Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 9, Number 1, pages 74–82, January 2020

Quantum Circuit Learning Using Error Backpropagation

Masaya Watabe
Engineering department,
The University of Electro-Communications
Tokyo, Japan

Koudai Shiba
Engineering department,
The Univdrsity of Electro-Communications
Tokyo, Japan

Katsuyoshi Sakamoto
Engineering deptpartment,
The University of Electro-Communications
Tokyo, Japan

Tomah Sogabe *
i-PERC &
Engineering department,
The University of Electro-Communications
Tokyo, Japan;
Grid, Inc.
Tokyo, Japan
sogabe@uec.ac.jp

***Abstract*— Quantum computing has the potential to outperform classical computers, and is expected to play an active role in various fields. On quantum machine learning, it is difficult to learn only on quantum computing. Classical-quantum hybrid algorithms are proposed in recent years. Classical computer is used for calculation of parameter tuning in quantum circuit. In this paper, we propose a backpropagation algorithm that can efficiently calculate gradient in optimization of parameter in quantum circuit, which outperforms the current parameter search algorithm while presents the same or even higher accuracy.**

Keywords—quantum computing, machine learning, error backpropagation, gradient

I. Introduction

There is the famous Di Vincenzo criterion for the question "What are the standards for quantum computers" or "What are the elements necessary for a" true "quantum computer?" [1]. Di Vincenzo criterion contains seven criteria and each one is being steadily cleared by the development of technology in recent years, for example, qubits and quantum gates have increased from tens to thousands with the development of superconducting technology, spin control technology, and microwave resonance technology. The Di Vincenzo criterion is important for all items, but the most difficult thing from the results of recent research and development is the third criterion, "Coherence time continues until quantum computation is completed". This condition has a deep meaning, and it can be said that it is a life-and-death problem of a quantum computer that is questioned by the completeness of physical conditions under which quantum superposition and quantum entanglement are the key elements of a quantum computer.

Quantum coherence refers to a coordinated and precise movement of qubits. However, in reality, qubits are very fragile and are subject to errors due to the phenomenon of decoherence. This is the biggest reason why it is difficult to realize quantum computers. It is difficult to precisely control the quantum state such as spin as expected, and bit flip and phase inversion occur due to the fluctuation in surrounding environment and noise. Such an error is called a quantum error. The number of quantum errors that potentially exist or the number of quantum errors that occur in a stochastic normal qubit is an important parameter that affects quantum computers, and has been actively studied in recent years. Quantum computer which possess considerable quantum errors, is called Noisy-Intermediate Scale Quantum computer (NISQ)[2]. Under the NISQ circumstance, it is necessary to develop fault-tolerant quantum computation methods that provide error resilience. There are two solutions to this problem. One is to perform quantum computing while correcting quantum errors in the presence of errors. Another approach is to develop hybrid quantum-classical algorithm which complete the quantum computing before the quantum error becomes fatal and shift the rest of task to classical computer when severe quantum error occurs. The latter approach has triggered a lot of algorithm such as quantum approximation optimization algorithm (QAOA) [3] and variation quantum eigensolver (VQE) [4] and many others [5].

A hybrid quantum-classical algorithm needs to build an efficient simulation channel to organically connect 'the quantum and the classic'. In QAOA, VQE or other hybrid NISQ algorithm, there exists a challenging task to optimize the model parameter. In a complete classical approach, the optimal parameter search is usually categorized as an mathematical optimization problem, where various approaches bother gradient based and non-gradient based have been widely utilized. For the quantum circuit learning, so far most of parameter searching algorithm are based on non-gradient ones such as Nelder–Mead method [6], SPSA [7] and most recently a difference method[8].

In this article, we propose an error backpropagation algorithm on quantum circuit learning to efficiently calculate the gradient required in parameter optimization. As mentioned above, current approaches to solve the parameter search is based on gradient free, which inevitably causes the execution time to

increase significantly as the number of quantum gates using in the quantum circuit increases. The error backpropagation method is an efficient method for calculating gradients in the field of deep neural network based machine learning when updating parameters using the gradient descent method [9]. By carefully examining the simulation process of a quantum circuit, if the input quantum state is $|\psi_{in}\rangle$ and a certain quantum gate $U(\theta)$ is applied, the output state $|\psi_{out}\rangle$ can be expressed by the dot product of the input state and quantum gate.

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

On the other hand, the calculation process of a fully connected neural network without activation function can be written as $\mathbf{Y} = \mathbf{W} \cdot \mathbf{X}$, where $\mathbf{X}$ is the input vector, $\mathbf{W}$ is the weight matrix of network, and $\mathbf{Y}$ is the output. It can be seen that the quantum gate $U(\theta)$ is very similar to the network weight matrix $\mathbf{W}$. This shows that backpropagation algorithms that is used for deep neural networks can be modified and be used in the simulation process of quantum circuit learning.

The method we proposed makes it possible to greatly reduce the time for gradient calculation when the number of qubits is increased or number of gates is increased. As a result, it is expected that using gradient based backpropagation in the NISQ hybrid algorithm facilitate parameter search when many qubits and deeper circuits are deployed.

II. Quantum Backpropagation Algorithm

The backpropagation method uses chain rule of a partial differential to propagate the gradient back from the network output and calculate the gradient of the weights. Owing to the chain rule, the backpropagation can be done only at the input / output relationship at the computation cost of a node. In the simulation of the quantum computing, the quantum state $|\psi\rangle$ and the quantum gates are represented by complex value. Hereafter, we will show the derivation details regarding the quantum backpropagation in complex valued vector space.

When the input of n qubits is $|\psi_{in}\rangle$ and the quantum circuit parameter network $W(\theta)$ is applied, the output $|\psi_{out}\rangle$ can be expressed as:

$$W(\theta)|\psi_{in}\rangle = \sum_{j=0}^{2^n-1} c_\theta^j |j\rangle = |\psi_{out}\rangle \quad (1)$$

where c_θ^j is the probability amplitude of state $|j\rangle$ and $\left|c_\theta^j\right|^2 = p_\theta^j$ is the observation probability of state $|j\rangle$. If loss function L can be expressed by using observation probability determined by quantum measurement, the gradient of the learning parameter can be described as:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial p_\theta^j} \cdot \frac{\partial p_\theta^j}{\partial \theta} \quad (2)$$

Since

$$p_\theta^j = \left|c_\theta^j\right|^2 = c_\theta^j \overline{c_\theta^j} \quad (3)$$

where $\overline{c_\theta^j}$ is the conjugate of c_θ^j, Therefore the gradient of observation probability can be further expanded as :

$$\frac{\partial p_\theta^j}{\partial \theta} = \frac{\partial c_\theta^j \overline{c_\theta^j}}{\partial \theta} = \overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta} + c_\theta^j \frac{\partial \overline{c_\theta^j}}{\partial \theta} \quad (4)$$

Formula(4) can be further expanded as:

$$\overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta} + c_\theta^j \frac{\partial \overline{c_\theta^j}}{\partial \theta} = \overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta} + \overline{\overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}} \quad (5)$$

Formula (5) contains complex value but can be nicely summed out as real value shown below:

$$\overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta} + \overline{\overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}} = 2\mathrm{Re}\left[\overline{c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}\right] = 2\mathrm{Re}\left[\frac{\partial p_\theta^j}{\partial c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}\right] \quad (4)$$

Therefore,

$$\frac{\partial L}{\partial \theta} = 2\mathrm{Re}\left[\frac{\partial L}{\partial p_\theta^j}\frac{\partial p_\theta^j}{\partial c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}\right]. \quad (5)$$

$\frac{\partial L}{\partial p_\theta^j}\frac{\partial p_\theta^j}{\partial c_\theta^j}\frac{\partial c_\theta^j}{\partial \theta}$ can be obtained by error backpropagation in the same way as the conventional calculation used in deep neural network [9]. Meanwhile, one advantage of the proposed method is that quantum gate matrix containing complex value is converted to real value. Gradient of the loss function with respect to θ can be obtained from the real part of the value of the complex vector space calculated by the conventional backpropagation. For more detailed derivation regarding backpropagation at each node using a computation graph. Is given in the Appendix for reference.

III. Experiment

Next we conducted the experiment with the supervised learning tasks including both regression and classification problems to verify the validity of proposed quantum backpropagation algorithm.

The quantum circuit consists of a unitary input gate $U_{in}(\boldsymbol{x})$

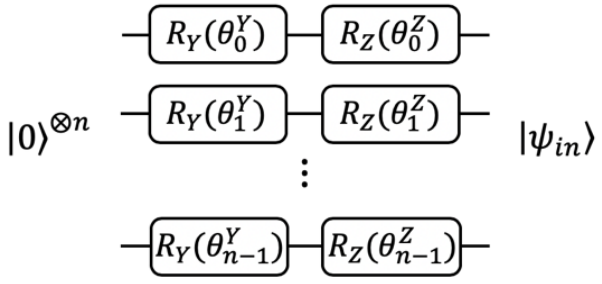

Fig.1 a unitary input gate $U_{in}(\boldsymbol{x})$

that creates an input state from classical input data $\boldsymbol{x}$ and a unitary gate $W(\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$. We use $U_{in}(\boldsymbol{x}) = \otimes_{j=0}^{n-1} R_Z(\theta_j^Z) R_Y(\theta_j^Y)$ as proposed in reference [8] a unitary input gate. (Fig. 1)

We use $W(\boldsymbol{\theta}) = U_{\text{loc}}^{(l)}(\theta_l)U_{\text{ent}} \cdots U_{\text{loc}}^{(1)}(\theta_1)U_{\text{ent}}U_{\text{loc}}^{(0)}(\theta_0)$ as proposed in reference [10] $U_{\text{loc}}^{(k)}(\theta_k) = \otimes_{j=0}^{n-1}U(\theta_{j,k})$. U_{ent} is entangling gates. We use controlled-Z gates (CZ) as U_{ent}. The overall quantum circuit is shown in Fig.2:

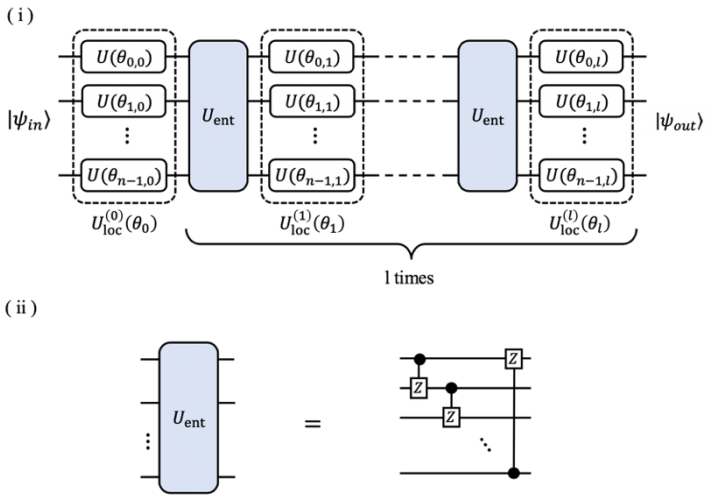

Fig. 2 (i) $W(\boldsymbol{\theta})$ is the variational form. l denotes depth of quantum circuit. (ii) U_{ent} gate is composed of CZ gates from qubit j to qubit (j+1) mod n, j $\in \{0, \ldots, n-1\}$.

A. *Regression*

In regression tasks, the circuit parameters were set to n = 3 and l = 3, that is to say, the number of qubit is 3 and depth of circuit is 4. The expected value of observable Z for the first qubit was obtained from the output state $|\psi_{out}\rangle$ of the circuit. One-dimensional data x is input by setting circuit parameters as

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

The target function $f(x)$ was regressed with the output of twice of the Z expected value. We performed nonlinear regression tasks to verify the effectiveness of the proposed approach. A conventional least square loss function was adopted in the current regression tasks.

$$L = \frac{1}{2}|2\langle Z\rangle - f(x)|^2 \qquad (6)$$

And the first deriviation becomes:

$$\delta = \frac{\partial L}{\partial p} = \frac{\partial L}{\partial \langle Z\rangle}\frac{\partial \langle Z\rangle}{\partial p} = \left(2\langle Z\rangle - f(x)\right)\frac{\partial \langle Z\rangle}{\partial p} \qquad (7)$$

Here, $\langle Z\rangle = 1 \cdot p^0 + (-1) \cdot p^1$. The error δ is the one for the backpropagation.

Before conducting the nonlinear regression, we have confirmed the validity of the proposed algorithm for linear regression task, which is not shown here but presented in the Appendix part for reference. In Fig.3(a) and (b) shows the two nonlinear task $f_1(x) = x^2$, which represent a single concave profile nonlinear problem, and $f_2(x) = \sin x$, which represents multi-concave-convex wavy profile for more complex

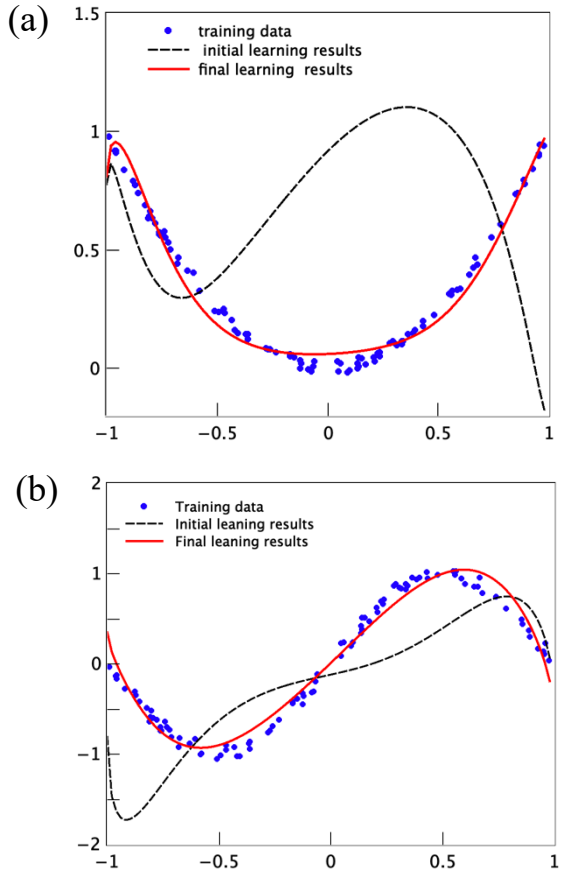

Fig. 3 (a) Regression results for target function $f_1(x) = x^2 + 0.015N(0,1)$, (b) Regression results for target function $f_1(x) = sin\, x + 0.015N(0,1)$,

problems. The noise was also added into the target function for realistic purpose and the number of training data was chosen as 100 in circuit learning for the two target functions. It can be clearly seen the quantum circuit based on error backpropagation performs very well in the regression task. At the initial learning stage, the results show large deviation from the target function and at the final leaning stage the regressed curve catches the main feature of the training data and shows very reasonable fitted curve. It is noticed at the Fig.3(a), the fitted curve showed deviation at the left edge of the regression profile. This deviation is considered as lack of training data at the boundary and can be either improved by increasing the number of training data or adding regularization term in the loss function, which are regularly used in the conventional machine learning tasks.

B. *Classification*

In the classification problem, we have modified the quantum circuit architecture to accommodate increased number of parameters for both qubit and circuit depth. The initial parameter set for classification problem were set for qubit n=4 and l=6 (number of layer is 7). Again, here we show only the results for nonlinear classification problem. The example of bbinary classification of the two-dimensional data are shown in Fig.7. Here the dataset was prepared by referring to the similar dataset

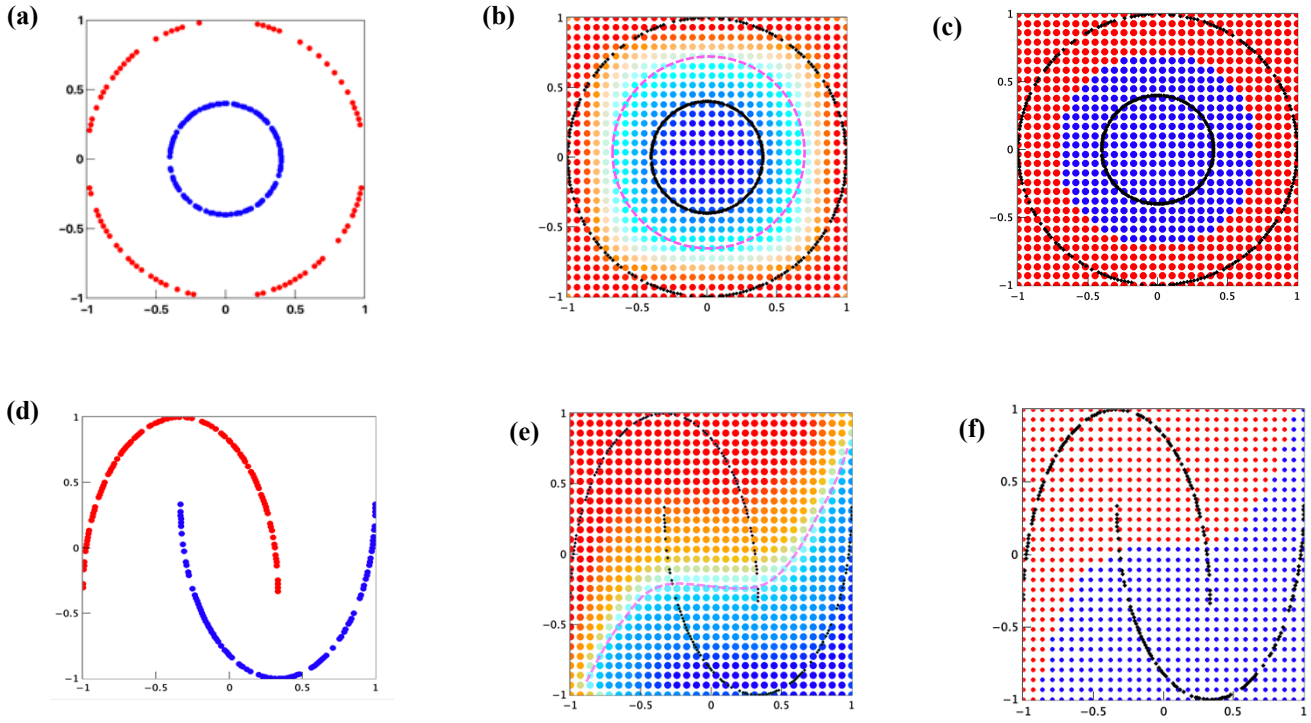

Fig. 4 Quantum circuit learning results using error backpropagation for nonlinear binary classification problem with 4 qubit and 7 layer depth: (a) Training data set for make_circles, red for label '0' and blue for label '1' ; (b) Test results using the learnt parameter using the 200 make_circles dataset, pink line corresponding to the median boundary of the continuous probability; (c) scikit-learn-SVM classification results using the learnt support vectors; (d) Training data set for make_moons, red for label '0' and blue for label '1' ;; (e) Test results using the learnt parameter under the 200 make_moon dataset, pink line corresponding to the median boundary of the continuous probability; (f) scikit-learn-SVM classification results using the learnt support vectors.

from scikit-learn[11]. We consider two representative nonlinear examples: one is dataset of make_circles and another one is make_moons and we consider the make_moons possess more complicated nonlinear feature than make_circles.

For the output state $|\psi_{out}\rangle$, we calculated the expected value $\langle Z_1\rangle$ and $\langle Z_2\rangle$ of observable Z using the first and second qubits. A softmax function was applied to the output for $\langle Z_1\rangle$ and $\langle Z_2\rangle$ and continuous probability between 0 and 1 is thus obtained. For the training purpose, a typical cross-entropy loss function was adopted to generate the error and was further backpropagated to update the learning parameter.

$$L = d_i \log[p] + (1 - d_i) \log[1 - p] \qquad (8)$$

The cross-entropy formula looks complicated but its first derivative upon probability p reduced to the form of error similar to the regression.

$$\delta = \frac{\partial L}{\partial p} = p - d_i \qquad (9)$$

For the proof of concept, limited number of training data was set as 200 and half of the data was labeled as '0' , the rest half of data was labeled as '1'. For comparison we have also applied the classical support vector machine (SVM), a toolkit attached in the scikit-learn package to the same datasets. The results from SVM are served as a rigorous reference for the validity verification of the proposed approach.

Two-dimensional data $\boldsymbol{x}$ is input by setting circuit parameters as:

$$\begin{cases} \theta_{2i}^{Z} = \cos^{-1} {x_1}^2, \\ \theta_{2i}^{Y} = \sin^{-1} x_1 \text{ or} \\ \theta_{2i-1}^{Z} = \cos^{-1} {x_2}^2, \\ \theta_{2i-1}^{Y} = \sin^{-1} x_2. \\ (i = 0, 1, \ldots, n-1) \end{cases}$$

Fig.4 shows the learnt results for two non-linear classification tasks. Fig.4 (a) and (e) show 2-dimensional training data with value ranged between (-1,1) were chosen as the training dataset. Here the noise was not added for simplicity and the training data with added noise will be presented elsewhere showing the similar tendency as reported here. Fig.4(b) shows the test results based on the parameter using the dataset from Fig.4(a). A multicolored contour-line like classification plane was found in Fig.4(b). The multicolored value corresponding to the continuous output of the probability from the sofmax function. a typical two-valued region can be easily determined by taking the median of the continuous probability as the classification boundary. This is shown in the Fig.4(b) with line colored by pink. Reference results simulated using scikit-learn-SVM is shown in Fig.4(c). Since SVM simulation treats the binary target discretely, the output shows exact two value based colormap of the test results. It can be easily seen here that the results shown

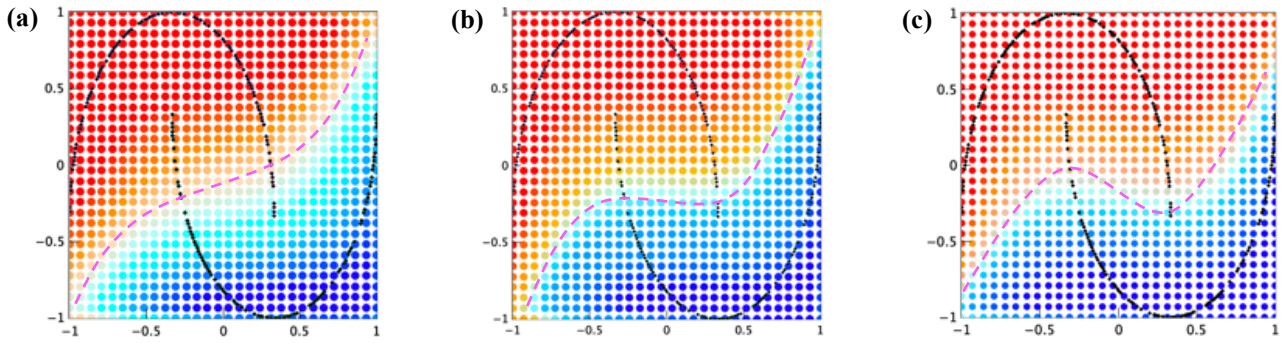

Fig. 5 Effect of quantum circuit depth on the classification accuracy. (a) 4 layer of quantum circuit with 4 qubits. (b) 7 layer of quantum circuit with 4 qubits ; (c) 10 layer of quantum circuit with 4 qubits

in Fig.4(b) is in great consistence with the SVM results. Especially the location of median boundary (pink line) corresponds exactly to the SVM results. For the dataset of make_moons, the situation become more complicated due to the increased nonlinearity in the training data. Fig.4(d), (e) and (f) showed the same simulation sequence as the data of make_circles. However, it is found both the results from error backpropagation and SVM showed misclassification. The classification mistake is usually occurred near the terminal edge area where the label '0' and label '1' overlapped with each other. Taking a closer look at the test results shown in Fig.4(e) and (f), it can be found that the misclassification presented in a different manner. For quantum circuit learning, the misclassification occurs mostly at the left side of label '0' in the overlapping area. For SVM, the misclassification is roughly equally distributed for both label '0' and label '1' indicating the intrinsic difference between these two simulation algorithms. Further investigation aiming at improving the test accuracy for the make_moons data were also conducted and the results are shown in Fig.5. We consider that one of reason for misclassification occurred in Fig.4(e) would be attributed to the limited representation ability due to limited depth of quantum circuit. This can be confirmed from Fig.5 where the quantum circuit with varied layer thickness ranging from 4, 7 and 10 are given. It can be seen that the by, 4 layer of circuit showed almost linear separation plane, but with the increase of the circuit layer thickness, the classification boundary (separation plane) becomes more nonlinear. The layer thickness of 10 showed great improvement of separation plane to reflect the hidden 'moon' feature of the training data. It should be mentioned here that the number of training data is kept at 200. It is apparent that classification accuracy will dramatically increase with increase of number of training data. Here we intentionally reduced the number for training data set so as to magnify the effect from quantum circuit depth.

C. Computation efficiency.

After having confirmed the validity of the proposed backpropagation on various regression and classification problem, we will show one great advantage of using backpropagation to perform parameter optimization over other approaches. It has been rigorously in deep neural network based machine learning field that the error backpropagation method shown several magnitude faster than finite difference method. In this work, we have also conducted benchmark test to verify where there is decisive advantage of using backpropagation algorithm in quantum circuit learning. Fig.6 shows the computation cost comparison among three methods: a finite difference method proposed in reference [8], the popular Nelder-Mead method from SciPy, which is mostly widely used in current quantum circuit leaning field and the proposed method based on backpropagation. This comparison was performed under 4 qubit. The horizontal axis is the circuit depth l, and the vertical axis is the execution time[sec] per 100 iterations. The number of parameters corresponding to the circuit depth l is given as :

$$N_{parameter} = (2\ rotation\ gates) \times (4qubit) \times (l+1).$$

We implemented the three method on the same make_moons dataset and record the computation time costed per 100 iterations. The depth of quantum circuit is varied from 5 to 20 at the interval of 5. It can be clearly seen there is dramatic difference in computation time costed for 100 iteration learning steps. Finite-difference method showed the worst computation

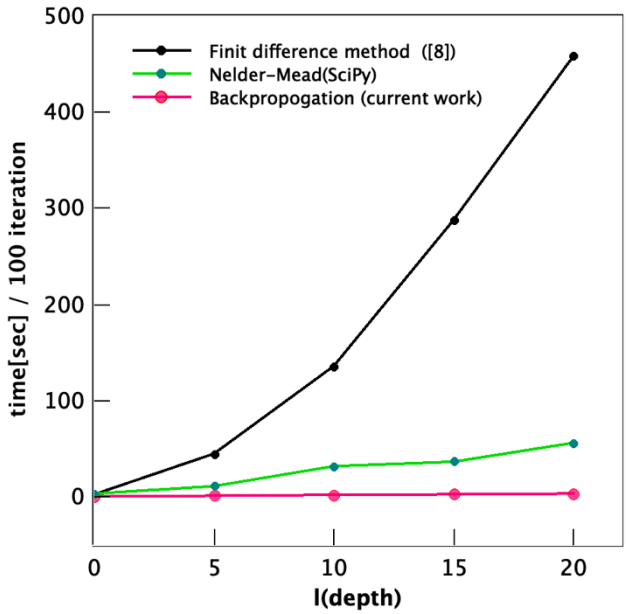

Fig. 6 Comparison of computation cost for different approaches .

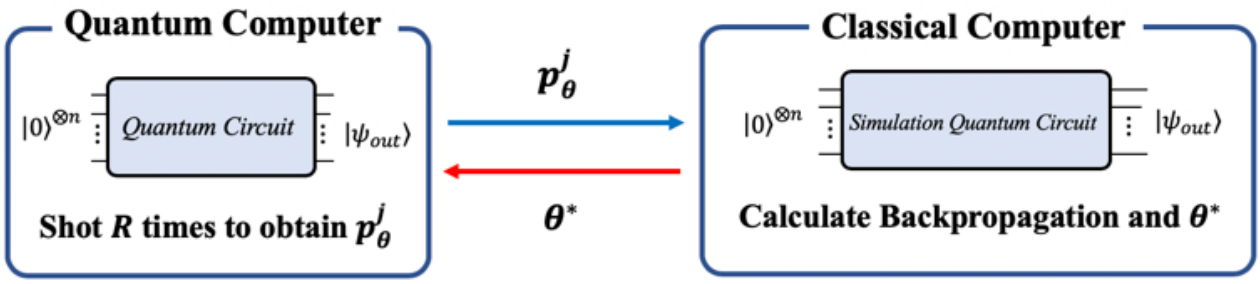

Fig.7 Implementation architecture of error backpropagation-based quantum circuit learning on the real NISQ type quantum computer

efficiency, as has been mentioned above and demonstrated in deep neural network related machine learning field. The computation cost rises exponentially as the thickness of the circuit increases, limiting its application based on its current form. The Nelder-Mead method is much more computationally efficient than finite difference method by showing a linear increase of computation cost for each 100 iteration learning steps. Since the computation is recorded at every 100 iterations. for a task like the make_moons dataset,

In contrast, the backpropagation method proposed here showed dramatic advantage over all other methods by shown an almost constant dependency on the depth of quantum circuit. The computation time is recorded at depth of 20 layer as 3.2 seconds, which is almost negligible when compared to the recorded value at the same 20 layer thickness : 56 seconds by using Nelder-Mead method and 458 seconds by using Finite difference method.

D. Impltemention scheme on real quantum computer.

So far we have focused our results on the simulation using quantum simulator. Implementing architecture when using a real machine such as NISQ type quantum is described in Fig.7. In order to use the error backpropagation method, a quantum state $|\psi\rangle$ is required to get prepared at the initial stage. Therefore, as shown in the figure, a quantum circuit having the same configuration as the real quantum circuit must be prepared as a quantum simulator on classical computer. It should be noticed here that his could not be considered as additional load for the quantum computing scientist since for fabricating a quantum computer at the hardware base, it needs its counterpart of quantum circuit simulator to monitor and diagnose the qubits and gate error. This is especially true since a quantum computer is not allowed to be disturbed during the working condition unlike the classical computer. Therefore, for a real quantum computer, it always requires a quantum simulator ready for use at any time. That means we can always access to the quantum simulator as shown at the right side of Fig.7 to examine and obtain detailed information regarding the performance of corresponding real quantum computer. Observation probability for each state $|\psi_j\rangle$ can be calculated by shooting R times at the real quantum computer side. The observation probability obtained from the real quantum machine is then passed to the classical computer, and the quantum circuit in the simulator for simulation is then used. The parameter θ can be updated using backpropagation since all the intermediate information is available at the simulator side. After the parameterθ^* is updated at the simulate side, it will return to the real quantum machine for next iteration quantum simulation. The implementing the backpropagation will be reported elsewhere.

IV. Conclusion

We proposed a backpropagation algorithm for quantum circuit learning. The proposed algorithm showed success in both linear and nonlinear regression and classification problem. Meanwhile dramatic computation efficiency by using the error backpropagation based gradient circuit learning rather than the gradient free method such as Finite difference method or Nelder-Mead method. The reduction of computing time is surprisingly up to several magnitude high when compared to the conventional method. A backpropagation embedded quantum circuit paves the path toward large scale and deep quantum circuit learning for complicated feature extraction problem and intractable optimization problem by designing quantum advantage oriented hybrid NISQ algorithm.

References

[1] DiVincenzo *Fortschritte der Physik*. 48 (9–11): 771–783 (2000)
[2] John Preskill, arXiv:1801.00862 (2018)
[3] Peruzzo, A. *et al. Nature Commun.* **5**, 4213 (2014).
[4] Farhi, E., et al., arxiv.org/abs/1411.4028 (2014).
[5] K, Shiba,et al., arXiv:1906.01196(2019)
[6] Nelder, John A.; R. Mead Computer Journal , 7,313 (1965).
[7] Spall, J. C. IEEE Transactions on Auto.Cont., 37, 332(1992).
[8] Mitarai, K. arxiv.org/abs/1803.00745 (2018).
[9] Rumelhart, David E. et al., Nature. 323 (6088): 533–536 (1986)
[10] Vojtch Havl´ıˇcek, et al., Nature 567, 209–212 (2019).
[11] Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

v. APPENDIX

A. *dot product node*

Backpropagation of dot product node.

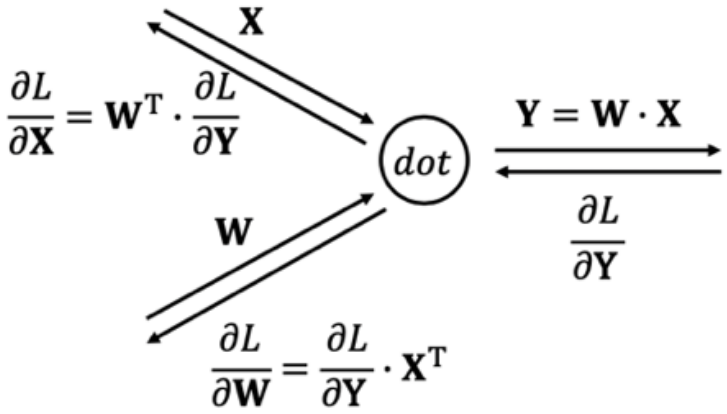

Fig.1 Graph of dot product node.

X is input state. The size of **X** corresponds to $\mathrm{N}(=2^n)$ of states with n qubits. **W** is the network weights and corresponds to a gate in the quantum circuit. The size is N × N.

$$\mathbf{X} = (x_1, x_2, \dots, x_N)^{\mathrm{T}}$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,N} \\ \vdots & \ddots & \vdots \\ w_{N,1} & \cdots & w_{N,N} \end{bmatrix}$$

Output **Y** is written as $\mathbf{Y} = \mathbf{W} \cdot \mathbf{X}$. Here, when there is a gradient $\frac{\partial L}{\partial \mathbf{Y}}$ with respect to **Y** of the loss function L, the gradient of each L with respect to **X** and **W** is calculated as follows.

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

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

B. *rotation gate $\boldsymbol{U}(\theta)$ node*

The weight **W** in the full-connected network is made to correspond to the rotation gate in the quantum circuit.

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} u_{1,1}(\theta) & u_{1,2}(\theta) \\ u_{2,1}(\theta) & u_{2,2}(\theta) \end{bmatrix} = \boldsymbol{U}(\theta)$$

Unlike conventional full-connected networks, the elements of the unitary gate $\boldsymbol{U}(\theta)$ matrix are not independent of each other and have a common parameter θ. Therefore, the gradient of L with respect to θ is obtained by backpropagation using the calculation graph shown in Fig. 2.

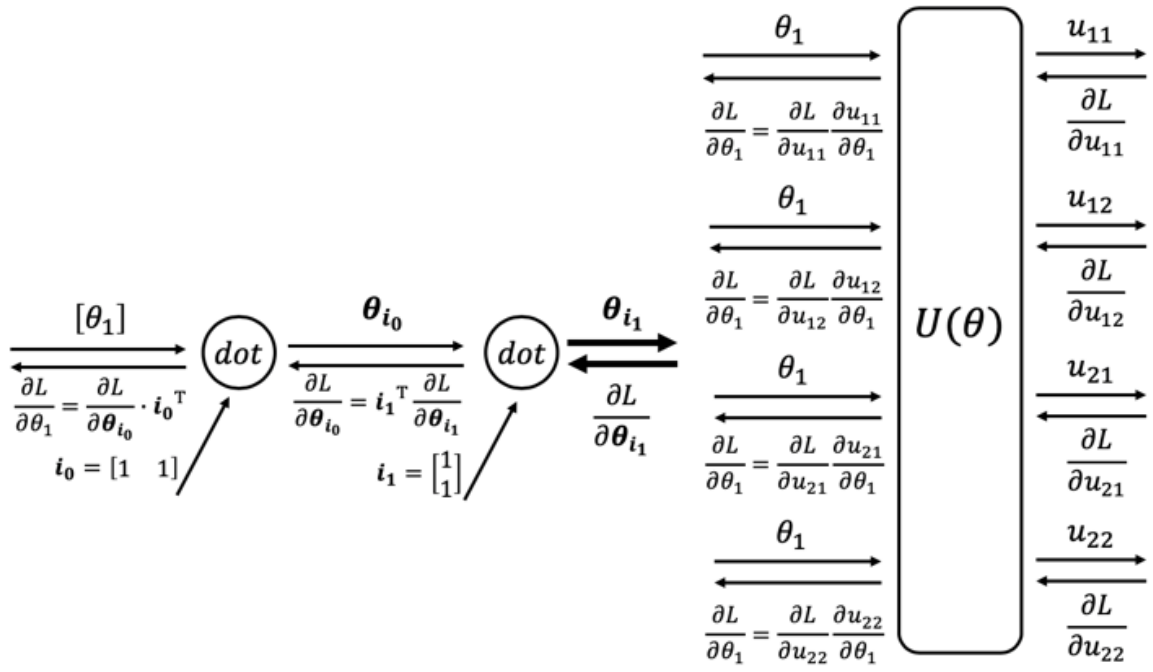

Fig. 2 Graph of rotation gate $\boldsymbol{U}(\theta)$ node

For example, we chose R_Y gate,

$$\boldsymbol{U}(\theta) = \mathbf{Ry}(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix},$$

$$\frac{\partial \mathbf{Ry}(\theta)}{\partial \theta} = \begin{bmatrix} \frac{\partial u_{1,1}}{\partial \theta} & \frac{\partial u_{1,2}}{\partial \theta} \\ \frac{\partial u_{2,1}}{\partial \theta} & \frac{\partial u_{2,2}}{\partial \theta} \end{bmatrix} = \begin{bmatrix} -\sin\frac{\theta}{2} & -\cos\frac{\theta}{2} \\ \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \end{bmatrix},$$

We define vector $\boldsymbol{i_0} = [1 \quad 1]$ and $\boldsymbol{i_1} = \boldsymbol{i_0}^{\mathrm{T}}$. The gradient is

$$\frac{\partial L}{\partial \boldsymbol{\theta_{i_1}}} = \begin{bmatrix} \frac{\partial L}{\partial \theta} & \frac{\partial L}{\partial \theta} \\ \frac{\partial L}{\partial \theta} & \frac{\partial L}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial u_{1,1}} \frac{\partial u_{1,1}}{\partial \theta} & \frac{\partial L}{\partial u_{1,2}} \frac{\partial u_{1,2}}{\partial \theta} \\ \frac{\partial L}{\partial u_{2,1}} \frac{\partial u_{2,1}}{\partial \theta} & \frac{\partial L}{\partial u_{2,1}} \frac{\partial u_{2,1}}{\partial \theta} \end{bmatrix},$$

$$\frac{\partial L}{\partial \boldsymbol{\theta_{i_0}}} = \boldsymbol{i_1}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \boldsymbol{\theta_{i_1}}},$$

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \boldsymbol{\theta}_{i_0}} \cdot \boldsymbol{i_0}^{\mathrm{T}}.$$

From above,

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \boldsymbol{\theta}_{i_0}} \cdot \boldsymbol{i_0}^{\mathrm{T}} = \left(\boldsymbol{i_1}^{\mathrm{T}} \cdot \frac{\partial L}{\partial \boldsymbol{\theta}_{i_1}}\right) \cdot \boldsymbol{i_0}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial u_{1,1}} \frac{\partial u_{1,1}}{\partial \theta} + \frac{\partial L}{\partial u_{1,2}} \frac{\partial u_{1,2}}{\partial \theta} + \frac{\partial L}{\partial u_{2,1}} \frac{\partial u_{2,1}}{\partial \theta} + \frac{\partial L}{\partial u_{2,2}} \frac{\partial u_{2,2}}{\partial \theta}$$

C. Observation Probability node

Output state $|\psi_{out}\rangle$ is

$$|\psi_{out}\rangle = c_0|0\rangle + c_1|1\rangle + \cdots + c_{N-1}|N-1\rangle,$$

where c_j is the probability amplitude and satisfies $|c_0|^2 + |c_1|^2 + \cdots + |c_{N-i}|^2 = 1$.

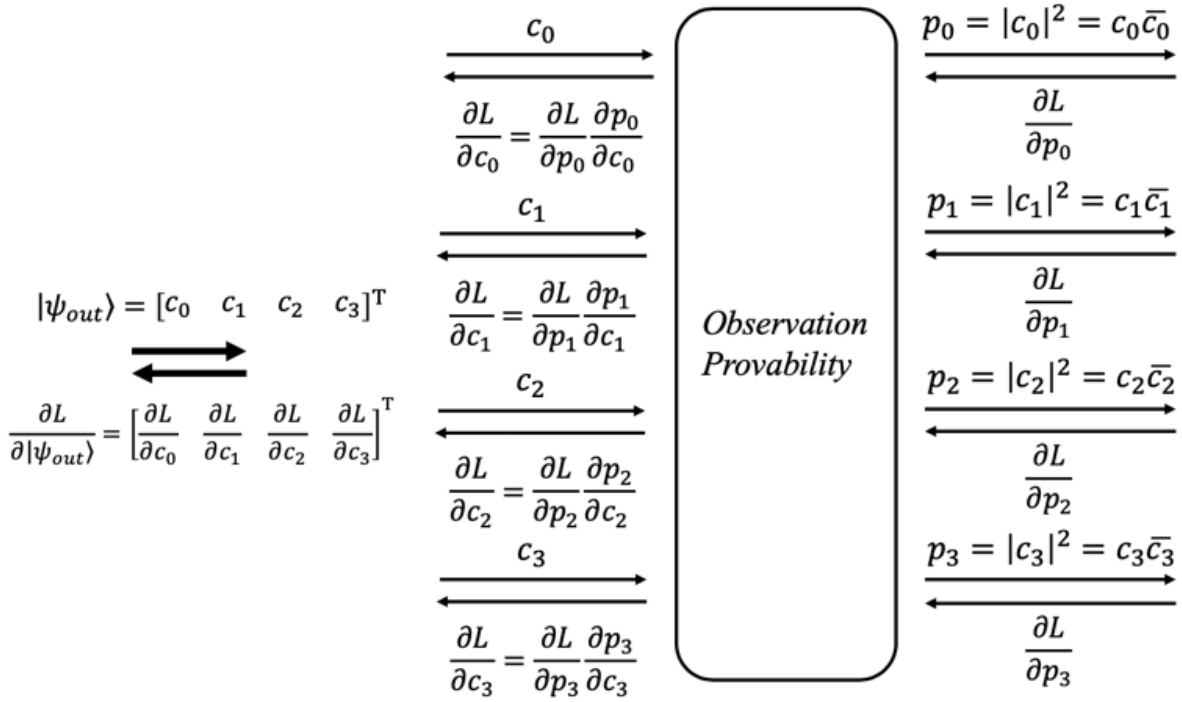

Fig. 3 Graph of the probability amplitude node

The gradient of p_i with respect to a probability amplitude c_j can be calculated as follows. $\bar{c_i}$ is the complex conjugate of c_j.

$$\frac{\partial p_j}{\partial c_j} = \frac{\partial}{\partial c_j}\left(c_j \bar{c_j}\right) = \bar{c_j}$$