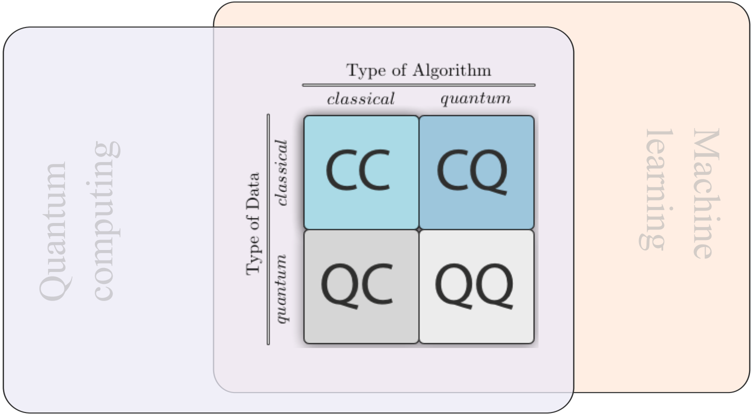
# Machine Learning with Quantum Computers

Maria Schuld

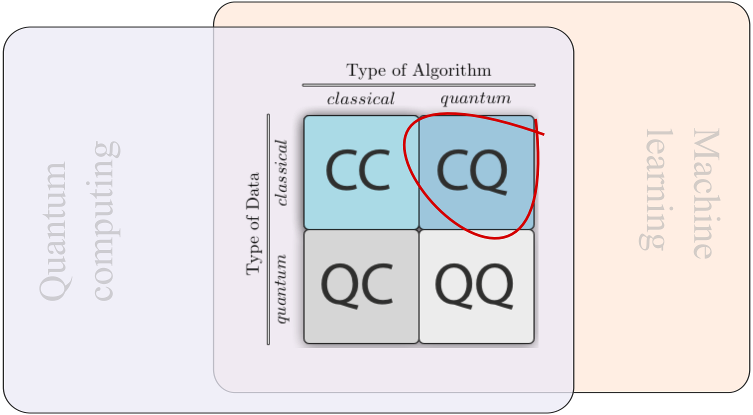
Xanadu and University of KwaZulu-Natal

DESY Workshop, August 2020

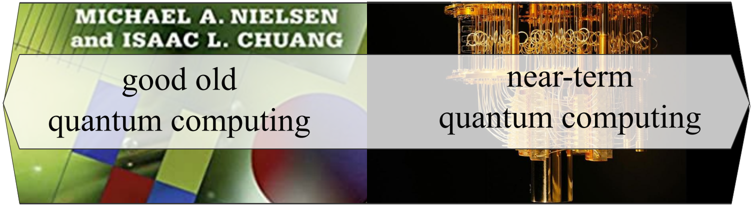
# The intersection of quantum computing and ML is rich.



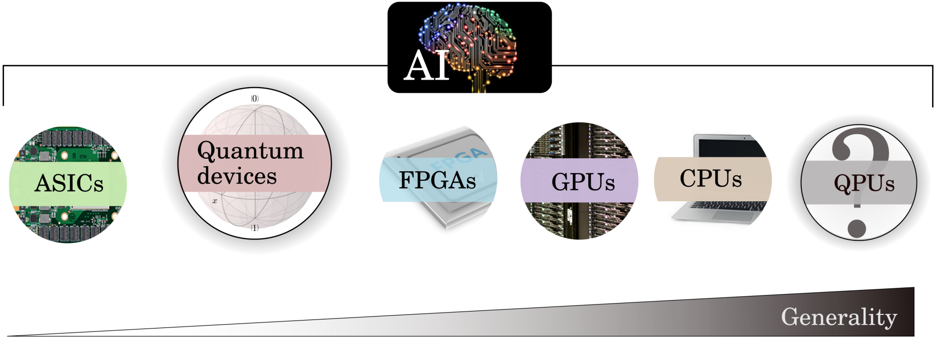
# The intersection of quantum computing and ML is rich.



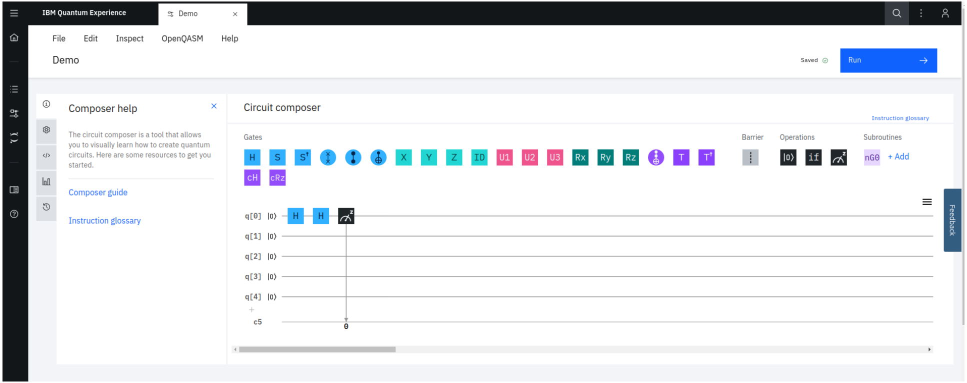
# Approaches range from traditional to near-term QC.



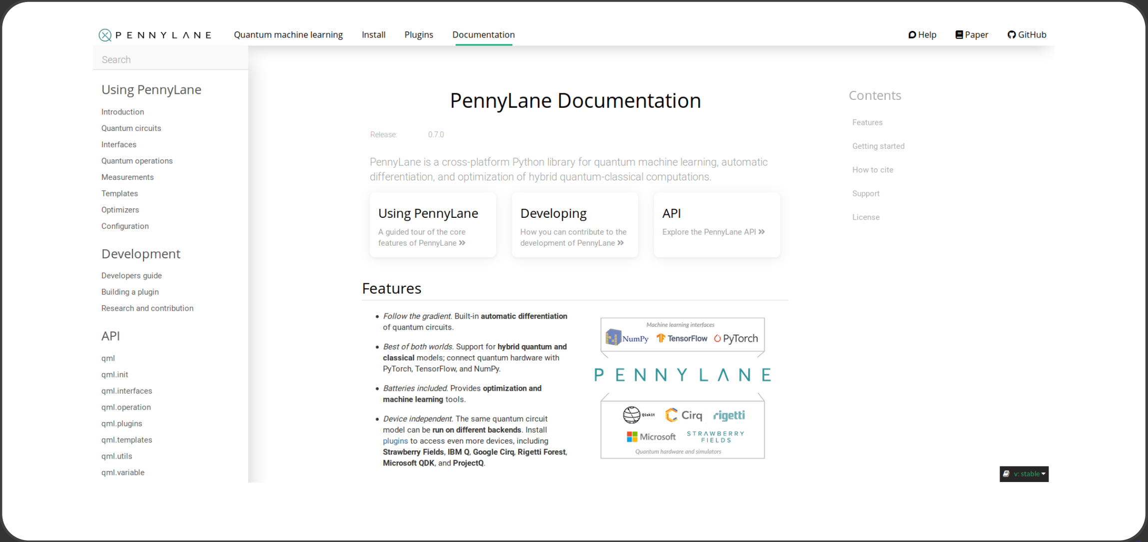
# Approaches range from traditional to near-term QC.



# Software plays a central role in near-term approaches.



# Software plays a central role in near-term approaches.


PENNYLANE | Quantum machine learning | Install | Plugins | Documentation | Help | Paper | GitHub

Search

## Using PennyLane

- Introduction
- Quantum circuits
- Interfaces
- Quantum operations
- Measurements
- Templates
- Optimizers
- Configuration

## Development

- Developers guide
- Building a plugin
- Research and contribution

## API

- qml
- qml.init
- qml.interfaces
- qml.operation
- qml.plugins
- qml.templates
- qml.utils
- qml.variable

# PennyLane Documentation

Release: 0.7.0

PennyLane is a cross-platform Python library for quantum machine learning, automatic differentiation, and optimization of hybrid quantum-classical computations.

**Using PennyLane**
A guided tour of the core features of PennyLane »

**Developing**
How you can contribute to the development of PennyLane »

**API**
Explore the PennyLane API »

## Features

- *Follow the gradient.* Built-in **automatic differentiation** of quantum circuits.
- *Best of both worlds.* Support for **hybrid quantum and classical** models; connect quantum hardware with PyTorch, TensorFlow, and NumPy.
- *Batteries included.* Provides **optimization and machine learning** tools.
- *Device independent.* The same quantum circuit model can be **run on different backends**. Install plugins to access even more devices, including **Strawberry Fields**, **IBM Q**, **Google Cirq**, **Rigetti Forest**, **Microsoft QDK**, and **ProjectQ**.



Contents

- Features
- Getting started
- How to cite
- Support
- License

v: stable

# How can quantum computers innovate ML?

machine intelligence = data/distributions + models + algorithm/hardware

# Agenda

- QML and HEP
- Variational quantum circuits

# QML and HEP

# A flavour of current work

FERMILAB-PUB-20-184-QIS

**Quantum Machine Learning in High Energy Physics**

**Wen Guan, Gabriel Perdue, Arthur Pesah, Maria Schuld, Koji Terashi, Sofia Vallecorsa, Jean-Roch Vlimant**

E-mail: `jvlimant@caltech.edu`

May 2020

**Abstract.** Machine learning has been used in high energy physics since a long time, primarily at the analysis level with supervised classification. Quantum computing was postulated in the early 1980s as way to perform computations that would not be tractable with a classical computer. With the advent of noisy intermediate-scale quantum computing devices, more quantum algorithms are being developed with the aim at exploiting the capacity of the hardware for machine learning applications. An interesting question is whether there are ways to combine quantum machine learning with High Energy Physics. This paper reviews the first generation of ideas that use quantum machine learning on problems in high energy physics and provide an outlook on future applications.



Guan, Perdue, Pesah, Schuld, Terashi, Vallecorsa, Vlimant,
*Quantum Machine Learning in High Energy Physics*, arxiv:2005.08582

# A flavour of current work

**Task:** *Distinguish pair of photons created by Higgs decay from uncorrelated background events*
**Features:** *8 measurements taken on the di-photon system*
**Quantum technology:** *Quantum annealer (hardware)*
**Quantum algorithm:** *Use QUBO to find best (0/1) weights to combine 36 simple ML models ("weak learners")*

Mott, Job, Vlimant, Lidar, & Spiropulu (2017), Nature, 550(7676), 375-379

# A flavour of current work

**Task:** *Particle track reconstruction*
**Features:** *Locations of hits + corresponding particles (TrackML challenge)*
**Quantum technology:** *Qubit-based quantum circuits (simulator)*
**Quantum algorithm:** *Represent hits as "tree-tensor network" quantum circuit and train gates in the network*

Tysz, Carminati, Demirkz, Dobos, Fracas, Novotny, ... & Vlimant (2020), arXiv:2003.08126

# A flavour of current work

**Task:** *Higgs coupling to top quark pairs (ttH)*
**Features:** *45 input events (+ PCA)*
**Quantum technology:** *Qubit-based quantum circuits (simulator + hardware)*
**Quantum algorithm:** *Variational circuit (SVM interpretation)*

Chan, Guan, Sun, Wang, Wu, Zhou, ... & Di Meglio (2019), PoS, LeptonPhoton2019, 49

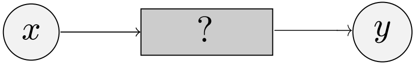
# A flavour of current work

**Task:** *Classification of signal predicted in Supersymmetry*
**Features:** *SUSY data set in the UC Irvine Machine Learning Repositiory*
**Quantum technology:** *Qubit-based quantum circuits (simulator + hardware)*
**Quantum algorithm:** *Variational circuit (NN interpretation)*

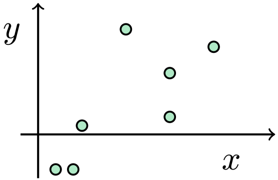
Terashi, Kaneda, Kishimoto, Saito, Sawada, & Tanaka (2020), arXiv:2002.09935

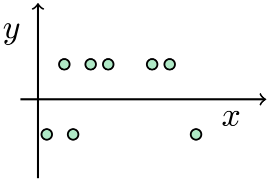
# Variational quantum circuits

# The first ingredient of ML is data.




$$\boxed{(x, y) \in \mathcal{X} \times \mathcal{Y}}$$

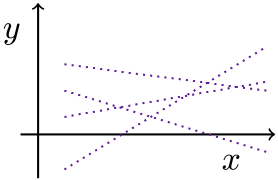


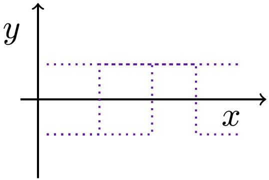



# The second ingredient of ML is a model family.

$$x \longrightarrow \boxed{f(x;\theta)} \longrightarrow y$$

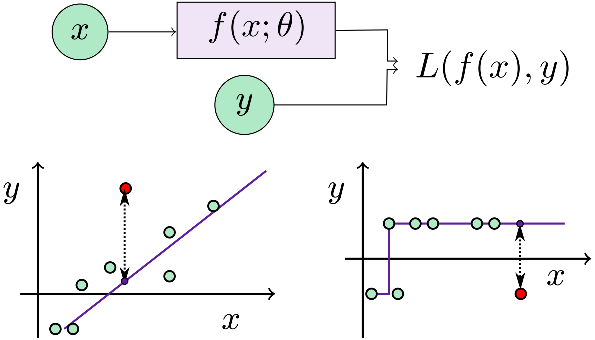
$$\boxed{f(x) \in \{\mathcal{F}\}}$$



# The third ingredient of ML is a loss.

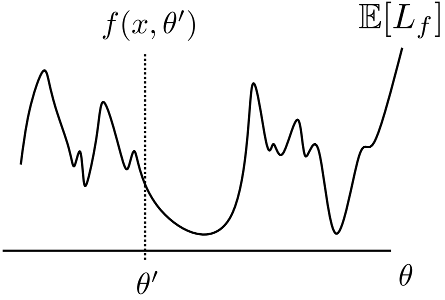
$$\boxed{L(f(x), y) \in \mathbb{R}}$$



# The goal of ML is to minimise the "average" loss of the model.

$$\mathbb{E}[L_f] = \int L(f(x), y)\, p(x, y)\, dxdy$$

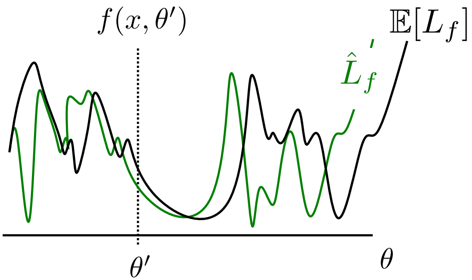
$$\boxed{f^* = \min_{f \in \{\mathcal{F}\}} \mathbb{E}[L_f]}$$



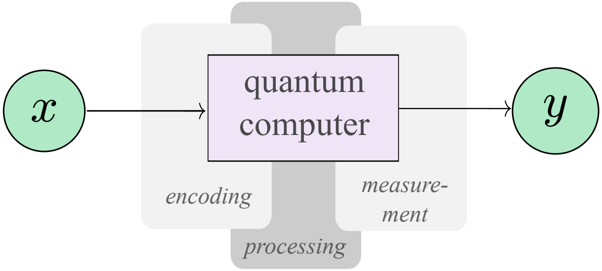
# We can only minimise the average loss on training data.

$$\hat{L}_f = \sum_{(x,y)\in\mathcal{D}} L(f(x), y)$$

$$\boxed{f^* = \min_{f\in\{\mathcal{F}\}} \hat{L}_f}$$



# Quantum circuits can be used as machine learning models.



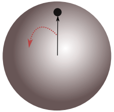

Farhi & Neven 1802.06002, Schuld et al. 1804.00633, Benedetti et al. 1906.07682

# Quantum circuits can be used as machine learning 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()
```

# The mathematics of quantum computers.



PHYSICAL CIRCUIT

$$n \left[ \begin{array}{c} |0\rangle \\ \vdots \\ |0\rangle \end{array} \right.$$

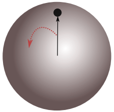
MATHEMATICAL DESCRIPTION

$$|1|^2 = p(0\ldots00)$$

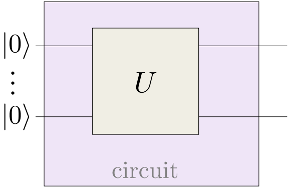
$$2^n \left[ \boxed{\begin{array}{c} 1+0i \\ 0+0i \\ \vdots \end{array}} \right.$$

$$|0|^2 = p(0\ldots01)$$

# The mathematics of quantum computers.



PHYSICAL CIRCUIT



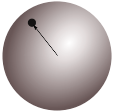

MATHEMATICAL DESCRIPTION

$$|1|^2 = p(0...00)$$

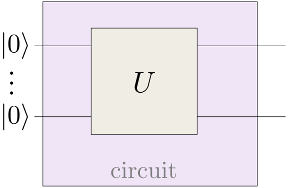
$$u_{ij} \quad \begin{array}{c} 1+0i \\ 0+0i \\ \vdots \end{array}$$

$$|0|^2 = p(0...01)$$

# The mathematics of quantum computers.



PHYSICAL CIRCUIT



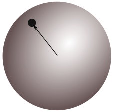

MATHEMATICAL DESCRIPTION

$$|\psi_1|^2 = p(0...00)$$

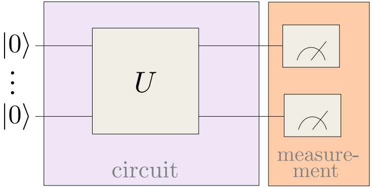
$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \end{pmatrix}$$

$$|\psi_2|^2 = p(0...01)$$

# The mathematics of quantum computers.



PHYSICAL CIRCUIT



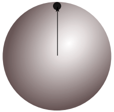

MATHEMATICAL DESCRIPTION

$$\begin{matrix} 10110\ldots1 \\ 11010\ldots0 \\ 00001\ldots0 \\ \vdots \end{matrix} \sim \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \end{bmatrix}$$

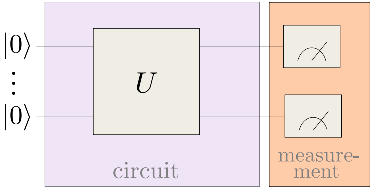
$|\psi_1|^2 = p(0\ldots00)$

$|\psi_2|^2 = p(0\ldots01)$

# The mathematics of quantum computers.



PHYSICAL CIRCUIT




MATHEMATICAL DESCRIPTION

$|\psi_1|^2 = p(0\ldots00)$

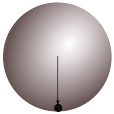
10110...1
11010...0
00001...0
$\vdots$

$\sim$

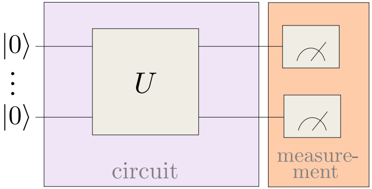
$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \end{bmatrix}$$

$|\psi_2|^2 = p(0\ldots01)$

# The mathematics of quantum computers.



PHYSICAL CIRCUIT




MATHEMATICAL DESCRIPTION

10110...1
11010...0
00001...0
$\vdots$

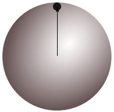
$\sim$

$|\psi_1|^2 = p(0...00)$

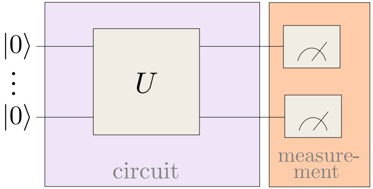
$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \end{bmatrix}$$

$|\psi_2|^2 = p(0...01)$

# The mathematics of quantum computers.



PHYSICAL CIRCUIT




MATHEMATICAL DESCRIPTION

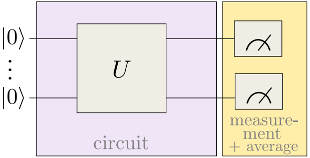
$$|\psi_1|^2 = p(0...00)$$

$$\begin{matrix} 10110...1 \\ 11010...0 \\ 00001...0 \\ \vdots \end{matrix} \sim \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \end{bmatrix}$$

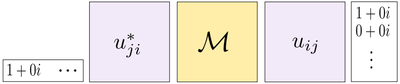
$$|\psi_2|^2 = p(0...01)$$

# The mathematics of quantum computers.

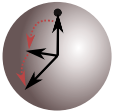
PHYSICAL CIRCUIT



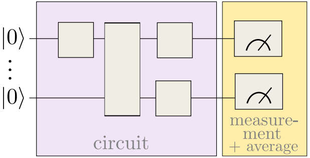

MATHEMATICAL DESCRIPTION



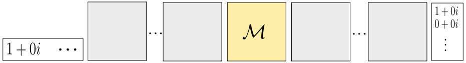
# The mathematics of quantum computers.



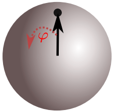
PHYSICAL CIRCUIT



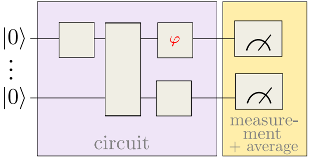

MATHEMATICAL DESCRIPTION



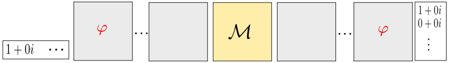
# 1. QCs perform trainable, modular linear operations.



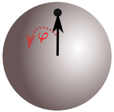

PHYSICAL CIRCUIT



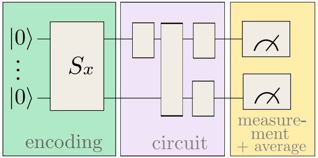

MATHEMATICAL DESCRIPTION



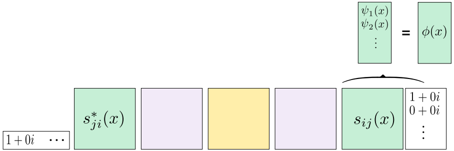
# 2. QCs map data to high-dimensional qstates.



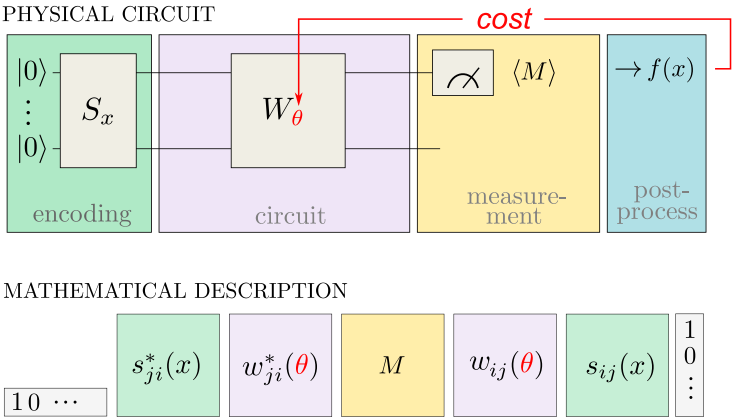

PHYSICAL CIRCUIT




MATHEMATICAL DESCRIPTION

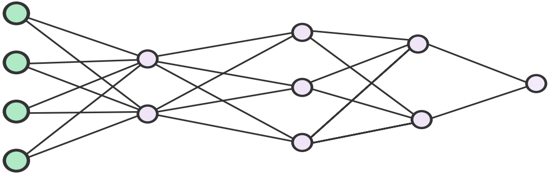


# 3. We can train QCs.

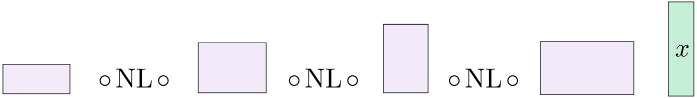


# Quantum models are linear neural nets in feature space.

MODEL

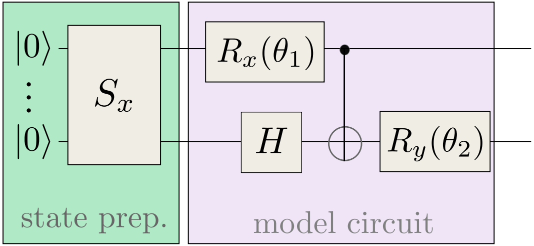


MATHEMATICAL DESCRIPTION



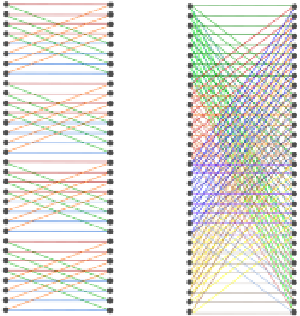
# Quantum models are linear neural nets in feature space.

PHYSICAL CIRCUIT



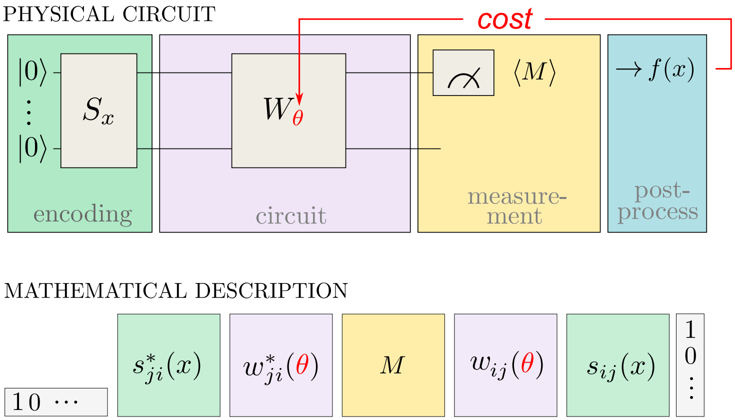

MATHEMATICAL DESCRIPTION

# Quantum models are linear neural nets in feature space.



Schuld & Petruccione, Springer 2018

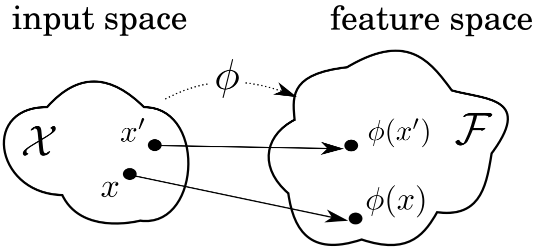
# Quantum models are natural kernel methods.



# Quantum models are natural kernel methods.

$$x \to |x\rangle = \phi(x)$$

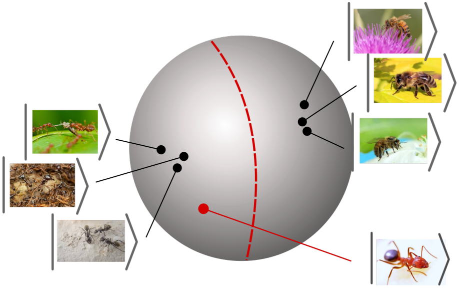
# Quantum models are natural kernel methods.




$$\kappa(x, x') = \langle \phi(x), \phi(x') \rangle$$

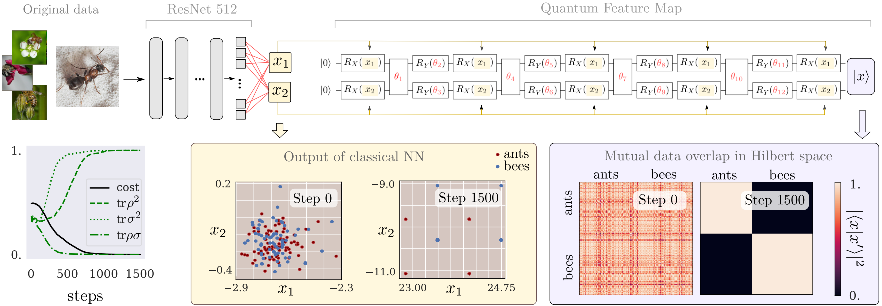
$$f(x) = \langle \phi(x), w \rangle$$

# Quantum models are natural kernel methods.



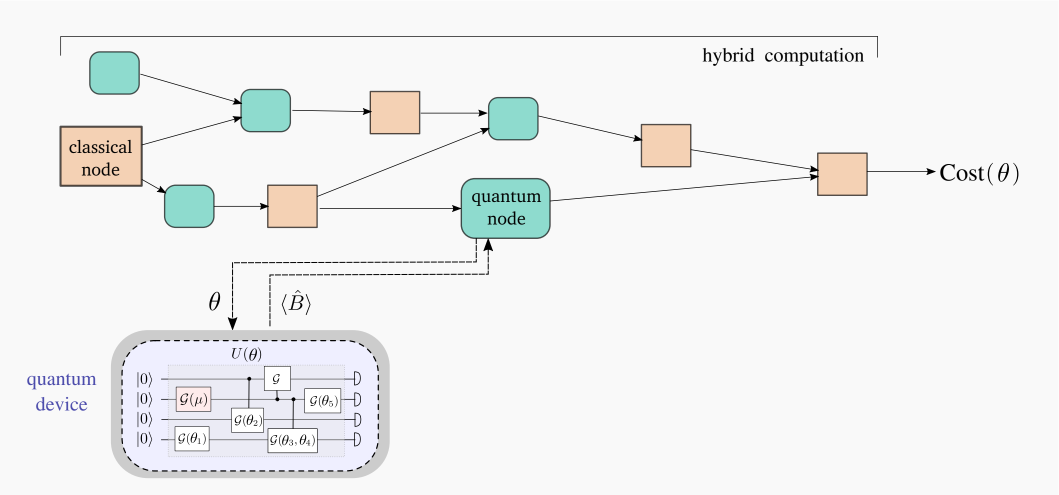
Lloyd et al. 2001.03622

# Quantum models are natural kernel methods.



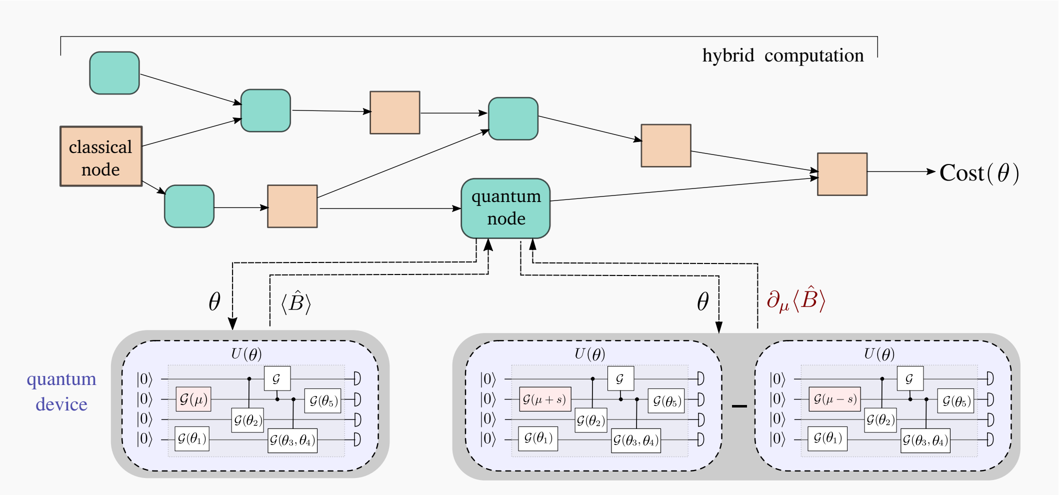

Lloyd et al. 2001.03622

# We can compute gradients.



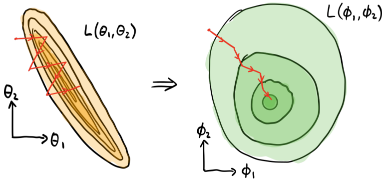

Guerreschi & Smelyanskiy 1701.01450, Mitarai et al. 1803.00745, Schuld et al. 1811.11184

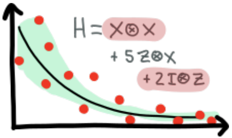
# We can compute gradients.




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

# We can compute gradients.




Stokes et al. 1909.02108, Kübler et al. 1909.09083, Sweke et al. 1910.01155, Ostaszewski et al. 1905.09692, ...

# We can compute gradients.

**Barren plateaus in quantum neural network training landscapes**

Jarrod R. McClean,[1,*] Sergio Boixo,[1,†] Vadim N. Smelyanskiy,[1,‡] Ryan Babbush,[1] and Hartmut Neven[1]
[1] *Google Inc., 340 Main Street, Venice, CA 90291, USA*
(Dated: March 30, 2018)

Many experimental proposals for noisy intermediate scale quantum devices involve training a parameterized quantum circuit with a classical optimization loop. Such hybrid quantum-classical algorithms are popular for applications in quantum simulation, optimization, and machine learning. Due to its simplicity and hardware efficiency, random circuits are often proposed as initial guesses for exploring the space of quantum states. We show that the exponential dimension of Hilbert space and the gradient estimation complexity make this choice unsuitable for hybrid quantum-classical algorithms run on more than a few qubits. Specifically, we show that for a wide class of reasonable parameterized quantum circuits, the probability that the gradient along any reasonable direction is non-zero to some fixed precision is exponentially small as a function of the number of qubits. We argue that this is related to the 2-design characteristic of random circuits, and that solutions to this problem must be studied.



Rapid developments in quantum hardware have motivated advances in algorithms to run in the so-called noisy intermediate scale quantum (NISQ) regime [1]. Many of the most promising application-oriented approaches are hybrid quantum-classical algorithms that rely on optimization of a parameterized quantum circuit [2–8]. The resilience of these approaches to certain types of errors and high flexibility with respect to coherence time and gate requirements make them especially attractive for NISQ implementations [3, 9–11].

The first implementation of such algorithms was de-

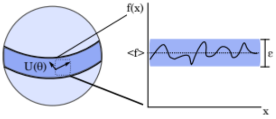
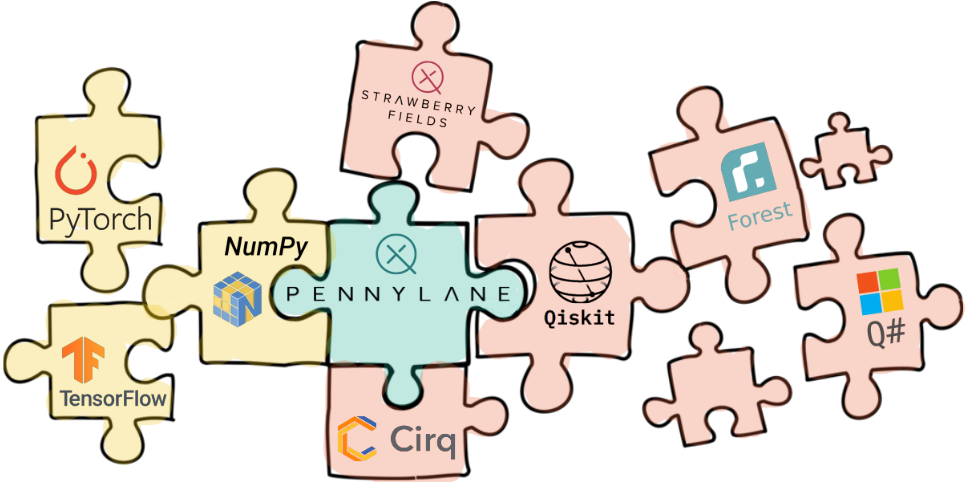



FIG. 1. A cartoon of the general geometric results from this work. The sphere depicts the phenomenon of concentration of

McClean et al. 1803.11173

# We can compute gradients.



# Open questions...

- What models are quantum circuits?
- Are they actually useful?
- Will they perform well on larger problem instances?
- Will they perform well under noise?
- What problems are they good for?
- Is there a problem where they are exponentially better?
- How should I design a quantum model?

# ...and some advice.

- Don't compare quantum models blindly to classical ML.
- Understand the features and models you use.
- Understand what feature map your model performs.
- Think of cutting out the intermediate measurements.
- Try continuous-variable quantum circuits for HEP?

# Thank you!

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