QOSF: Task 2 Encoding and Classifier Analysis

Wittmann Goh, Nadine Meister

12 March 2022

1 Encodings

To convert our numerical data into qubits, we implemented 2 different encodings: angle and amplitude. Each qubit starts in the $|0\rangle$ and are transferred into the desired states through rotation matrixes.

1.1 Data preparation

To reduce data bias due to uneven scaling of data, we had to do some preprocessing. First, since columns 1 and 2 contain data that differ exponentially from each other, $\log_{10}$ was applied to these two columns. After this step, the values in column 0 were much bigger than the values in the other 3 columns. Therefore we normalized each column according to its mean. The result of this was that each column contained values in the interval $[0, 3]$.

1.2 Angle Encoding

In angle encoding, each column of the dataset is stored as the angle in the qubit. Thus, for our dataset, we have 4 qubits, and the necessary transformations are

$$S_x|0\,|\rangle = \bigotimes_{i=1}^{4}(\cos(x_i)\,|0\rangle + \sin(x_i)\,|1\rangle)$$

We normalized the input to $[-\pi, \pi]$ and applied a single R_x with this angle.

1.3 Amplitude Encoding

The idea of amplitude encoding is to store data in the amplitudes of each basis state in the system. For concreteness, suppose our data vector is $\mathbf{x} = (x_0, x_1, x_2, x_3)$, columns 0 through 3 in the mock

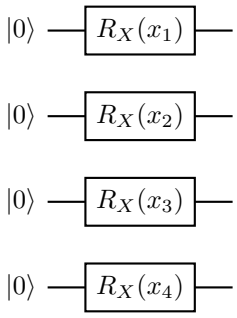

Figure 1: Angle encoding circuit

dataset. Then the state that encodes this information would be

$$|\mathbf{x}\rangle = \frac{x_0}{||\mathbf{x}||}|00\rangle + \frac{x_1}{||\mathbf{x}||}|01\rangle + \frac{x_2}{||\mathbf{x}||}|10\rangle + \frac{x_3}{||\mathbf{x}||}|11\rangle$$

To accomplish this encoding, we apply a series of R_Y and controlled R_Y gates to obtain the necessary relative amplitudes. The circuit we use is the following:

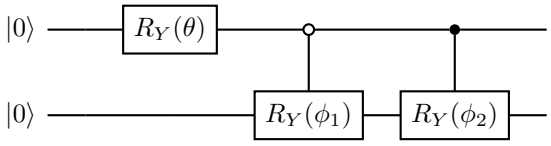

Figure 2: Amplitude encoding circuit

Intuitively, the first R_Y gate distributes the amplitude between the first two datapoints and the last two datapoints, and then the controlled R_Y gates adjust the relative amplitudes between each pair. The angles θ, ϕ_1, ϕ_2 are computed using the input data x_0 through x_3.

2 Layers in the Variational Circuit

After encoding the data, we use a variational circuit to classify the data. We did this for each of the encodings described above.

For our first attempt, we simply added adding a R_X, R_Y, and R_Z to each qubit in the variational circuit. After training the circuit, the amplithe accuracy tude-encoding model achieved only 70% accuracy, while the angle-encoding model achieved only 50%. After adding a CNOT layer (consisting of $CNOT(i, j)$ on all ordered pairs (i, j) of qubits), the accuracy for amplitude encoding the accuracy improved to 96%, while the angle encoding stayed at around 50%.

Adding additional layers only increased the training time but did not improve accuracy. We saw that after adding another full layer of $R_X/R_Y/R_Z$/CNOT gates, the accuracy did not change noticeably.

Sierra-Sosa et. al aimed to show that amplitude encoding is beneficial to the accuracy, as compared to angle encoding and other methods [1]. Our results were similar, showing that amplitude encoding had a higher accuracy than angle encoding.

3 Future

To make the variational circuit more robust, we would implement an arbitrary unitary matrix and train the model to learn the optimal parameters of the matrix.

We found that the best accuracy when directly measuring one of the input quits. However, in the future, we would also explore having a readout qubit which is connected to the input qubits through gates.

It is unclear why the angle-encoding performs so poorly. This is something that we would also like to understand and explain in the future.

References

[1] D. Sierra-Sosa, M. Telahun, and A. Elmaghraby. Tensorflow quantum: Impacts of quantum state preparation on quantum machine learning performance. *IEEE Access*, 8:215246–215255, 01 2020.