CPEN 400Q / EECE 571Q Lecture 13
Introducing variational algorithms

Tuesday 1 March 2022

Announcements

- Assignment 3 available
  - second-last assignment
  - please read grading details
  - due Friday 11 March 23:59
- Meetings *next week* for project prototypes (schedule selection starting tomorrow)

Quiz 6 after class today.

★ Remind me to put up survey before end of class.

Last time

We learned about RSA, implemented Shor's algorithm, and used it to decompose numbers into their prime factors.

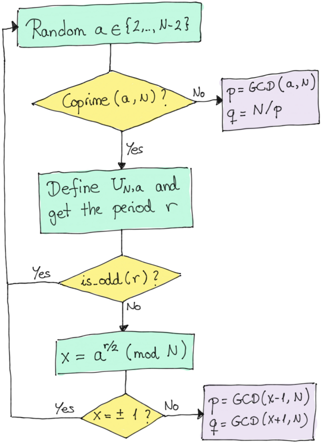
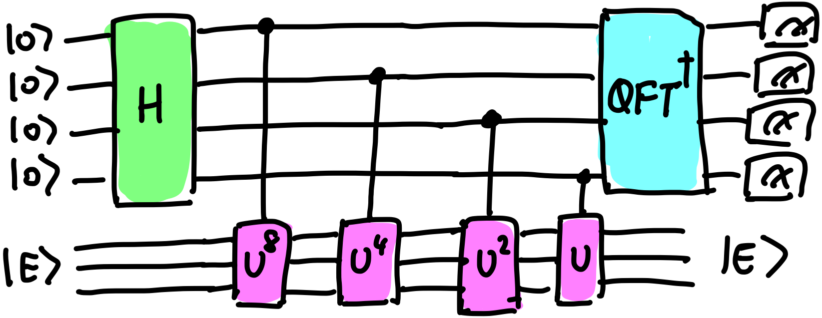


Image credit: Xanadu Quantum Codebook node S.4

Learning outcomes

- Describe the main structural elements of a variational quantum algorithm
- Compute gradients of variational circuit parameters using the parameter-shift rule
- Find optimal parameters of a variational circuit in PennyLane

Consider an algorithm like QPE...



Why variational algorithms?

"Full-size algorithms" like QPE, Shor, Grover, etc.:

- Use many qubits
- Require dense qubit *connectivity*
- Have high circuit depth

Today's quantum computers today aren't really suitable for these...

Why variational algorithms?

[A NISQ-era device, for exemplary purposes]

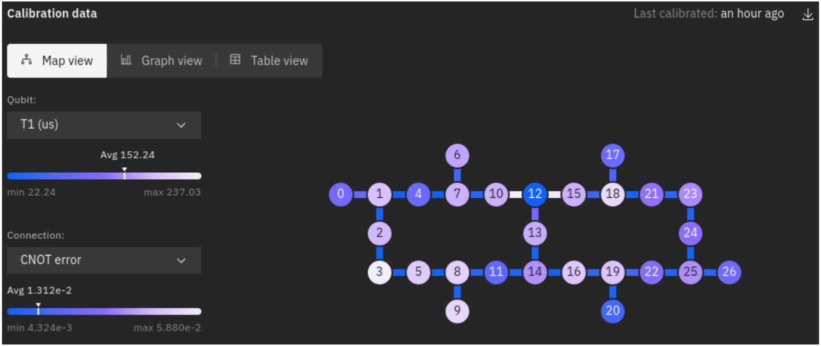


Image credit: IBM Q Auckland, screen capture 2022-03-01.

`https://quantum-computing.ibm.com/services?services=systems&system=ibm_auckland`

Why variational algorithms?

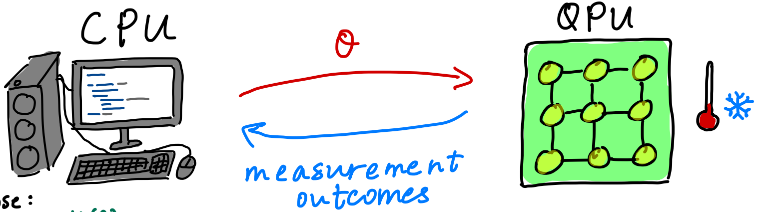
What *can* we do with a NISQ device?

Suitable algorithms should:

- Not be too long
- Fit the processor architecture well
- Use a quantum computer to do something non-trivial
- Still solve an interesting problem

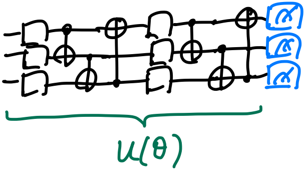
Variational algorithms

Feature an iterative exchange between classical and quantum devices. (Sometimes called "hybrid" quantum-classical algorithms)



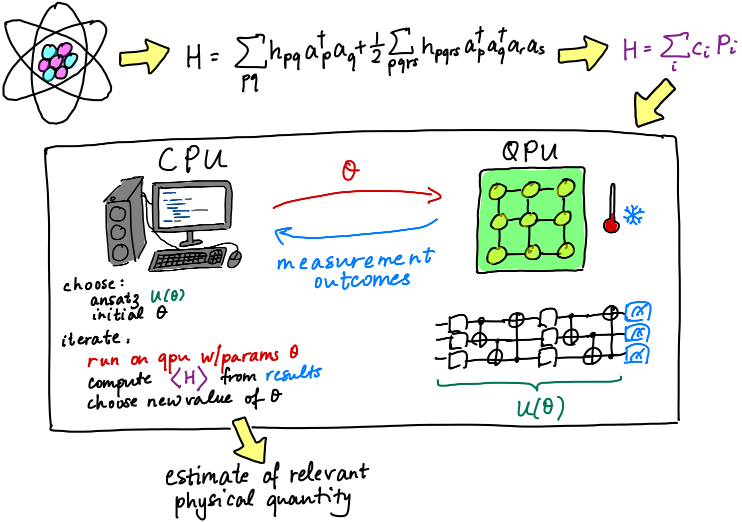
choose:
ansatz $U(\theta)$
initial θ

iterate:
run on qpu w/params θ
compute cost from results
choose new value of θ



Variational algorithms

Useful in many domains: quantum chemistry, quantum machine learning, optimization, etc.



Variational algorithms

We will cover in the rest of the course:

- The basics of setting up and running variational algorithms
  - Quantum gradients
  - Parametrized quantum circuits and variational ansaetze
  - Cost functions
- A number of common variational algorithms
  - Variational quantum classifier (VQC)
  - Variational quantum eigensolver (VQE)
  - Quantum approximate optimization algorithm (QAOA)
- Challenges and solutions for running such algorithms on noisy quantum hardware

Parametrized quantum circuits

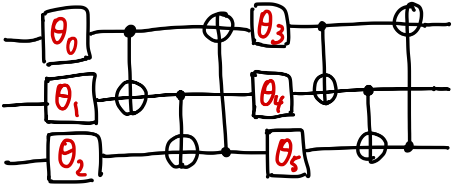
Some quantum operations depend on real-valued *parameters*.

These can be passed as arguments to a quantum function.

```
def circuit(x, y, z):
    qml.RX(x, wires=0)
    qml.RY(y, wires=1)
    qml.RZ(z, wires=2)
```

We call these **parametrized quantum circuits**.

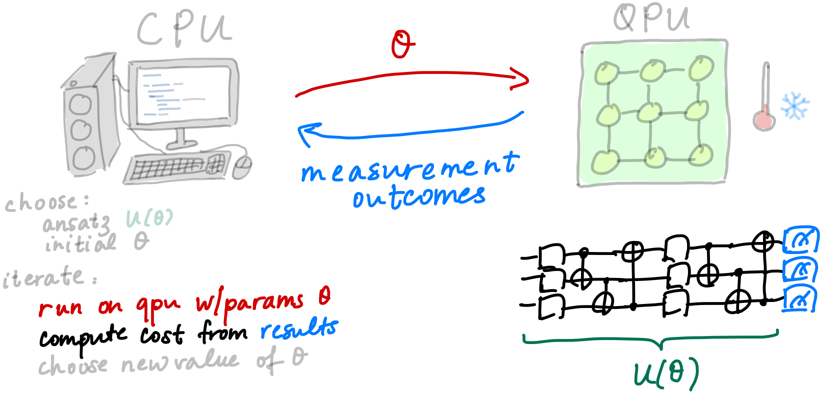
Parametrized quantum circuits




```
def parametrized_circuit(theta):
    qml.RX(theta[0], wires=0)
    qml.RX(theta[1], wires=1)
    qml.RX(theta[2], wires=2)
    qml.CNOT(wires=[0, 1])
    qml.CNOT(wires=[1, 2])
    qml.CNOT(wires=[2, 0])
    qml.RX(theta[3], wires=0)
    qml.RX(theta[4], wires=1)
    qml.RX(theta[5], wires=2)
    qml.CNOT(wires=[0, 1])
    qml.CNOT(wires=[1, 2])
    qml.CNOT(wires=[2, 0])
```

Parametrized quantum circuits

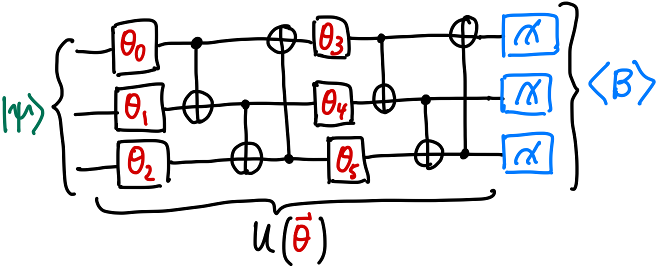
Parametrized circuits are used to assist in evaluation of a cost function which represents a particular problem.



Parametrized quantum circuits

We are trying to *find optimal values* for these parameters in order to minimize the cost, which represents the solution to the problem.

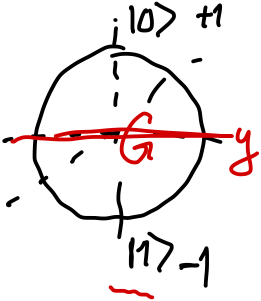
Expectation values are often used to construct a cost function.

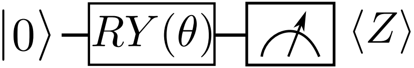


$$\min_{\vec{\theta}} \langle B \rangle = \min_{\vec{\theta}} \langle \psi | U^\dagger(\vec{\theta}) B U(\vec{\theta}) | \psi \rangle$$

Expectation values and objective functions

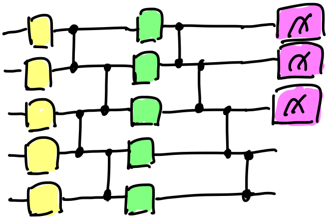
Example: find the value of θ which minimizes $\langle Z \rangle$.

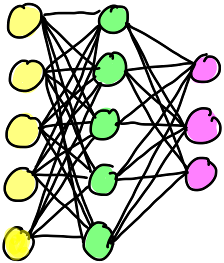




Easy to solve by hand ($\theta^* = \pi$). How can we *train* the quantum circuit to learn the optimal value?

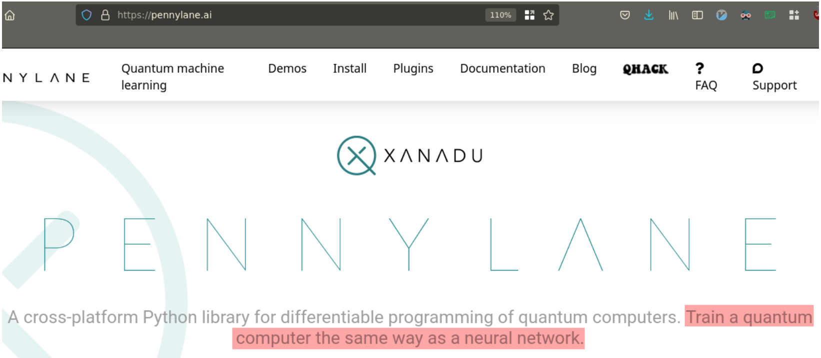
Working with variational quantum circuits is a lot like working with neural networks (architecture and layer design, training to determine optimal weights, etc.)





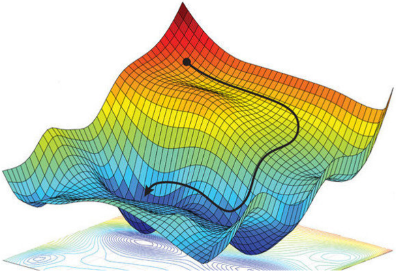
Variational circuits are often termed *quantum neural networks*.

Training variational quantum circuits



Training variational quantum circuits

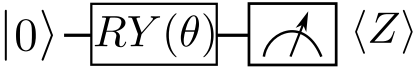
Circuits can be trained using standard optimization techniques *on a classical computer* such as gradient descent.



... but how do we compute the gradient of a quantum circuit?

Image credit: A. Amini, A. P. Soleimany, S. Karaman, D. Rus. *Spatial Uncertainty Sampling for End-to-End Control*. NIPS 2017.

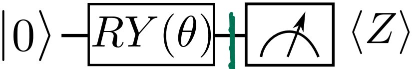
Gradients of quantum circuits



Key point: the expectation values measured at the end are *functions* of the variational parameters, i.e.,

$$\langle Z \rangle = f(\theta)$$

We can compute such functions, then differentiate them.



$B, \; |\psi\rangle$

$\langle B\rangle = \langle\psi|B|\psi\rangle$

$|\psi\rangle = RY(\theta)|0\rangle$

Let's compute the analytical expression for $\langle Z\rangle$:

$$RY(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

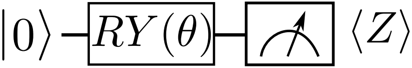
$$\langle Z\rangle = \langle 0|RY^\dagger(\theta)\cdot Z\cdot \underbrace{RY(\theta)\,|0\rangle}$$

$$= \langle 0|RY^\dagger(\theta)\, Z \left(\cos\tfrac{\theta}{2}|0\rangle + \sin\tfrac{\theta}{2}|1\rangle\right)$$

$$= \underbrace{\langle 0|RY^\dagger(\theta)}\left[\cos\tfrac{\theta}{2}|0\rangle - \sin\tfrac{\theta}{2}|1\rangle\right]$$

$$\langle 0|RY^\dagger(\theta) = \left[RY(\theta)\,|0\rangle\right]^\dagger$$

$$= \left[\cos\tfrac{\theta}{2}\langle 0| + \sin\tfrac{\theta}{2}\langle 1|\right]\left[\cos\tfrac{\theta}{2}|0\rangle - \sin\tfrac{\theta}{2}|1\rangle\right]$$



Let's compute the analytical expression for $\langle Z \rangle$:

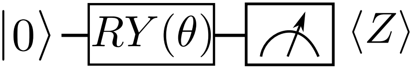
$$\left[\cos\frac{\theta}{2}\langle 0| + \sin\frac{\theta}{2}\langle 1|\right]\left[\cos\frac{\theta}{2}|0\rangle - \sin\frac{\theta}{2}|1\rangle\right]$$

$$= \cos^2\frac{\theta}{2}\underbrace{\langle 0|0\rangle}_{=1} + \cancel{\sin\frac{\theta}{2}\cos\frac{\theta}{2}\langle 1|0\rangle}$$

$$- \cancel{\cos\frac{\theta}{2}\sin\frac{\theta}{2}\langle 0|1\rangle} - \sin^2\frac{\theta}{2}\underbrace{\langle 1|1\rangle}_{=1}$$

$$= \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2}$$

$$\langle Z \rangle = \cos\theta$$



In order to train the circuit, we can use gradient descent; just compute the derivative of the function!

$$\langle Z \rangle = \cos\theta$$

$$\frac{\partial \langle Z \rangle}{\partial \theta} = -\sin\theta$$

But obviously, we don't want to do this by hand... use automatic differentiation instead! PennyLane will do this for us.

`qml.grad` is a *transform*: apply to a QNode to obtain a function that computes the *gradient* of that QNode.

```
@qml.qnode(dev)
def pqc(theta):
    qml.RY(theta, wires=0)
    return qml.expval(qml.PauliZ(0))

grad_fn = qml.grad(pqc)
grad_fn(theta)
```

Easy to do in software, but what about on hardware?

To train using gradient descent, then we'd have to:

1. guess an initial value for θ
2. run a circuit that computes the gradient at θ
3. use those results to produce an updated value for θ
4. repeat 2-3 until converged

You don't actually need a different circuit: you can use a circuit to compute its own gradient by running the circuit multiple times at different values, and combining the results.

The parameter-shift rule

Our circuit implements the function

$$f(\theta) = \cos\theta$$

The gradient of this function is

$$\frac{\partial f(\theta)}{\partial \theta} = -\sin\theta$$

Consider the following:

$$\frac{1}{2}\left[\cos\left(\theta + \frac{\pi}{2}\right) - \cos\left(\theta - \frac{\pi}{2}\right)\right]$$

The parameter-shift rule

Let's simplify this by noting that

$$\cos\left(\theta+\frac{\pi}{2}\right) = -\sin\theta$$
$$\cos\left(\theta-\frac{\pi}{2}\right) = \sin\theta$$

Then:

$$\frac{1}{2}\left[\cos\left(\theta+\frac{\pi}{2}\right) - \cos\left(\theta-\frac{\pi}{2}\right)\right] = \frac{1}{2}\left[-\sin\theta - \sin\theta\right]$$
$$= -\sin\theta$$
$$= \frac{\partial f}{\partial\theta}$$

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{1}{2}\left[\cos\left(\theta + \frac{\pi}{2}\right) - \cos\left(\theta - \frac{\pi}{2}\right)\right]$$

This is an example of a parameter-shift rule: we can compute gradients with respect to parameters of a circuit by evaluating them at shifted versions of those parameters!

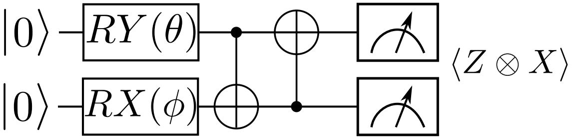
Next time

More generally, for all single-qubit rotation gates $U(\theta)$,

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{1}{2}\left[f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) \right]$$

where $f(\theta)$ is the function implemented by the *whole circuit*, with every other parameter held constant.

Let's try an example with more than one parameter.



Next time

Content:

- Embedding data in variational circuits
- The variational quantum classifier

Action items:

1. Assignment 3 (can do all problems)
2. Start working on prototype implementation for project

Recommended reading:

- QML glossary entries (`https://pennylane.ai/qml/glossary.html`):
  - Quantum differentiable programming
  - Parameter-shift rules
  - Quantum gradients
  - Variational circuit
- `https://arxiv.org/abs/2012.09265v2` (review paper)