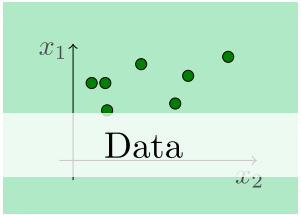
Quantum Machine Learning

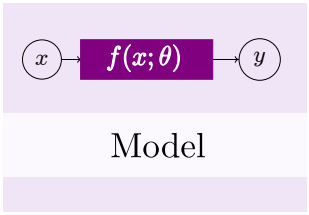
Maria Schuld, Xanadu

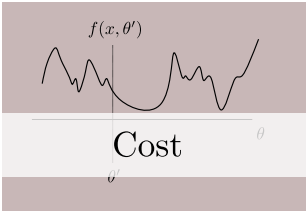
Hackaday Meetup, October 2020

XANADU

Machine Learning

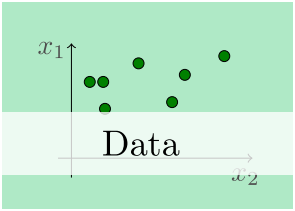


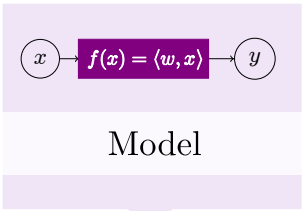


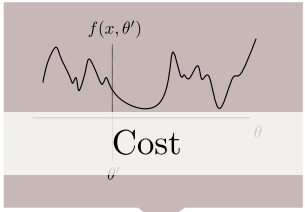


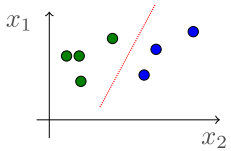
Use data samples
to construct model
that minimises cost
on unseen data.

Linear models

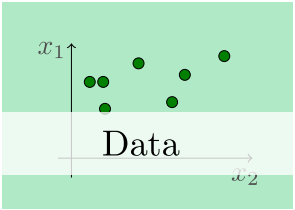


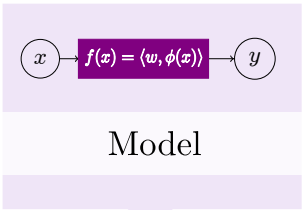


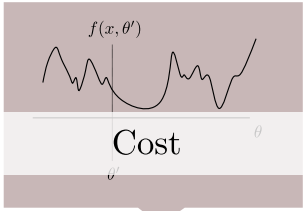


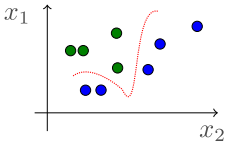


Kernel methods

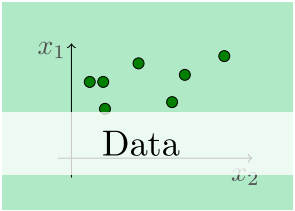


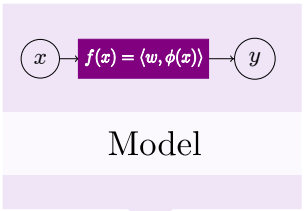


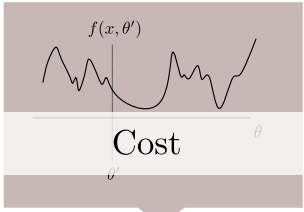


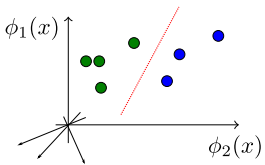


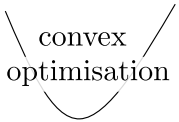
Kernel methods



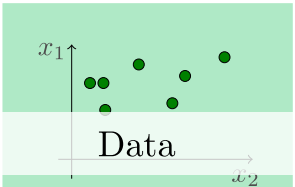


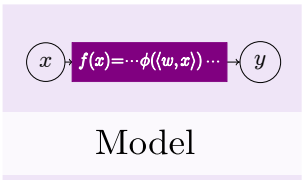


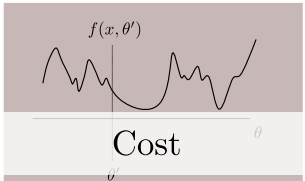




Deep learning

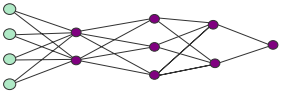






Big

Composable & differentiable



nonconvex optimisation

- gradient descent
- high performance hardware
- special purpose software

How could quantum computing help?

- Data
  - use fewer data samples (Arunachalam 1701.06806)
  - process “quantum data“ (folklore)

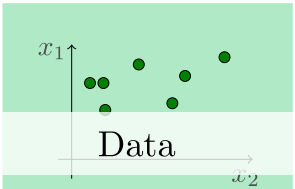
How could quantum computing help?

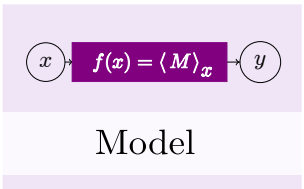
- Data
  - use fewer data samples (Arunachalam 1701.06806)
  - process “quantum data“ (folklore)
- Optimisation
  - speed up optimisation (Wiebe et al. 1204.5242, Rebentrost et al. 1307.0471, Denil & Freitas ~ 2012 `cs.ubc.ca/~nando/`)
  - find better solutions (?)

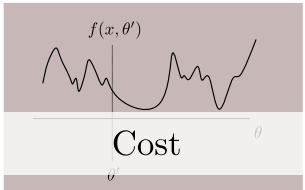
How could quantum computing help?

- Data
  - use fewer data samples (Arunachalam 1701.06806)
  - process “quantum data“ (folklore)
- Optimisation
  - speed up optimisation (Wiebe et al. 1204.5242, Rebentrost et al. 1307.0471, Denil & Freitas ~ 2012 cs.ubc.ca/~nando/)
  - find better solutions (?)
- Model
  - speed up existing models (Pararo et al. 1401.4997, Low et al. 1402.7359, Allcock et al. 1812.03089)
  - design new models (Amin et al. 1601.02036, Benedetti et al. 1906.07682)

We can train quantum circuits like neural nets.

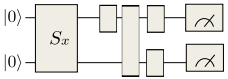






Big

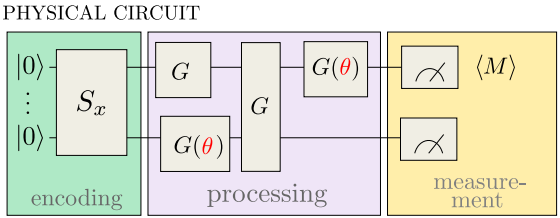
Composable & differentiable



nonconvex optimisation

- gradient descent
- high performance hardware
- special purpose software

Variational circuits as composable & differentiable models.

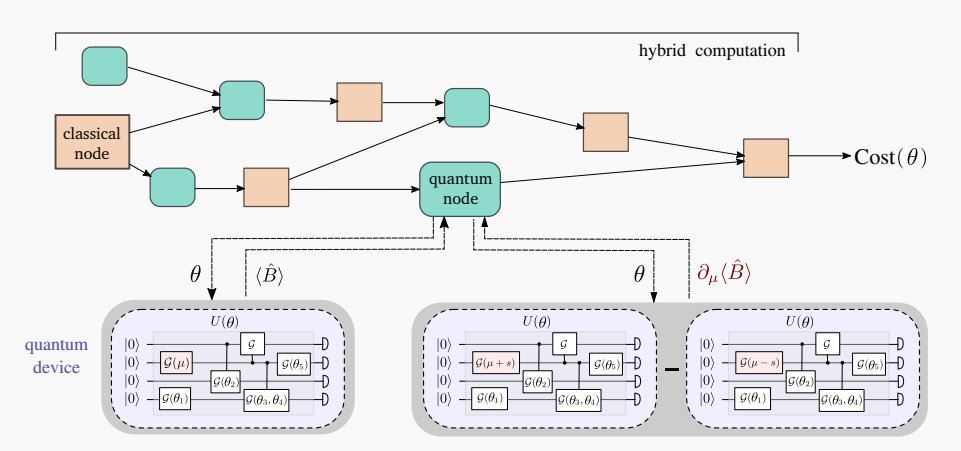


MATHEMATICAL DESCRIPTION

$$10 \quad s^*_{ji}(x) \quad g^*_{ji} \quad \cdots \quad g^*_{ji}(\theta) \quad M \quad g_{ji}(\theta) \quad \cdots \quad g_{ji} \quad s_{ij}(x) \quad \begin{matrix}1\\0\end{matrix}$$

Farhi & Neven 1802.06002, Schuld et al. 1804.00633

Variational circuits as composable & differentiable models.



Guerreschi & Smelyanskiy 1701.01450, Mitarai et al. 1803.00745, Schuld et al. 1811.11184

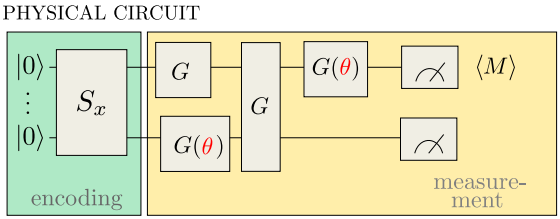
Variational circuits as composable & differentiable models.

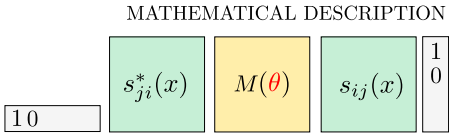
```python
import torch
from torch.autograd import Variable

data = torch.tensor([(0., 0.), (0.1, 0.1), (0.2, 0.2)])



def model(phi, x=None):
    return x*phi



def loss(a, b):
    return torch.abs(a - b) ** 2

def av_loss(phi):
    c = 0
    for x, y in data:
        c += loss(model(phi, x=x), y)
    return c

phi_ = Variable(torch.tensor(0.1), requires_grad=True)
opt = torch.optim.Adam([phi_], lr=0.02)

for i in range(5):
    l = av_loss(phi_)
    l.backward()
    opt.step()
```

```python
from pennylane import *
import torch
from torch.autograd import Variable

data = [(0., 0.), (0.1, 0.1), (0.2, 0.2)]

dev = device('default.qubit', wires=2)

@qnode(dev, interface='torch')
def circuit(phi, x=None):
    templates.AngleEmbedding(features=[x], wires=[0])
    templates.BasicEntanglerLayers(weights=phi, wires=[0, 1])
    return expval(PauliZ(wires=[1]))

def loss(a, b):
    return torch.abs(a - b) ** 2

def av_loss(phi):
    c = 0
    for x, y in data:
        c += loss(circuit(phi, x=x), y)
    return c

phi_ = Variable(torch.tensor([[0.1, 0.2],[-0.5, 0.1]]), requires_grad=True)
opt = torch.optim.Adam([phi_], lr=0.02)

for i in range(5):
    l = av_loss(phi_)
    l.backward()
    opt.step()
```

pennylane.ai

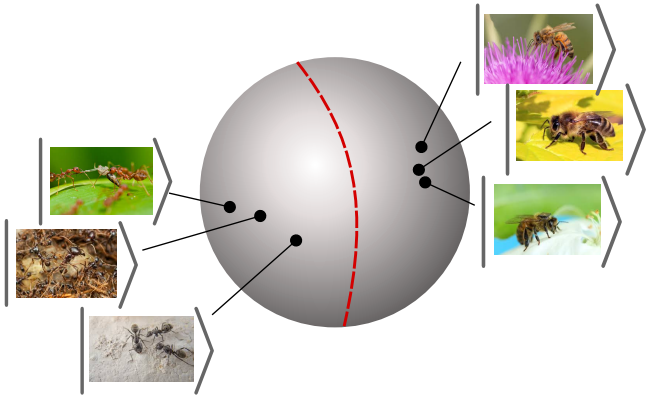
Quantum models are similar to kernel methods.





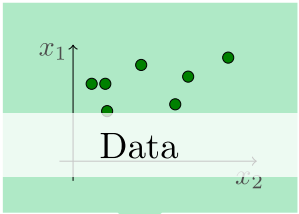
Schuld & Killoran 1803.07128, Havlicek et al. 1804.11326, Lloyd et al. 2001.03622

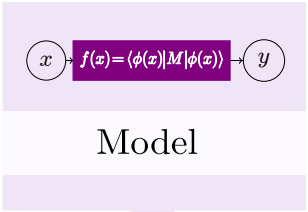
Quantum models are similar to kernel methods.

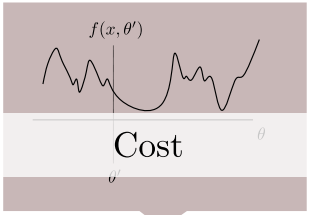


Lloyd et al. 2001.03622

Quantum models are similar to kernel methods.

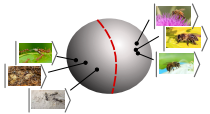






Big

Composable & differentiable



nonconvex optimisation

OR

convex optimisation

Thank you!

www.pennylane.ai
www.xanadu.ai
@XanaduAI