CPEN 400Q / EEСE 571Q Lecture 14
Parameter-shift rules, and the variational quantum classifier

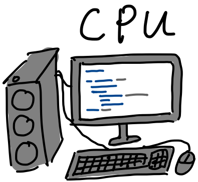
Thursday 3 March 2022

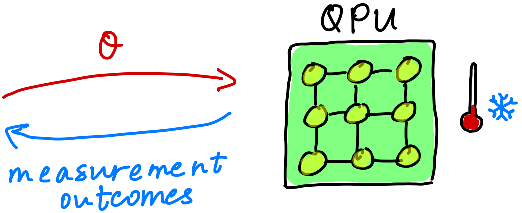
Announcements

- Schedule prototype project meeting for next week (15 minutes; informal; virtual or in-person)
- Scheduling final presentations: 9 groups for 8 slots (which day to do extra?)

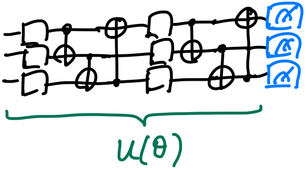
Last time

We started looking at variational algorithms.

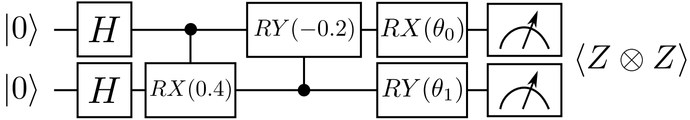




choose:
ansatz $U(\theta)$
initial θ
iterate:
run on qpu w/params θ
compute cost from results
choose new value of θ



We optimized some small parametrized quantum circuits.




```
@qml.qnode(dev)
def qnode(theta):
    # ...
    qml.RX(theta[0], wires=0)
    qml.RY(theta[1], wires=1)
    return qml.expval(qml.PauliZ(0) @ qml.PauliZ(1))

theta_opt = np.array([0.0, 0.0], requires_grad=True)

opt = qml.GradientDescentOptimizer(stepsize=0.1)

for _ in range(300):
    theta_opt = opt.step(qnode, theta_opt)
```

Last time

We saw how to compute gradients of parametrized quantum circuits using parameter-shift rules.

Parameter-shift rules tell us how to evaluate the gradient of a circuit by:

- running the *circuit itself* at different, shifted values
- combining the results

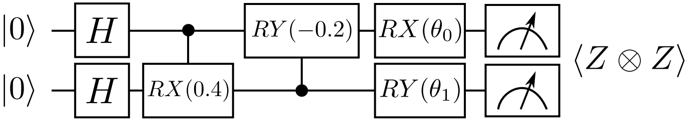
The "standard" two-term shift rule is:

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

where $f(\theta)$ is an expectation value obtained from running some circuit $U(\theta)$.

Last time

We showed in code that this works even for circuits with multiple parameters, e.g.,



$$\frac{\partial f(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{2}\left(f(\theta_0 + \pi/2,\ \ \theta_1) - f(\theta_0 - \pi/2,\ \ \theta_1)\right)$$
$$\frac{\partial f(\theta_0, \theta_1)}{\partial \theta_1} = \frac{1}{2}\left(f(\theta_0,\ \ \theta_1 + \pi/2) - f(\theta_0,\ \ \theta_1 - \pi/2)\right)$$

where $f(\theta_0, \theta_1)$ is the expectation value's analytical function.

- Derive the two-term parameter-shift rule to compute quantum gradients of single-qubit Pauli rotations
- Describe common embedding strategies for loading classical data into a quantum circuit
- Implement a variational quantum classifier

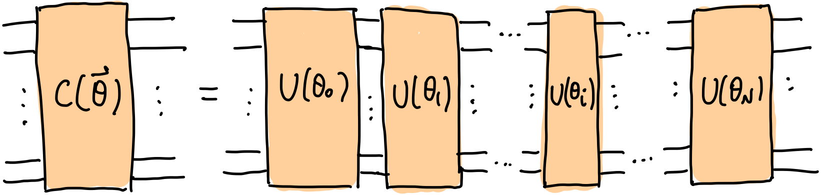
This kind of seems like magic... where does it come from?

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

We will derive this for the case of *single-qubit Pauli rotations* (which, if you recall, are universal for single-qubit operations).

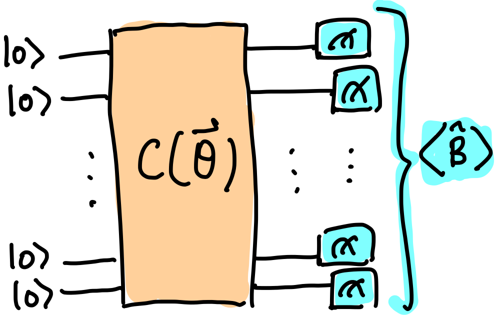
The derivation can also be done in a more general way (`https://arxiv.org/abs/1811.11184`).

Let $C(\theta) = C(\theta_0, \theta_1, \ldots, \theta_N)$ be a quantum circuit.



Suppose we want to compute the gradient with respect to θ_i.

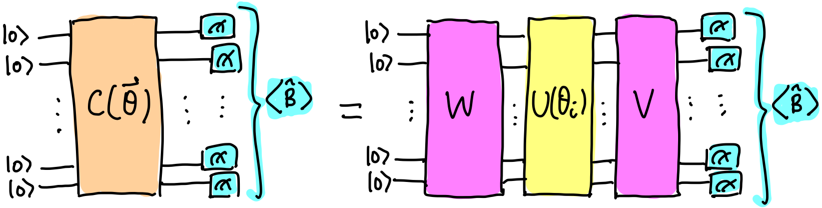
We differentiate the *expectation value* of some observable, $\hat{B}$, as a function of the circuit parameters.



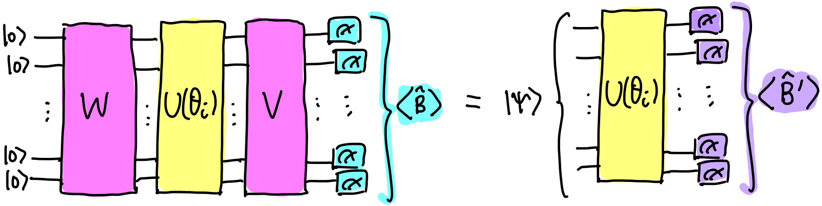
We want to compute $\frac{\partial f(\theta)}{\partial \theta_i}$.

The two-term parameter-shift rule

Let's isolate the parameter of interest:



Let's also tidy things up and group things that don't depend on θ_i:



where $|\psi\rangle = W|0\rangle$, and $\hat{B}' = V^\dagger \hat{B} V$ is still a Hermitian observable.

The two-term parameter-shift rule

Take the partial derivative of this function using the *chain rule*:

Note that the two terms are Hermitian conjugates, i.e., A general property of unitary operations is that they can be expressed in terms of a Hermitian generator, i.e.,

for G Hermitian, t some real-valued coefficient.

We know this is true for the Pauli rotations:

The two-term parameter-shift rule

Consider that our $U(\theta_i)$ is a Pauli rotation:

We can compute the derivative of this operation w.r.t. θ_i:

Let's put this back in our earlier equation:

Let's make one more substitution for now, $|\psi'\rangle = U(\theta_i)|\psi\rangle$:

Now we make use of the following identity: for any two operators P, Q,

(try proving it yourself!).

We have the expression

Set

The two-term parameter-shift rule

$$\langle\psi| P^\dagger \hat{M} Q|\psi\rangle + \langle\psi| Q^\dagger \hat{M} P|\psi\rangle = \frac{1}{2}[\langle\psi| (P+Q)^\dagger \hat{M}(P+Q)|\psi\rangle \\ - \langle\psi| (P-Q)^\dagger \hat{M}(P-Q)|\psi\rangle]$$

Setting $P = \frac{1}{\sqrt{2}}$, $Q = -i\frac{G}{\sqrt{2}}$ we get

The two-term parameter-shift rule

Recall that $U(\theta) = e^{-i\theta \frac{G}{2}}$ for G a Pauli. Evaluate this at $\theta = \frac{\pi}{2}$:

Can similarly show that .

So from

$$\frac{\partial f(\theta)}{\partial \theta_i} = \frac{1}{2}[\langle \psi' | \left(\frac{I}{\sqrt{2}} - i\frac{G}{\sqrt{2}} \right)^\dagger \hat{B}' \left(\frac{I}{\sqrt{2}} - i\frac{G}{\sqrt{2}} \right) |\psi'\rangle - \langle \psi' | \left(\frac{I}{\sqrt{2}} + i\frac{G}{\sqrt{2}} \right)^\dagger \hat{B}' \left(\frac{I}{\sqrt{2}} + i\frac{G}{\sqrt{2}} \right) |\psi'\rangle],$$

we obtain

The two-term parameter-shift rule

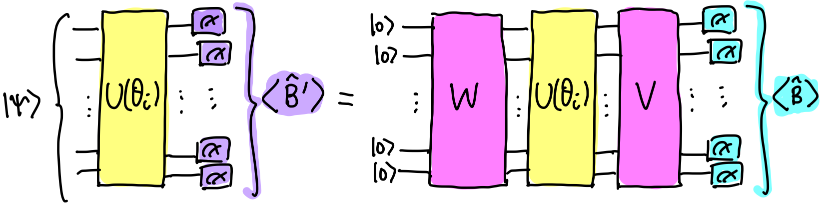
Earlier, we defined $|\psi'\rangle = U(\theta_i)|\psi\rangle$. So, we can rewrite

$$\frac{\partial f(\theta)}{\partial \theta_i} = \frac{1}{2}[\langle \psi'| \, U\left(\frac{\pi}{2}\right)^\dagger \hat{B}' U\left(\frac{\pi}{2}\right) |\psi'\rangle \\ - \langle \psi'| \, U\left(-\frac{\pi}{2}\right)^\dagger \hat{B}' U\left(-\frac{\pi}{2}\right) |\psi'\rangle],$$

as

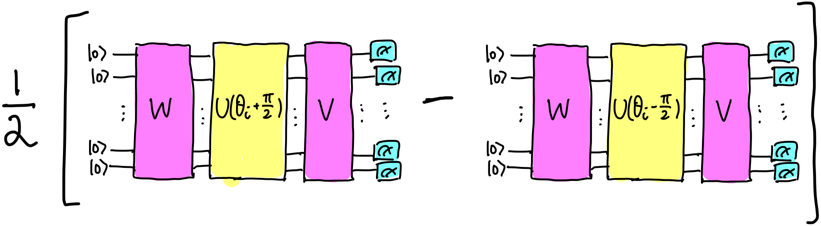
Now we can merge these as they are Pauli rotations...

But recall we had made some other substitutions...



The two-term parameter-shift rule

So we've recovered the parameter-shift rule!



Since you asked: generalized parameter-shift rules

Not all operations admit the simple two-term shift rule. For example, $CRX(\theta), CRY(\theta), CRZ(\theta)$ have a four-term rule.

If $CRX(\theta)$ is in a circuit with output function $f(\theta)$,

$$\frac{\partial f(\theta)}{\partial \theta} = c_+[f(\theta+\pi/2)-f(\theta-\pi/2)]-c_-[f(\theta+3\pi/2)-f(\theta-3\pi/2)]$$

where

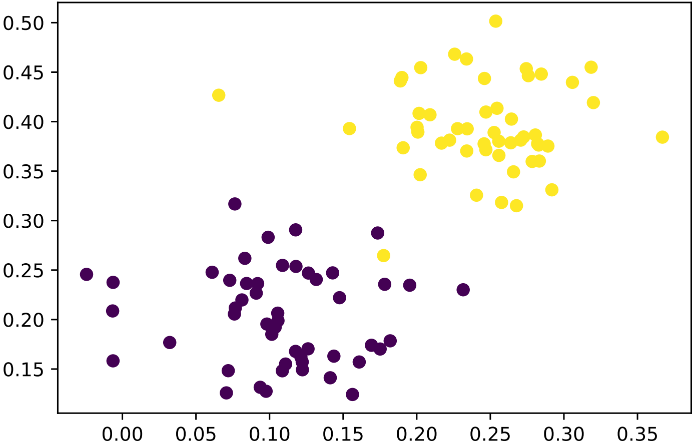
$$c_\pm = \frac{\sqrt{2}\pm 1}{4\sqrt{2}}$$

For more info: `https://arxiv.org/abs/2104.05695`, `https://arxiv.org/abs/2107.12390`

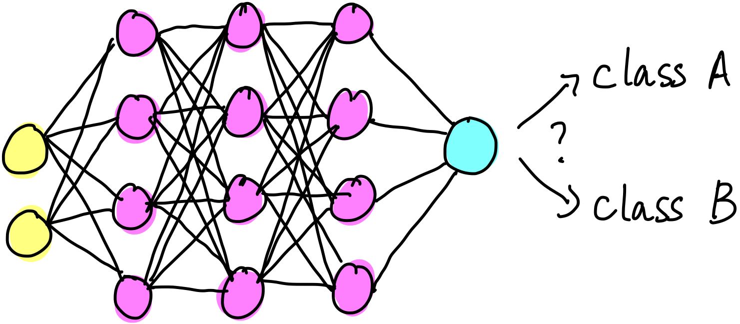
Variational quantum classifier

Overarching problem: binary classification

Suppose we have some 2-dimensional data:

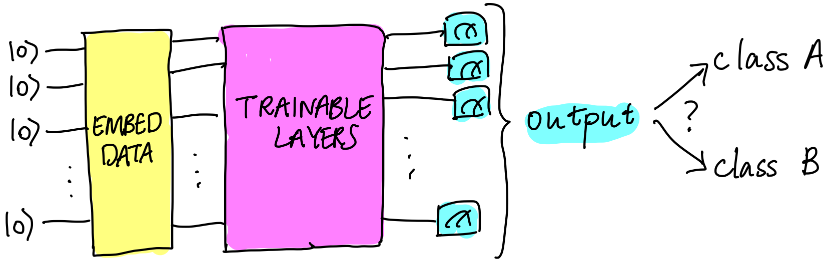


Consider how classification can be done with a neural network:



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:

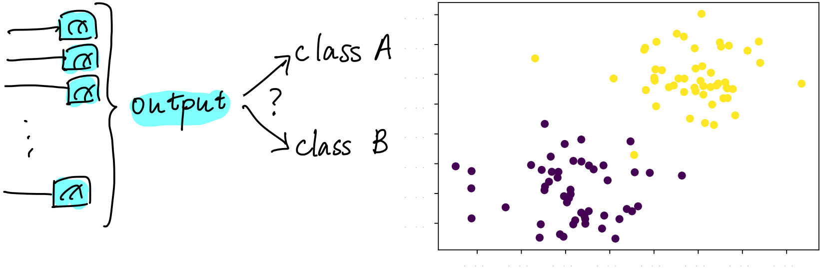
1. How to set up a cost function: what to measure, and how to use it to determine classes
2. How to get the data into the circuit
3. What the trainable part of the circuit should look like

(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

Measurements and cost functions

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



Measurements and cost functions

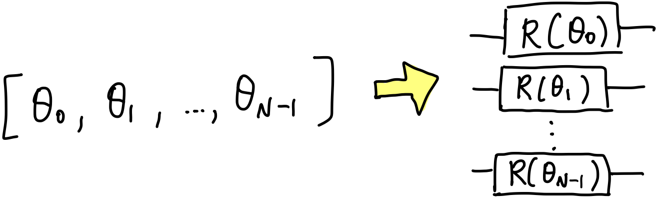
Use a simple least-squares fit: minimize the difference between the computed expectation values and the true values:

```python
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

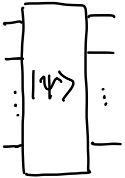
1. How to set up a cost function: what to measure, and how to use it to determine classes
2. How to get the data into the circuit
3. What the trainable part of the circuit should look like

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



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

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



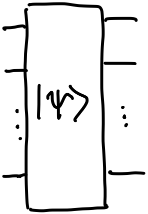
$$|\psi\rangle \approx \theta_0 |0\rangle + \theta_1 |1\rangle + \cdots + \theta_{N-1} |N-1\rangle$$

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

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

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

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



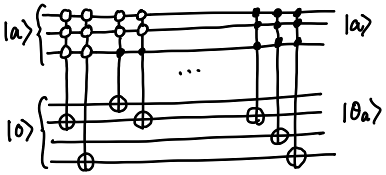
$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\theta_i\rangle$$

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

QROM/QRAM

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

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



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

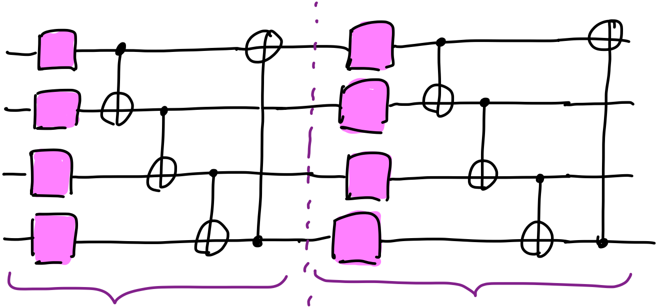
Upper bound the number of gates by $m \cdot 2^n$, which is *linear* in the amount of data (but they are all multi-controlled Toffolis which would need to be decomposed).

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

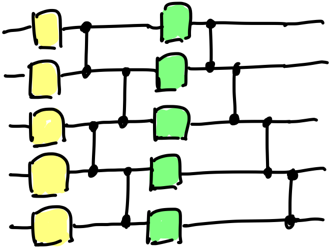
Parametrized circuits (often called variational ansaetze in this context) come in many shapes and sizes. Often they have a *layered* structure, but this can depend on the problem at hand.



Layers typically alternate between single-qubit operations, and sequences of entangling gates.

Architecture of parametrized circuits

Option: "hardware-efficient" ansatz



Pros:

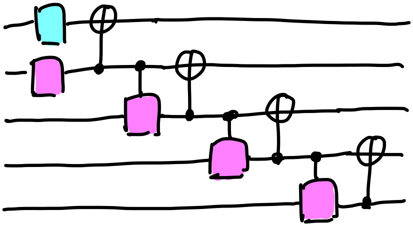
- simple, flexible structure
- can be adapted to fit perfectly on hardware
- expressive (good coverage of the Hilbert space)

Cons:

- does not take advantage of any problem structure

Architecture of parametrized circuits

Option: problem-specific, or physically-motivated ansatz



Pros:

- takes advantage of problem structure: better use of available resources, can consider symmetries, etc.

Cons:

- may not fit the hardware architecture
- requires information about the solution
- not very expressive

Barren plateaus are areas in the cost landscape where:

- The *gradient* approaches 0
- The *variance of the gradient* approaches 0

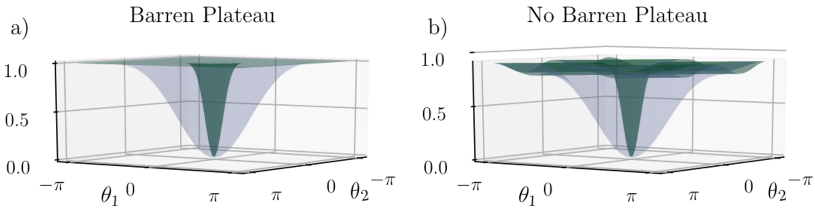


Image credit: A. Arrasmith, Z. Holmes, M. Cerezo, and P. J. Coles. *Equivalence of quantum barren plateaus to cost concentration and narrow gorges*. arXiv 2104.05868 [quant-ph]

Expressibility and barren plateaus

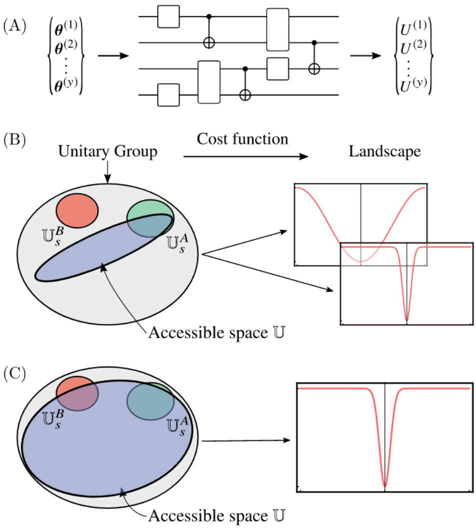
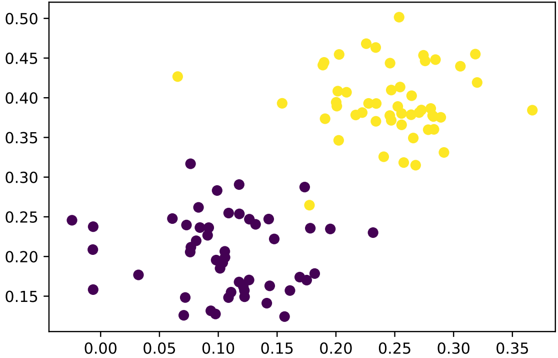


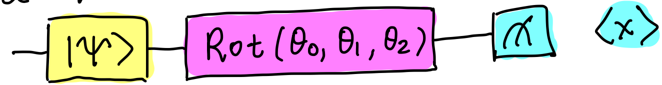
Image credit: Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles. *Connecting ansatz expressibility to gradient magnitudes and barren plateaus.* arXiv 2101.02138 [quant-ph]

Back to the overarching problem

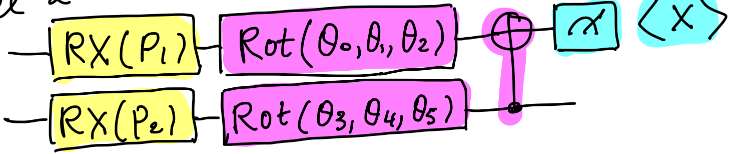


Let's try a few different models for a VQC. How well can we do?

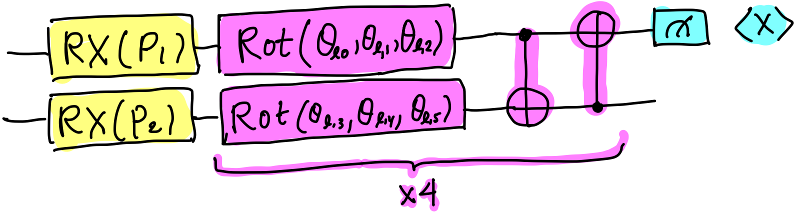
Model 1:



Model 2:



Model 3:



Next time

Content:

- Starting with Hamiltonians and the variational quantum eigensolver

Action items:

1. Prototype implementation for project
2. Assignment 3

Recommended reading:

- QML glossary entries (`https://pennylane.ai/qml/glossary.html`)
  - Quantum embedding
  - Quantum feature map
- Schuld & Petruccione, *Supervised learning with quantum computers* (chapter 4 about data embeddings)
- Variational classifier demo `https://pennylane.ai/qml/demos/tutorial_variational_classifier.html`