

# Quantum Machine Learning using Covalent

---

A QAOA application

# Anna Hughes, PhD

## Quantum Software Engineer

Ph.D., UBC, Dept. of Physics & Astronomy

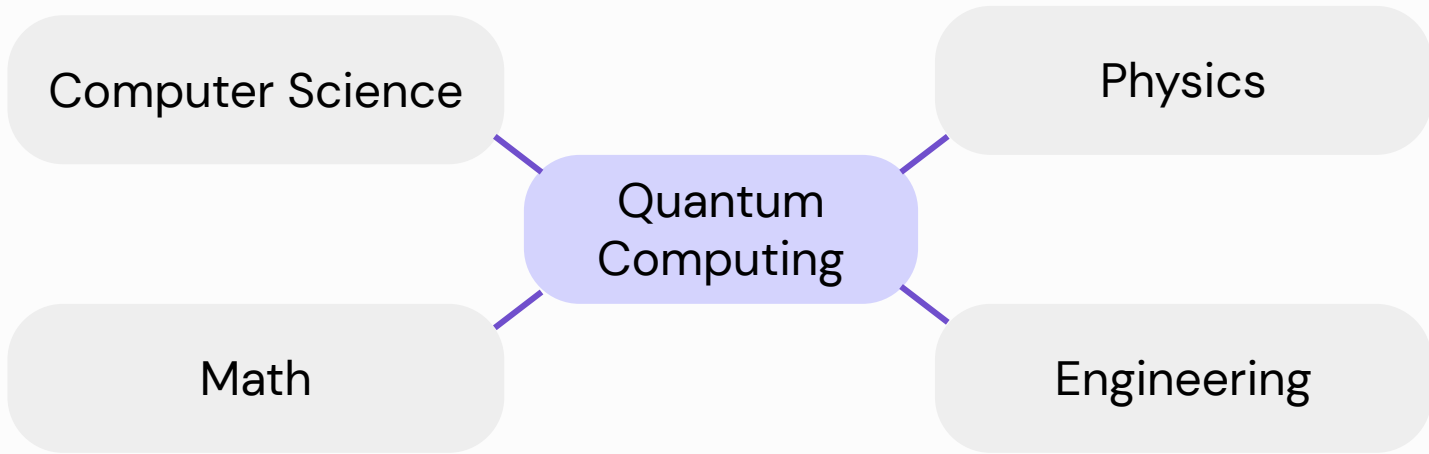
BSc, RPI, Dept. of Physics

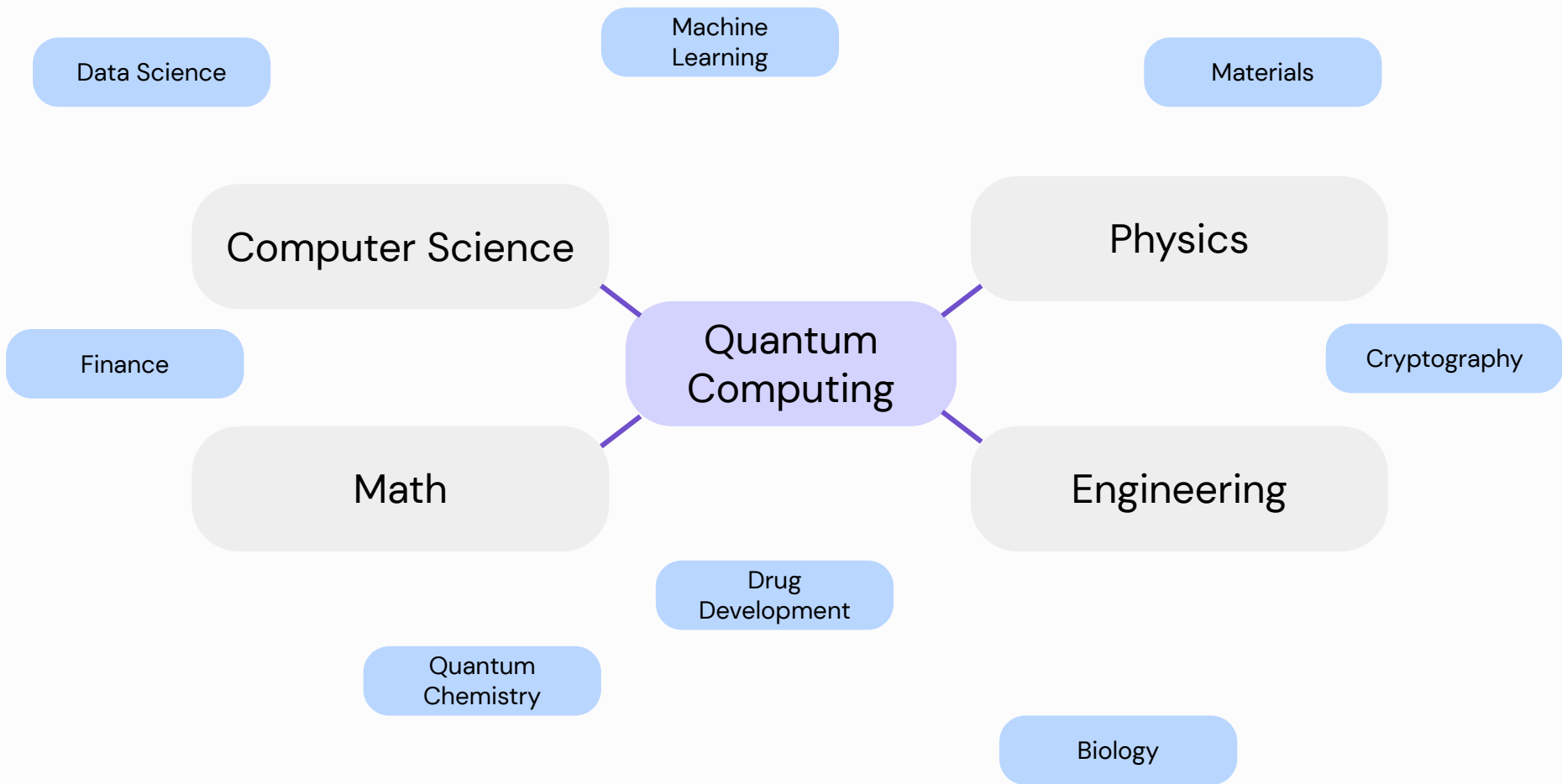




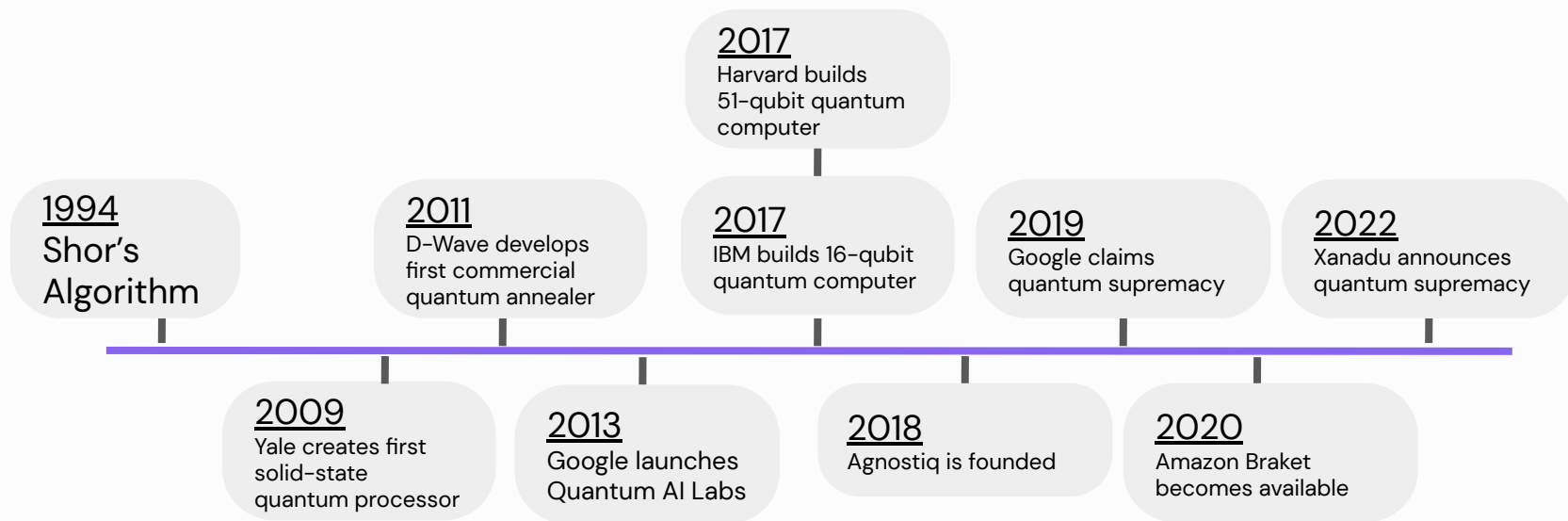
# Introduction to Quantum Computing

---





# A Timeline of Quantum Computing



# A Timeline of Quantum Computing

1945  
First programmable  
digital computer

1994  
Shor's Algorithm

2009  
Yale creates first  
solid-state quantum  
processor

2011  
D-Wave develops first  
commercial quantum  
annealer

2013  
Google launches  
Quantum AI Labs

2017  
Harvard builds 51-qubit  
quantum computer

2017  
IBM builds 16-qubit  
quantum computer

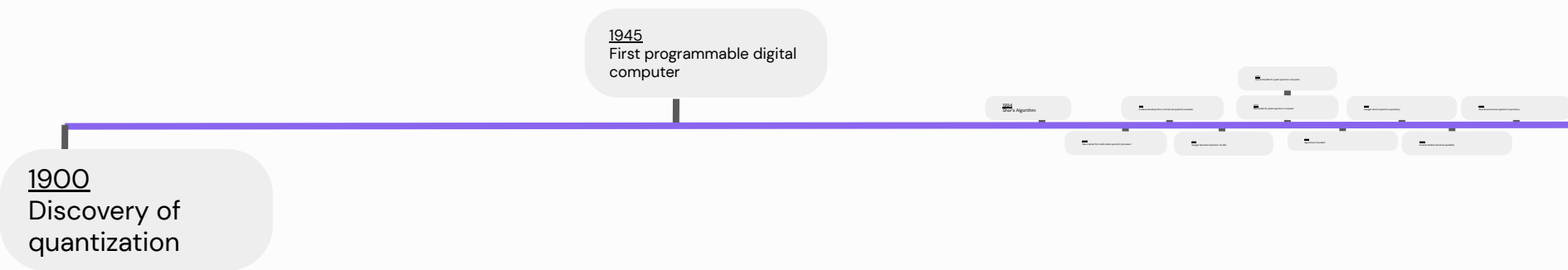
2018  
Agnostiq is founded

2019  
Google claims quantum  
supremacy

2020  
Amazon Braket  
becomes available

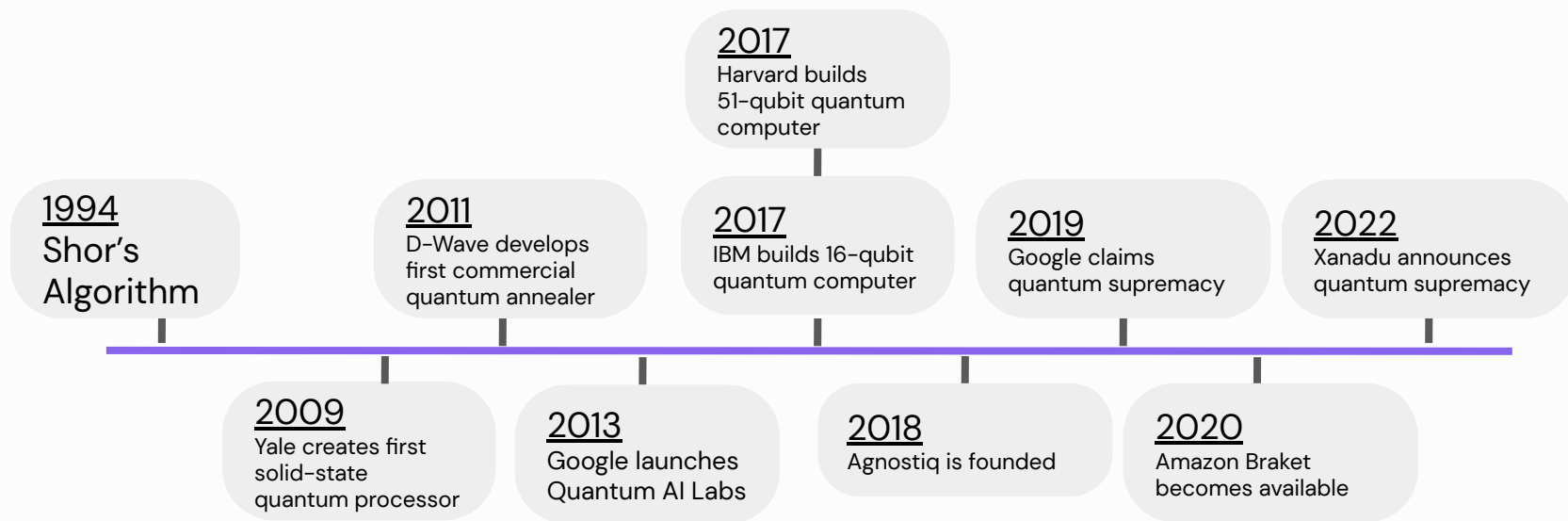
2022  
Xanadu announces  
quantum supremacy

# A Timeline of Quantum Computing



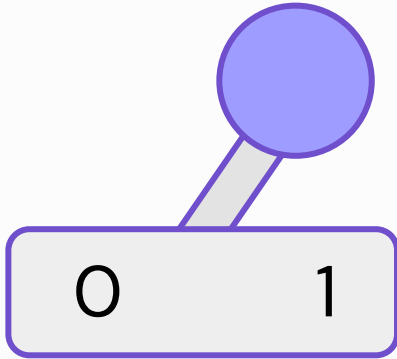


# A Timeline of Quantum Computing



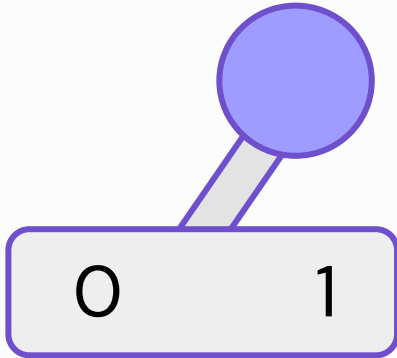
# Classical Computers

Composed of **bits**, which can take on values of 0 or 1



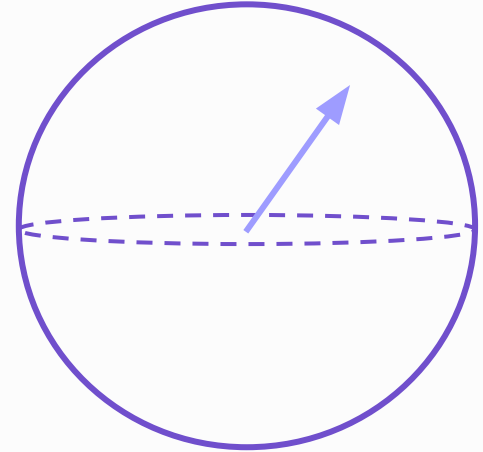
# Classical Computers

Composed of **bits**, which can take on values of 0 or 1

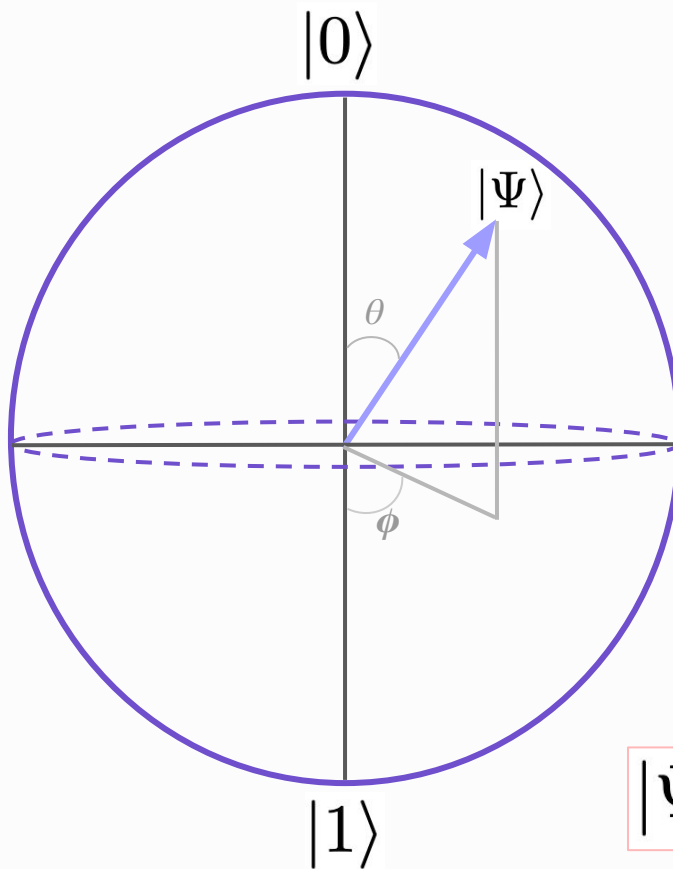


# Quantum Computers

Composed of **qubits**, which can be in a superposition of 0 and 1



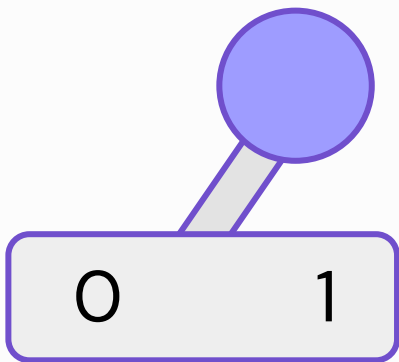
# Bloch Sphere



$$|\Psi\rangle = c_0 |0\rangle + c_1 |1\rangle$$

# Classical Computers

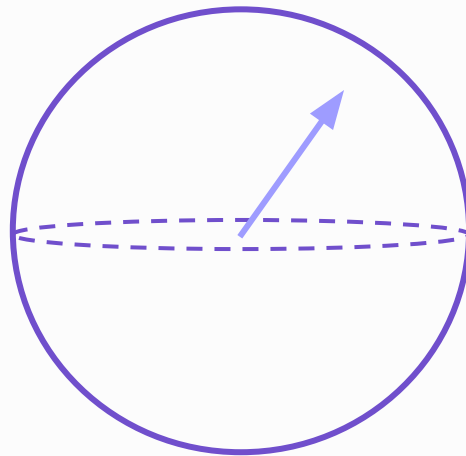
Composed of **bits**, which can take on values of 0 or 1



Deterministic measurements

# Quantum Computers

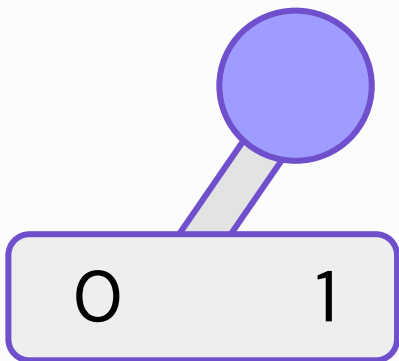
Composed of **qubits**, which can be in a superposition of 0 and 1



Probabilistic measurements

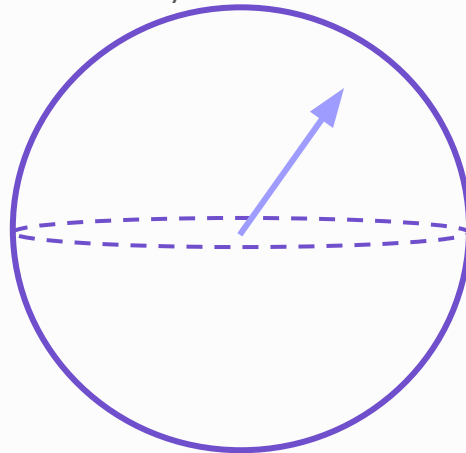
# Classical Computers

If you have  $N$  bits, you have  $2^N$  states that you can only execute 1 at a time (or in parallel)

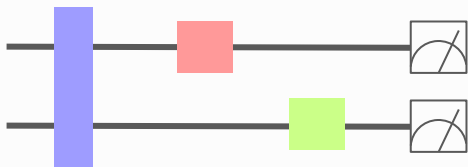


# Quantum Computers

If you have  $N$  qubits, you can encode all  $2^N$  components into one state simultaneously

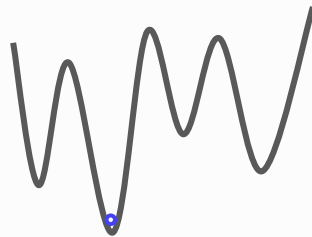


# Types of Quantum Computers.



## Gate-Based

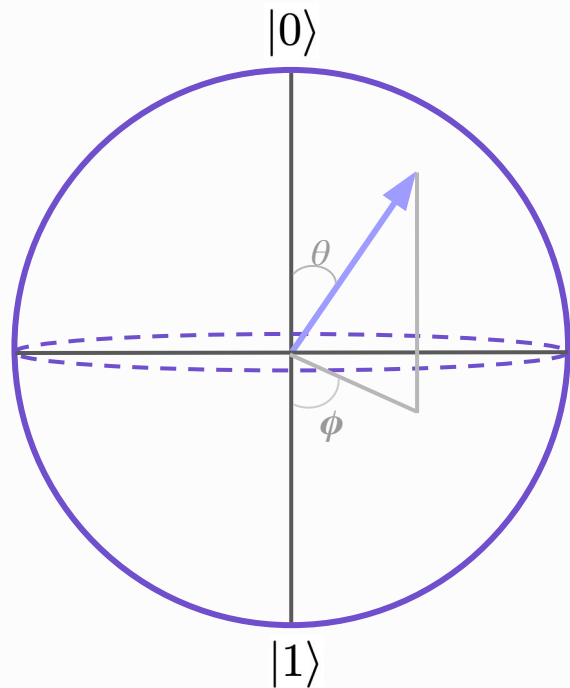
- Broad applications
- Apply gates, or circuit operations, to quantum state
- Universal computer



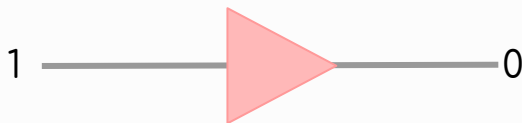
## Quantum Annealers

- Search an energy landscape for the lowest-energy solution
- Problem encoded as a Hamiltonian
- Can only solve optimization problems

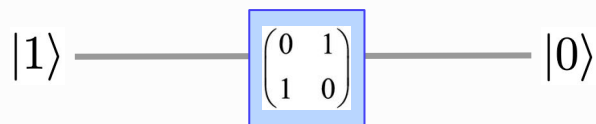
# Quantum Gates



○ Classical NOT Gate

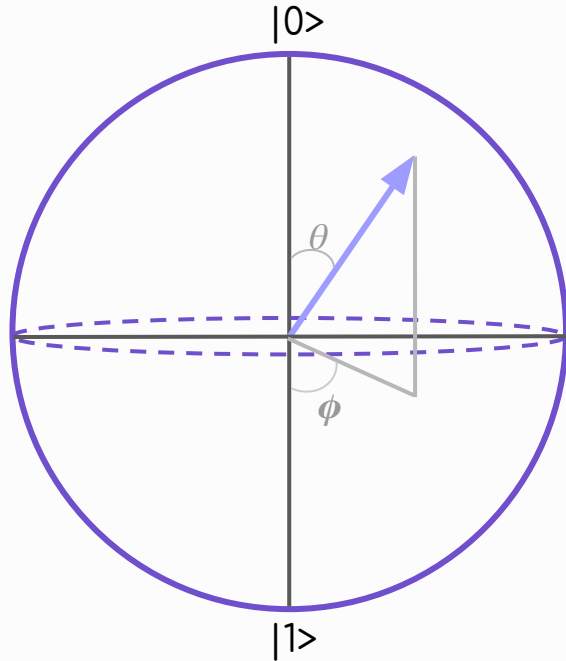


○ Pauli X Gate





# Quantum Gates



- Pauli X Gate

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

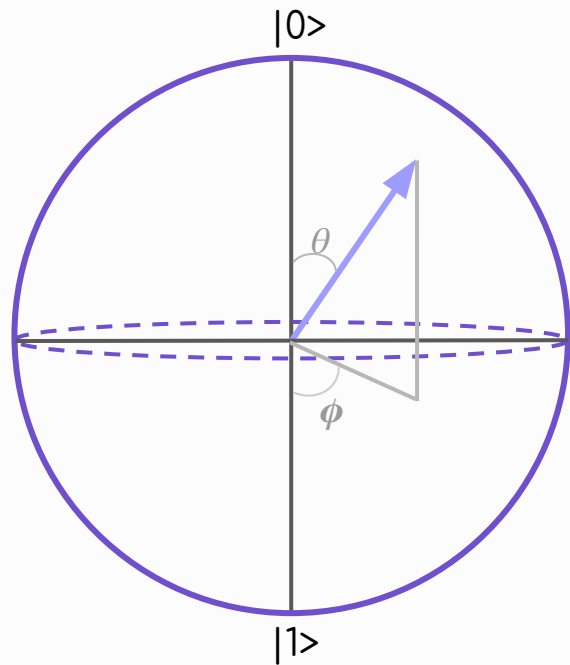
- Pauli Y Gate

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

- Pauli Z Gate

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

# Quantum Gates



- Pauli X Gate

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- Pauli Y Gate

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

- Pauli Z Gate

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

covalent.xyz

- Hadamard Gate

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

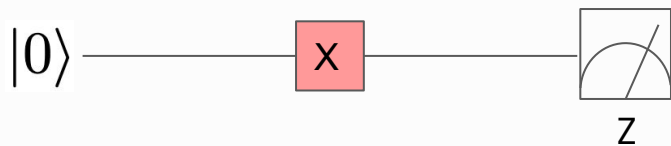
- Controlled NOT Gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

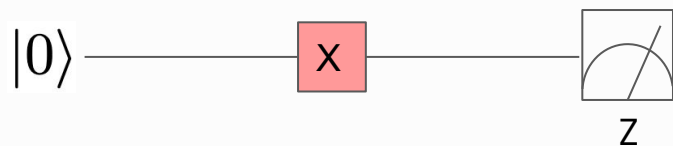
- Toffoli Gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

# Building a Quantum Circuit

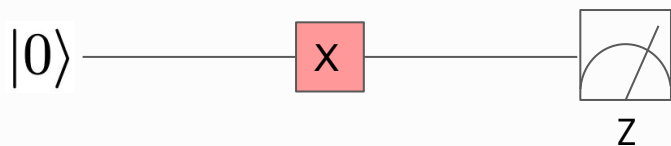


# Building a Quantum Circuit



$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \xrightarrow{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \xrightarrow{\begin{bmatrix} 0 & 1 \end{bmatrix}} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

# Building a Quantum Circuit

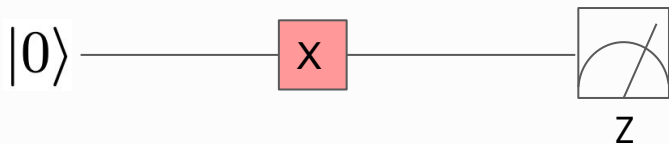


$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \xrightarrow{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \xrightarrow{\begin{bmatrix} 0 & 1 \end{bmatrix}} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$|0\rangle \xrightarrow{\sigma_x} |0\rangle \xrightarrow{\langle 1 | \sigma_z | 1 \rangle}$$

$$= |1\rangle$$

# Building a Quantum Circuit

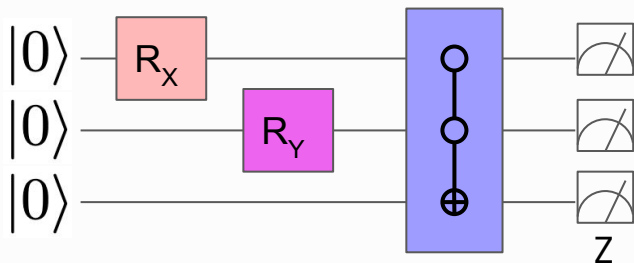


```
import pennylane as qml

dev1 = qml.device("default.qubit", wires=1)

@qml.qnode(dev1)
def circuit():
    qml.PauliX(wires=0)
    return qml.expval(qml.PauliZ(0))
```


# Building a Quantum Circuit



```
import pennylane as qml

dev1 = qml.device("default.qubit", wires=3)

@qml.qnode(dev1)
def circuit(params):
    qml.RX(params[0], wires=0)
    qml.RY(params[1], wires=1)
    qml.CNOT(wires=[0,1])
    qml.CNOT(wires=[1,2])
    return qml.expval(qml.PauliZ(0)),
    qml.expval(qml.PauliZ(1)), qml.expval(qml.PauliZ(2))
```



# Introduction to Quantum Machine Learning

---



# Machine Learning



A **model** is developed to describe and make predictions about data



Model **parameters** are tuned using a **training dataset**



The model is assessed by making predictions about a **test dataset**

# Machine Learning



A **model** is developed to describe and make predictions about data

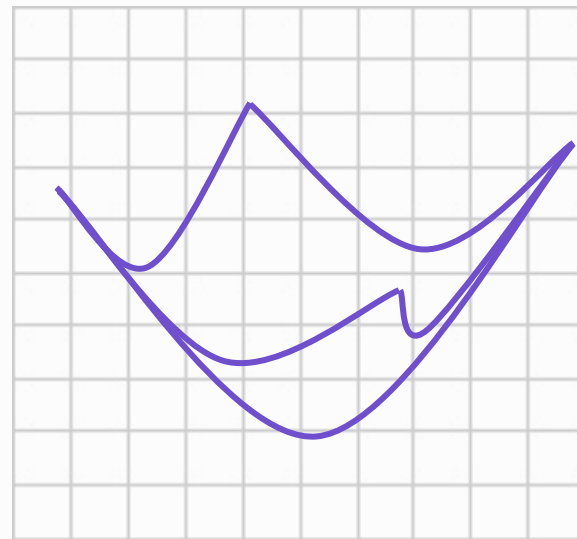


Model **parameters** are tuned using a **training dataset**



The model is assessed by making predictions about a **test dataset**

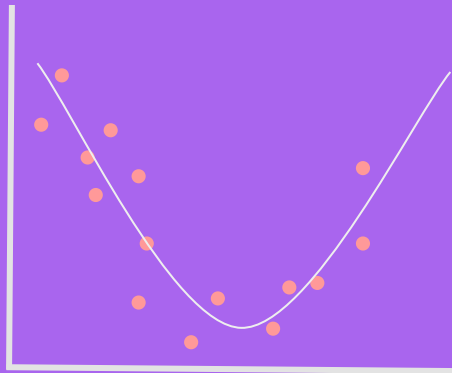
## Optimizing a cost function



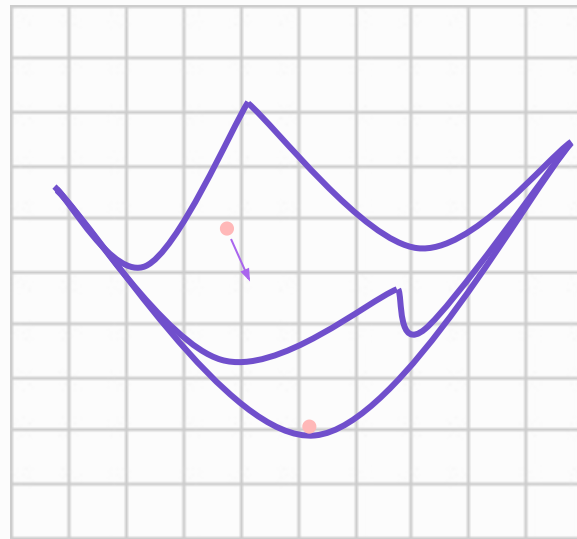
# Machine Learning Example



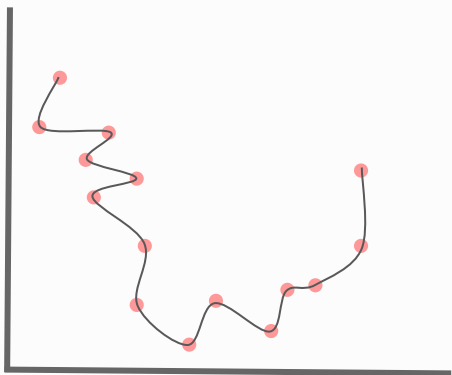
A **model** is developed to describe and make predictions about data



## Optimizing a cost function

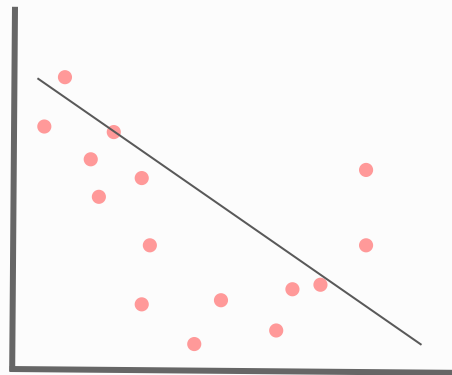


# Machine Learning Example



## Overfitting.

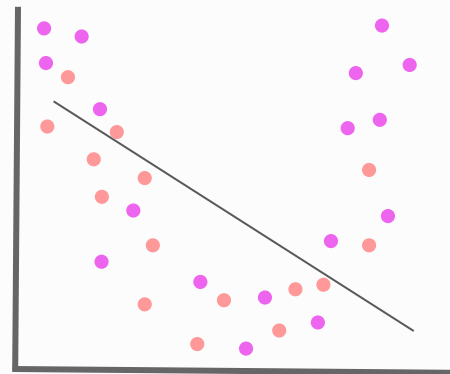
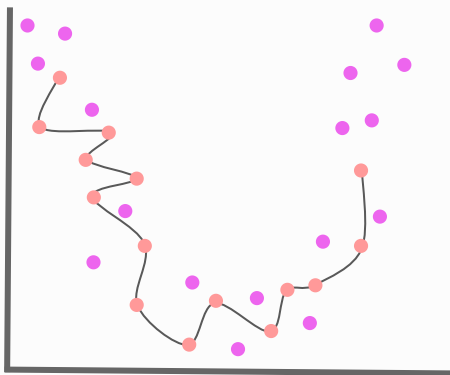
Model is too tailored to the training data and does not generalize to test data



## Underfitting.

Model is too **general** and does not well represent the training or test data

# Machine Learning Example



## Overfitting.

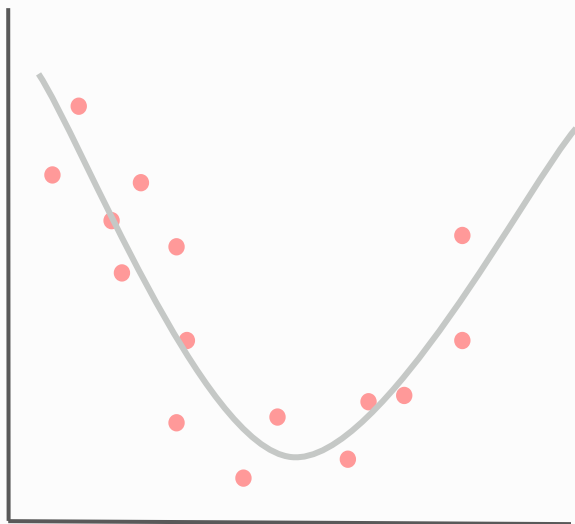
Model is too tailored to the training data and does not generalize to test data



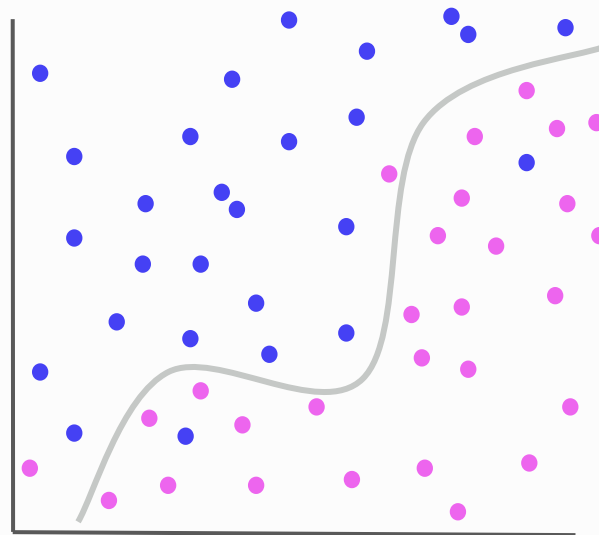
## Underfitting.

Model is too **general** and does not well represent the training or test data

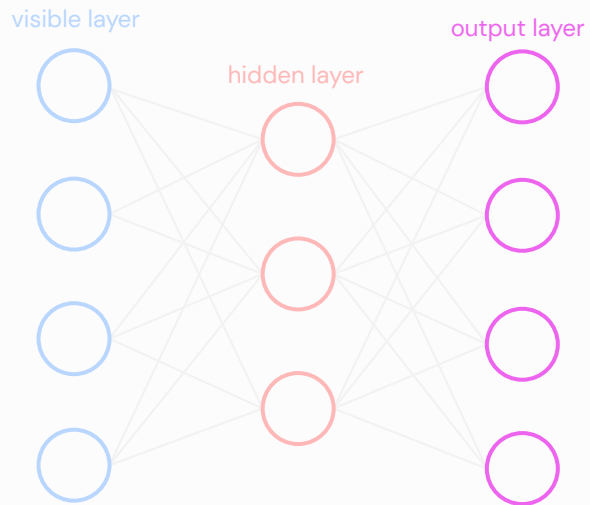
## Regression



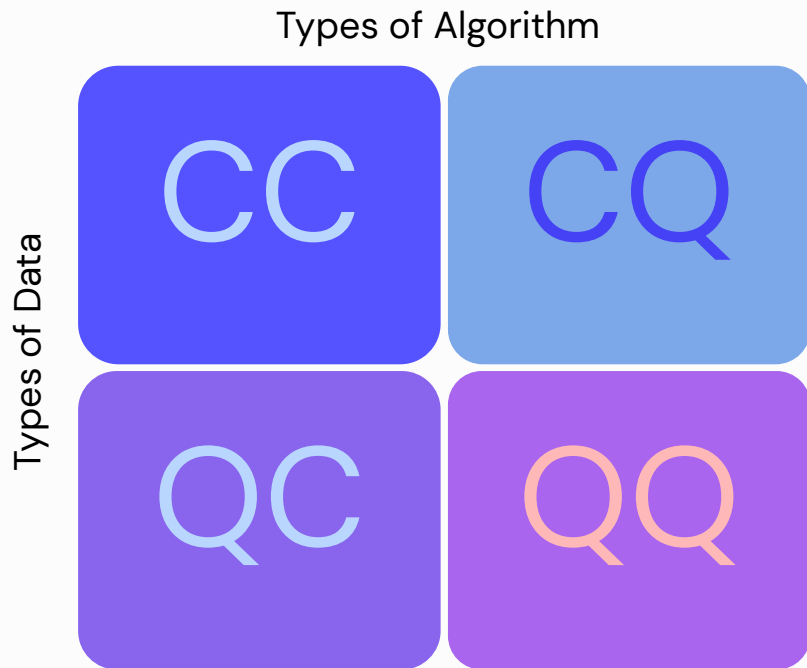
## Classification



# Neural Networks



# Quantum Machine Learning





# Quantum Machine Learning

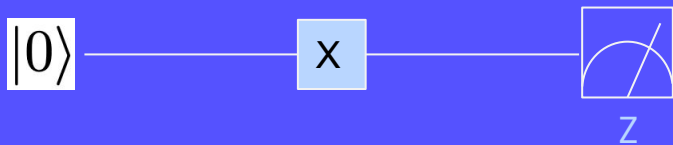
Types of Algorithm

Types of Data

CQ

classical data with  
quantum algorithms

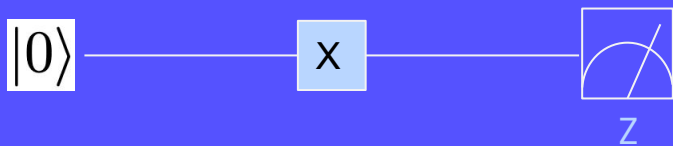
# Parameterized Quantum Circuits



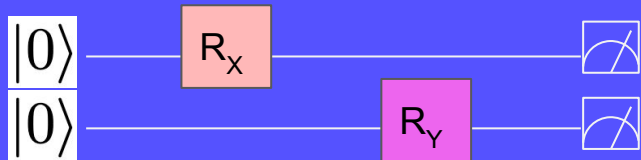
This circuit does not have tunable parameters



# Parameterized Quantum Circuits



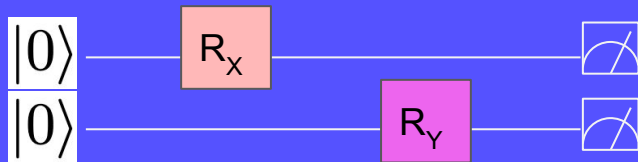
This circuit does not have tunable parameters



The input angles in  $R_x$  and  $R_y$  are tunable parameters



# Parameterized Quantum Circuits



The input angles in  $R_x$  and  $R_y$  are tunable parameters

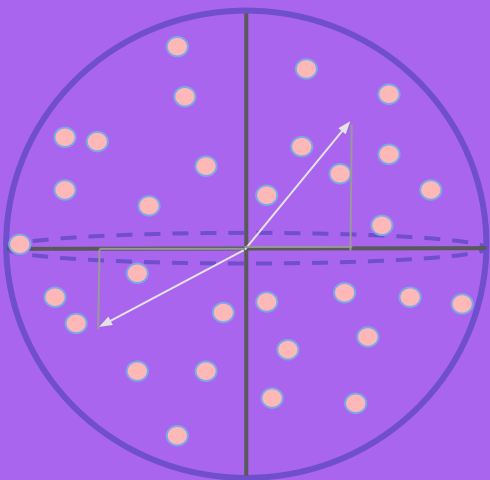
A quantum circuit with tunable parameters consists of unitary operations  $U(\theta)$  performed on  $n$  qubits

# Parameterized Quantum Circuits



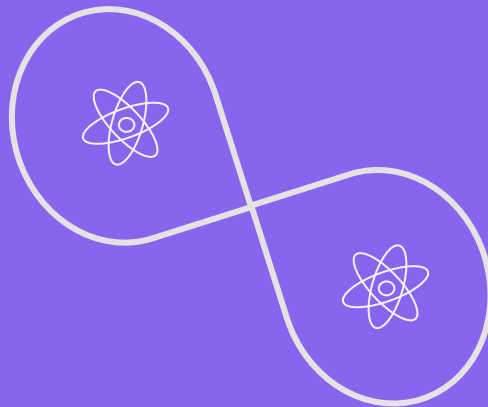
## Expressibility.

We want our quantum circuit to be able to span a wide subset of Hilbert Space!



## Entanglement.

Entangled qubits are difficult to simulate using a classical simulator.

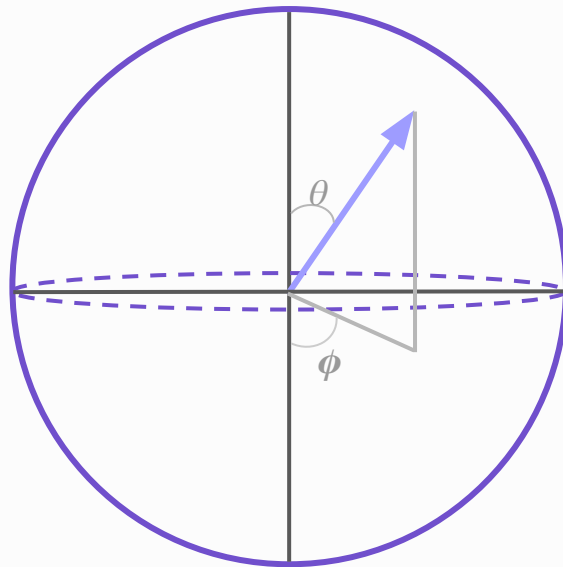


# Parameterized Quantum Circuits



## Expressibility.

We want our quantum circuit to be able to span a wide subset of Hilbert Space!

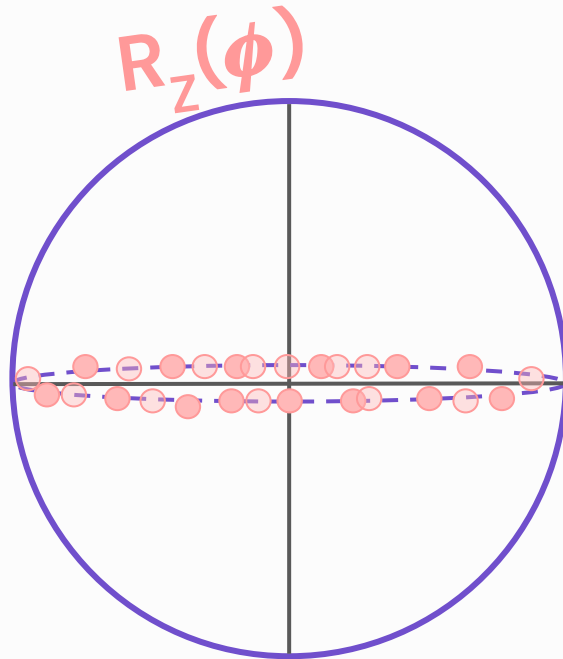


# Parameterized Quantum Circuits



## Expressibility.

We want our quantum circuit to be able to span a wide subset of Hilbert Space!

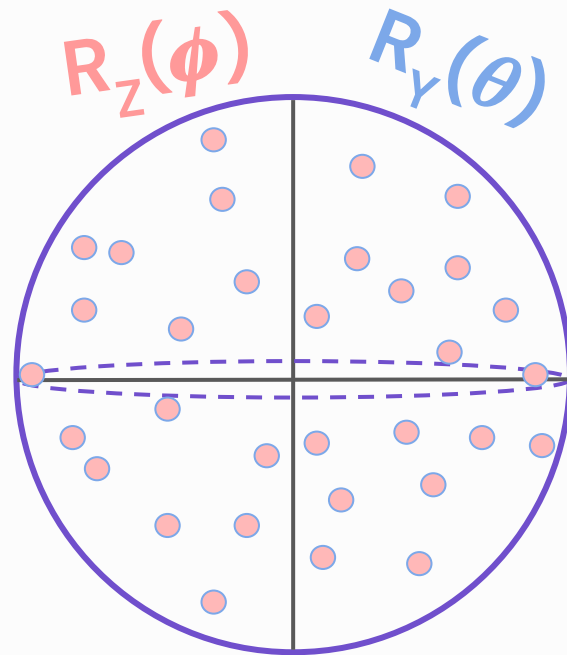


# Parameterized Quantum Circuits



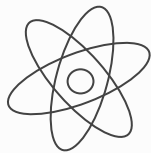
## Expressibility.

We want our quantum circuit to be able to span a wide subset of Hilbert Space!



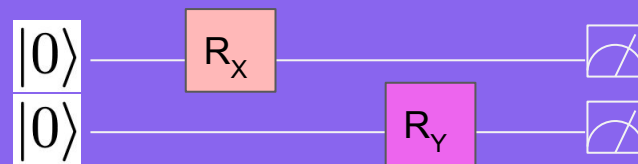


# Parameterized Quantum Circuits

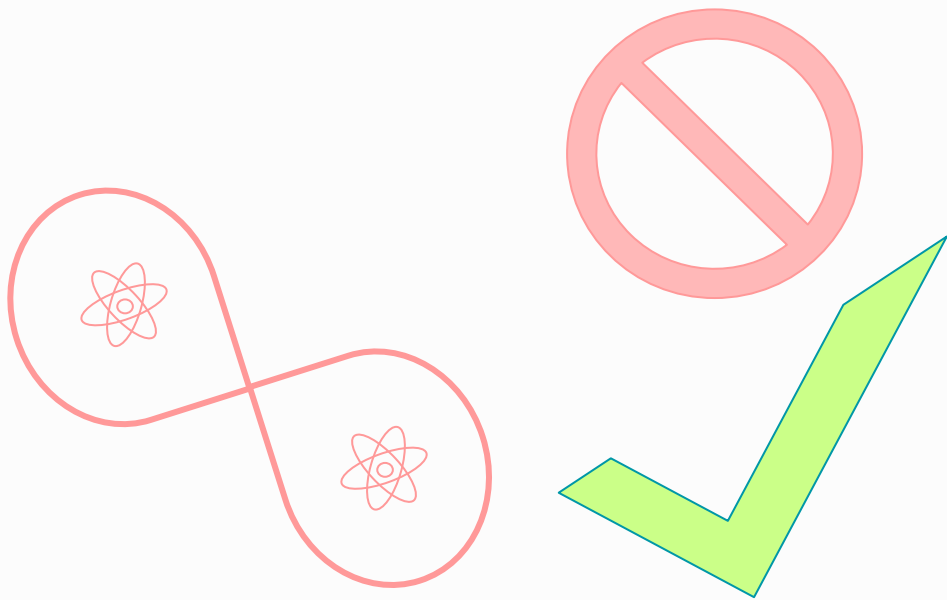


## Entanglement •

Entangled qubits are difficult to simulate using a classical simulator.

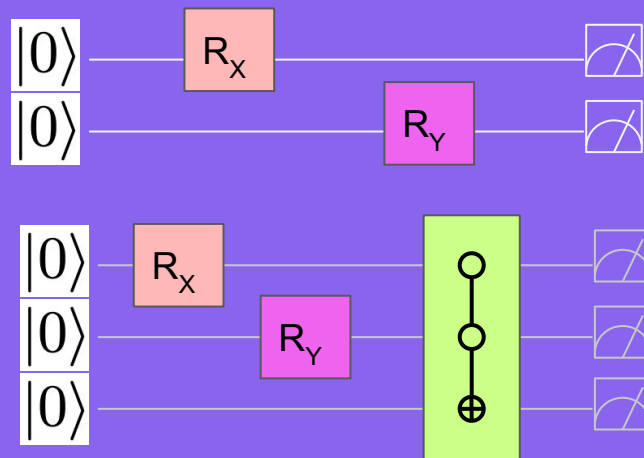


# Parameterized Quantum Circuits



## Entanglement

Entangled qubits are difficult to simulate using a classical simulator.



# Limitations in the NISQ Era.

## Coherence Time

The time a qubit is able to maintain its quantum state before it breaks down due to noise

## Qubit Connectivity

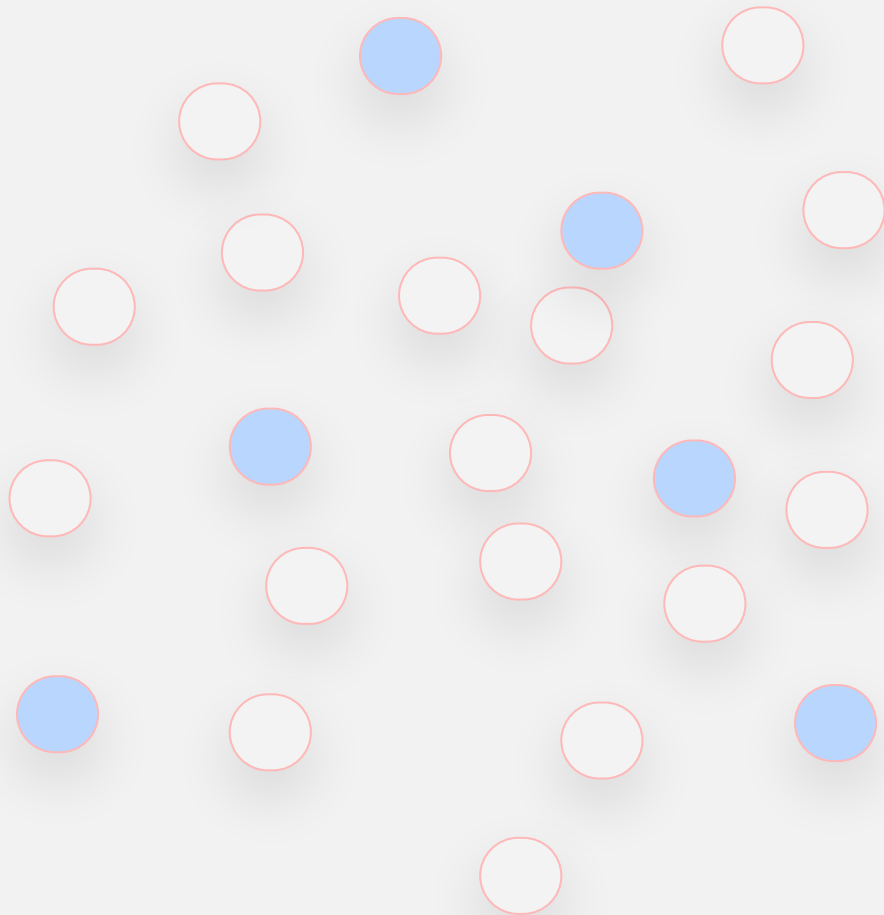
In today's quantum computers, not all qubits are able to interact with each other

# The Quantum Approximation Optimization Algorithm

---

# QAOA

- Combinatorial optimization problems are computationally expensive to solve exactly
- QAOA is a mathematical method to find approximate solutions
- It is simple to implement and can be run on NISQ devices



# Combinatorial Optimization Example

## The Traveling Salesman Problem

Given a list of cities and the distance between each pair of cities, what is the shortest possible route that visits each city exactly once and returns home at the end?



# QAOA problems involve...

$$Z \in \{0, 1\}^n$$

○ Bit String  $Z$

- Elements in the string are **binary-valued**
- $n$  is the total number of elements

$$C_\alpha = \begin{cases} 1, & \text{if } z \text{ satisfies clause } C_\alpha \\ 0, & \text{otherwise.} \end{cases}$$

○ Clause  $C$

- For all elements in  $Z$ , if the element satisfies  $C_\alpha$  it is given a value of **1**

$$C(z) = \sum_{\alpha=1}^m C_\alpha(z)$$

○ Clause Satisfaction

- $m$  is the total number of clauses
- The greater the value of  $C(z)$ , the better the overall **clause satisfaction**

# Hamiltonians

$$U(H, t) = e^{-iHt/\hbar}$$

- Using a Hamiltonian to construct a quantum circuit
- Hamiltonians can be evolved using the time evolution operator



## Cost Hamiltonian

The **expectation value** of the cost hamiltonian is the cost function to be optimized.

## Mixer Hamiltonian

A layer to increase the **mix** the quantum state so the angles  $\gamma$  and  $\beta$  can be optimized.

## Cost Hamiltonian

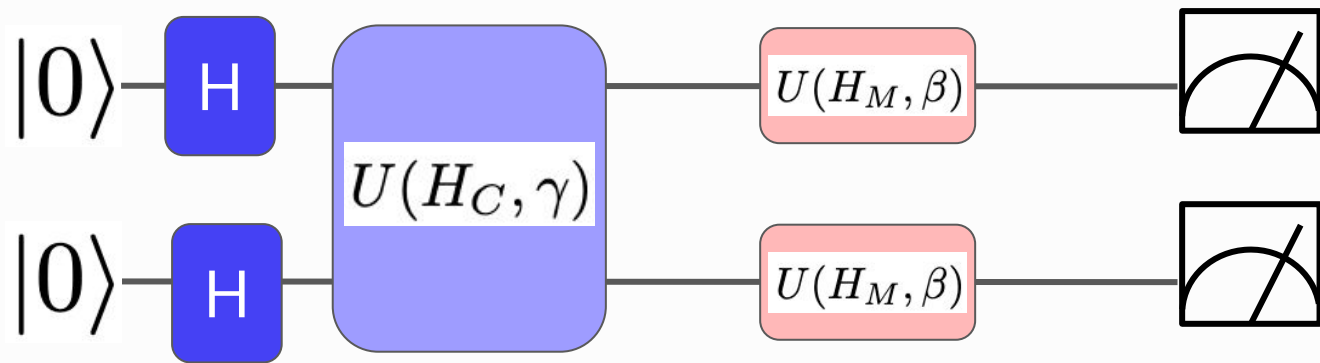
The **expectation value** of the cost hamiltonian is the cost function to be optimized.

$$U(H_C, \gamma) \equiv e^{-i\gamma C}$$

## Mixer Hamiltonian

A layer to increase the **complexity** of the quantum circuit.

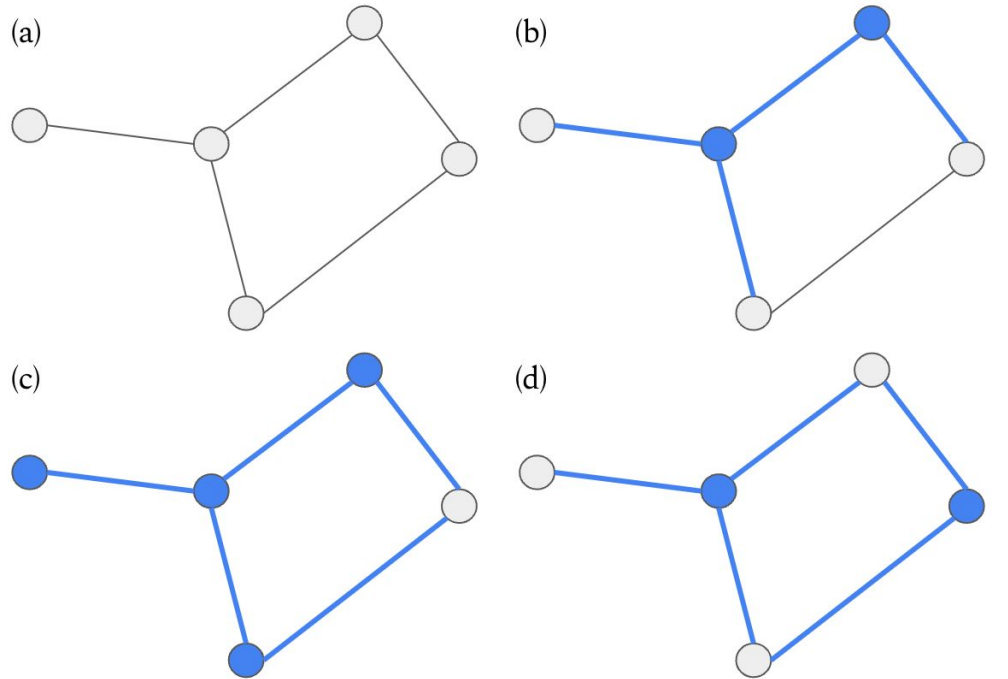
$$U(H_M, \beta) \equiv e^{-i\beta\sigma_x}$$



# The Minimum Vertex Cover Problem

A **combinatorial optimization** problem to find the **minimum** number of vertices needed to cover all edges in a graph

There are no exact solutions to the minimum vertex cover problem that can be found in **polynomial time**



# Solving QAOA Problems with Quantum Circuits

Define a cost and a mixer hamiltonian

Construct the cost and mixer layers

Construct a circuit by alternating cost and mixer layers

Use classical techniques to optimize the circuit parameters

Get the approximate solutions by measuring the circuit output

## Define a cost and a mixer hamiltonian

```
from pennylane import qaoa
```

```
cost_h, mixer_h = qaoa.min_vertex_cover(graph, constrained=False)
```

## Construct the cost and mixer layers

```
def qaoa_layer(gamma, alpha):  
    qaoa.cost_layer(gamma, cost_h)  
    qaoa.mixer_layer(alpha, mixer_h)
```

## Construct a circuit by alternating cost and mixer layers

```
import pennylane as qml

wires = range(4)

depth = 2

def circuit(params, **kwargs):
    for w in wires:
        qml.Hadamard(wires=w)

    qml.layer(qaoa_layer, depth, params[0], params[1])
```



# Use classical techniques to optimize the circuit parameters

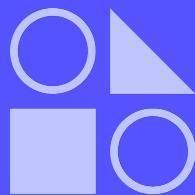
```
dev = qml.device("lightning.qubit", wires=qubits)

@qml.qnode(dev)
def cost_function(params):
    circuit(params)
    return qml.expval(cost_h)

optimizer = qml.GradientDescentOptimizer()
steps = 70
params = np.array([[0.5, 0.5], [0.5, 0.5]],
                  requires_grad=True)
```

# Get the approximate solutions by measuring the circuit output

```
for i in range(steps):  
    params = optimizer.step(cost_function, params)  
  
print params
```



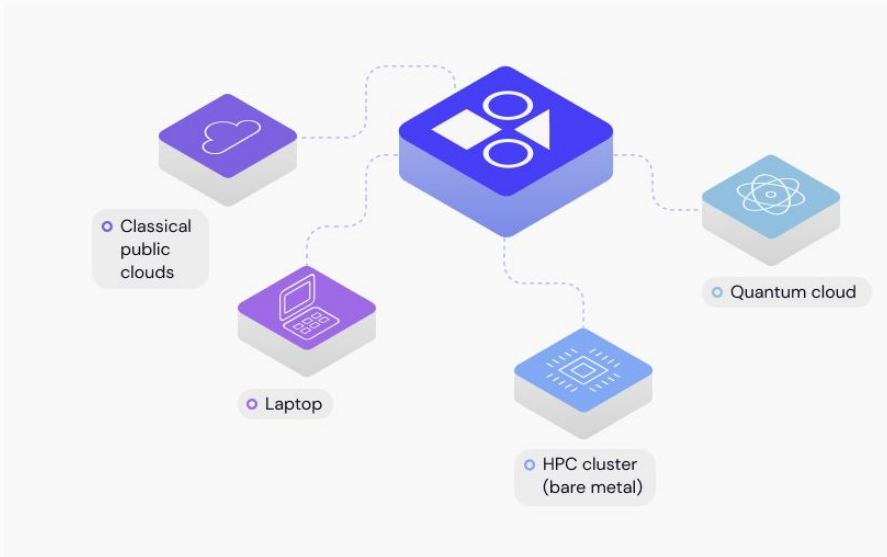
**Covalent.**

covalent.xyz

# Covalent is an open source workflow orchestration platform for quantum and high performance computing.

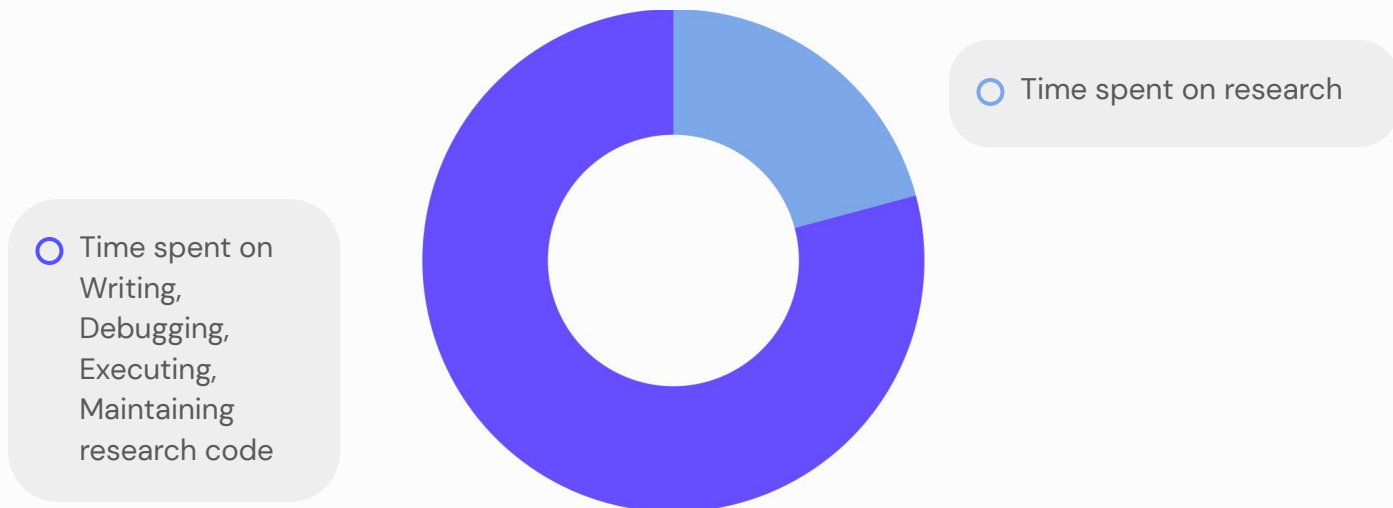
Covalent is designed to make your experiments:

- Modular
- Scalable
- Reproducible



# Why does Covalent exist?

Computational research



# Real-time monitoring

## Visual overview

Visualize and share your workflow to transfer knowledge as fast as possible

## Status/Error updates

Get real time Updates on errors and completion

## Checkpoints

Stores anything and everything automatically without a single line of code

## Meta-data

Maintains details from environment to hardware used

## Parameter

Never forget the hyper-parameters that worked

## Interactive

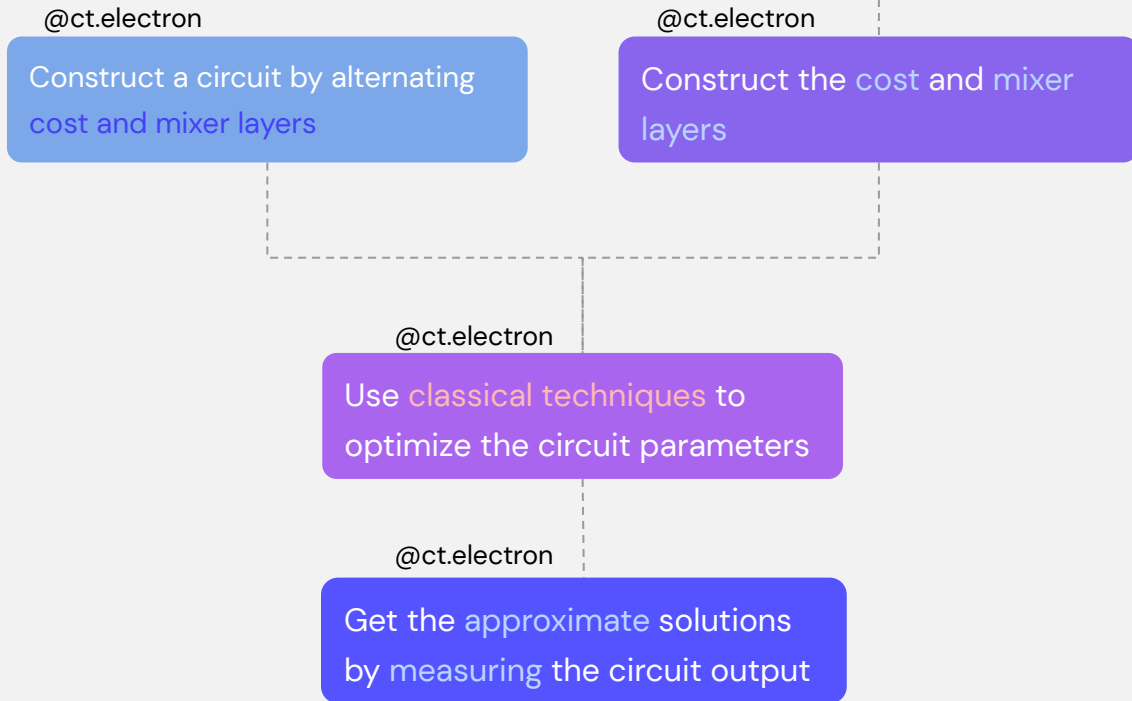
Start/stop/manage your HPC jobs right from the UI

## Beautiful and interactive UI

bring your workflows to life!

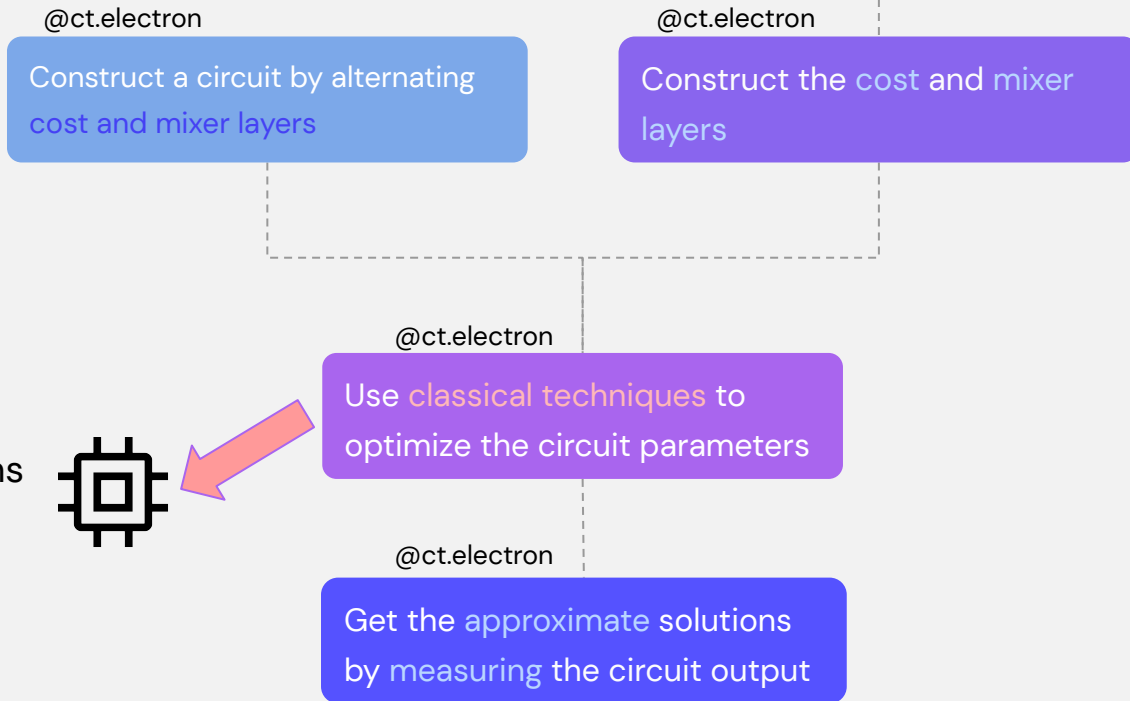
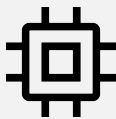


# Modularize your experiments.



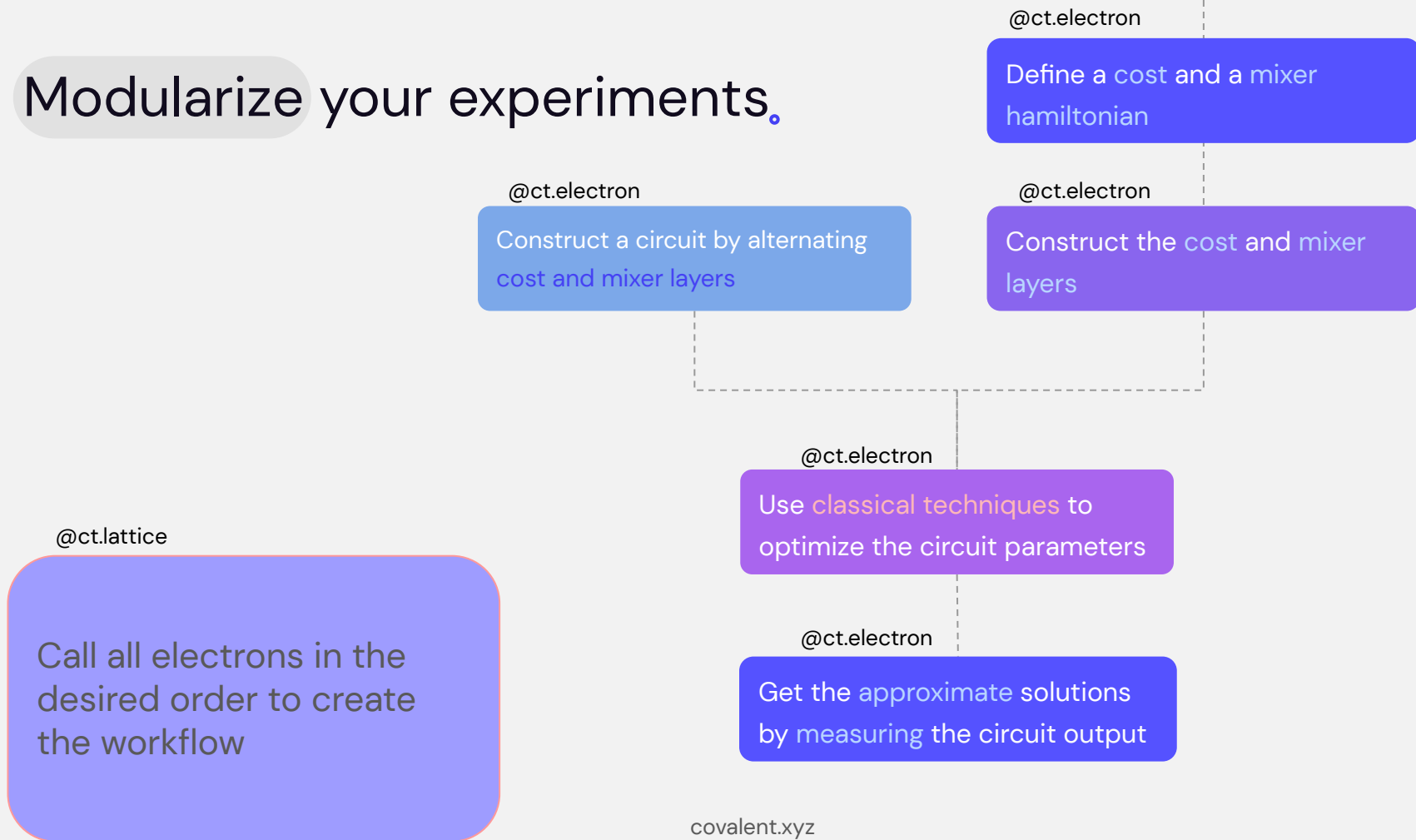
# Modularize your experiments.

Can dispatch specific electrons to remote devices





# Modularize your experiments.



# Thank You

---



[covalent.xyz](https://covalent.xyz)



[agnostiqHQ/covalent](https://github.com/agnostiqHQ/covalent)



[@covalentxyz](https://twitter.com/covalentxyz)