Quantum Computing Machine Learning

Alex Wu 301248931
Cai Yu Su 301283106
Chuan Zhang 301360028
Siyuan Wu 301313026

August 2020

Motivation

Nowadays quantum computing has acquired attention in a wide range of areas and the research of quantum algorithms has been focused on how to extend exponential or geometric acceleration to a wider range of applications. Although qubits can store more information than bits, each observation can access only one of the information from qubit according to the probability distribution. Therefore quantum algorithms would have a big difference with the classical algorithm.

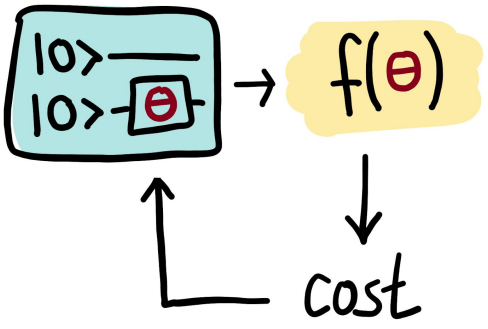
Introduction

The purpose of machine learning is to involve computer learning from data and outputting correct predictions. Many of the steps in the classical machine learning algorithm could be replaced by quantum computation in order to speed up the calculation or increase the results coverage.

In our project, we would study machine learning and use quantum computing to implement the classical machine learning algorithm. We would design a quantum circuit and attempt to optimize the time/space complexity of the existing algorithm.

Basic Idea

Our quantum machine learning system structure is based on the figure shown below, and has three different parts:



Picture Cite: https://pennylane.ai/qml/glossary/variational_circuit.html

1. **Variant circuit**

   The quantum variant circuit is acting as the role of feed-forward neural networks. The input data will be pre-processed and normalized to fit into an initial state by turning according to the z-axis. In this circuit we have a series of angles to train such that the initial state could be modified to a designated or expected entanglement state. The whole transformation procedure could be represented by unitary matrices acting on input states ($f(\theta)$) thus we could improve it by adjusting the parameters.

2. **Inferred function**

   We need a surjective function from a part of possible measurement results to valid labels. In our point of view, one set of measurement results mapping to a certain label should be reasonable to appear evenly in an entanglement state and the sum of them need to be the majority of all measurements. However, for each training set, this inferred function is not unique. Any of them could be used in the training procedure as though it is appropriate.

3. **Cost function**

   The cost function belongs to the classical part of QML. We need to design a function which could be minimized and highly related to the correctness of classification. Here a usual way in classical machine learning is to use L2 regression and compute the partial gradient of each parameter through it. We'll discuss our cost function later in this report.

As we know, in classical machine learning we need to train the network by backpropagation and using SGD for efficiency. On the contrary, a quantum circuit could perform GD directly on the same predicting circuit by phase shifting if we use the measurement as the cost function. (Mitarai, et al, 2018) Here we are trying to use another minimization method (Powell's method) which is a generic method and does not need to compute gradients, thus we do not need to calculate the gradient of cost function.

Methods

We would use the supervised learning method. First is our training process

1. Set an inferred function mapping quantum circuit output to labels
2. Input one training data into the quantum circuit.
3. Measure the result many times and get the frequency distribution.
4. Input the frequency distribution into the inferred function and get the predicted result.
5. Compare the predicted result and the correct result in training data.
6. Calculate the loss function.
7. Repeat step 2-6 with rest training data.
8. Calculate the cost, which is the sum of all loss divide number of data
9. Update the quantum circuit, try to find the minimum cost.
10. Repeat step 7-9 with whole training data.
11. Get the result quantum circuit which has min-loss.

Dataset

We will implement the Iris Flower Classification problem.

The dataset contains a set of 150 records of 3 types of species: 50 Iris setosa, 50 Iris virginica, and 50 Iris versicolor. Each species has 4 features: the length and width of the petals, and the length and width of the sepals.

Our quantum computing program would first learn from the training dataset which species are already known, then we want to predict the species for the testing data.

Measurement

First, we have to apply the classical problem into quantum computing.

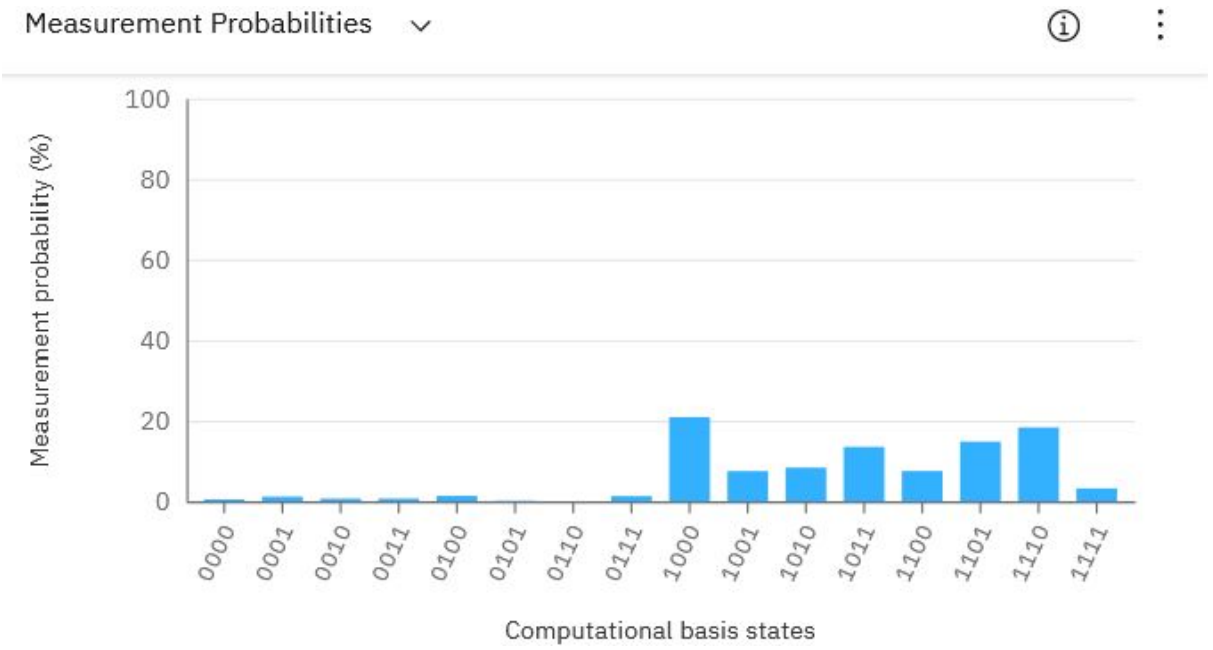
We input the training data into the quantum circuit and measure the state which indicates the species. The number of qubits we need depends on the input data features. Each qubit could represent one feature.

For example, if the data has 4 features,

state: 0001, 0010, 0100, 1000, it represents 1 species.

state: 0011, 0101, 1010, 1001, it represents 1 species.

state: 0111, 1011, 1101, 1110, it represents 1 species.



This is the results tested by IBM Quantum Experience, it's clear to see that 0111, 1011, 1101, 1110 has more than 50% count, which is greater than two other species.

Quantum Circuit

The circuit includes H-gate, RY-gate, RZ-gate, and CNOT-gate.

The Ry gate and Rz gate is about rotation. The Ry rotates a single-qubit around the y-axis, and Rz rotates around the z-axis.

The CNOT gate operates on two-qubit, where the one qubit is the control qubit and the other qubit is the target qubit. The target qubit rotates through pi radians around the x-axis only if the control qubit is 1. We use CNOT to entangle all qubits.

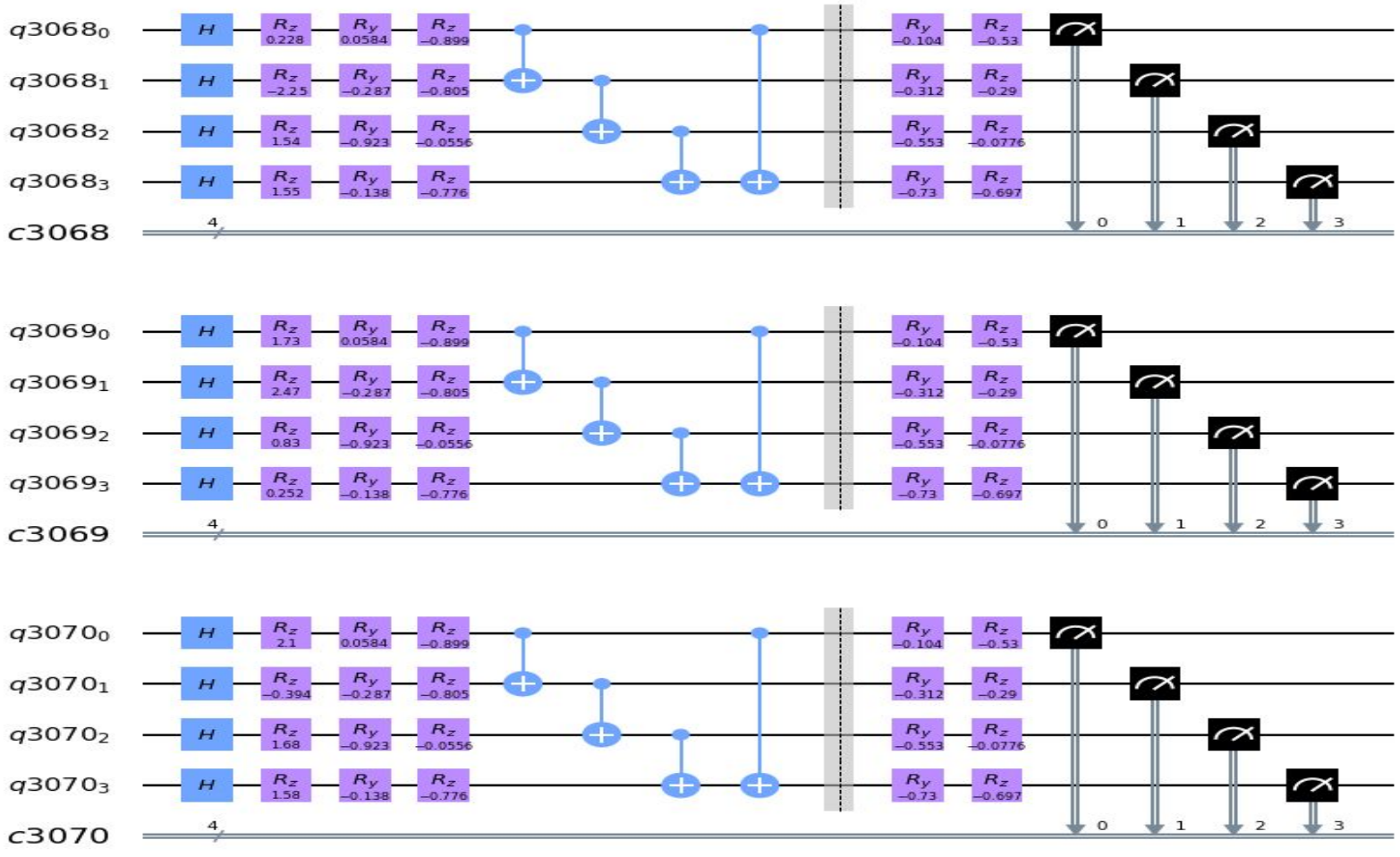
The Hadamard gate operates on a single-qubit operation. It creates an equal superposition.

Our data have 4 features, each feature is one-dimension space. Each flower species is a vector in its particular four-dimension space, represented by the quantum state. Our goal is to find an appropriate way to distinguish these states.

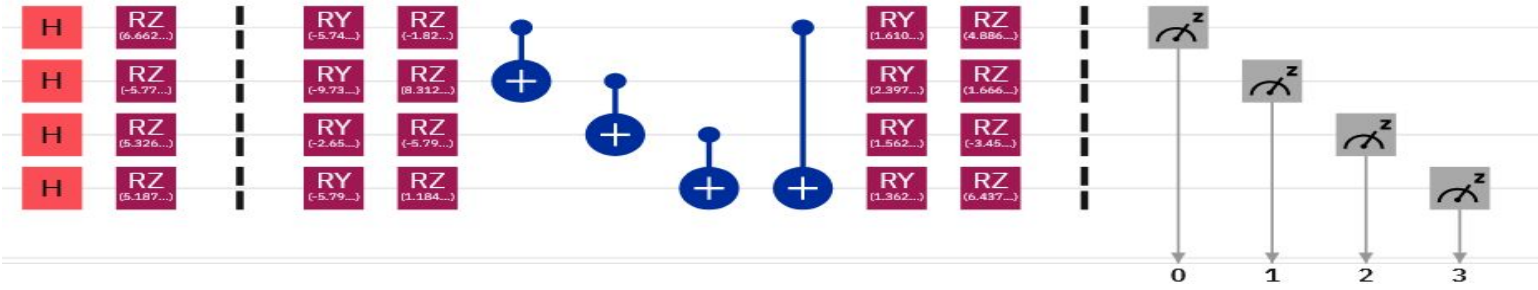
First, we set an initial quantum circuit with random rotation on the y-axis and the z-axis. After each training, we calculate rotation with min-loss on prediction, so that we could update our circuit with that new rotation. In this assignment we used Powell optimization from the scipy module. This optimization function will iteratively call the user defined function and return the parameter which minimizes the function.

During the optimization process, we update RY-gate and RZ-gate with different rotations, so that data from the same species could be sent to highly related positions in the 4-D hyperplane and we could better divide the hyperplane to classify the input.

The circuit during training: we can see the rotations are changing.



The circuit after training:



Accuracy

The measured result we get is the probability of the actual result. To increase the accuracy of the measured result, we need to measure the result for multiple times to get the most frequent result. Compare the predicted result and the correct result, then update our quantum circuit depending on the current accuracy.

Thus, we need a good loss function to $\sum_{j}^{k} \binom{N}{j} p_a^j (1-p_a)^{N-j}$ improve accuracy.

Set ya is the predicted result, Yi is the correct result, yb and yc is other two species

Loss(i)= p(ya≠Yi),

N is for the number of times that the result has been measured.

N=Nya+Nyb+Nyc= Nya +2(max(Nyb,Nyc)) - |Nyb-Nyc|

When we cannot have a right predicted label, Nya must be less than max(Nyb,Nyc):

Nya < k = (N- |Nyb-Nyc|)/3

We have

p(ya≠Yi)= p[Nya< max(Nyb,Nyc)]=

Using approximation from binomial coefficient to integral and take the probability into a sigmoid function we can get our loss function

loss_value =1/(1+exp(-(sqrt(N)*(max(Nyb,Nyc)-freq[Nya])/sqrt((Nyb+Nyc)*freq[Nya]))))

Discussion

One problem we have encountered is that our program had a low success prediction rate. At first, we only get a 30% success rate. After improving the training process, we get approximately 80-90% success rate. We have also met problems on the framework we used to build the program. Originally we try pyQPanda, which is a python wrapper of QPanda(a quantum programming environment for realistic quantum computing). There are many problems with its library which makes the program unable to compile. Thus we use Qiskit to implement the program.

Then we have encountered another problem which is that our training progress takes time to run. It usually takes a few hours to run the Iris Flower dataset in Qiskit. In theory, quantum machine learning has higher efficiency than machine learning(LOURIZ, 2019). However, the same training progress in classical machine learning takes only a few minutes. A possible reason is that our data set is small with O(1) complexity, thus the classical machine learning algorithm takes advantage of running this simple data set. Another possible reason is that we are using the Qiskit simulator to run a quantum program. We cannot compare the quantum machine learning algorithm against the classical machine learning algorithm in real-world problems.

Conclusion and Future Work

We have implemented a quantum machine learning program to solve the Iris Flower Classification problem using the supervised learning method to develop the training process. We convert the classical data set into quantum data, then input these data into the quantum circuit. To implement the training process in our quantum program, we design a quantum circuit and update the circuit with calculation to improve the prediction. The program runs as expected. Through the implementation process, we study machine learning, understand the quantum classifier model, and learn how to design the quantum circuit.

In this project we designed a variant circuit to perform quantum machine learning. However, our result did not show the advantage using quantum computing compared to the classical method. We are considering improving our work by optimizing our cost function and trying to adopt the phase shifting method to find the parameters quicker and make the most of computation dealt with the quantum part. On the other hand, we'd like to try other classification methods such as 1-vs-all classification by a O(n) depth quantum circuit (n is the number of different species). At last, we could try more difficult datasets on the network.

References

Farhi, Edward & Neven, Hartmut. (2018). Classification with Quantum Neural Networks on Near Term Processors.

Hadamard gate. Retrieved August 11, 2020, from https://www.quantum-inspire.com/kbase/hadamard/

LOURIZ, R. (2019, September 25). Highlighting Quantum Computing for Machine Learning. Retrieved August 11, 2020, from https://medium.com/meetech/highlighting-quantum-computing-for-machine-learning-1f1abd41cb59

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum Circuit Learning. Bulletin of the American Physical Society, 2018.

Schuld, Maria & Bocharov, Alex & Svore, Krysta & Wiebe, Nathan. (2018). Circuit-centric quantum classifiers. Physical Review A. 101. 10.1103/PhysRevA.101.032308.

Trindade, F. (2019, October 18). Start to learn Machine Learning with the Iris flower classification challenge. Retrieved August 11, 2020, from https://medium.com/gft-engineering/start-to-learn-machine-learning-with-the-iris-flower-classification-challenge-4859a920e5e3

Pyqpanda. Retrieved August 11, 2020, from https://pypi.org/project/pyqpanda/

Pennylane. Retrieved August 11, 2020, from https://pennylane.ai/qml/glossary/variational_circuit.html

Wikipedia contributors. (2020, June 12). Iris flower data set. In *Wikipedia, The Free Encyclopedia*. Retrieved August 11, 2020, from https://en.wikipedia.org/w/index.php?title=Iris_flower_data_set&oldid=962157652