CPEN 400Q Lecture 08
Hands-on with the variational quantum classifier (VQC)

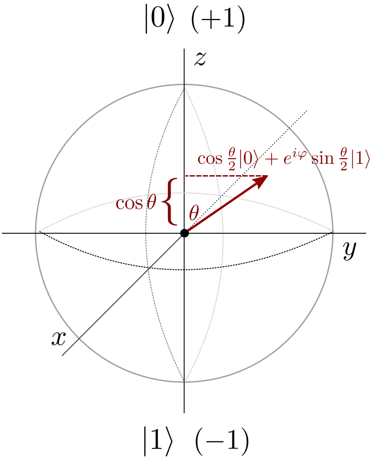
Friday 3 February 2023

Announcements

- Quiz 4 Monday at beginning of class
- Assignment 1 due Monday
- Literacy assignment grading in progress

Last time

We measured expectation values of observables, and related them to projective measurements / the Bloch sphere for a single qubit.



Last time

We computed expectation values of observables by hand.

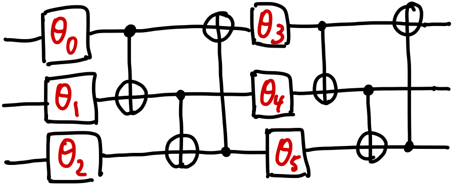
$$\langle B \rangle = \langle \psi | B | \psi \rangle$$

We computed expectation values of observables in PennyLane.

```
dev = qml.device('default.qubit', wires=1)

@qml.qnode(dev)
def measure_z():
    qml.RX(2*np.pi/3, wires=0)
    return qml.expval(qml.PauliZ(0))
```

Last time

We introduced parametrized quantum circuits.

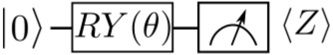


We computed gradients of expectation values w.r.t. parameters.

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

grad_fn = qml.grad(pqc)
grad_value = grad_fn(0.2)
```

Last time

We used PennyLane to apply gradient descent and find the optimal parameter than minimizes an expectation value.

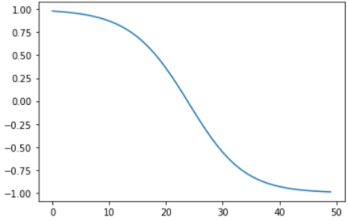


Demo 3: training a small PQC

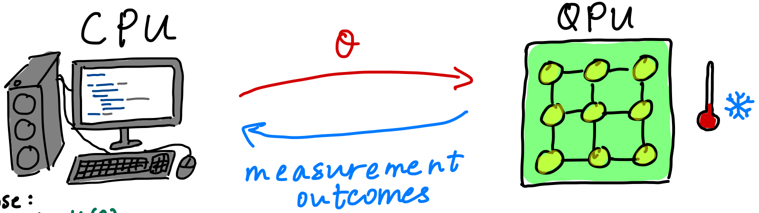
[30]:
```
opt = qml.GradientDescentOptimizer(stepsize=0.1)

num_iterations = 50

storage = []

init_param = np.array(0.2)
params = init_param.copy()

for _ in range(num_iterations):
    params, _cost = opt.step_and_cost(pqc, params)
    storage.append(_cost)
```

[31]:
```
plt.plot(storage)
```

[31]: [<matplotlib.lines.Line2D at 0x7f92e339ce50>]

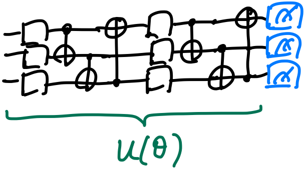


We introduced the idea of variational algorithms.



choose:
ansatz $U(\theta)$
initial θ

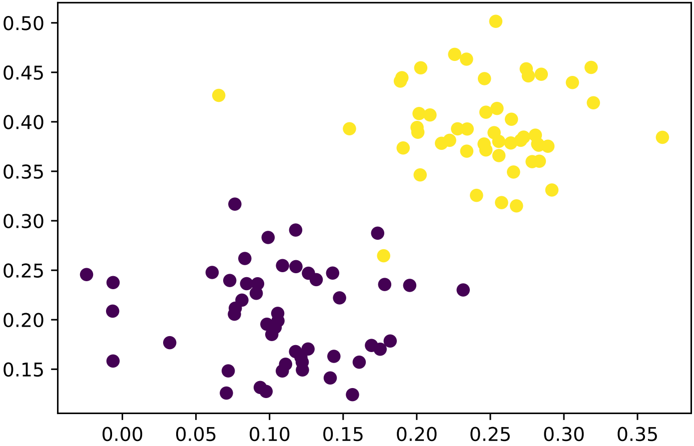
iterate:
run on qpu w/params θ
compute cost from results
choose new value of θ



Learning outcomes

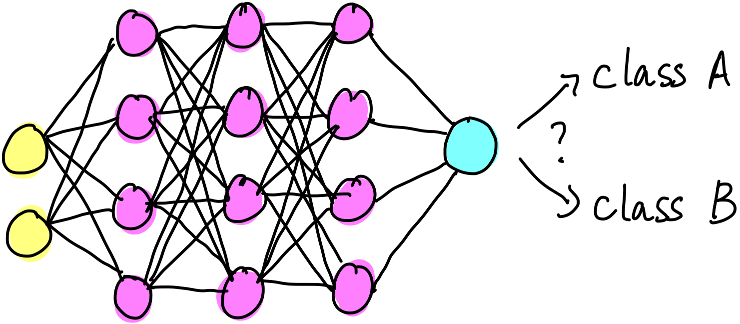
- Describe 3 different ways to embed data into a variational quantum classifier
- Classify real data with the VQC!

Overarching problem: binary classification



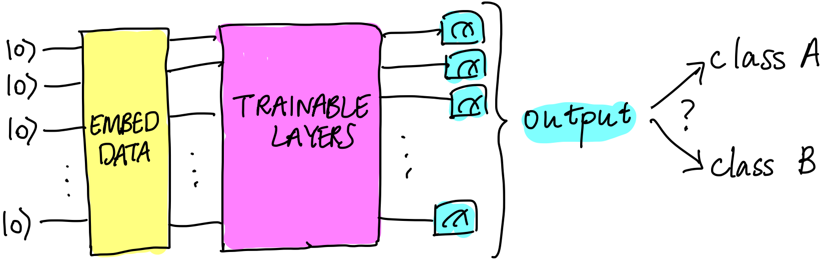
Overarching problem: binary classification

Consider how classification can be done with a neural network:



We are going to train a quantum circuit to *classify* this data.

The general structure of our model is:



Building a quantum machine learning model

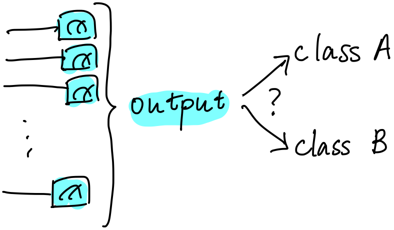
Need to figure out:

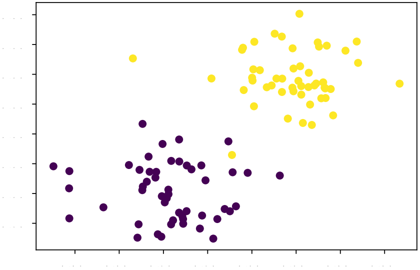
1. How to set up a cost function: what to measure, and how to use it to determine classes
2. How to get the data into the circuit
3. What the trainable part of the circuit should look like

(These are loosely ordered in terms of difficulty)

1. How to set up a cost function: what to measure, and how to use it to determine classes
2. How to get the data into the circuit
3. What the trainable part of the circuit should look like

Running the quantum circuit gives us an expectation value: we can use this to design a meaningful cost function.





Measurements and cost functions

Use a simple least-squares fit: minimize the difference between the computed expectation values and the true values:

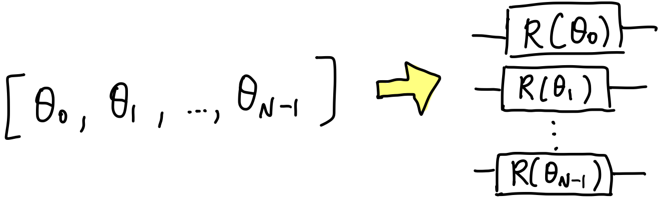
```
def cost(data, true_labels):
    total = 0.0

    for data_point, label in zip(data, true_labels):
        computed_exp_val = circuit(data_point)
        total += (computed_exp_val - label) ** 2

    return total / len(data)
```

Overarching problem

1. How to set up a cost function: what to measure, and how to use it to determine classes
2. How to get the data into the circuit
3. What the trainable part of the circuit should look like

Angle embedding

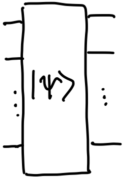
N features $\rightarrow$ N qubits and N gates; simple encoding scheme.



Amplitude embedding

N features $\to \lceil \log_2 N \rceil$ qubits.

$$[\theta_0,\ \theta_1,\ \ldots,\ \theta_{N-1}]$$



$$|\psi\rangle \approx \theta_0|0\rangle + \theta_1|1\rangle + \cdots + \theta_{N-1}|N-1\rangle$$

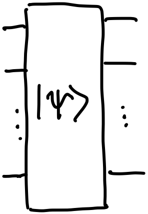
Circuits can be designed that perform this using $O(N)$ gates (this is what `qml.MottonenStatePreparation` does).

Basis embedding

N m-bit features $\rightarrow$ m qubits, N terms in the superposition.

$$[\theta_0, \theta_1, \ldots, \theta_{N-1}] \Rightarrow$$

$$\theta_i \in \{0, 1\}^m$$

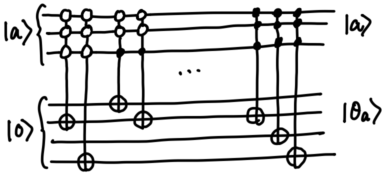


$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\theta_i\rangle$$

Circuit construction methods exist that use $O(Nm)$ gates (and require auxiliary qubits).

$N = 2^n$ n-bit addresses and m-bit (*binary*) data $\rightarrow$ $n + m$ qubits.

$\lvert a\rangle$	$\lvert \theta_a\rangle$
$\lvert 000\rangle$	$\lvert 0101\rangle$
$\lvert 001\rangle$	$\lvert 1100\rangle$
$\vdots$	$\vdots$
$\lvert 111\rangle$	$\lvert 0111\rangle$



$$\sum_a c_a |a\rangle|0\rangle \longmapsto \sum_a c_a |a\rangle|\theta_a\rangle$$

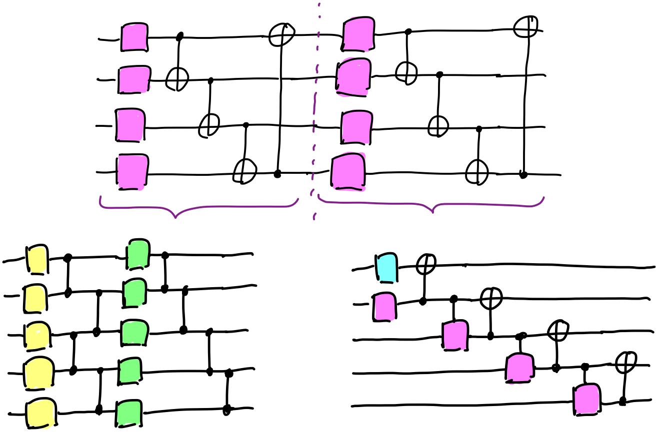
Upper bound the number of gates by $m \cdot 2^n$, which is *linear* in the amount of data (but they are all multi-controlled Toffolis which would need to be decomposed).

Overarching problem

1. How to set up a cost function: what to measure, and how to use it to determine classes
2. How to get the data into the circuit
3. What the trainable part of the circuit should look like

Architecture of parametrized circuits

Parametrized circuits (variational ansaetze) come in many forms.
Often have a *layered* structure, but depends on the problem.



Let's try a few different models for a VQC. How well can we do?

Recap

- Describe 3 different ways to embed data into a variational quantum classifier
- Classify real data with the VQC!

Next time

Content:

- Towards Shor's algorithm: the Quantum Fourier Transform

Action items:

1. Assignment 1 (can do all problems now)

Recommended reading for next time:

- Codebook nodes F.1-F.3