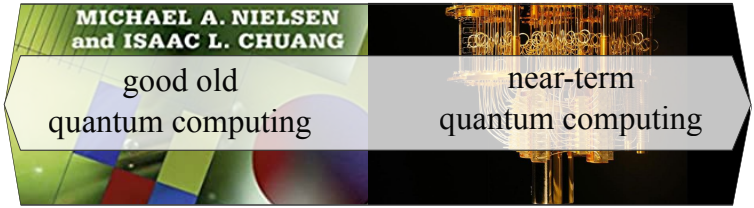
Quantum Machine Learning

Maria Schuld

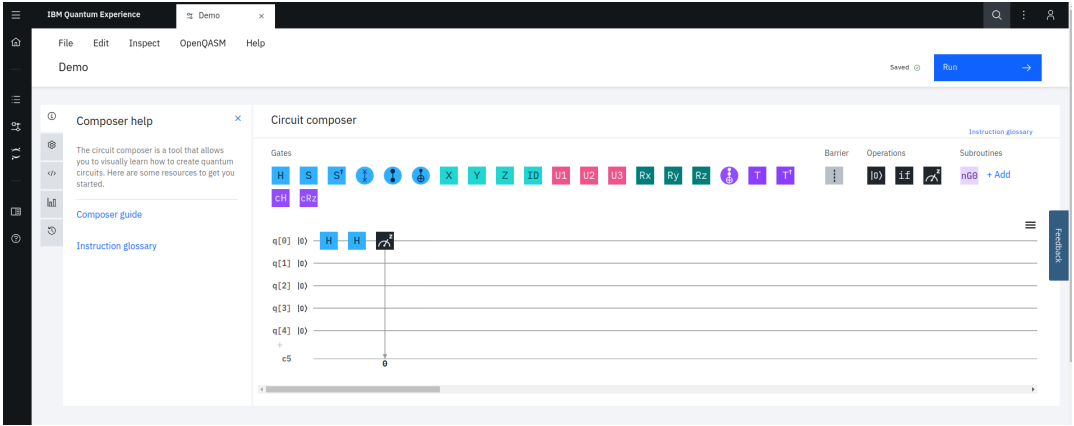
Xanadu and University of KwaZulu-Natal

MLSS, July 2020

Quantum computing is an emerging technology.



Quantum computing is an emerging technology.



Quantum computing is an emerging technology.

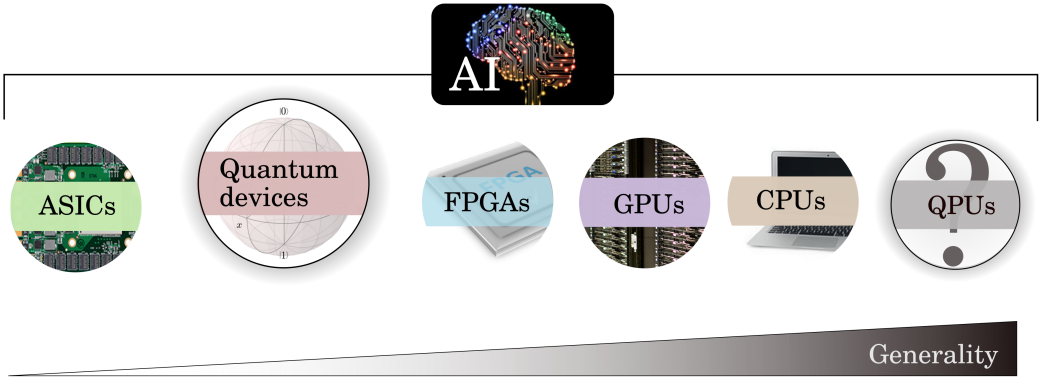
Quantum computing is an emerging technology.

WANTED

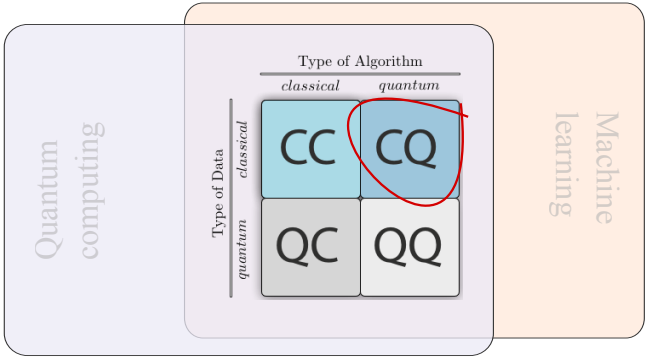
Application which

- doesn't care about noise
- gives us access to multi-billion $ markets
- attracts young researchers

Quantum computing is an emerging technology.



How can quantum computers innovate ML?



How can quantum computers innovate ML?

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

Agenda

- Quantum computing
- Data
- Algorithms
- Models

QUANTUM COMPUTING

Quantum theory predicts expectations of measurements.

- A quantum state $|\psi\rangle$ lives in a **Hilbert space** $\mathcal{H}$ with scalar product $\langle\psi|\,\psi\rangle$.
- An **observable** is represented by a Hermitian operator O on $\mathcal{H}$. The eigenvectors of O form an orthonormal basis of $\mathcal{H}$ with real eigenvalues. Every $|\psi\rangle \in \mathbb{C}^N$ can hence be expressed in O's eigenbasis $\{|\psi_i\rangle\}_{i=1\ldots N}$, $|\psi\rangle = \sum_{i=1}^{N} a_i|\psi_i\rangle$, where the $a_i \in \mathbb{C}$ are the **amplitudes**.
- The effect of applying O to an element $|\psi\rangle \in \mathbb{C}^N$ is fully defined by the eigenvalue equations $O|\psi_i\rangle = \lambda_i|\psi_i\rangle$ with eigenvalues λ_i. **Expectation values** of the observable property are calculated by $\mathbb{E}(O) = \langle\psi|O|\psi\rangle$.
- The dynamic evolution of a quantum state is represented by a **unitary operator** $U = U(t_2, t_1)$ mapping $|\psi(t_1)\rangle$ to $U(t_2, t_1)|\psi(t_1)\rangle = |\psi(t_2)\rangle$ with $U^\dagger U = 1$. U is the solution of the corresponding **Schrödinger equation** $i\hbar\partial_t|\psi\rangle = H|\psi\rangle$ with **Hamiltonian** H.

Schuld & Petruccione, Springer 2018

Quantum theory predicts expectations of measurements.

1. Consider a random variable M (measurement) that can take the values (observations) $\{m_1, ..., m_N\}$.
2. Assign probabilities $\{p_1, ..., p_N\}$ to these values quantifying our knowledge on how likely an observation is to occur.
3. The expectation value of the random variable is defined as

$$\langle M \rangle = \sum_{i=1}^{N} p_i m_i,$$

Schuld & Petruccione, Springer 2018

Quantum theory predicts expectations of measurements.

Use the notation

$$q = \begin{pmatrix} \sqrt{p_1} \\ \vdots \\ \sqrt{p_N} \end{pmatrix} = \sqrt{p_1} \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} + \ldots + \sqrt{p_N} \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix}, \qquad M = \begin{pmatrix} m_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & m_N \end{pmatrix}.$$

The expectation value can now be written as

$$\langle M \rangle = q^T M q = \sum_{i=1}^{N} p_i m_i$$

Schuld & Petruccione, Springer 2018

Quantum theory predicts expectations of measurements.

$$q \to \psi = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} \in \mathbb{C}^N, \quad |\alpha_i|^2 = p_i$$

$$M \to O_{\text{hermitian}} \in \mathbb{C}^{N \times N}, \quad \text{eig}[\mathrm{O}] = \{\mathrm{m}_1, \ldots, \mathrm{m}_\mathrm{N}\}$$

$$\langle M \rangle = \psi^T M \psi = \langle \psi | M | \psi \rangle = \sum_{i=1}^{N} p_i m_i$$

Schuld & Petruccione, Springer 2018

Quantum theory predicts expectations of measurements.

$$\begin{pmatrix} s_{11} & \dots & s_{1N} \\ \vdots & \ddots & \vdots \\ s_{N1} & \dots & s_{NN} \end{pmatrix} \begin{pmatrix} p_1 \\ \vdots \\ p_N \end{pmatrix} = \begin{pmatrix} p'_1 \\ \vdots \\ p'_N \end{pmatrix}, \qquad \sum_{i=1}^{N} p_i = \sum_{i=1}^{N} p'_i = 1$$

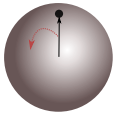
Schuld & Petruccione, Springer 2018

Quantum theory predicts expectations of measurements.

$$\begin{pmatrix} u_{11} & \dots & u_{1N} \\ \vdots & \ddots & \vdots \\ u_{N1} & \dots & u_{NN} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \alpha'_1 \\ \vdots \\ \alpha'_N \end{pmatrix}, \qquad \sum_{i=1}^{N} |\alpha_i|^2 = \sum_{i=1}^{N} |\alpha'_i|^2 = 1$$

Schuld & Petruccione, Springer 2018

Quantum computers perform linear algebra.



PHYSICAL CIRCUIT

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

MATHEMATICAL DESCRIPTION

$$2^n \left[\begin{array}{c} 1 \\ 0 \\ \vdots \end{array} \right]$$

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

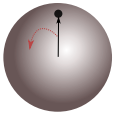
$|0|^2 = p(0...01)$

Quantum computers perform linear algebra.

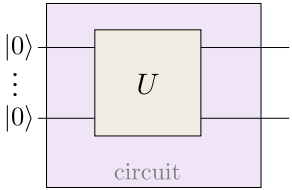
```python
from pennylane import *

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

@qnode(dev)
def circuit():
    return probs(wires=[0, 1])

print(circuit()) # [1. 0. 0. 0.]
print(dev.state) # [1.+0.j 0.+0.j 0.+0.j 0.+0.j]
```

pennylane.ai

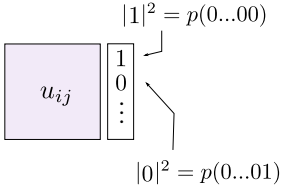
Quantum computers perform linear algebra.



PHYSICAL CIRCUIT

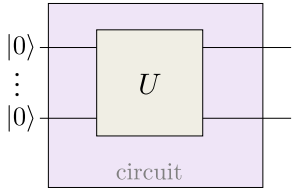


MATHEMATICAL DESCRIPTION



Quantum computers perform linear algebra.

PHYSICAL CIRCUIT



MATHEMATICAL DESCRIPTION

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

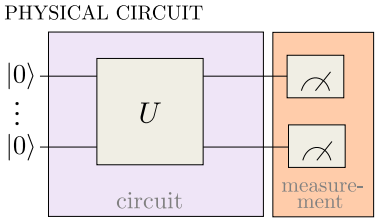
ψ_1
ψ_2
$\vdots$

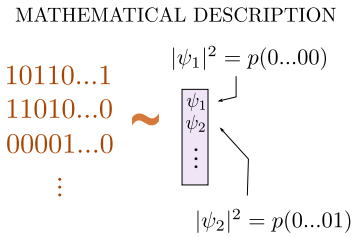
$|\psi_2|^2 = p(0...01)$

Quantum computers perform linear algebra.

```
from pennylane import *
import numpy as np

dev = device('default.qubit', wires=2)
U = np.array([[0.,   -0.70710678, 0.,    0.70710678],
              [0.70710678, 0.,   -0.70710678, 0.   ],
              [0.70710678, 0.,    0.70710678, 0.   ],
              [0.,   -0.70710678, 0.,   -0.70710678]])

@qnode(dev)
def circuit():
    QubitUnitary(U, wires=[0, 1])
    return probs(wires=[0, 1])

print(circuit()) # [0. 0.5 0.5 0.]
print(dev.state) # [0.+0.j 0.707+0.j 0.707+0.j 0.+0.j]
```

Quantum computers perform linear algebra.

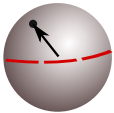




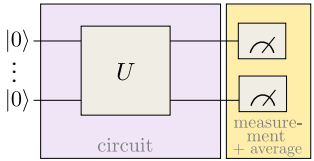
Quantum computers perform linear algebra.

```python
from pennylane import *
import numpy as np

dev = device('default.qubit', wires=2, shots=1)
U = np.array([[0.,   -0.70710678, 0.,    0.70710678],
              [0.70710678, 0.,   -0.70710678, 0.   ],
              [0.70710678, 0.,    0.70710678, 0.   ],
              [0.,   -0.70710678, 0.,   -0.70710678]])

@qnode(dev)
def circuit():
    QubitUnitary(U, wires=[0, 1])
    return sample(PauliZ(wires=0)), sample(PauliZ(wires=1))

print(circuit()) # [[-1], [ 1]]
print(dev.state) # [0.+0.j 0.707+0.j 0.707+0.j 0.+0.j]
```

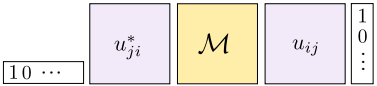
Quantum computers perform linear algebra.



PHYSICAL CIRCUIT



MATHEMATICAL DESCRIPTION

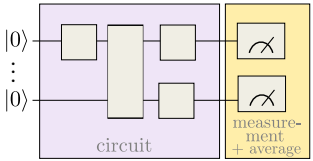


Quantum computers perform linear algebra.

```python
from pennylane import *
import numpy as np

dev = device('default.qubit', wires=2, shots=1)
U = np.array([[0.,   -0.70710678, 0.,    0.70710678],
              [0.70710678, 0.,   -0.70710678, 0.   ],
              [0.70710678, 0.,    0.70710678, 0.   ],
              [0.,   -0.70710678, 0.,   -0.70710678]])

@qnode(dev)
def circuit():
    QubitUnitary(U, wires=[0, 1])
    return expval(PauliZ(wires=0)), expval(PauliZ(wires=1))

print(circuit()) # [0., 0.]
print(dev.state) # [0.+0.j 0.707+0.j 0.707+0.j 0.+0.j]
```

Quantum computers perform linear algebra.

PHYSICAL CIRCUIT



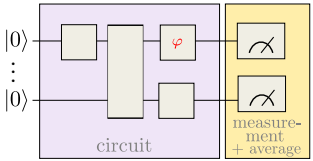
MATHEMATICAL DESCRIPTION

Quantum computers perform linear algebra.

```python
from pennylane import *

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

@qnode(dev)
def circuit():
    PauliX(wires=0)
    CNOT(wires=[0, 1])
    Hadamard(wires=0)
    PauliZ(wires=1)
    return expval(PauliZ(wires=0)), expval(PauliZ(wires=1))

print(circuit()) # [0., -1.]
print(dev.state) # [ 0.+0.j -0.70710678+0.j  0.+0.j  0.70710678+0.j]
```

Quantum computers perform linear algebra.

PHYSICAL CIRCUIT

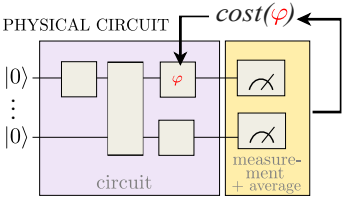


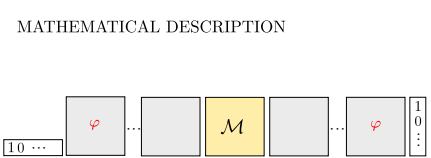
MATHEMATICAL DESCRIPTION

Quantum computers perform linear algebra.

```python
from pennylane import *

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

@qnode(dev)
def circuit(phi):
    RX(phi, wires=0)
    CNOT(wires=[0, 1])
    Hadamard(wires=0)
    PauliZ(wires=1)
    return expval(PauliZ(wires=0)), expval(PauliZ(wires=1))

print(circuit(0.2)) # [0. 0.98006658]
print(dev.state) # [0.70+0.j 0.+0.07j 0.70+0.j 0.-0.070j]
```

Quantum computers perform linear algebra.



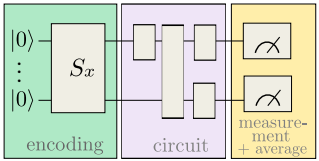


Quantum computers perform linear algebra.

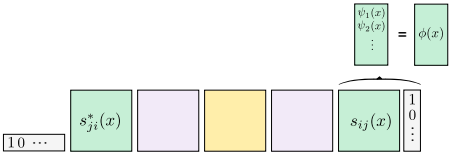
```
from pennylane import *

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

@qnode(dev)
def circuit(phi):
    Hadamard(wires=0)
    RY(phi, wires=0)
    return expval(PauliZ(wires=0))

phi = 0.2
opt = GradientDescentOptimizer(stepsize=0.2)

for i in range(5):
    phi = opt.step(circuit, phi)

    print(phi)
    # 0.39601331556824826
    # 0.5805345472544579
    # 0.7477684644009802
    # 0.8944100937922911
    # 1.0196058853338432
```

Quantum computers perform linear algebra.

PHYSICAL CIRCUIT



MATHEMATICAL DESCRIPTION

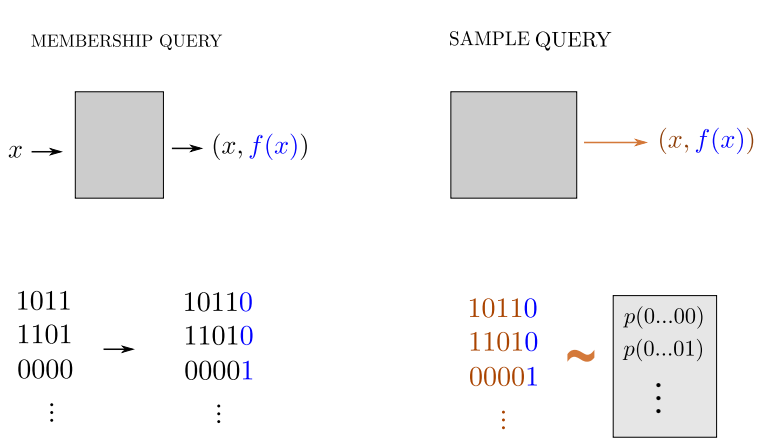


Quantum computers perform linear algebra.

```python
from pennylane import *

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

@qnode(dev)
def circuit(phi, x=None):
    RX(x, wires=[0])
    CNOT(wires=[0, 1])
    RY(phi, wires=[1])
    return expval(PauliZ(wires=[1]))

print(circuit(0.2, x=0.1)) # 0.975
print(circuit(0.2, x=0.5)) # 0.860
```

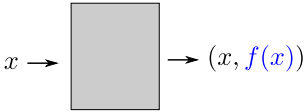
DATA

Quantum computers cannot learn from "exponentially less" data.

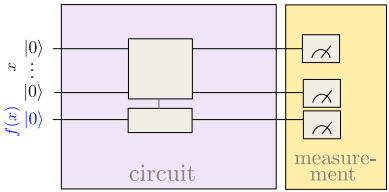


Quantum computers cannot learn from “exponentially less” data.

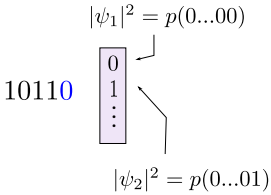
MEMBERSHIP QUERY



PHYSICAL CIRCUIT

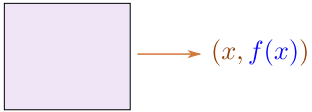


MATHEMATICAL DESCRIPTION

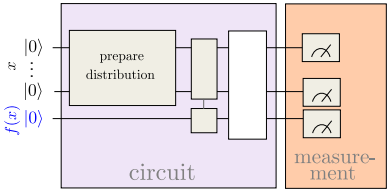


Quantum computers cannot learn from “exponentially less” data.

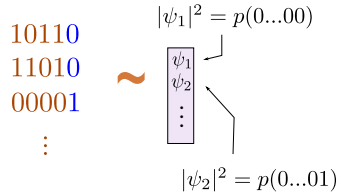
QUANTUM SAMPLE ORACLE



PHYSICAL CIRCUIT



MATHEMATICAL DESCRIPTION



Quantum computers cannot learn from “exponentially less” data.

A Survey of Quantum Learning Theory

Srinivasan Arunachalam* Ronald de Wolf†

Abstract

This paper surveys quantum learning theory: the theoretical aspects of machine learning using quantum computers. We describe the main results known for three models of learning: exact learning from membership queries, and Probably Approximately Correct (PAC) and agnostic learning from classical or quantum examples.

Arunachalam 1701.06806

Quantum computers cannot learn from “exponentially less” data.

Exact learning. In this setting the goal is to learn a target concept from the ability to interact with it. For concreteness, we focus on learning target concepts that are Boolean functions: the target is some unknown $c : \{0,1\}^n \to \{0,1\}$ coming from a known concept class $\mathcal{C}$ of functions,[2] and our goal is to identify c exactly, with high probability, using *membership queries* (which allow the learner to learn $c(x)$ for x of his choice). If the measure of complexity is just the number of queries, the main results are that quantum exact learners can be polynomially more efficient than classical, but not more. If the measure of complexity is *time*, then under reasonable complexity-theoretic assumptions some concept classes can be learned much faster from quantum membership queries (i.e., where the learner can query c on a superposition of x's) than is possible classically.

Arunachalam 1701.06806

Quantum computers cannot learn from "exponentially less" data.

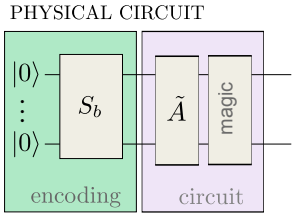
PAC learning. In this setting one also wants to learn an unknown $c : \{0,1\}^n \to \{0,1\}$ from a known concept class $\mathcal{C}$, but in a more passive way than with membership queries: the learner receives several *labeled examples* $(x, c(x))$, where x is distributed according to some unknown probability distribution D over $\{0,1\}^n$. The learner gets multiple i.i.d. labeled examples. From this limited "view" on c, the learner wants to generalize, producing a *hypothesis* h that probably agrees with c on "most" x, *measured according to the same* D. This is the classical Probably Approximately Correct (PAC) model. In the quantum PAC model [BJ99], an example is not a random sample but a *superposition* $\sum_{x \in \{0,1\}^n} \sqrt{D(x)}|x, c(x)\rangle$. Such quantum examples can be useful for some learning tasks with a fixed distribution D (e.g., uniform D) but it turns out that in the usual distribution-independent PAC model, quantum and classical sample complexity are equal up to constant factors, for every concept class $\mathcal{C}$. When the measure of complexity is *time*, under reasonable complexity-theoretic assumptions, some concept classes can be PAC learned much faster by quantum learners (even from classical examples) than is possible classically.

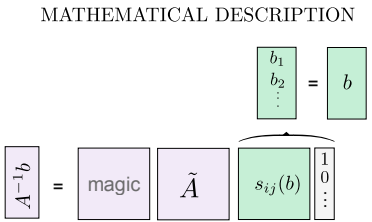
Arunachalam 1701.06806

ALGORITHMS

Quantum computers can invert exponentially large matrices.*

Assumption: the bottleneck of my ML algorithm is matrix inversion: $Ax = b \rightarrow x = A^{-1}b$.





Wiebe et al. 1204.5242, Rebentrost et al. 1307.0471, Zhao et al. 1803.10520, Kerenidis et al. 1603.08675

Quantum computers can invert exponentially large matrices.*

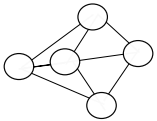
1. Prepare ψ_b.
2. Apply $\tilde{A}$ ψ_b (where $\tilde{A} = e^{-iA}$, but that is not so important).

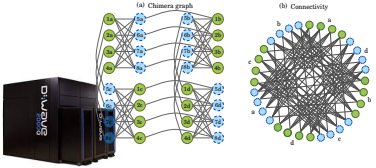
$$\begin{aligned}
\tilde{A}\psi_b &= \tilde{A}\ (\langle a_1, \psi_b\rangle a_1 + \cdots + \langle a_N, \psi_b\rangle a_N) \\
&= \langle a_1, \psi_b\rangle \lambda_1 a_1 + \cdots + \langle a_N, \psi_b\rangle \lambda_N a_N \\
&\Rightarrow \langle a_1, \psi_b\rangle \frac{1}{\lambda_1} a_1 + \cdots + \langle a_N, \psi_b\rangle \frac{1}{\lambda_N} a_N \\
&= \psi_x
\end{aligned}$$

3. Use ψ_b to do something interesting.

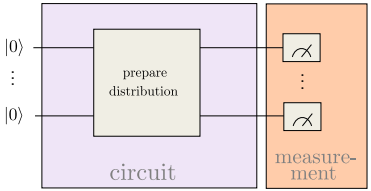
Wiebe et al. 1204.5242, Rebentrost et al. 1307.0471, Zhao et al. 1803.10520, Kerenidis et al. 1603.08675

Quantum computers can train Boltzmann machines.*

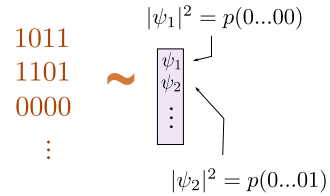




PHYSICAL CIRCUIT



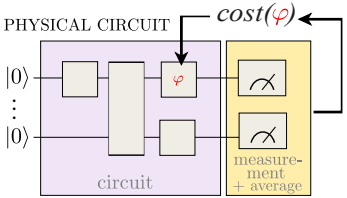
MATHEMATICAL DESCRIPTION

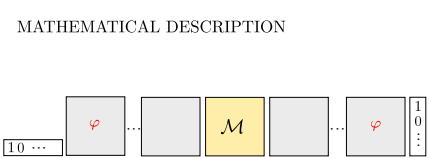


Denil & Freitas 2012(?) `https://www.cs.ubc.ca/~nando/papers/quantumrbm.pdf`

MODELS

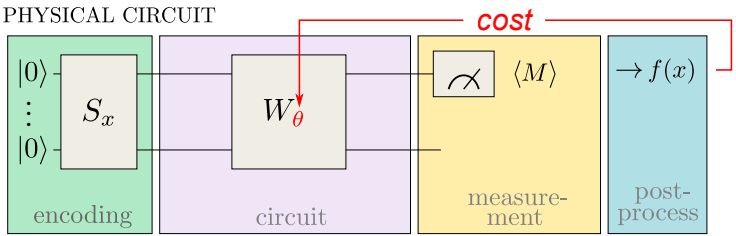
We can train quantum computations.





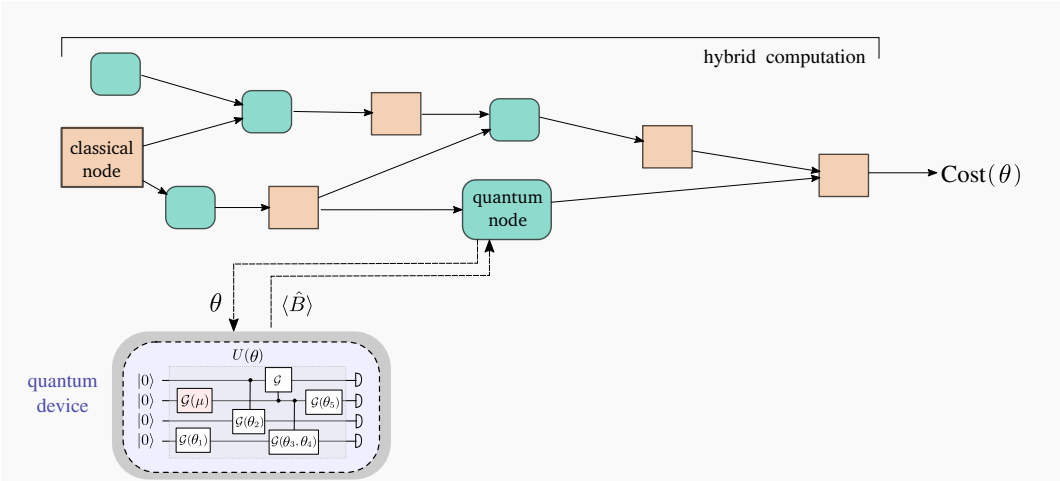
Farhi & Neven 1802.06002, Schuld et al. 1804.00633, Benedetti et al. 1906.07682

We can train quantum computations.



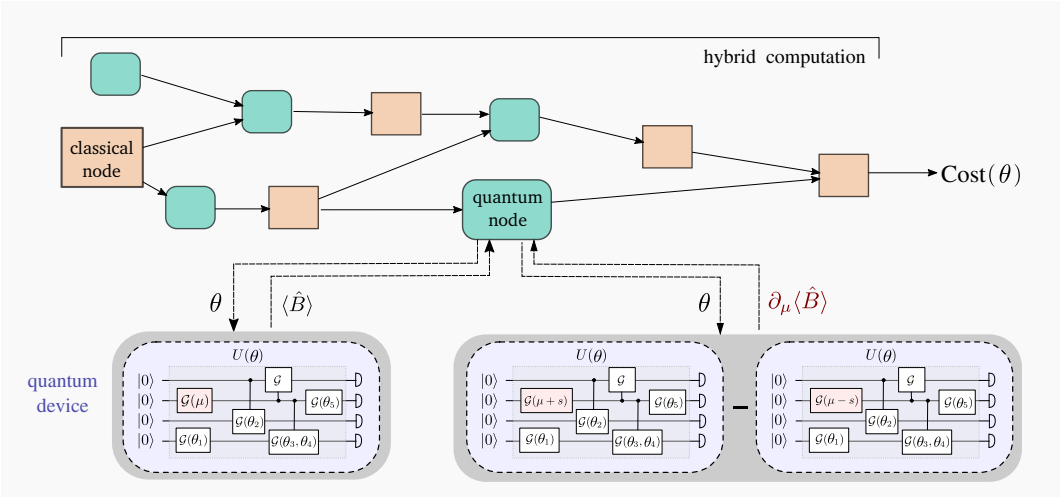
Farhi & Neven 1802.06002, Schuld et al. 1804.00633, Benedetti et al. 1906.07682

We can do gradient descent on variational circuits.



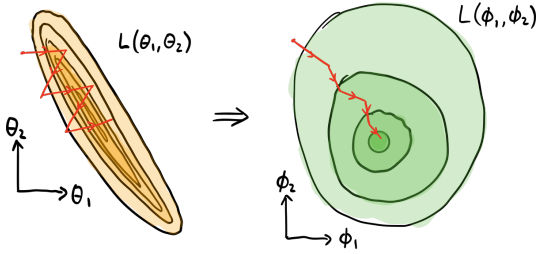
Guerreschi & Smelyanskiy 1701.01450, Mitarai et al. 1803.00745, Schuld et al. 1811.11184

We can do gradient descent on variational circuits.



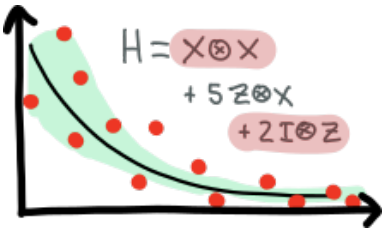
Guerreschi & Smelyanskiy 1701.01450, Mitarai et al. 1803.00745, Schuld et al. 1811.11184

We can do gradient descent on variational circuits.



Stokes, Izaac, Killoran, Carleo 1909.02108

We can do gradient descent on variational circuits.



Kübler et al. 1909.09083, Sweke et al. 1910.01155

We can do gradient descent on variational circuits.

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-

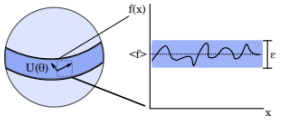


FIG. 1. A cartoon of the general geometric results from this work. The sphere depicts the phenomenon of concentration of

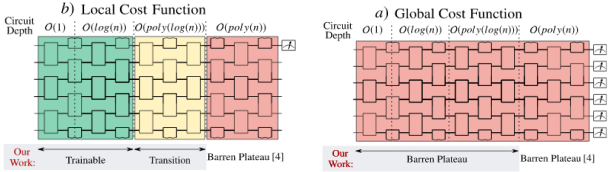
McClean et al. 1803.11173

We can do gradient descent on variational circuits.

Cost-Function-Dependent Barren Plateaus in Shallow Quantum Neural Networks

M. Cerezo,[1,2] Akira Sone,[1,2] Tyler Volkoff,[1] Lukasz Cincio,[1] and Patrick J. Coles[1]
[1] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM, USA.*
[2] *Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM, USA*

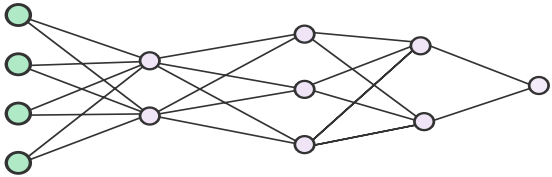
Variational quantum algorithms (VQAs) optimize the parameters $\boldsymbol{\theta}$ of a quantum neural network $V(\boldsymbol{\theta})$ to minimize a cost function C. While VQAs may enable practical applications of noisy quantum computers, they are nevertheless heuristic methods with unproven scaling. Here, we rigorously



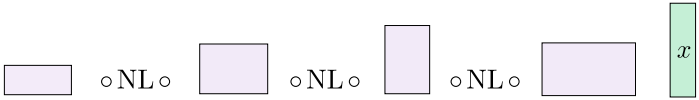
Cerezo et al. 2001.00550

Quantum circuits are unitary neural nets in feature space.

MODEL

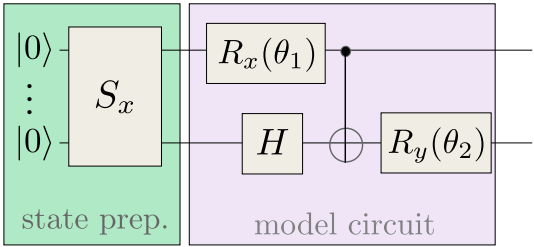


MATHEMATICAL DESCRIPTION



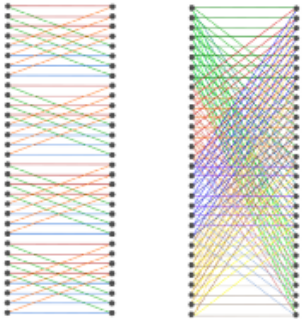
Quantum circuits are unitary neural nets in feature space.

PHYSICAL CIRCUIT



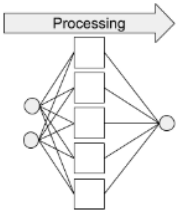
MATHEMATICAL DESCRIPTION

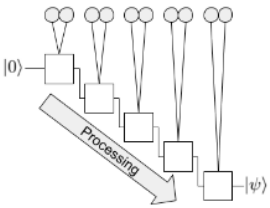
Quantum circuits are unitary neural nets in feature space.

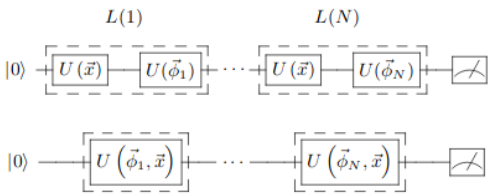


Schuld & Petruccione, Springer 2018

Quantum circuits are unitary neural nets in feature space.

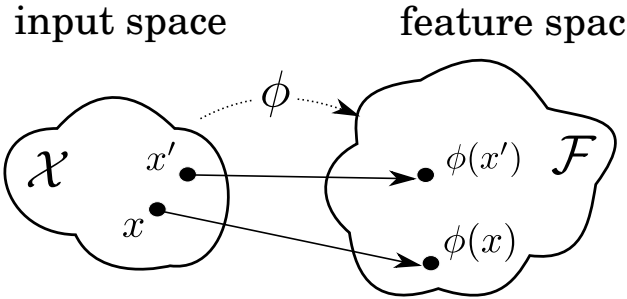






Pérez-Salinas et al. 1907.02085

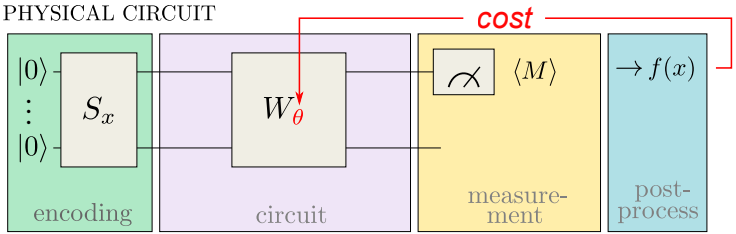
Quantum circuits are kernel methods.



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

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

Quantum circuits are kernel methods.



Schuld & Killoran 1803.07128, Havlicek et al. 1804.11326

Quantum circuits are kernel methods.

Quantum feature map:

$$x \to \phi(x) \in \mathbb{C}^{2^{n_{\text{qubits}}}}$$

Measurement:

$$\phi(x)^\dagger M \phi(x)$$

Quantum circuits are kernel methods.

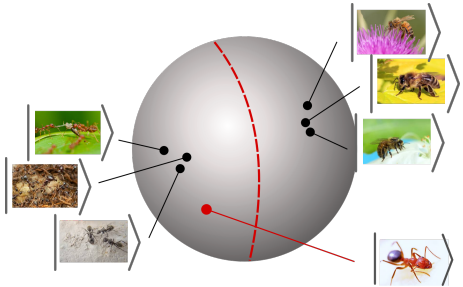
Quantum feature map:

$$x \to \phi(x) \in \mathbb{C}^{2^{n_{\text{qubits}}}}$$

Measurement:

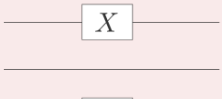
$$\begin{aligned} & \phi(x)^T M \phi(x) \\ &= \phi(x)^\dagger \; w w^\dagger \; \phi(x) \\ &= |\phi(x)^\dagger w|^2 \end{aligned}$$

Quantum circuits are kernel methods.



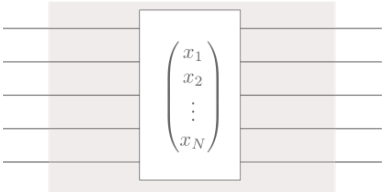
Data encoding defines a “quantum kernel”.

$$x \to \phi(x) = \begin{pmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix}$$



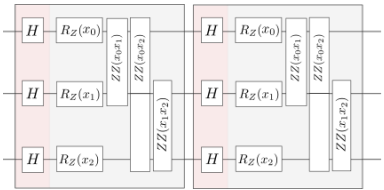
Data encoding defines a “quantum kernel”.

$$x \to \phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{pmatrix}$$

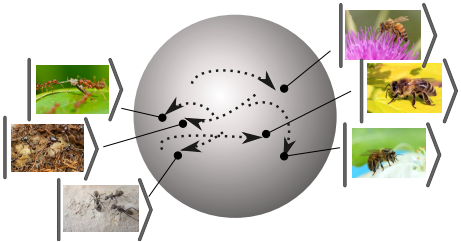


Data encoding defines a “quantum kernel”.

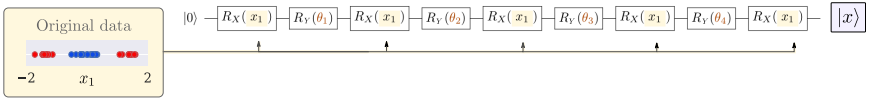
$$x \to S(x) \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$



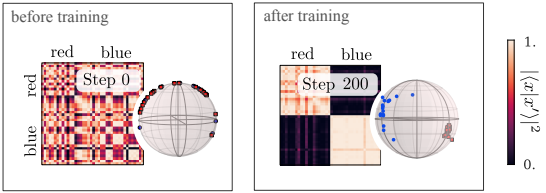
We can engineer/train our features.



We can engineer/train our features.

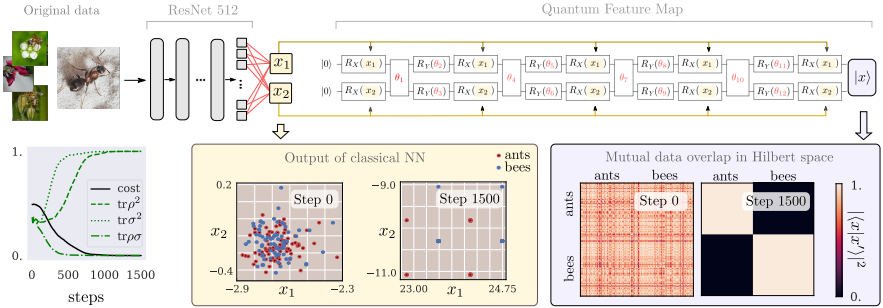


Mutual data overlap in Hilbert space



Lloyd et al. 2001.03622

We can engineer/train our features.



Lloyd et al. 2001.03622

OPEN PROBLEMS

Other questions than “are quantum computers better?”:

- Can we get stronger guarantees for 2-sample tests with quantum distributions?
- What measurement corresponds to a quantum maximum margin classifier?
- How can we benchmark quantum models if simulations are limited and devices too small?
- Is there a useful connection between quantum theory and deep learning?
- What function classes or kernels do quantum models give rise to?
- What is a good framework to study generalisation of quantum models?
- Will barren plateaus kill us?

Thank you!

quantum.ukzn.ac.za
www.pennylane.ai
www.xanadu.ai
@XanaduAI