CALCULATING GRADIENTS IN QUANTUM NEURAL NETWORK

F08222011 Chen, Yi-An
Department of Physics
National Taiwan University
r08222011@gmail.com

ABSTRACT

It is hard to calculate the gradient for Variational Quantum Circuit (VQC) with classical differentiation method. Especially in NISQ era, the noise of gates or measurements will cause a huge error on classical differentiation method. We use Parameter-Shift Rule (PSR) proposed by Crooks [2019] to calculate the gradient in a more quantum-efficient way.

Keywords Quantum Machine Learning · Variational Quantum Circuit · Gradient Descent

1 Introduction

Deep learning have been used in many fields, such as data classification, Natural Language Processing (NLP), regression, etc. Those different problems use different structures of deep neural network, but the basic ideas are the same: reducing the loss function by tuning parameters. There are several ways to reduce the loss, probably the most familiar one is to use Gradient Descent (GD) method. GD requires the gradient of the tunable parameters (training parameters), and it can be calculated easily with classical computers, e.g. finite difference method or back propagation.

In recent years, researchers have tried to use quantum computers to implement machine learning. The most intuitive way is to implement machine learning using quantum circuits, one of the example is to implement deep neural network with a quantum computer, which is the basic idea of "Variational Quantum Circuit" (VQC) by Peruzzo et al. [2014]. However, if we want to implement "Gradient Descent Method" with a quantum computer, it's hard to calculate the gradients in NISQ era with classical differentiation method such as finite difference method. We now introduce a method called "Parameter-Shift Rule" (PSR) proposed by Crooks [2019]. Unlike most of the classical differentiation method, PSR gives an analytic solution based on the natural structure of Pauli rotation gates. In the following sections, we give a simple derivation of PSR.

2 Problem and Methodology

In this section, we first briefly review the basic ideas of classical neural network. Then we will see how we implement neural network on a quantum computer, e.g. VQC, using quantum circuits. Finally, we use PSR to calculate the gradients.

2.1 Review of Classical Neural Network

Neural network consists of neurons (circles on Fig.1), briefly speaking, one can think of the number of neurons as the ability or complexity of the function we can learn. Each neuron consists of certain number of parameters θ, we call them tunable parameters or trainable parameters, or simply parameters if there is no confusion in the context. The initial values of parameters is randomly chosen, e.g., can be generated from uniform distribution or Gaussian distribution.

In the following sections, we deal with classification problems. Suppose we have a dataset $\{(x^i, y^i)\}$ for i from 1 to N, where x^i is the feature vector of i-th data, x^i can be multi-dimension (the features will be denoted as x^i_1, x^i_2, ...). The y^i is usually called the true value or label of the i-th data. When we feed a data (x^i in Fig.1) into the neural network, it will feed forward the data and output a predicted value depends on the parameters θ. One can define a loss function to

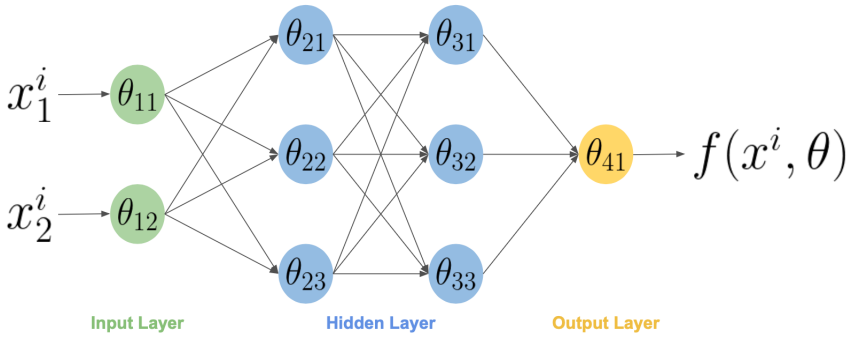

Figure 1: Basic structure of a classical neural network. The circles are the neurons, each neuron will have one or more tunable (trainable) parameters. When feeding in a data x^i, it will output a value $f(x^i, \theta)$

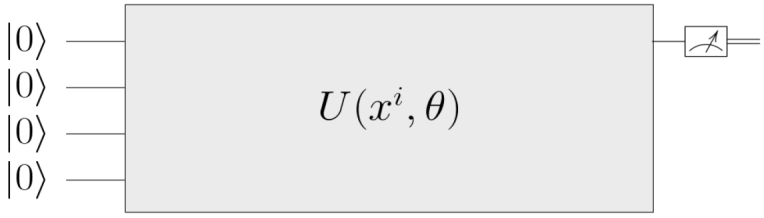

Figure 2: The basic idea of VQC is to implement the quantum circuit with unitary gates that contain parameters. To analog the classical neural net work, we can take the measurement as the output value.

quantify the error of the predicted values. For instance, we use the squared loss function defined as

$$L(\theta) = \sum_{i=1}^{N} (f(x^i, \theta) - y^i)^2. \tag{1}$$

Intuitively, the smaller the loss, the more accurate the function f. In order to tune the parameters to reduce the loss, we use the famous "Gradient Descent" (GD) method. The basic idea is to update the parameters by

$$\theta \to \theta - \lambda \nabla L(\theta) \tag{2}$$

where λ is a positive real number called the learning rate, it controls how fast we update our parameters. We can further expand the gradient as

$$\nabla L(\theta) = \sum_{i=1}^{N} 2(f(x^i, \theta) - u^i) \frac{\partial f}{\partial \theta}. \tag{3}$$

2.2 Introduction to Quantum Neural Network

One of the most famous example to apply the idea of neural network using quantum computers is "Variational Quantum Circuit" (VQC), proposed first in the work of Peruzzo et al. [2014]. The basic idea of VQC is to use unitary gates with tunable parameters (see Fig.2). Analogous to classical neural network, one can take the measurement as the output value (prediction). An example will be showed in our experiment.

However, if we want to optimize the parameters using GD method, we need to calculate the gradients. But since in the NISQ era, quantum computers may have noise, which cause a bad precision and a huge error on calculating gradients using classical differentiation method such as finite different method. To circumvent this problem, we certainly need a new way to calculate gradient on quantum computers. Fortunately, if our parameters are encoded with some special structures, e.g., Pauli rotation gates, then we can calculate the gradient with high precision. This method is called "Parameter-Shift Rule"(PSR) proposed by Crooks [2019]. Suppose we have a Pauli rotation gate $R(\theta) = e^{-i\frac{\theta}{2}\sigma}$ for some Pauli matrix σ with a parameter θ in our circuit. Squeeze other gates into A and $|\psi\rangle$, and focus on this Pauli

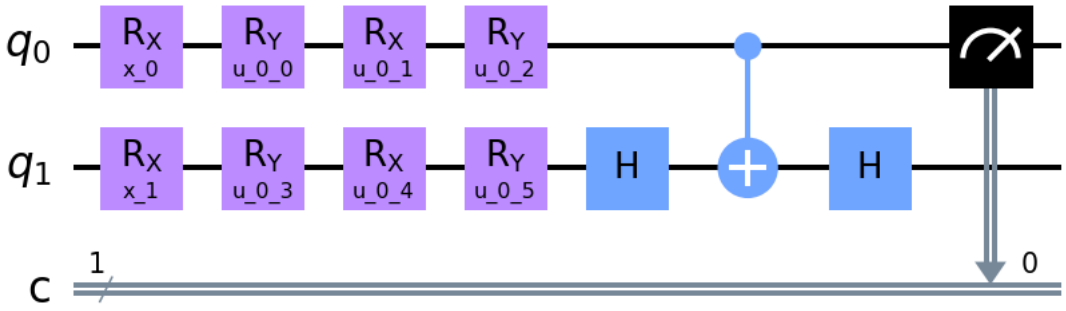

Figure 3: Structure of VQC. The data (x_1^i and x_2^i are the two PCA features of each data) is encoded in the angle of rotation gates. Then the circuit will be followed by some parametrized gates (with tunable parameters θ) and entangling gates. Finally, we only measure the first qubit in the Z-basis as the output value.

Table 1: Hyper-parameters we use in our experiment.

measurement shots	128 and 1024
number of qubits	2
training epochs	10
learning rate λ	1
noise rate	0.1

rotation gate, we can write our function as

$$f(x^i, \theta) = \langle\psi| R^\dagger(\theta) A R(\theta) |\psi\rangle , \tag{4}$$

where $|\psi\rangle$ is the state operated by gates before $R(\theta)$, and A is the gates and measurements after $R(\theta)$. Surprisingly, the gradient of f can be calculated by

$$\frac{\partial f}{\partial \theta} = \frac{1}{2}[f(x^i, \theta + \frac{\pi}{2}) - f(x^i, \theta - \frac{\pi}{2})]. \tag{5}$$

The important thing is that this is an exact value, not an approximation! What's more? We plus and minus the parameter with $\pi/2$, this is a huge difference with finite difference method, and can be used in the NISQ era. Finally, we plug Eq.5 into Eq.3 in order to update the parameters using GD method.

3 Result and Analysis

3.1 Experiment Setup

We demonstrate the power of PSR by training a small model. We use the traditional dataset known as Iris Data, provided by Dua and Graff [2017]. There are 3 different classes in Iris dataset, each with 50 data. Each data has 4 features, we use Principal Component Analysis (PCA) to reduce the dimension of features from 4 to 2. We then choose 2 of the classes to setup a classification problem. The dataset will be split into 90 training data and 10 testing data (each class will has 45 training data and 5 testing data). The structure of our VQC is provided in Fig.3. We use rotation angle to encode our data. Since the argument for rotation gates has a period in $[0, 2\pi]$, we shift and scale the Iris data into $[0, \pi]$, so the state will be prepared between $|0\rangle$ and $|1\rangle$. After encoding gates, we operate a series of Pauli rotation gates with tunable parameters, then we also apply Hadamard gates with CNOT gate to get a controlled-Z gate. Finally, the first qubit will be measured in the Z-basis. We set the output value to be the expectation value of number of measurements in $|1\rangle$, so the output value will be restricted between $[0, 1]$.

The VQC will be trained with total 10 epochs. For optimization, we use "Stochastic Gradient Descent" (SGD) method, simply a simplified version of GD method. In SGD, we randomly choose 5 data out of training data and calculate the mean loss gradient to optimize the parameters θ. Other hyper-parameters will be listed in Table 1.

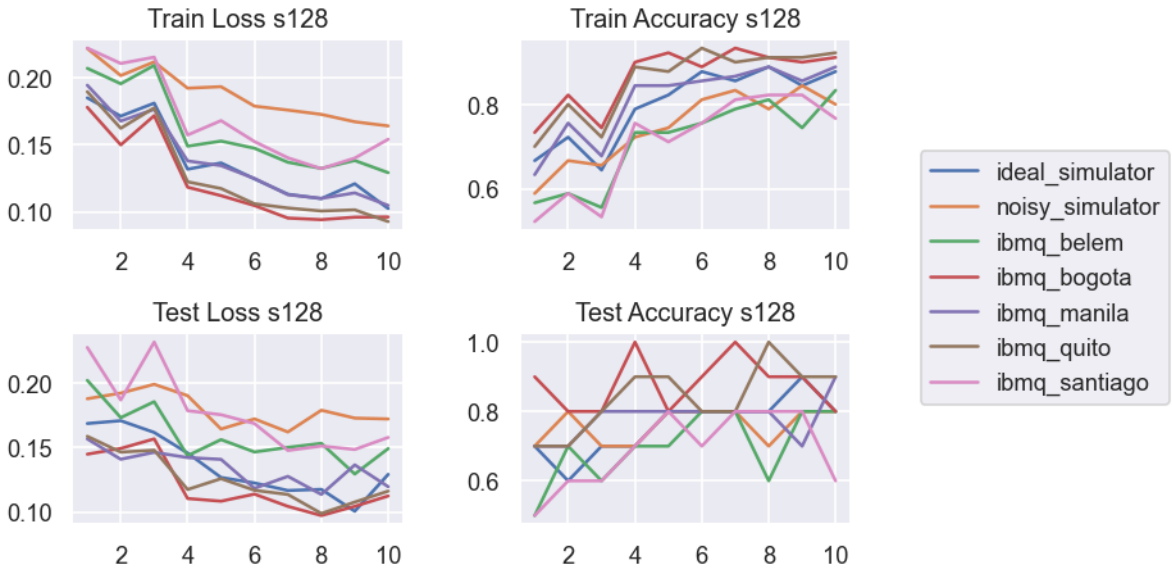

Figure 4: Results of training with shots=128

Figure 5: Results of training with shots=1024

3.2 Experiment Result

We use Qiskit "qasm_simulator" for our simulation, including ideal simulator (noise-free) and noisy simulator (for detail, see github `https://github.com/r08222011/Qiskit_Parameter_Shift`). For real quantum computers, we choose 5-qubit quantum computers. We have successfully test on "ibmq_belem", "ibmq_bogota", "ibmq_manila", "ibmq_santiago" and "ibmq_quito". The only one "ibmq_lima" has too many queues to be wait, so we skip these two quantum computers.

Training results for shots 128 and shots 1024 can be found in Fig.4 and Fig.5 respectively. We see that PSR works well both in simulator and real quantum computers, since they can both be trained successfully (meaning that the loss is reduced and the accuracy is increased). One interesting thing to be noticed is that "ibmq_bogota" and "ibmq_quito" perform even better than "ideal_simulator", we think that it was just a coincidence.

4 Discussion and Conclusion

PSR is certainly a candidate for training a VQC, however, the (time and space) complexity should be discussed. Unlike the classical method such as back propagation, we can only calculate the gradient for a parameter at a time, which means the training time complexity is polynomial in the number of parameters. Also, to use PSR technique, one should obey the special structure of gates we use, such as Pauli rotation gates. The greatest advantage of PSR is that we calculate the gradient by shifting the parameters by $\pi/2$, unlike finite difference method, this is exact a value!

There are other techniques for training VQC such as data re-upload proposed by Pérez-Salinas et al. [2019], we also implement this technique in our code. However, due to the restriction of the usage of IBM quantum computers, we didn't use this technique in our experiment.

References

Gavin E. Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition, 2019.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5 (1):4213, Jul 2014. ISSN 2041-1723. doi:10.1038/ncomms5213. URL `https://doi.org/10.1038/ncomms5213`.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. 2019. doi:10.22331/q-2020-02-06-226.