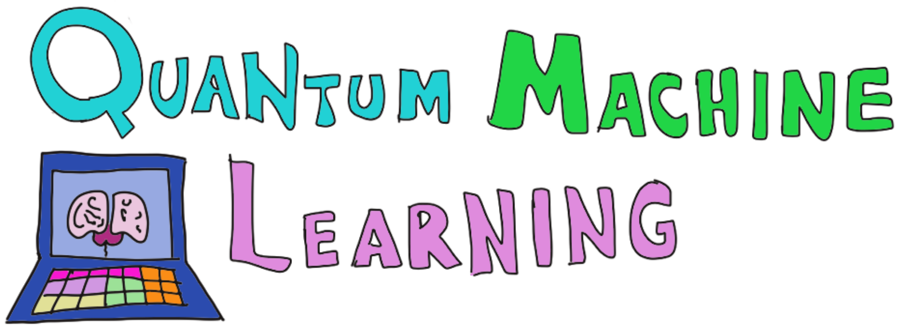

QML: Building a qRAM [400 points]

Version: 1

Quantum Machine Learning

In this set of challenges, you'll explore methods and applications for training variational quantum circuits and quantum neural networks. Both play critical roles in quantum machine learning. They typically have a layered structure, and a set of tunable parameters that are learned through training with a classical optimization algorithm.

The circuit structure, optimization method, and how data is embedded in the circuit varies heavily across algorithms, and there are often problem-specific considerations that affect how they are trained. In the **Quantum Machine Learning** challenges, you'll explore how to construct and optimize a diverse set of circuits from the ground up, and become acquainted with some of today's popular quantum machine learning and optimization algorithms.

Problem statement [400 points]

A quantum random access memory (qRAM) is a data structure that allows us to store a series of n-qubit states accessible via an index. Let's imagine that we want to store eight different states, each of which is generated through an operator A_i. The qRAM encodes these different states by generating the following state:

$$\frac{1}{\sqrt{8}}\left(|0\rangle A_0|0\rangle^{\otimes n} + |1\rangle A_1|0\rangle^{\otimes n} + ... + |7\rangle A_7|0\rangle^{\otimes n}\right)$$

In this case, since we will need to store eight elements in memory, we need three qubits to encode the eight indices. Remember that $|6\rangle$, for example, is a way of simplifying its binary representation $|110\rangle$. The potential of this structure lies in the fact that we can work with all the data simultaneously in an overlapping state, achieving, for example, more complex searches using Grover search or obtaining better performance in different QML algorithms.

In the following exercise, we will implement a qRAM in the case of $n = 1$. Here, A_i will be the gate $R_Y(\theta_i)$, so the input to your code will be a list of 8 elements storing each of the angles θ_i.

In the provided template called `building_QRAM_template.py`, there is a function called `qRAM` that you need to complete. Within this function, construct a quantum circuit that prepares the state mentioned above given the list of angles θ_i for $R_Y(\theta_i)$ gates as input. Your quantum circuit will return `qml.state()`.

Input

- `list(float)`: A list of the eight angles needed to define the R_Y gates.

Output

- `list(float)`: A list of coefficients corresponding to the output of the `qram` function (`qml.state()`).

Acceptance Criteria

In order for your submission to be judged as "correct":

- The outputs generated by your solution when run with a given `.in` file must match those in the corresponding `.ans` file to within the `0.001` tolerance specified below. To clarify, your answer must satisfy

$$\text{tolerance} \geq \left| \frac{\text{your solution} - \text{correct answer}}{\text{correct answer}} \right| .$$

- Your solution must take no longer than the `60s` specified below to produce its outputs.

You can test your solution by passing the `#.in` input data to your program as stdin and comparing the output to the corresponding `#.ans` file:

```
python3 {name_of_file}.py < 1.in
```

WARNING: Don't modify the code outside of the `# QHACK #` markers in the template file, as this code is needed to test your solution. Do not add any print statements to your solution, as this will cause your submission to fail.

Specs
Tolerance: **0.001**
Time limit: **60 s**

Version History

Version 1: Initial document.