	0.8489
110	0.0616
120	1.0519

Fig. 7 conducting $f(x) = x$ on the various random seed

Comparing to the result in Masaya Watabe team's work, which the circuit performed the curve fitting with the linear function and the quadratic function well [4] , the differences of experiment may cause from the distinct interpretation or parameter setting or missing calculation. More than 120-time of training is not performed caused of time consuming. However, the better way of backpropagation and error mitigation and the limitation of fitting quadratic function should be discussed.

VIII. CONCLUSION

This paper reported the experiment of quantum learning circuit. We provided the error calculation that are not defined in the original paper, according to our interpretation. We provided the modification of circuit for error calculation. We provided two forms of error, one value and n-value average. We conducted the experiment to test the circuit to perform the curve fitting task. From the experiment, learning the linear function is better than quadratic function in 120 times of training. The initiate weight random parameter effectes the number of round of training. The inaccuracy may occurred from missing calculation or our interpretation of error correcting circuit which would be discussed in the future work.

REFERENCES

[1] S. Lloyd, M. Mohseni and P. Rebentrost, 2013, "Quantum algorithms for supervised and unsupervised machine learning," *arXiv:1307.0411 [quant-ph]*

[2] I. Cong, S. Choi and M. D. Lukin, 2019, "Quantum convolutional neural networks," *arXiv:1810.03787 [quant-ph]*

[3] E. Farhi and H. Neven, 2018, "Classification with Quantum Neural Networks on Near Term Processors," *arXiv:1802.06002 [quant-ph]*

[4] M. Watabe, K. Shiba, M. Sogabe, K. Sakamoto and T. Sogabe, 2019, "Quantum Circuit Parameters Learning with Gradient Descent Using Backpropagation," *arXiv:1910.14266 [quant-ph]*

[5] V. Havlicek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow and J. M. Gambetta, 2018, "Supervised learning with quantum enhanced feature spaces," *arXiv:1804.11326 [quant-ph]*

978-1-6654-1197-4/21/$31.00 ©2021 IEEE