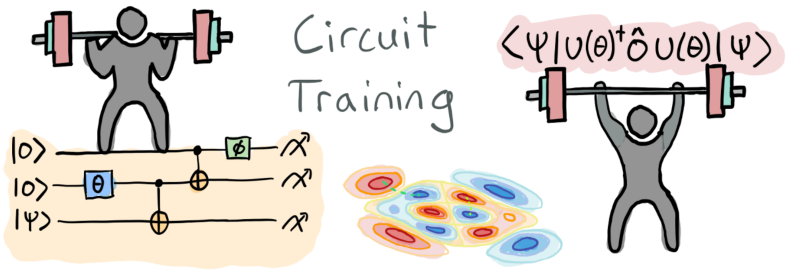

Circuit Training: Optimizing a Quantum Circuit [100 points]

Version: 1

NOTE: Coding templates are provided for all challenge problems at this link. You are strongly encouraged to base your submission off the provided templates.

Overview: Circuit Training challenges

In this set of challenges, you'll explore methods and applications for training variational quantum circuits. Variational circuits play a role in quantum machine learning akin to that of neural networks in classical machine learning. They typically have a layered structure, and a set of tunable parameters that are learned through training with a classical optimization algorithm. Data is input to the circuit by embedding it into the state of a quantum device, and measurement results inform the classical optimizer how to adjust the parameters.

The circuit structure, optimization method, and how data is encoded in the circuit varies heavily across algorithms, and there are often problem-specific considerations that affect how they are trained. In this set of 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 [100 points]

For this first challenge, we'll keep things simple. You will be provided with a variational quantum circuit in PennyLane that depends on a set of trainable parameters. The circuit outputs a single number as the expectation value of a fixed measurement. Your objective is to find the minimum expectation value this circuit can produce by optimizing its parameters. This will require converting the

circuit into a QNode. You can either code up your own optimizer by calculating the gradient of the QNode, or you can use one of the provided PennyLane optimizers.

Input

The `variational_circuit` function contains the quantum circuit and has a `params` argument for specifying the trainable parameters. These are the parameters that need to be updated by the optimizer.

You must fill in the `optimize_circuit` function so that it minimizes the variational circuit. The input to this function is a `params` argument that will specify to you the initial parameters to use when optimizing the variational circuit. Within `optimize_circuit`, you will need to convert the variational circuit into an executable QNode using `qml.QNode()`.

The `variational_circuit` function concludes with the measurement of the expectation value of a Hamiltonian. The Hamiltonian is specified by the problem input data, with each Hamiltonian resulting in a different minimum for the variational circuit. An unknown Hamiltonian will be used to judge your submission.

Output

The output of your `optimize_circuit` function should be a single floating-point number giving the minimum expectation value of the variational circuit that you have found.

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 `Tolerance` specified below.
- Your solution must take no longer than the `Time limit` 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 circuit_training_100_template.py < 1.in
```

WARNING: Don't modify any of the code in the template outside of the `# QHACK #` markers, 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: **0.05 (5%)**
Time limit: **60 s**

Version History

Version 1: Initial document.