Quantum Machine Learning

Devarshi Joshi (202003007)
Arpan Shingala (202003013)

Introduction

Quantum computing is an innovative approach that uses quantum physics principles to process and alter data. Unlike traditional computers, which utilize bits to encode information as a 0 or a 1, quantum computers use quantum bits or qubits, which can exist in several states at the same time due to the phenomena of superposition.

This unique property of qubits enables quantum computers to conduct computations in parallel, potentially providing a massive speedup in tackling complicated problems when compared to classical systems. Researchers have examined the convergence of quantum physics with artificial neural networks as the area of quantum computing advances, giving rise to an exciting new field known as quantum neural networks (QNNs).

These networks have enormous potential for handling computationally difficult tasks and pushing the boundaries of machine learning and artificial intelligence.

Classical Neural Networks

Artificial neural networks are computing systems inspired by the biological neural networks that make up the human brains. ANN consists of several nodes which act as neurons. These neurons work together to make certain decisions which dictate the result.

A neural network is made up of interconnected neurons, where each neuron's input consists of weighted outputs from other neurons from the previous layer. The first layer in the network is called the input layer, and the last layer is called the output layer. The layers in between the input and output layers are called hidden layers.

The interconnections allow neurons to exchange data or information. However, the data only flows from the previous layer to next, i.e the data only travels in one direction. This is called Forward Propagation. We train the data this way and find how far off we are from the actual answer. We then tweak the parameters a bit and run the program again. This is called Backward Propagation.

Tasks to do while working with CNNs:

Data Preparation

Data preparation involves taking in some initial input data and applying certain transformations to it for better efficiency.

The data may undergo some basic transformations such as normalization or scaling.

Data Processing

The processed data flows through a series of layers within the network, where each layer performs a specific transformation on the data.

These transformations are dependent on parameters that are optimized during training.

Data output

Finally, the processed data is outputted through a final layer, resulting in the network's prediction or classification.

Quantum Neural Networks

Quantum neural networks (QNNs) represent a promising frontier in the field of quantum computing and machine learning. QNNs combine the principles of quantum mechanics with artificial neural networks to harness the potential of quantum systems for enhanced computational capabilities.

In QNNs, quantum bits or qubits replace classical bits as the fundamental units of information processing. These qubits can exist in superposition, allowing for parallel processing and the exploration of multiple states simultaneously.

Additionally, quantum entanglement plays a crucial role in QNNs, enabling the creation of highly correlated qubit states and facilitating the representation and processing of complex patterns and relationships within data.

Similar to CNNs, the three tasks we need to do are :

1. Data Preparation
2. Data Processing
3. Data Output

Data Preparation in QNNs:

QNNs, like CNNs, require properly prepared and preprocessed data to produce accurate and useful outputs. However, because conventional and quantum systems are fundamentally different, data preparation in QNNs brings distinct challenges and considerations.

To begin, because quantum computers run on qubits that can represent complicated quantum states, data encoding becomes crucial. Encoding strategies that maintain information while projecting it to the quantum realm must be carefully considered when transforming classical data into quantum states.

Furthermore, data normalization algorithms must be customized to quantum systems, taking into account qubit-specific requirements and limits. Capturing and retaining entanglement patterns in data can improve QNN performance and efficiency.

Angle Encoding

Angle encoding is a critical technique used in data preparation for quantum neural networks (QNNs) that aids in the mapping of classical data onto quantum states.

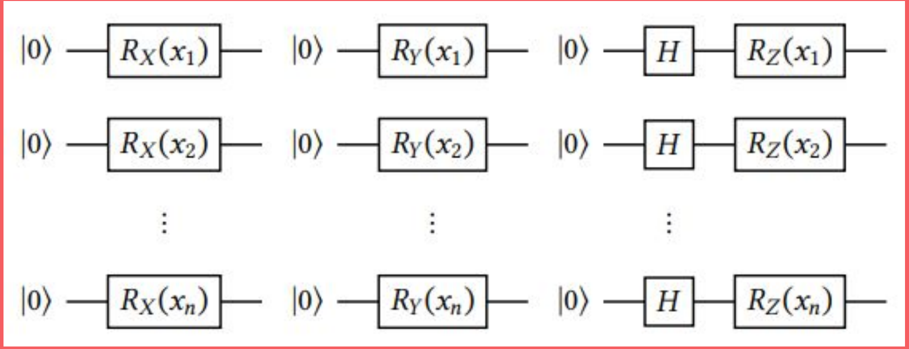
Angle encoding represents information in a quantum system by angles rather than simply recording data values. This method uses qubits' natural features, like phase and amplitude, to store and process data.

Angle encoding has various advantages in QNNs, including better qubit utilization and resistance to noise and mistakes. QNNs may use the full expressive capacity of quantum systems by mapping data into angles, effectively describing complex patterns and relationships within the data.

This feature map can accept up to n numerical inputs x1,..., xn when utilized on an n-qubit circuit. Its circuit operates by applying a rotation gate to each qubit j that is parametrized by the value xj.

The xj values are used as angles in the rotations in this feature map, hence the name of the encoding.

In angle encoding, we can utilize any rotation gate of our choosing. However, if we employ gates and use |0⟩as our beginning state, the action of our feature map will have no effect, as the definition of RZ clearly shows. As a result, when RZ gates are employed, Hadamard gates acting on each qubit are usually utilized before them.



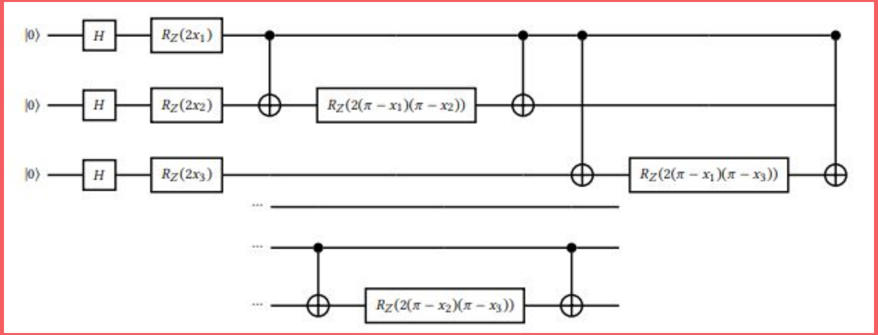
ZZ featuremap

The ZZFeatureMap is a well-known method for encoding classical data into quantum states when preparing data for quantum neural networks (QNNs).

This feature map is highly suited for problems where the relationships between variables are critical because it is specifically made to capture pairwise interactions between input characteristics. By building a quantum circuit out of controlled-Z gates, where each gate represents the interaction between two features, the ZZFeatureMap works.

The ZZFeatureMap generates entanglement patterns that encode the interactions between the features by applying these gates to the qubits that represent the input data. It has been demonstrated that the ZZFeatureMap works well in a variety of settings, including simulations of molecular chemistry and pattern recognition tasks. The following steps are done to achieve ZZFeatureMap:

1. Apply a Hadamard gate on each qubit.
2. Apply, on each qubit j, a rotation $RZ(2xj)$.
3. For each pair of elements $\{j, k\} \subseteq \{1, \dots, n\}$ with $j < k$, do the following:
   a. Apply a CNOT gate targeting qubit k and controlled by qubit j.
   b. Apply, on qubit k, a rotation $RZ\,(2(\pi - xj)(\pi - xk))$.
   c. Repeat step 3a



|0⟩
|0⟩
|0⟩
H
H
H
$R_Z(2x_1)$
$R_Z(2x_2)$
$R_Z(2x_3)$
$R_Z(2(\pi - x_1)(\pi - x_2))$
$R_Z(2(\pi - x_1)(\pi - x_3))$
$R_Z(2(\pi - x_2)(\pi - x_3))$

Data Processing in QNNs:

Data processing, which involves altering and analyzing incoming data to derive valuable insights and help machine learning tasks, is a basic component of quantum neural networks (QNNs). Data processing in QNNs involves a number of crucial phases that are specifically designed to take into account the special properties of quantum systems.

First, using methods like angle encoding or feature maps, data encoding is carried out to map classical data onto quantum states. This action successfully prepares the data for further processing by enabling the representation of complicated information in the quantum realm.

Quantum operations and calculations are performed on the encoded data after data encoding. These processes use quantum mechanical concepts like superposition and entanglement to process data concurrently and identify complex patterns and relationships in the data. To execute computations, optimise, and alter the data to support learning and decision-making processes, quantum gates and algorithms are used. The quantum results are then interpreted and converted back into classical information by data decoding and post-processing, enabling analysis, inference, and performance evaluation of the QNN.

2 Local Variational Form

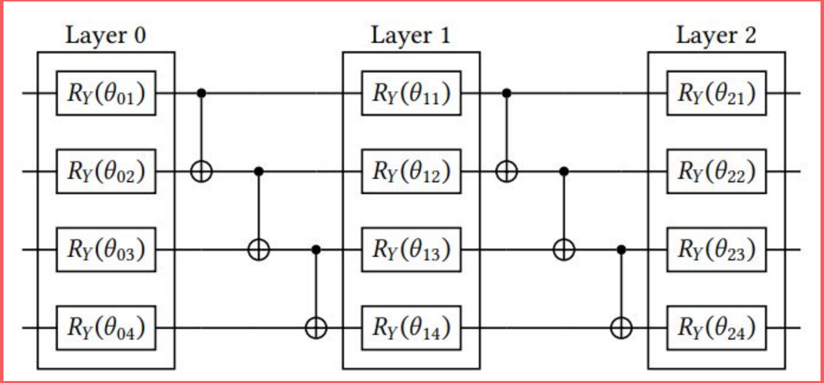
A crucial method used in the data processing stage of quantum neural networks (QNNs) is the local variational form.

The local variational form concentrates on building quantum circuits by utilising local operations that capture the local features and interactions contained in the data, such as single-qubit rotations and two-qubit interactions. The local variational form enables the optimisation process to modify the circuit's structure to suit the particular dataset and learning goal at hand by parameterizing the circuit with trainable parameters. This method improves the performance and generalisation skills of QNNs by efficiently teaching them complicated patterns and relationships within the data.

The local variational form offers versatility in quantum state representation, allowing the QNN to adjust and develop as it absorbs new information from the data. The algorithm for 2 Local Variational form is as follows:

TwoLocal(n, k, θ):

```
for all $r = 0, \dots, k$ do
    for all $j = 1, \dots, n$ do
        Apply a $RY(\theta r j)$ gate on qubit $j$.
        if $r < k$ then
            for all $t = 1, \dots, n - 1$ do
                Apply a CNOT gate with
            control on qubit $t$ and target
            on qubit $t + 1$.
```



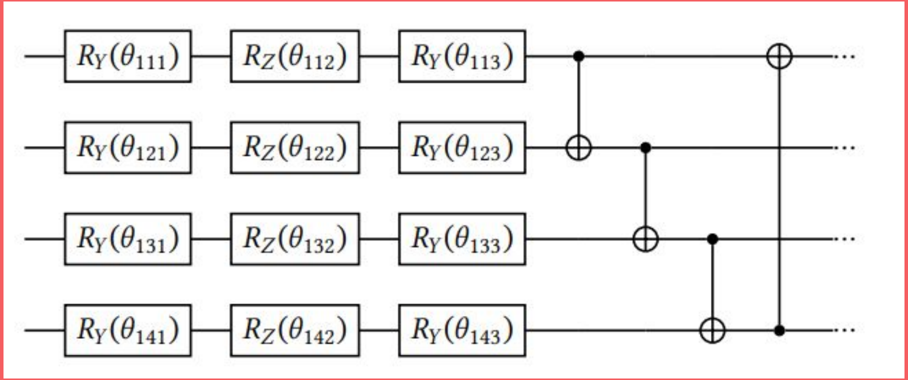
Strongly entangling layers

Strongly entangling layers are an important component in the data processing phase of QNNs. These layers are made to create entanglement and strong correlations among the qubits in a quantum circuit.

Strongly entangling layers improve the network's capacity to detect complicated and non-linear correlations within data by integrating entangling gates, such as controlled-Z or controlled-X gates. The generation of entanglement in these layers enables the QNN to take advantage of the inherent parallelism and quantum effects of quantum systems, processing information in a highly expressive and effective way.

The algorithm for Strongly entangling layers is as follows:

```
StronglyEntangl ingLayers($n, k, r, \theta$)
for all $l = 1, \ldots, k$ do
    for all $j = 1, \ldots, n$ do
        Apply a rotation $RY(\theta lj1)$ on qubit $j$.
        Apply a rotation $RZ(\theta lj2)$ on qubit $j$.
        Apply a rotation $RY(\theta lj3)$ on qubit $j$.
    for all $j = 1, \ldots, n$ do
        Apply a CNOT operation controlled by
    qubit $j$ and with target on qubit$[(j + rl - 1)$
    mod $N] + 1$.
```



Data Output in QNNs:

Data output in quantum neural networks (QNNs) refers to the final result or prediction generated by the network based on the input data. Depending on the exact job or learning purpose, a QNN's output can change.

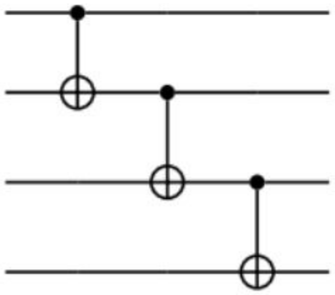
The QNN may generate class labels or probabilities associated with various classes in classification tasks. The prediction of continuous values or parameters may be involved in regression activities. Applying the final set of qubits or measurement operations to the quantum circuit that was trained using the input data yields the output.

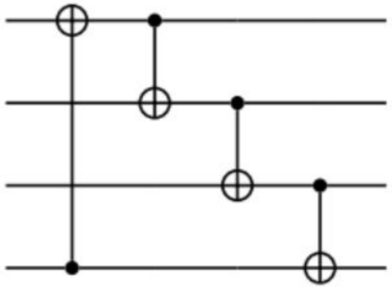
The desired information is subsequently extracted by measuring or sampling the quantum state that results from the calculation. It is crucial to remember that a QNN often produces probabilistic results because of the fundamental properties of quantum physics. As a result, the output could include probability distributions or quantum states that illustrate the likelihood of various occurrences. To analyze and improve the QNN output, post-processing techniques like decision thresholds or statistical analysis may be used.

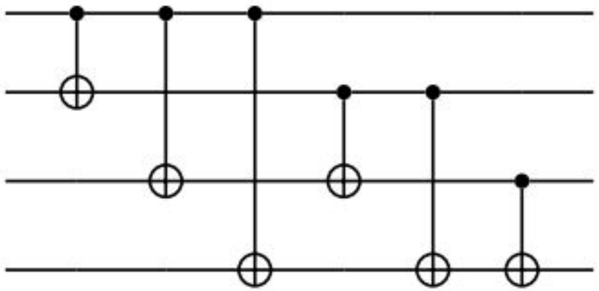
Measurement in QNNs:

In quantum neural networks (QNNs), measurement is a crucial function that is important for obtaining meaningful information from the quantum states generated during computation. The final stage in a QNN is to measure the quantum states to produce observable results after the quantum circuit has processed the incoming data.

Based on the fundamental properties of quantum physics, measurement converts the superposition of quantum states into particular classical states, producing probabilistic conclusions. Measurement can be thought of as reversing the entanglement which was achieved earlier. The 3 kinds of entanglement majorly used in QNNs are:


(a) Linear


(b) Circular


(c) Full

In order to reverse these entanglements, we can use any one of the below 2 operators:

M =

1	0
0	0

Z =

1	0
0	-1

$M = 1|0\rangle\langle 0| + 0|1\rangle\langle 1|$

$Z= 1|0\rangle\langle 0| - 1|1\rangle\langle 1|$

Implementation

Classical Neural Network

- Input layer: 30 neurons (parameters)
- 1st hidden layer: 9 neurons (relu - activation function)

$$f(x) = \max(x, 0)$$

- 2nd hidden layer: 1 neuron (sigmoid - activation function)

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Output layer: 1 neuron(output)

Quantum Neural Network - 1st Model

Data Preparation

- Using Principal component analysis to reduce the 30 parameters into 4 parameter
- Apply the zzfeaturemap to convert classical data into quantum data

Data Processing

- Input layer: 4 quantum qubit
- 1st hidden layer: 4 quantum qubits to do the Ry(θ) operation and entanglement operation
- 2nd hidden layer: 4 quantum qubits to do the Ry(θ) operation

Data Output

- Output layer: measurement operator M to measure output

Quantum Neural Network - 2nd Model

Data Preparation

- Using Principal component analysis to reduce the 30 parameters into 4 parameter
- Apply the zzfeaturemap to convert classical data into quantum data

Data Processing

- Input layer: 4 quantum qubit
- 1st hidden layer: 4 quantum qubits to do the rotation operation and entanglement operation
- 2nd hidden layer: 4 quantum qubits to do the rotation operation and entanglement operation

Data Output

- Output layer: measurement operator M to measure output

Some points to keep in mind:

Make wise choices

Depending on the QNN model we are working on, we need to wisely choose various methods we studied earlier.

To summarize, we have to pick a feature map, a variational form, and a measurement operation optimally.

Size matters

The power of the resulting quantum neural network will be directly related to the number of optimizable parameters it has.

Use too many parameters, and you may have a model that overfits. Use very few, and your model may end up underfitting.

Feed your QNN properly

Proper data preprocessing is important for effective QNN training. This includes normalizing the data according to the feature map's requirements and using dimensionality reduction techniques for high-dimensional data.

References

- Elías F. Combarro, Samuel González Castillo. A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms

Thank You!