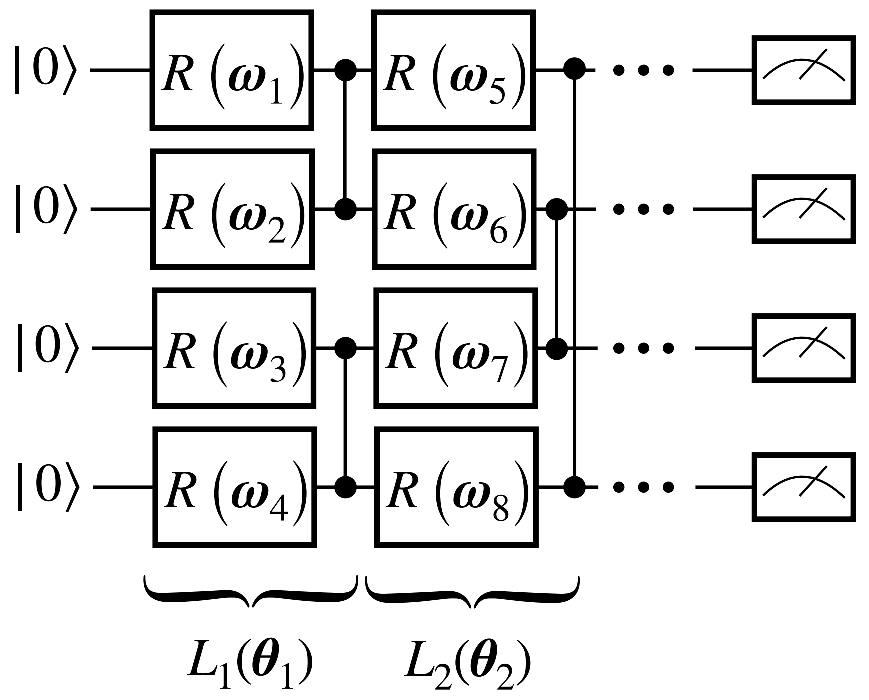
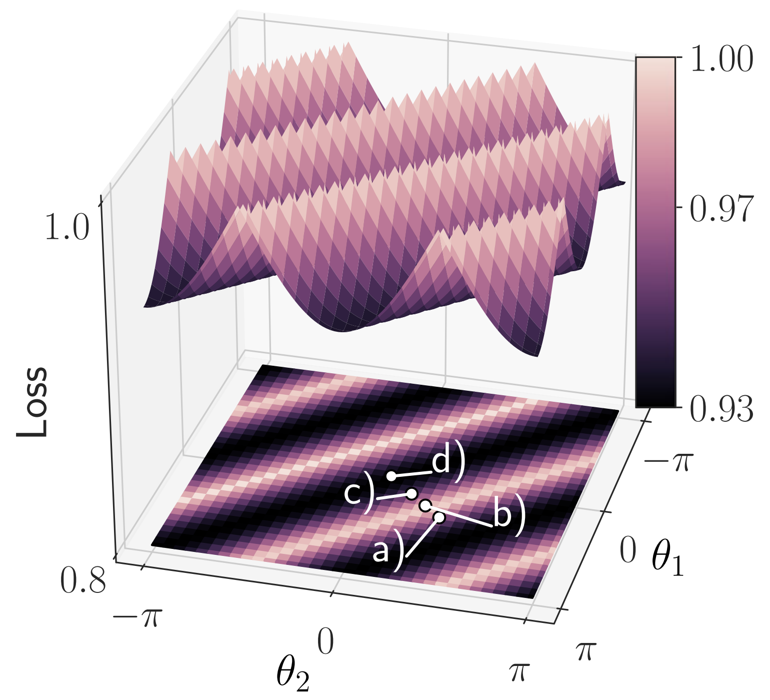
Introduction to Quantum Machine Learning

Paolo Stornati

paolo.stornati@icfo.eu



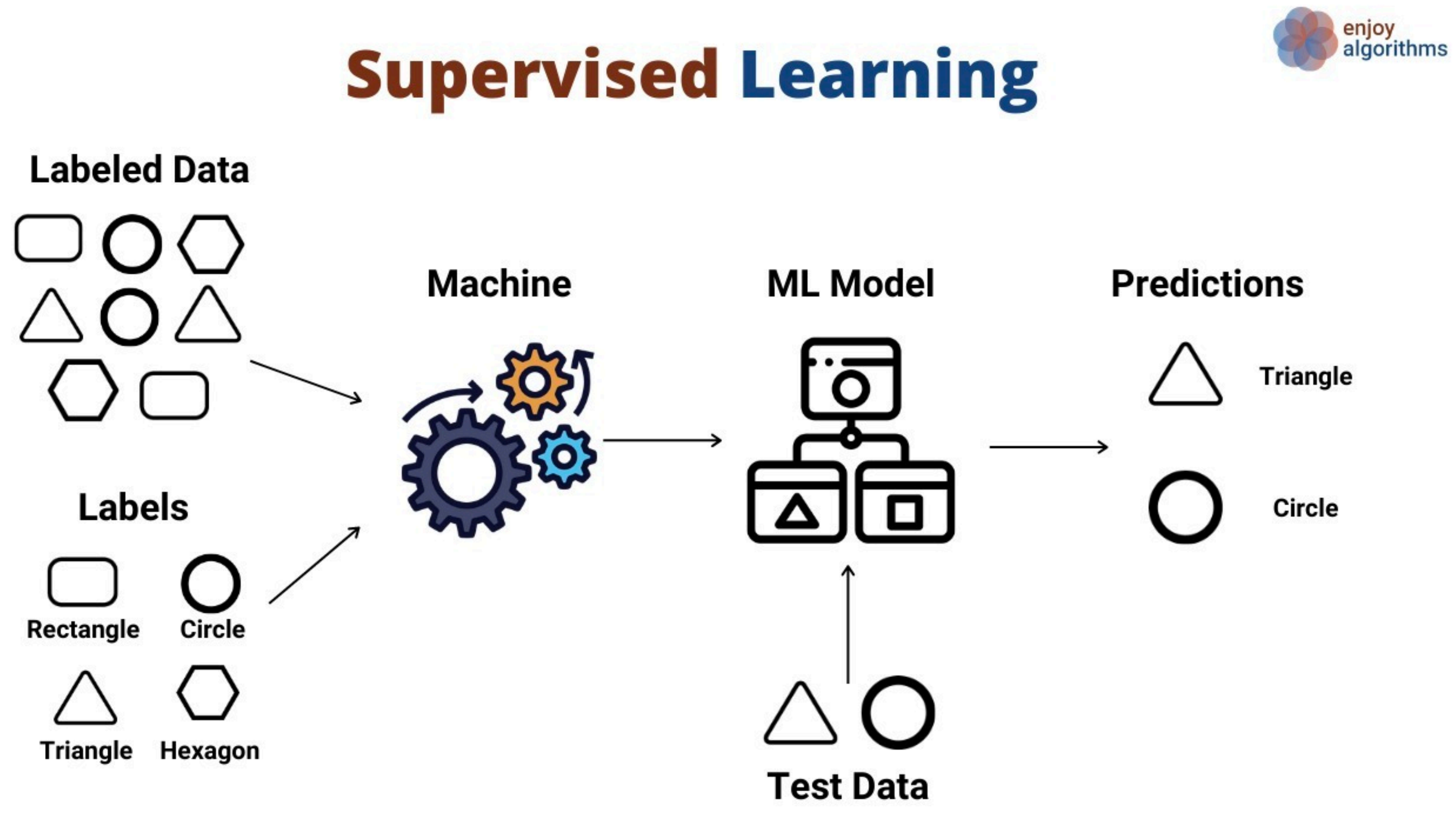


Spring School on Open-Source Tools for Quantum Computing & Simulation

Barcelona, March 2023

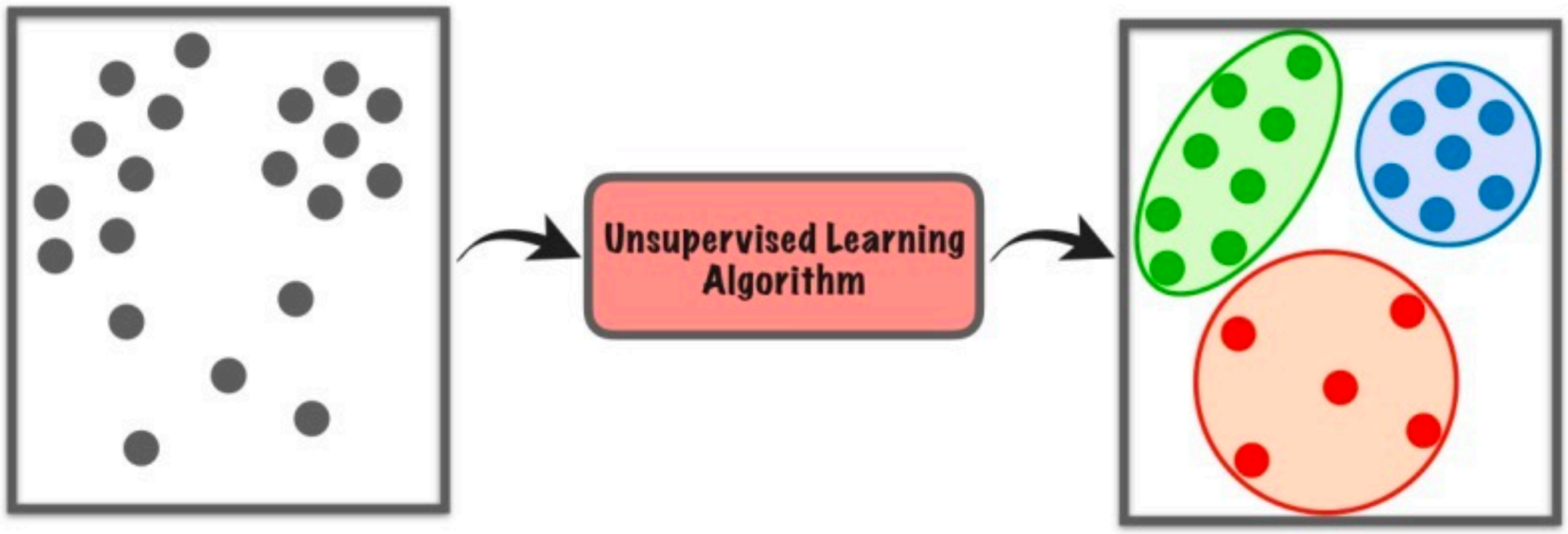
Supervised Learning

Source data are labelled aim to predict unlabelled data

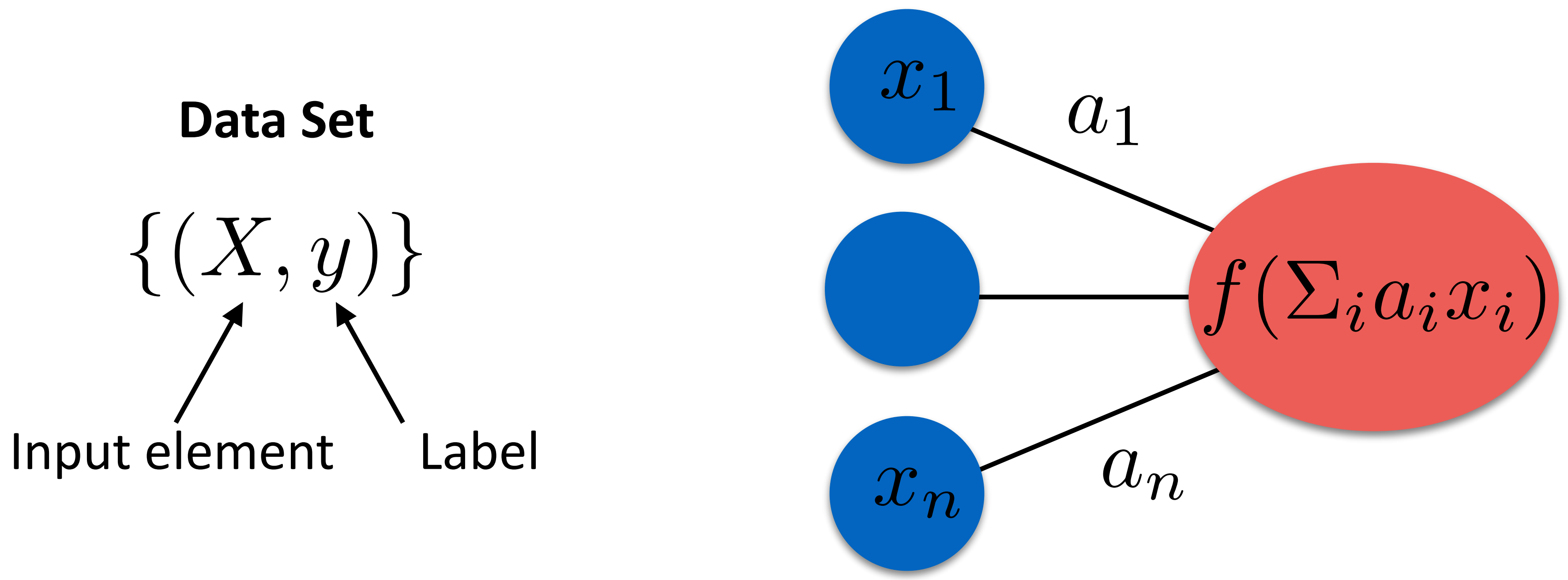


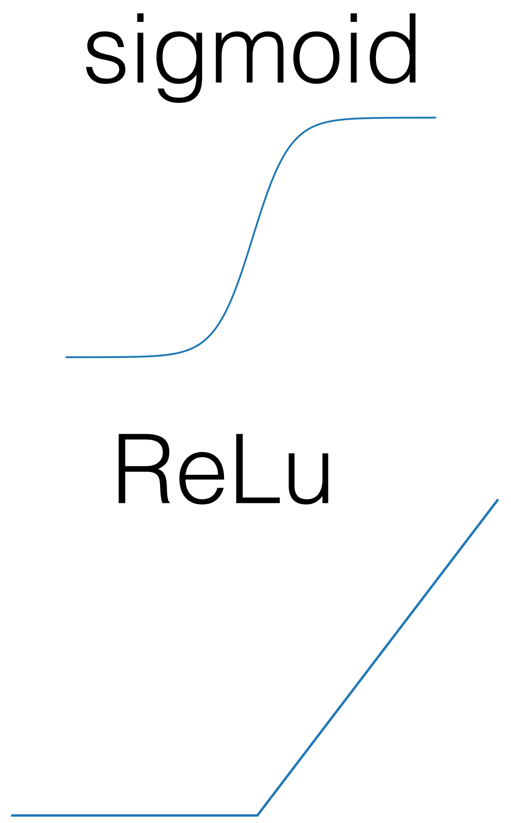
Unsupervised Learning

Source data are labelled aim to predict unlabelled data



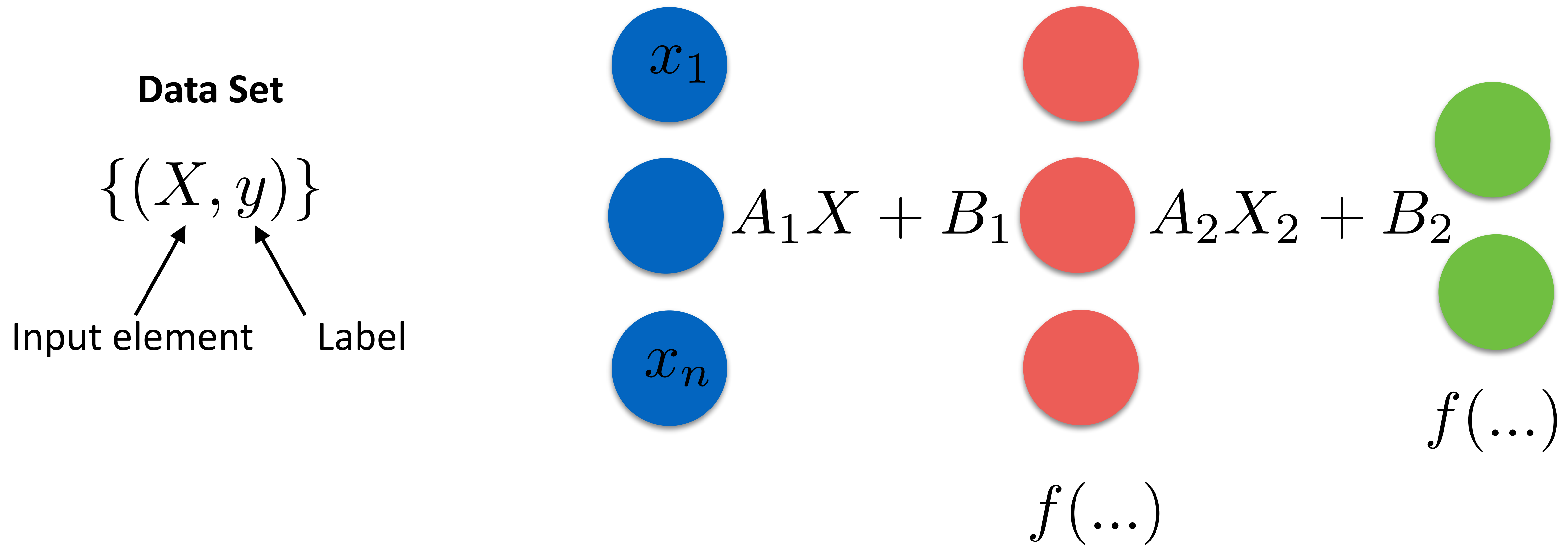
Neural Networks





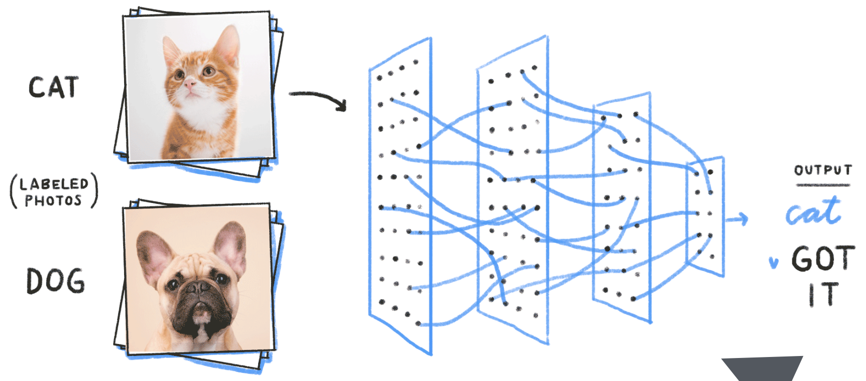
- f is an activation function.
- if f(x)=Θ(x) (Heaviside), then one recovers the perceptron.
- The choice of the activation function depends on the problem.
- The activation function allows onto have nonlinearities.
- In modern neural networks, the most used activations functions are the sigmoid and ReLu.

Neural Networks

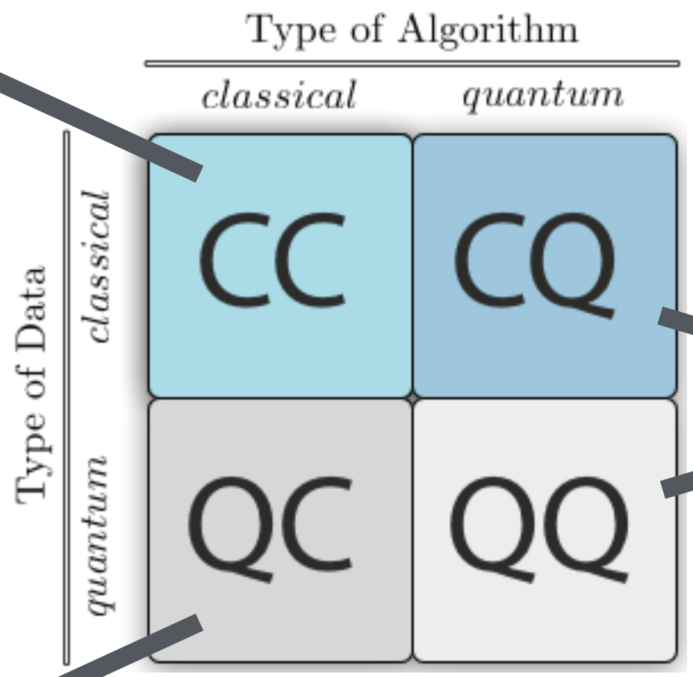


- Cost function: $C = \frac{1}{N}\Sigma_j(y_{\text{pred},j} - y_j)^2$
- Goal: minimize the cost function given the training set.
- Adjust the weights A_i and B_i to minimize the cost function.

Classical/Quantum Data VS Classical/ Quantum Algorithms

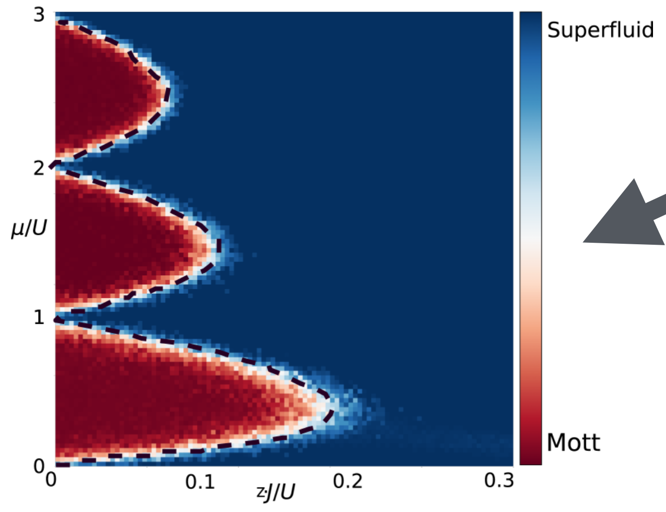


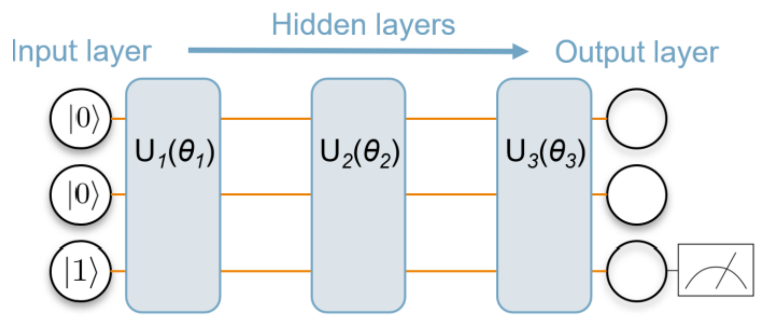
CLASSICAL MACHINE LEARNING



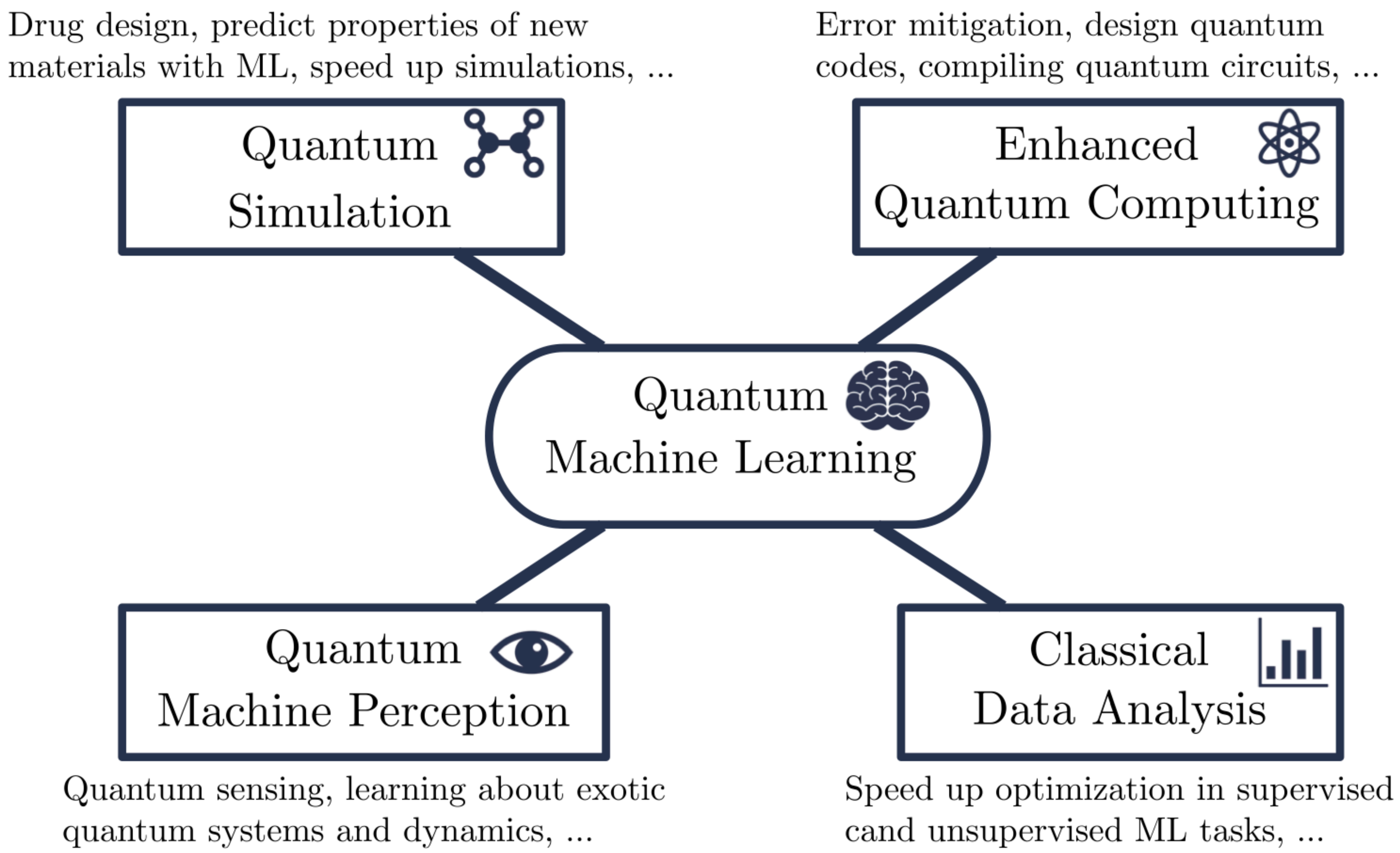
QUANTUM MACHINE LEARNING

PHASE TRANSITION CLASSIFICATION





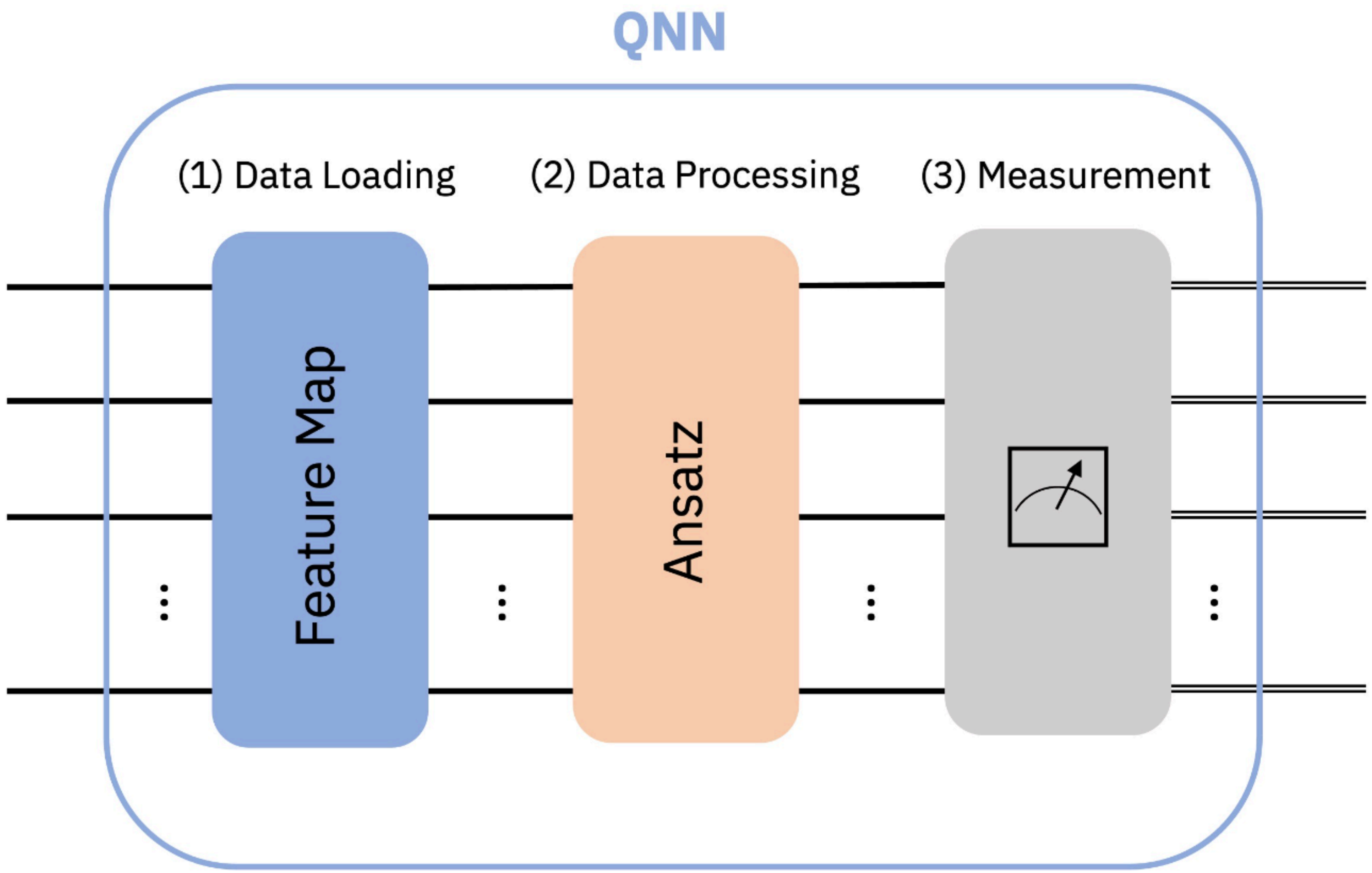
Applications of QML


Drug design, predict properties of new materials with ML, speed up simulations, ...
Quantum Simulation
Error mitigation, design quantum codes, compiling quantum circuits, ...
Enhanced Quantum Computing
Quantum Machine Learning
Quantum Machine Perception
Quantum sensing, learning about exotic quantum systems and dynamics, ...
Classical Data Analysis
Speed up optimization in supervised cand unsupervised ML tasks, ...

Picture adapted from arxiv.org/2303.09491

A quantum neural network

- QNN are quantum algorithms based on parametrized quantum circuits
- QNN are global unitary that depends on certain parameters θ



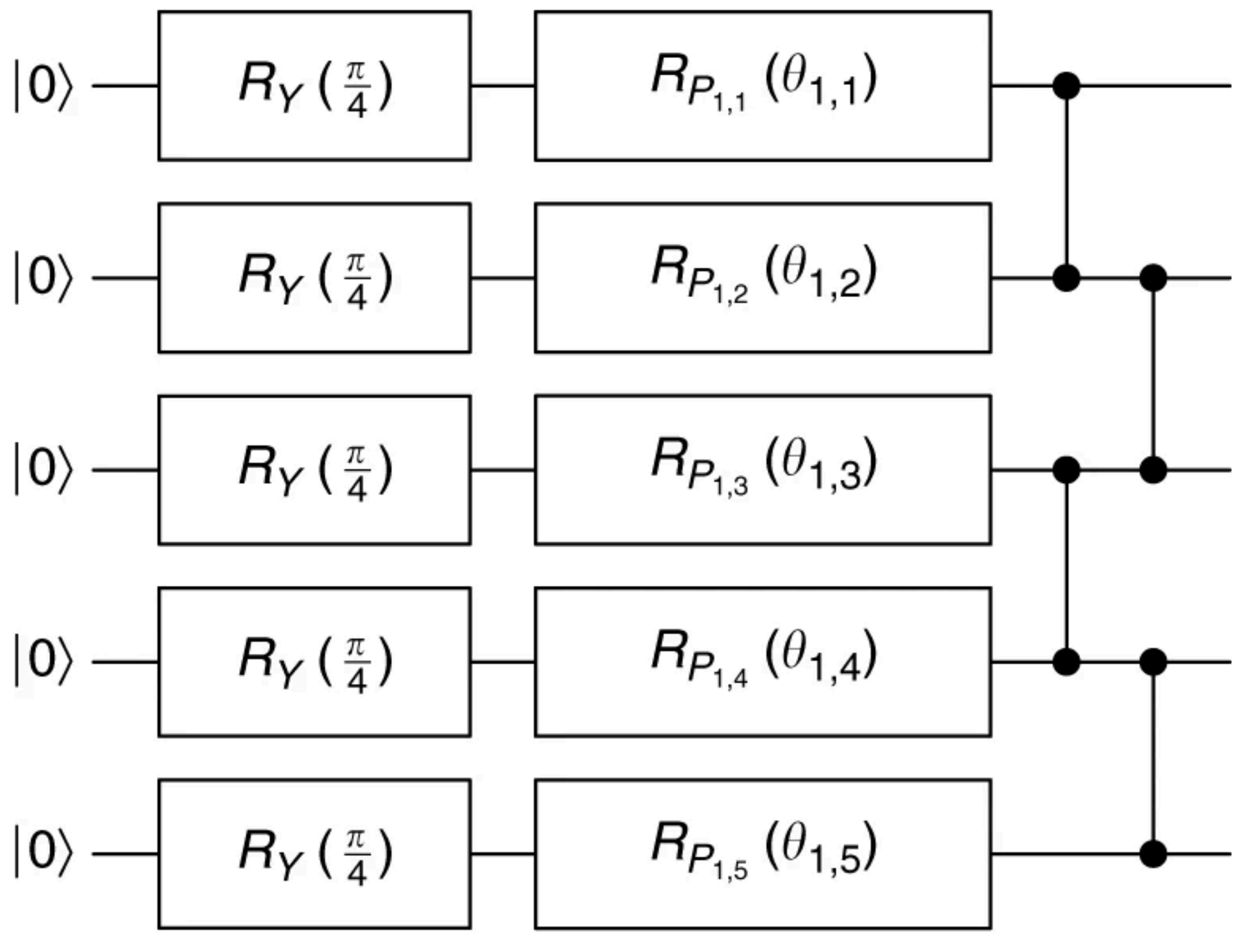
A quantum neural network

Variational quantum circuit ares defined as:

$$|\psi> = U(\theta)|\psi_0>$$

$U(\theta)$ is defined by a series of unitary operation: e.g.

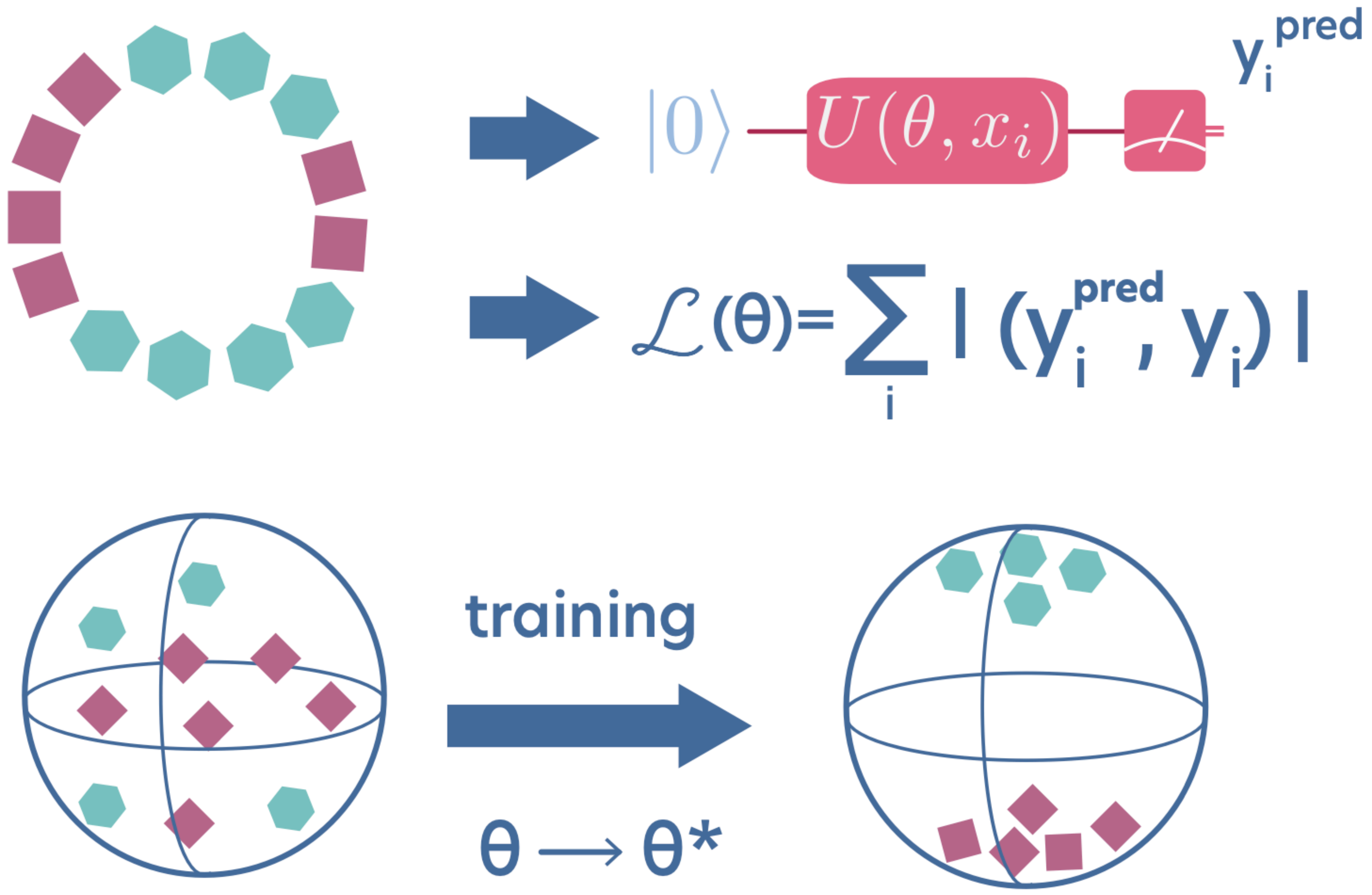
$$R_x(\theta) = e^{i\frac{\theta}{2}\sigma_x}$$



$|0\rangle$
$R_Y(\frac{\pi}{4})$
$R_{P_{1,1}}(\theta_{1,1})$
$R_{P_{1,2}}(\theta_{1,2})$
$R_{P_{1,3}}(\theta_{1,3})$
$R_{P_{1,4}}(\theta_{1,4})$
$R_{P_{1,5}}(\theta_{1,5})$

$U(\theta)$ is completely positive trace preserving (CPTP) map

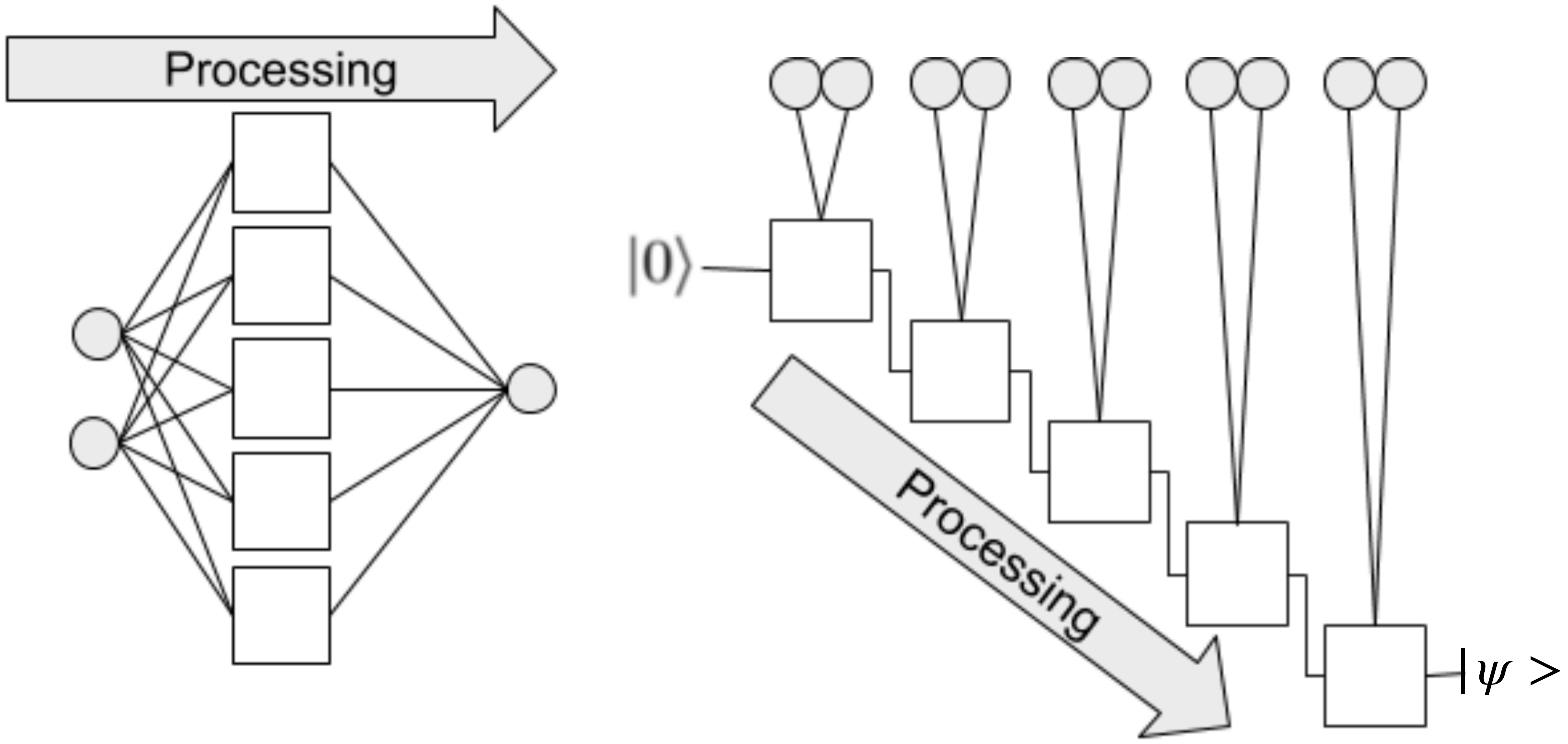
A Quantum Machine Learning algorithm



$|0\rangle$
$U(\theta, x_i)$
y_i^{pred}
$\mathcal{L}(\theta)=\sum_i l\,(y_i^{pred}, y_i)\,l$
training
$\theta \longrightarrow \theta^*$

Data re-uploading with QNN

Includes non linearity in the input data



Re-upload the data in the quantum circuit

Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

- Classical data are loaded as rotation in the qubit
- Three dimensional data $\vec{x}$ can be loaded using one rotation
- We call the uploading unitary $U(\vec{x})$
- The processing unitaries are one qubit rotation $U(\overrightarrow{\phi})$
- The layer in our QNN is:

$$L(i) = U(\vec{x})U(\overrightarrow{\phi}_i)$$

Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

- The wave function is constructed:

$$|\psi(\vec{x}, \overrightarrow{\phi}_i) > = U(\vec{x})U(\overrightarrow{\phi}_i)|\psi_0 >$$

- For multilayer architecture:

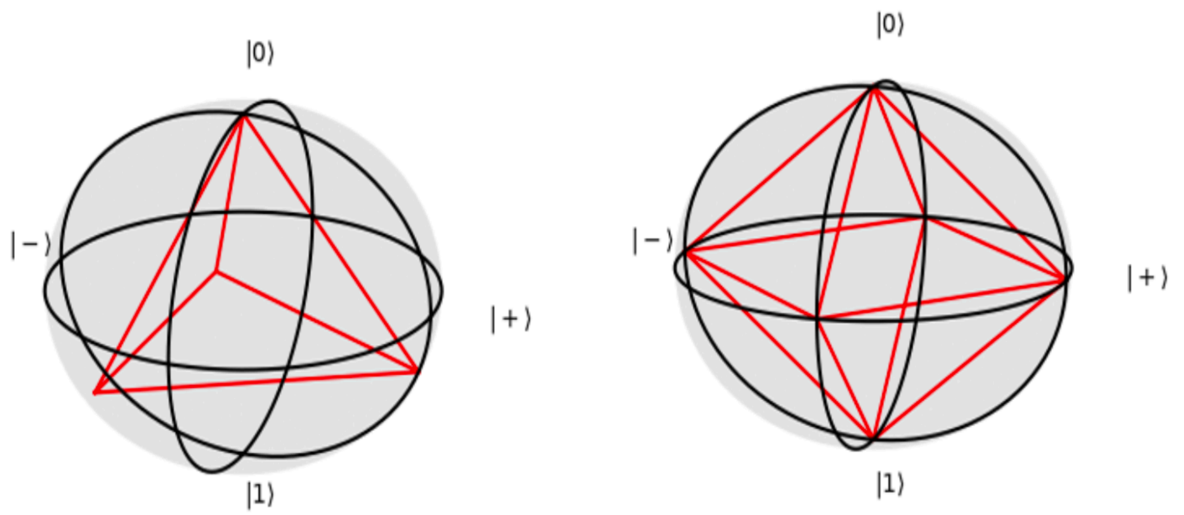
$$|\psi(\vec{x}, \overrightarrow{\phi}) > = \prod L(i)|\psi_0 >$$

- We need a cost function for our classification task
- The more layers the more representation capabilities the circuit

Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

- Let’s assume we have a binary classification problem
- We can have $P(0)$ for $|0>$ and $P(1)$ for $|1>$
- We can put also a trashed value λ:
- The output is 0 if $P(0) > \lambda$
- For more dimensional classification we can have different λ



Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

- Given a reference state in the block state ψ_c, we can write the cost function as:

$$C = \sum_{\mu}^{M} \left(1 - \left| < \psi_\mu | \psi(\vec{x}_\mu, \vec{\phi}) > \right|^2 \right)$$

- ψ_μ is the corresponding labelled state to the μ data point
- A classifier with this shape is universal
- Universal Approximation Theorem for classical NN

Perez Salinas et al. Quantum 4, 226 (2020)

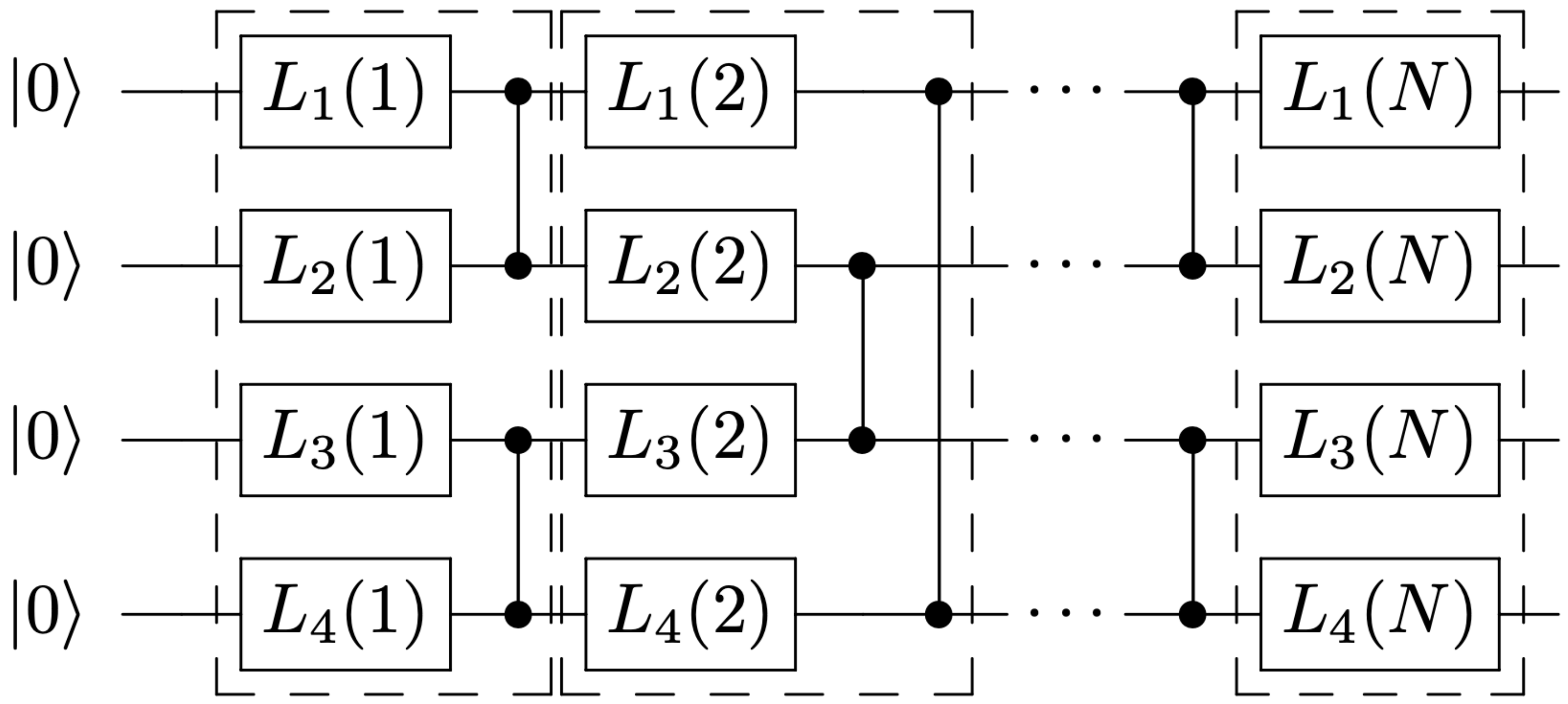
Data re-uploading with QNN

- Single qubit classifier cannot produce any quantum advantage
- A very large number of layer is needed to approximate a complex function
- The introduction of multiple qubits entangled may improve its performance
- The idea is to re upload the data in different cubit and entangle them

Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

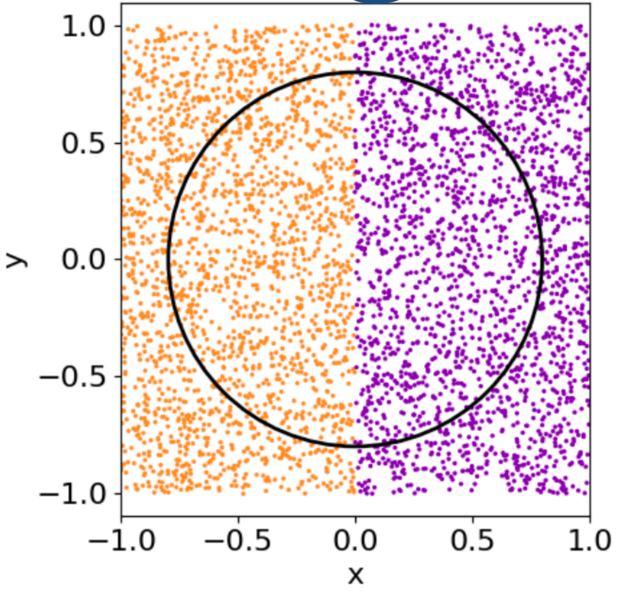
- After every upload a entangling layer is introduced
- The classification is performed only on one qubit
- Not clear wether this prove advantage



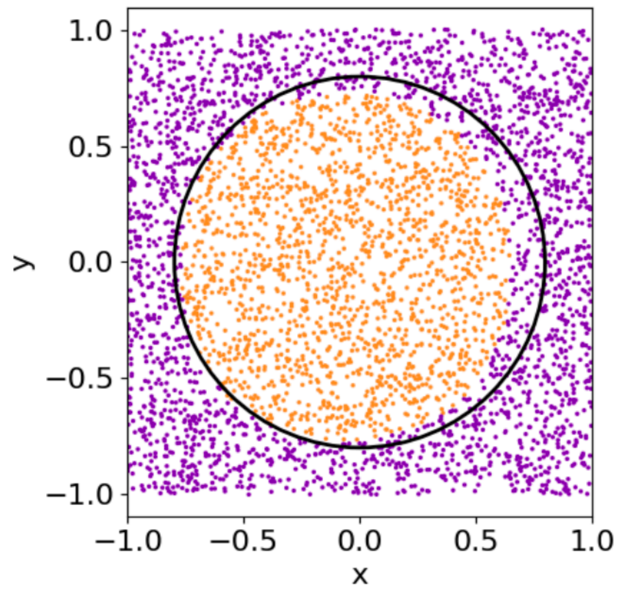
Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

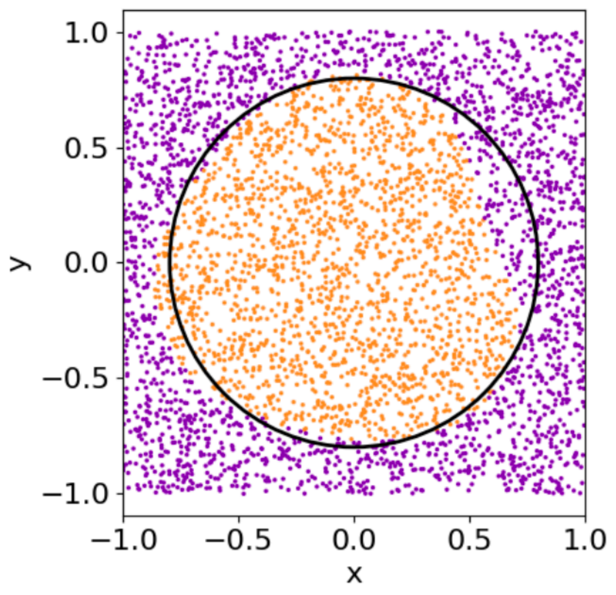
- Example on a simple classification on a circle
- The area of the two colours is the same
- Good approximation with 4 layer



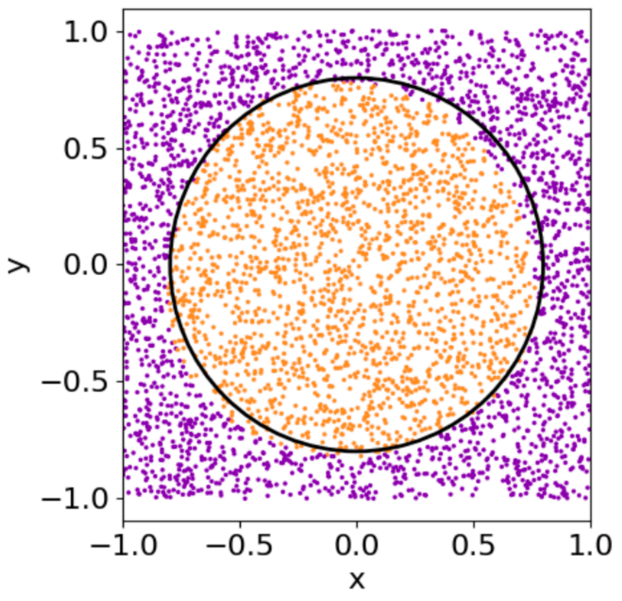
(a) 1 layer



(b) 2 layers



(c) 4 layers

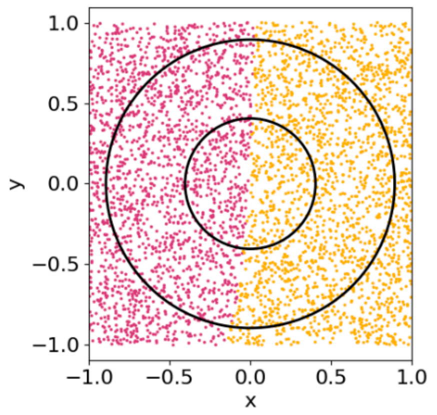


(d) 8 layers

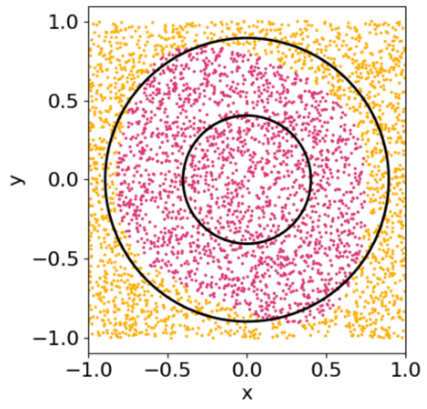
Perez Salinas et al. Quantum 4, 226 (2020)

Data re-uploading with QNN

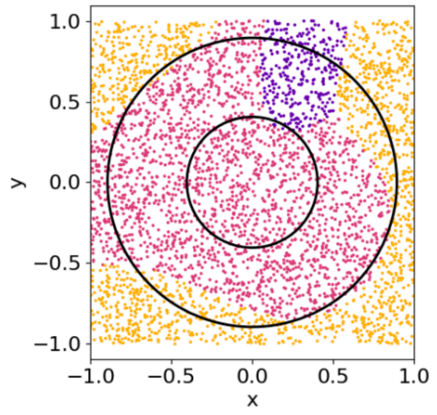
- Optimisation of a non convex function in 3 class problem
- Multiple entangled qubit improve the result



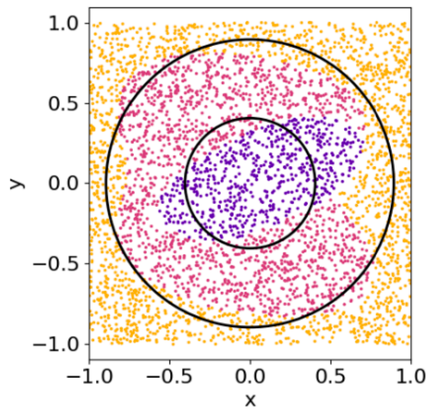
(a) 1 layer



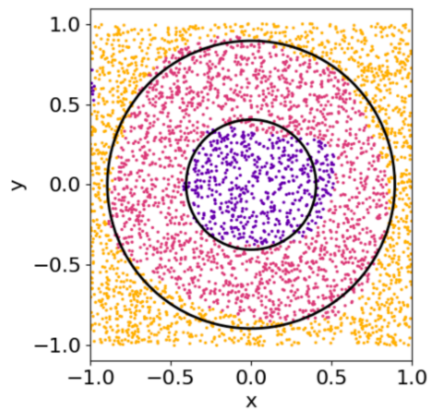
(b) 2 layers



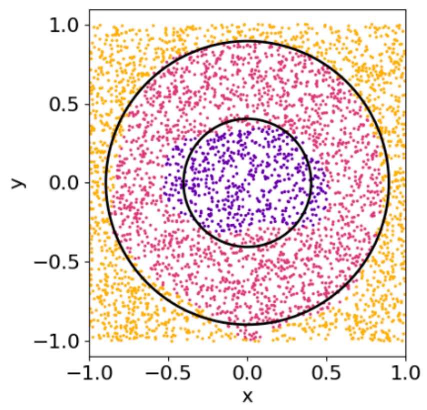
(c) 3 layers



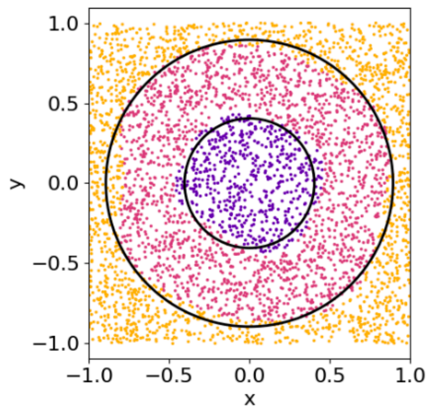
(d) 4 layers



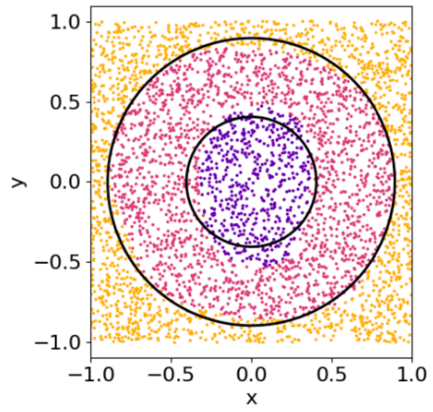
(e) 5 layers



(f) 6 layers



(g) 8 layers

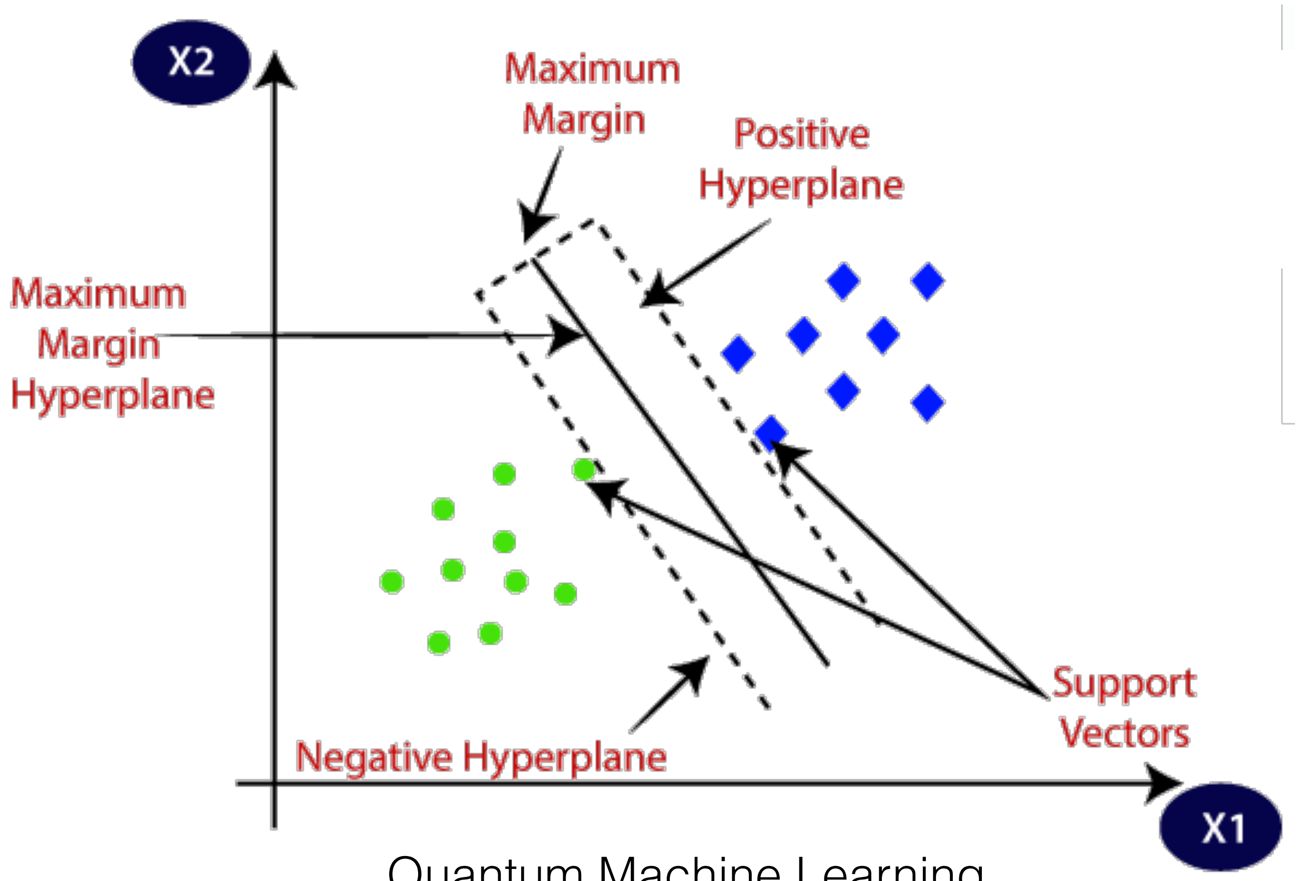


(h) 10 layers

Perez Salinas et al. Quantum 4, 226 (2020)

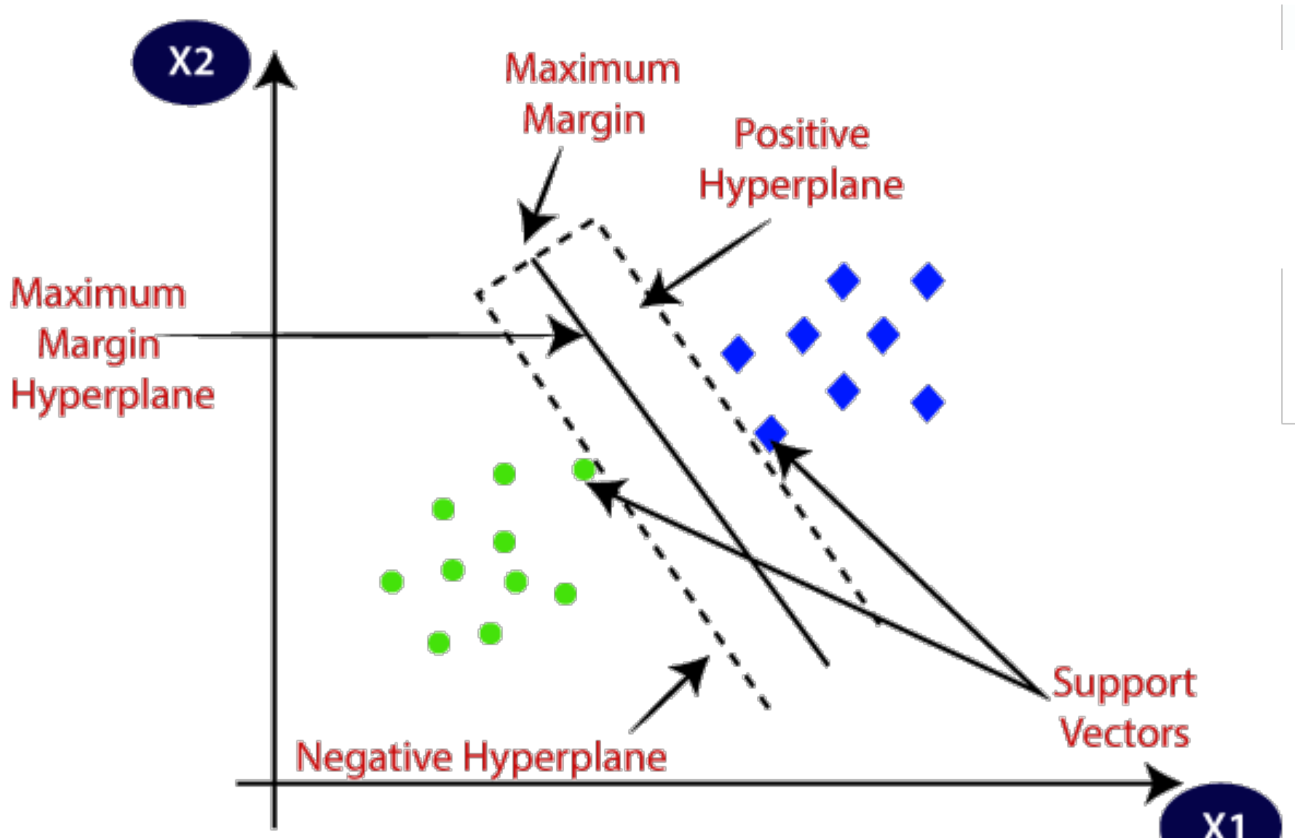
Support Vector Machine

- Linear Classifier
- Constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification,
- The class attributed on the test data points depends on the side of the plane.
- How to choose the line? By maximising the distance between the two clusters.



Support Vector Machine

- We are given a training dataset of points of the form (x_1, y_1) … (x_n, y_n) where the y_i are either 1 or −1
- We want to find the "maximum-margin hyperplane" that divides the group of points for which from the group of points for which



Support Vector Machine

- The SVM problem boils down to the minimisation of the cost function

$$L = \left[\frac{1}{n}\sum_i max(0, 1 - y_i(\beta^T x_i - b))\right] + \lambda |\beta|^2$$

- This reduce to a classical quadratic programming problem

- You can always write a linear system of equation to be solved to exactly find the global minimum of this function

- This is very costly for high dimensional data

Quantum Support Vector Machine

- The problem therefore boils down to finding the solution of a linear system of equations.
- For dense matrices, finding the solution of such a system of N equations is of the order $\mathcal{O}(N^3)$.
- Here, quantum mechanics and quantum computers can help to have a speed-up.
- In this case, the role of the computer is to perform this matrix inversion.
- How? We sketch in the next slide the details of the algorithm introduced by Harrow, Hassidim and Lloyd.

Rebentrost et al, Phys. Rev. Lett. 113, 130503 (2014)

Quantum Support Vector Machine

- A very famous algorithm has been designed to solve linear system of equation:

$$\mathbf{A}x = \mathbf{B}$$

- Where $\mathbf{A}$ is a non sparse matrix with condition number k
- The algorithm find x with $O(log(n)k^2)$ steps
- There is a debate on the speed up due to QRAM for loading data
- Could impure classical support vector machine

Quantum Support Vector Machine

- Schematic representation of the circuit for the HHL algorithm
- Not feasible for NISQ devices

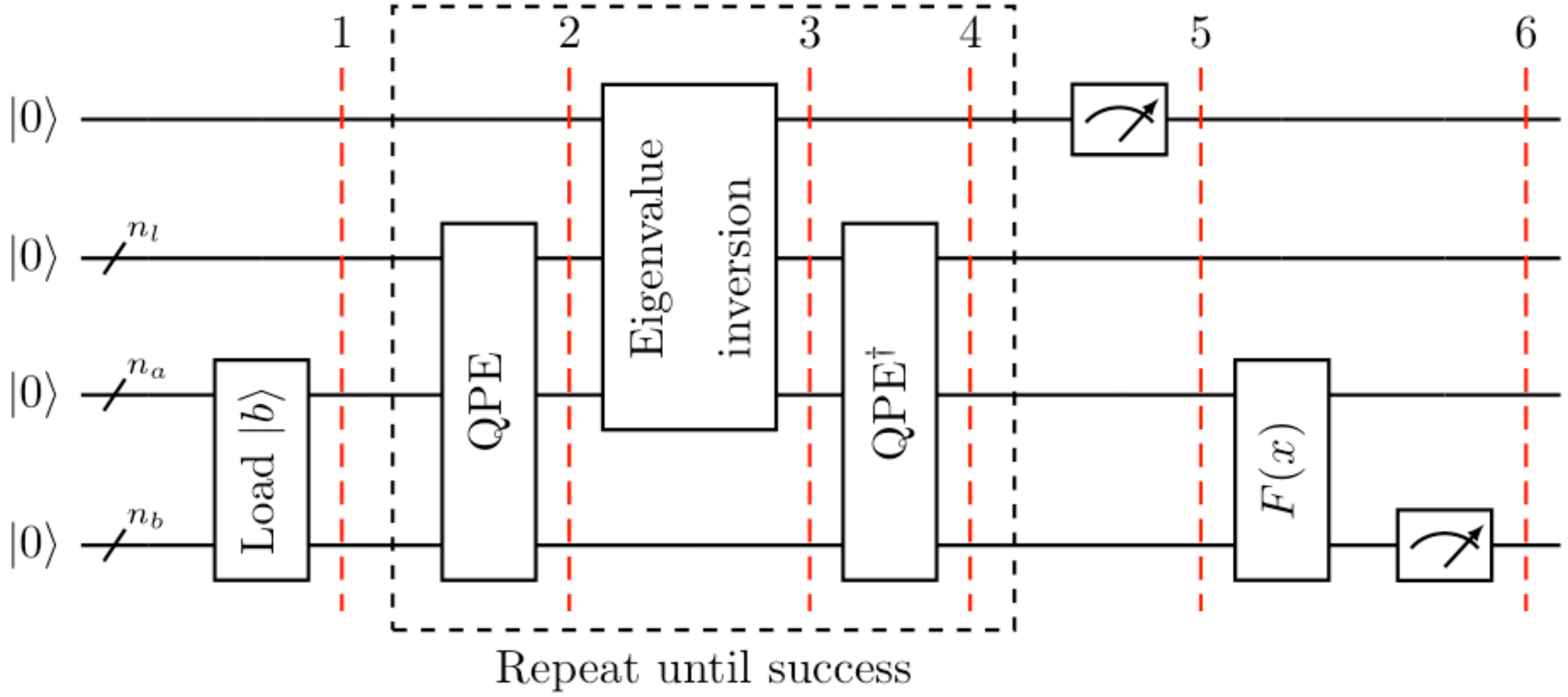
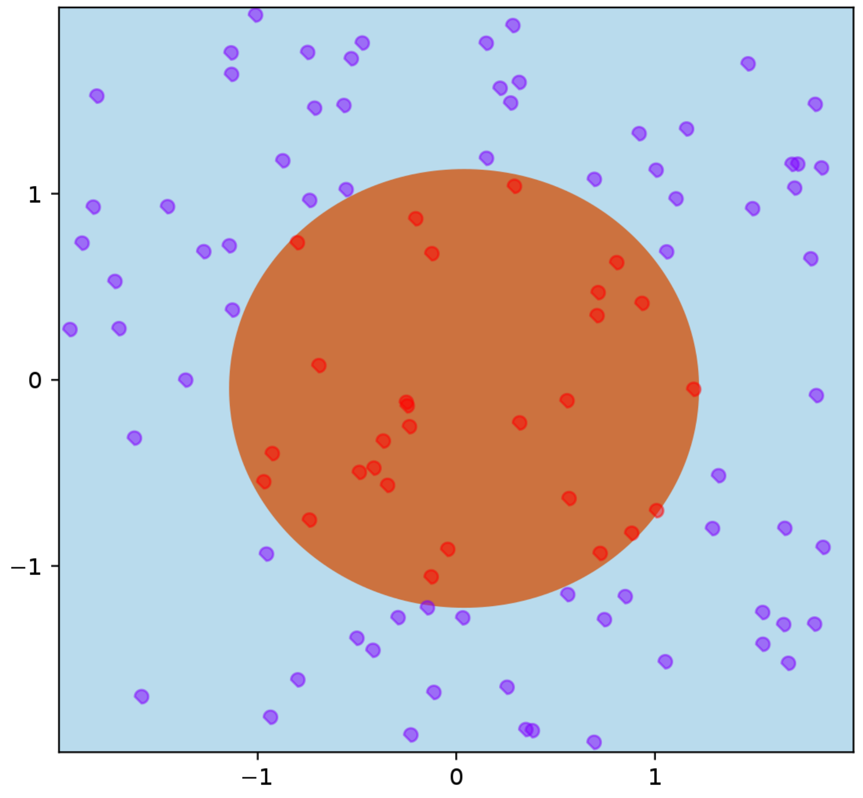
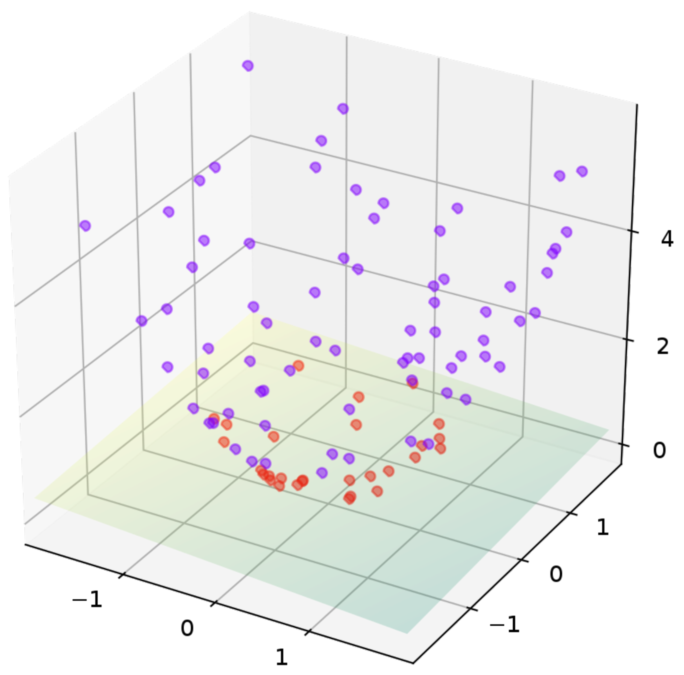


Image from qiskit tutorial

Classical kernel method

- Embed the data in a Higher dimensional system
- Construct a Kernel function that is your “distance ”



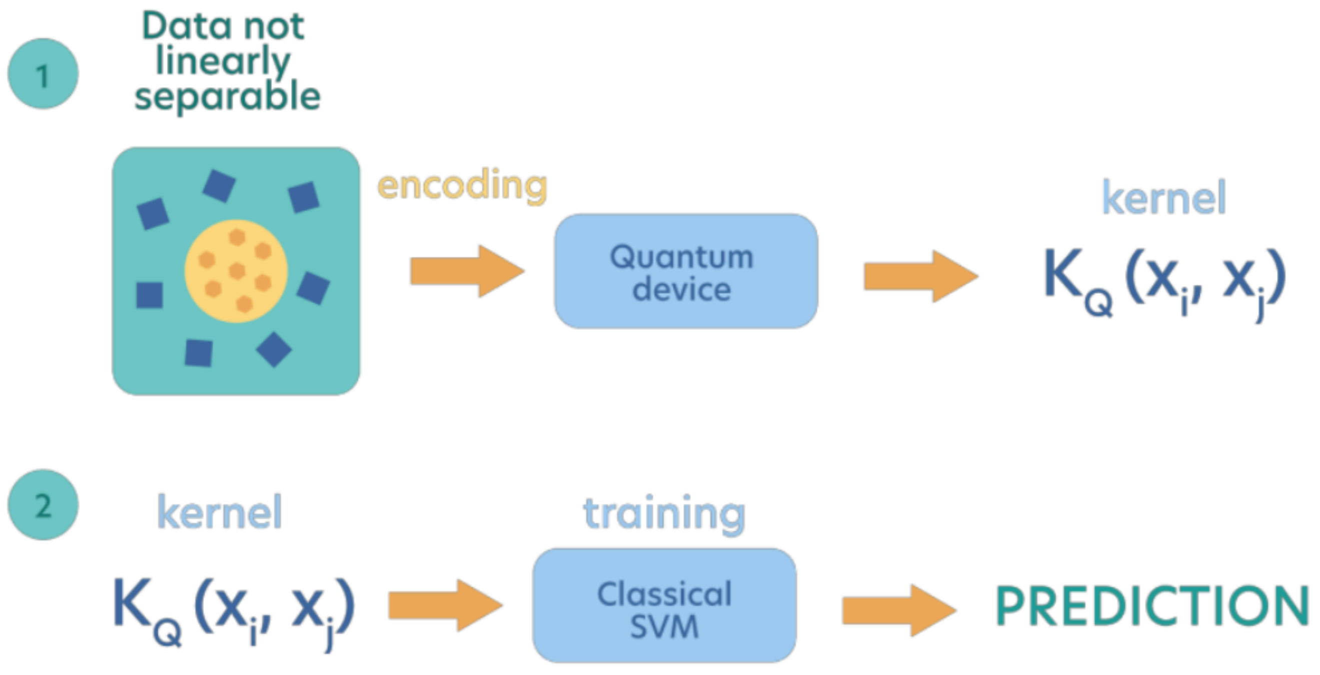


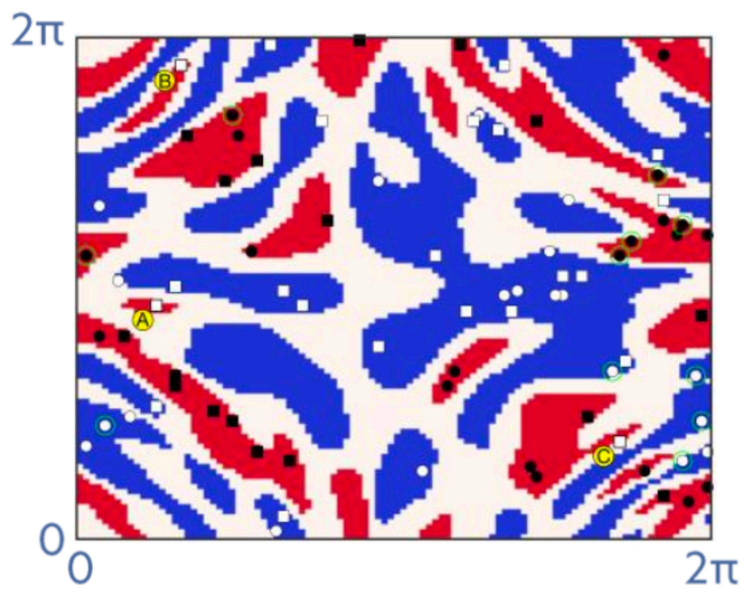
- It is easier to find the hyperplane in higher dimensions

Quantum Support Vector Machine

The data are encoded in a quantum device,
such as a quantum circuit, which computes the kernel.
These kernels are then used in classical SVM.

On the right an example of a dataset to show the capacity of quantum kernels. Blue (red) regions correspond to label 1 (0).





Havlícek et al, Nature 567(7747), 209 (2019)

Quantum SVM with Kernel method

In machine learning, a “kernel” is usually used to refer to the kernel trick, a method of using a linear classifier to solve a non-linear problem. It entails transforming linearly inseparable data like to linearly separable ones.

The image space of a quantum computer is very large. We can leverage this feature

Encode the classical data in a quantum computer:

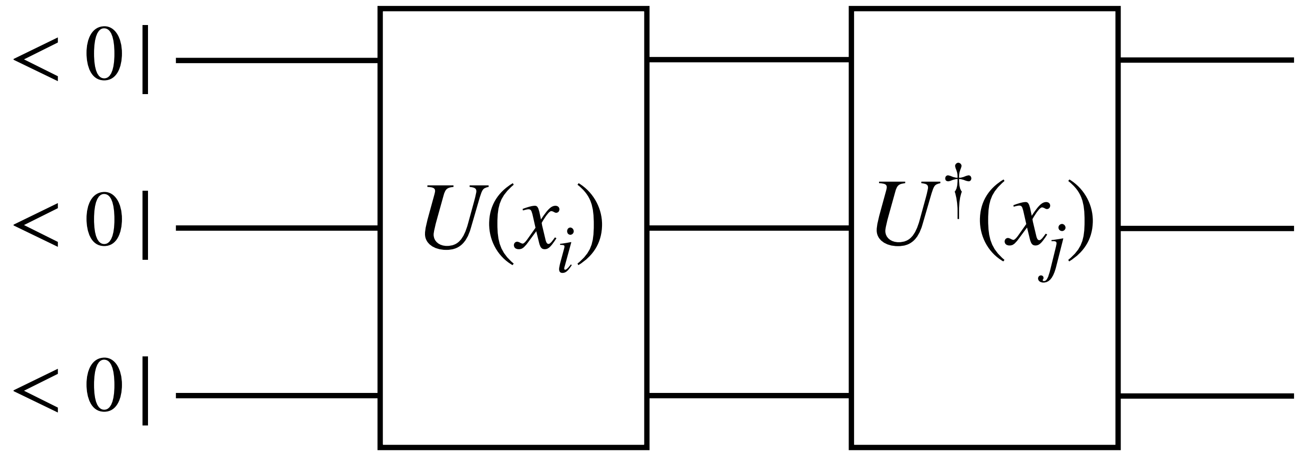
$$x_i \rightarrow |\Phi(\theta, x_i) > \qquad |\Phi(\theta, x_i) > = U(\theta, x_i)|0 >$$

The kernel is defined as:

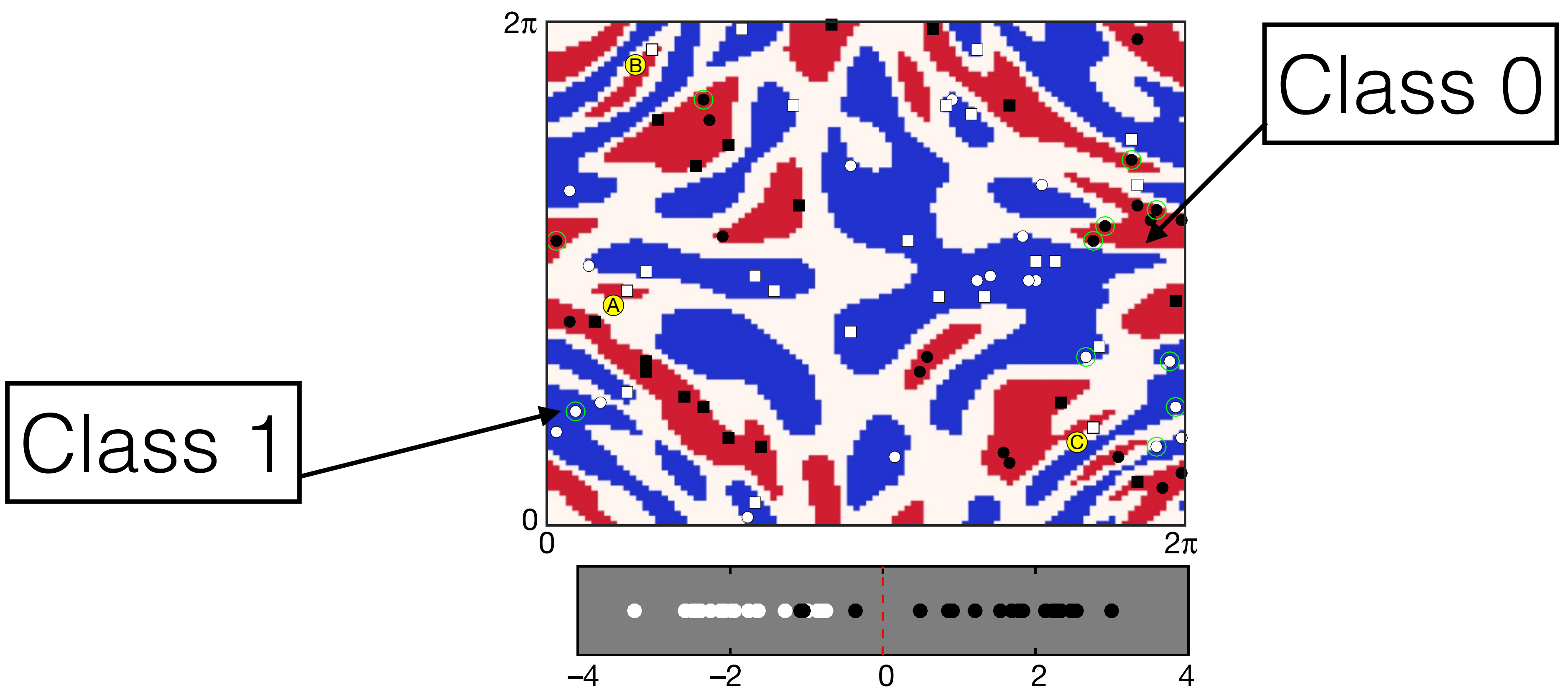
$$K(x_i, x_j) = < \Phi(x_i)|\Phi(x_j) >$$

Quantum SVM with Kernel method

We can use the reverse trick to compute the kernel:



$$= \quad U^\dagger(\theta, x_j)U(\theta, x_i)\,|0>$$



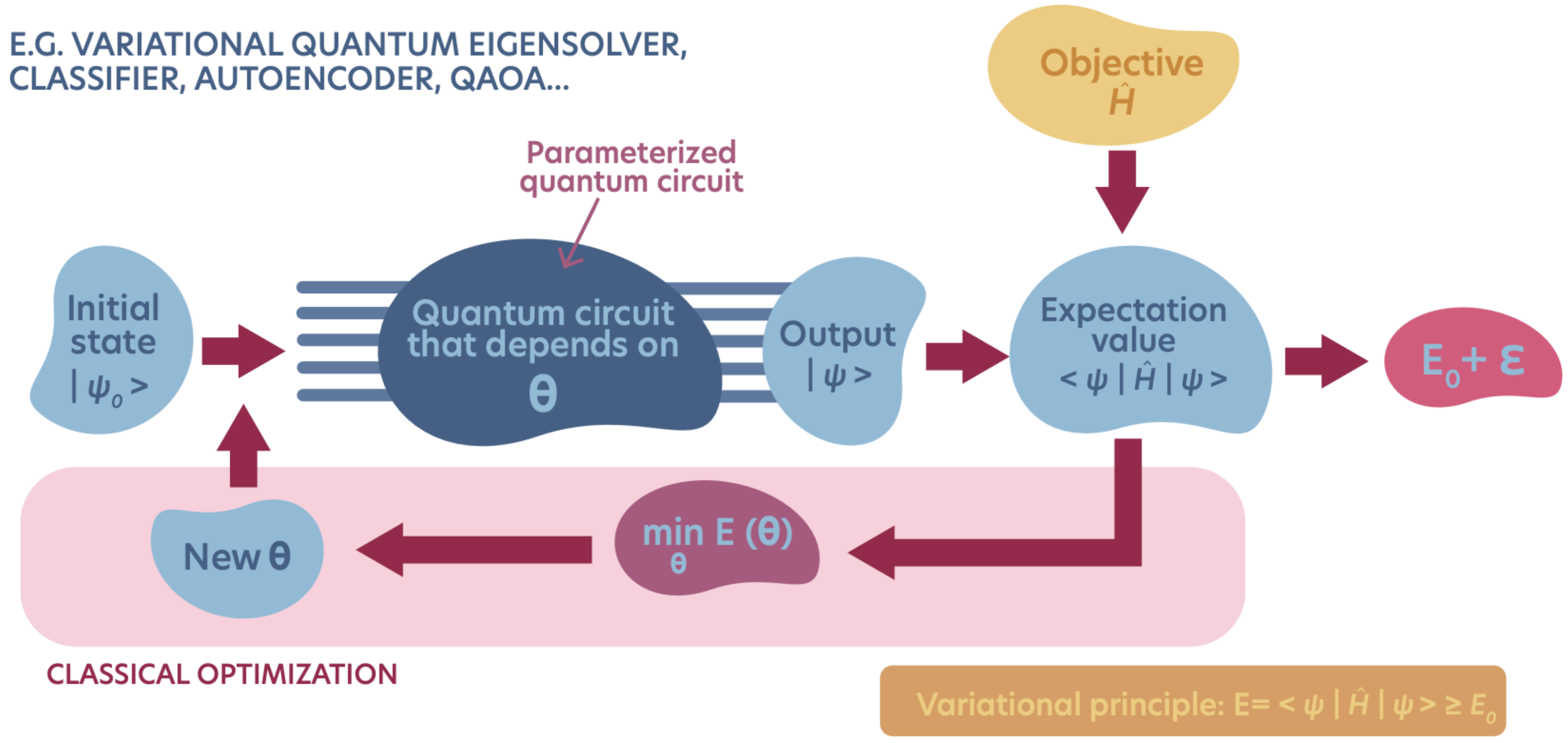
Havlícek et al, Nature 567(7747), 209 (2019)

QSVM vs Quantum enhanced SVM

Quantum Support vector machine		Quantum enhanced Support vector machine
Data: classical	$=$	Data: classical
Role of the quantum computer: Perform the matrix inversion	$\neq$	Role of the quantum computer: Evaluation of the kernel

Variational quantum simulation

Hybrid quantum classical approach



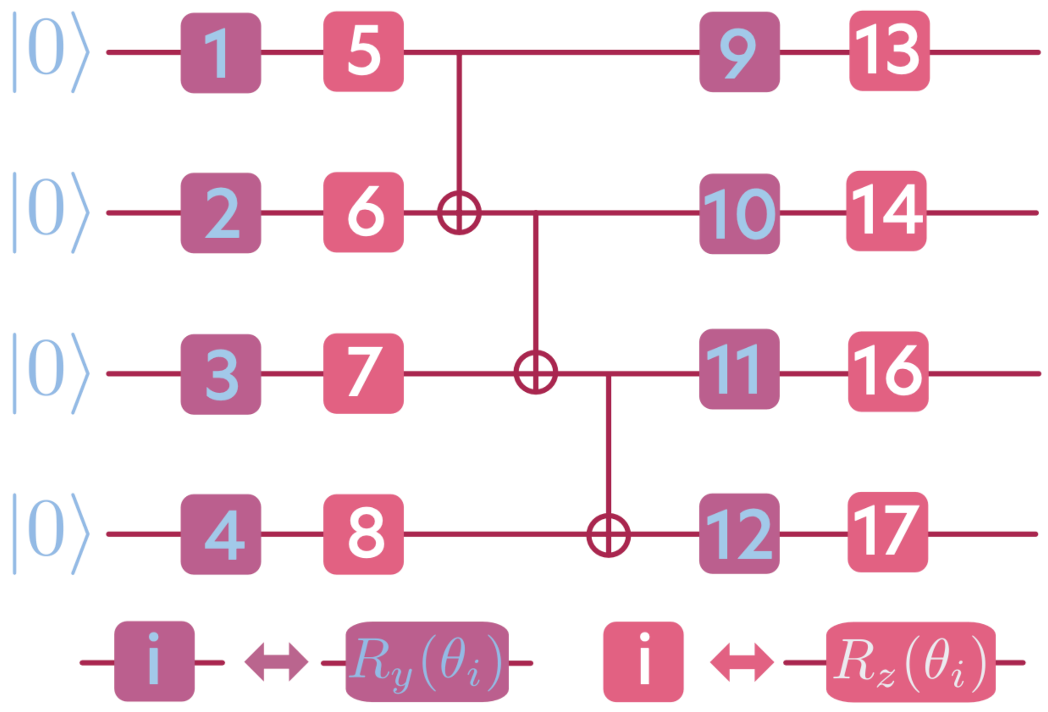
E.G. VARIATIONAL QUANTUM EIGENSOLVER, CLASSIFIER, AUTOENCODER, QAOA...
Objective $\hat{H}$
Parameterized quantum circuit
Initial state $|\psi_0>$
Quantum circuit that depends on θ
Output $|\psi>$
Expectation value $<\psi|\hat{H}|\psi>$
$E_0+\varepsilon$
New θ
$\min_\theta E(\theta)$
CLASSICAL OPTIMIZATION
Variational principle: $E=<\psi|\hat{H}|\psi>\geq E_0$

Variational quantum simulation

- Goal: Minimise the energy of a given Hamiltonian
- How: using the variational principle:

$$E = < \psi(\theta) | H | \psi(\theta) >$$

- We want to find the values of θ^* that minimise the energy



Peruzzo et al, nat commun 5, 4213 (2014)

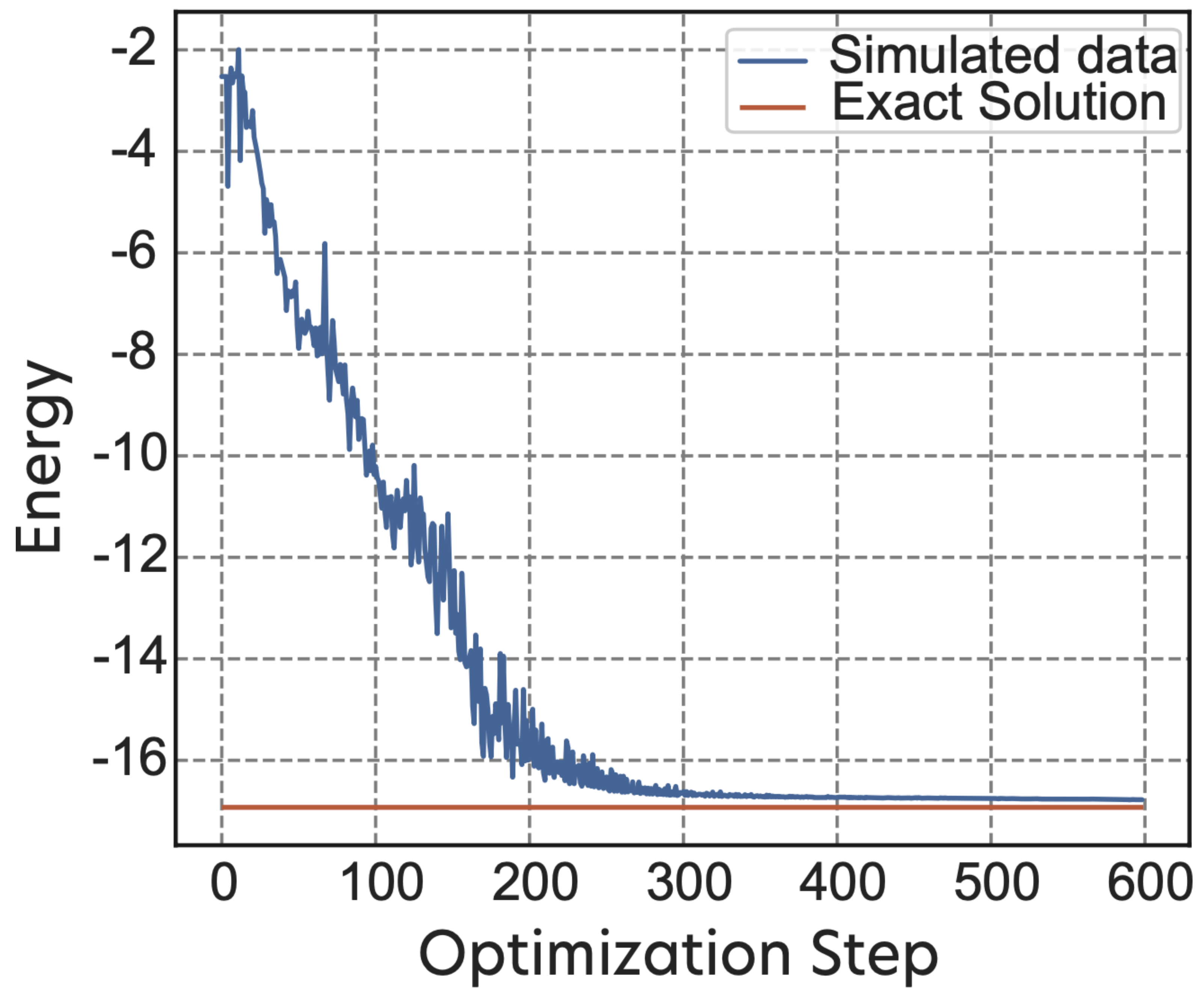
Variational quantum simulation

- We consider the Heisenberg Hamiltonian:

$$\hat{H} = \sum_{i=1}^{3} J_1 \sigma_i^x \sigma_{i+1}^x + J_2 \sigma_i^y \sigma_{i+1}^y + J_2 \sigma_i^z \sigma_{i+1}^z + \sum_{j=1}^{4} h_1 \sigma_i^x + h_2 \sigma_i^y + h_3 \sigma_i^z \,.$$

- Feedback from quantum and classical device
- Training done with gradient descent
- Gradient computed with parameter shift rule

Variational quantum simulation



Energy of the variational state as a function of the optimisation step

Classical optimization of cost function

Highly non linear non-convex optimisation problem

- The circuit ansatz is: $|\Psi(\theta)\rangle = U(\theta)|\Psi_0\rangle$
- The cost function can be written as: $C(\theta) = \langle\Psi(\theta)|H|\Psi(\theta)\rangle$
- If the dependence of θ is through rotations:
- $b_M^{(N)}$ is a set of 3^N unknown coefficients

$$C(\theta) = b_M^{(N)} \left[\bigotimes_{i=1}^{N} \left(\cos\theta_i, \sin\theta_i, 1\right) \right]^T$$

Parameter shift rule

- How do we perform the training?
- Minimisation of the loss function
- First approach: Gradient Free methods (Nelder Mead, etc…)
- Second Approach: Gradient descent.
- How do we compute the gradient?
- We do not have back-propagation (yet?) on quantum circuit
- Parameter shift rule!

Parameter shift rule

We want to compute the expectation value of this formula:

$$f(x,\theta) = < 0 | U^\dagger(\theta) B(x) U(\theta) | 0 >$$

$$M(\theta) = U^\dagger(\theta) B(x) U(\theta)$$

$$\partial_\theta f(x,\theta) = < 0 | \partial_\theta M(\theta) | 0 >$$

$$\partial_\theta M_\theta = \partial_\theta U^\dagger(\theta) B(x) U(\theta) + U^\dagger(\theta) B(x) \partial_\theta U(\theta)$$

Let us assume that the dependence on the parameters is:

$$U(\theta) = e^{-i\frac{\theta}{2}\sigma_i} \qquad \partial_\theta U(\theta) = -\frac{i}{2}\sigma_i e^{-i\frac{\theta}{2}\sigma_i}$$

Parameter shift rule

This imply that:

$$\partial_\theta M_\theta = \frac{i}{2}\sigma_i U^\dagger(\theta)B(x)U(\theta) - \frac{i}{2}U^\dagger(\theta)B(x)\sigma_i U(\theta)$$

$$\partial_\theta M_\theta = \frac{i}{2}U^\dagger(\theta)\left[B(x), \sigma_i\right] U(\theta)$$

For Pauli operators, one can show that:

$$\partial_\theta M_\theta = \frac{1}{2}U^\dagger(\theta + \frac{\pi}{2})B(x)U(\theta + \frac{\pi}{2}) - \frac{1}{2}U^\dagger(\theta - \frac{\pi}{2})B(x)U(\theta - \frac{\pi}{2})$$

Therefore:

$$\partial_\theta f(x, \theta) = < 0 | M(\theta + \frac{\pi}{2}) | 0 > - < 0 | M(\theta - \frac{\pi}{2}) | 0 >$$

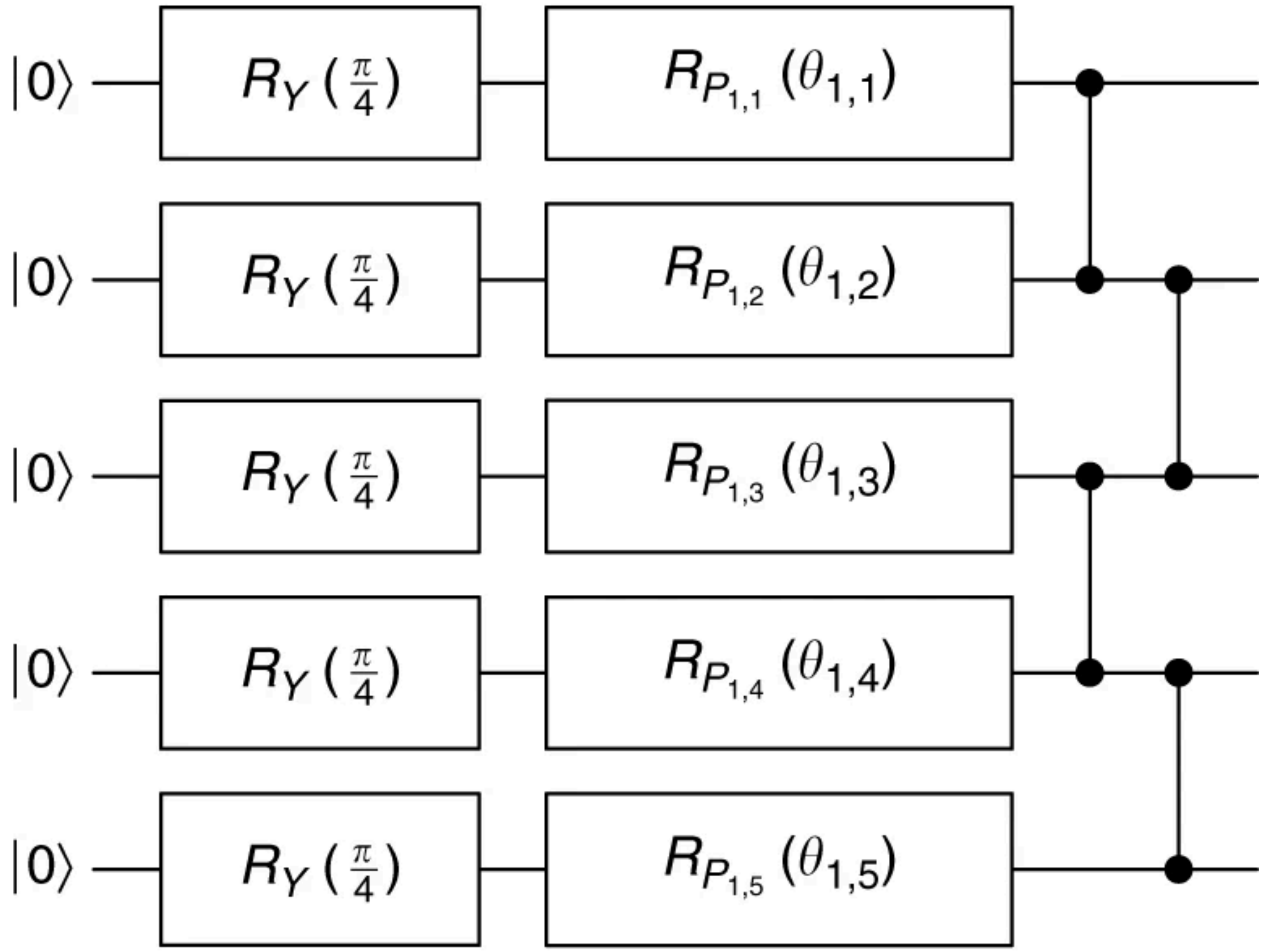
Training a quantum neural network

Consider the circuit:

$$C = < \psi_0 \,|\, U^\dagger(\theta)\, H\, U(\theta) \,|\, \psi_0 >$$

The circuit is sufficiently random

It approach a unitary 2-design



$|0\rangle$
$R_Y(\frac{\pi}{4})$
$R_{P_{1,1}}(\theta_{1,1})$
$R_{P_{1,2}}(\theta_{1,2})$
$R_{P_{1,3}}(\theta_{1,3})$
$R_{P_{1,4}}(\theta_{1,4})$
$R_{P_{1,5}}(\theta_{1,5})$

Training a quantum neural network

- For Harr random unitaries on n qubits:

$$\partial_\theta(C) = 0$$

- The Harr unitaries means that the circuit prepare states uniformly in the $2^n - 1$ dimensional space
- The gradient of the cost function is always
- Lipschitz continuous
- Easy to demonstrate from the analytical form of C

Training a quantum neural network

- The problem is even worse
- For a circuit is of sufficient depth
- The variance of the gradient:

$$Var[\partial_\theta(C)] \propto 2^{-n}$$

- Zero variance means difficulty into training
- This limit is approached when approaching the unitary two design

Training a quantum neural network

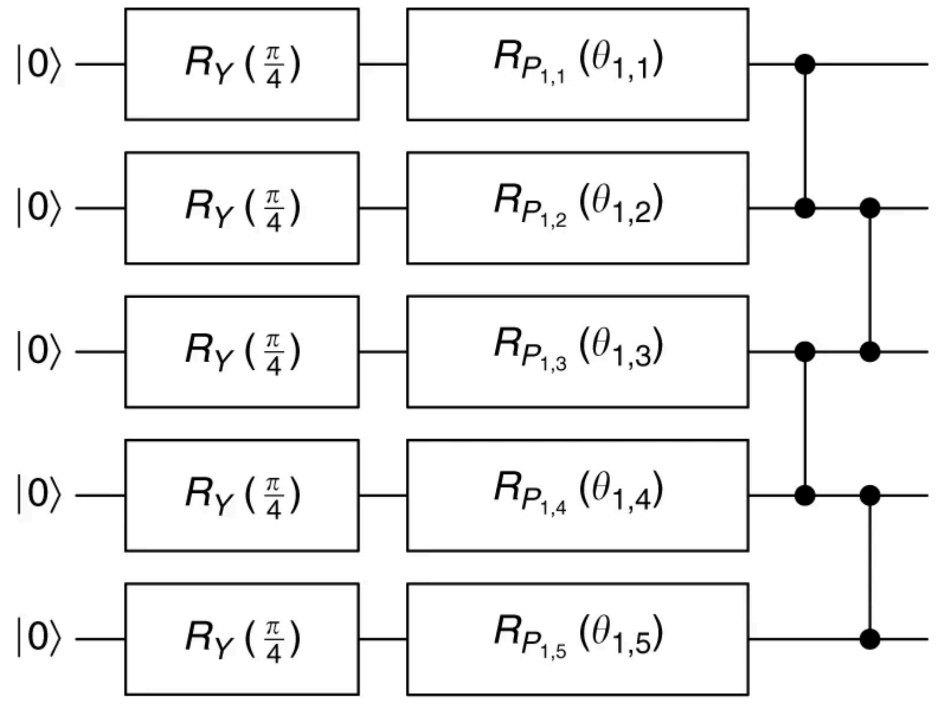
Numerica example:

The hamiltonian is:

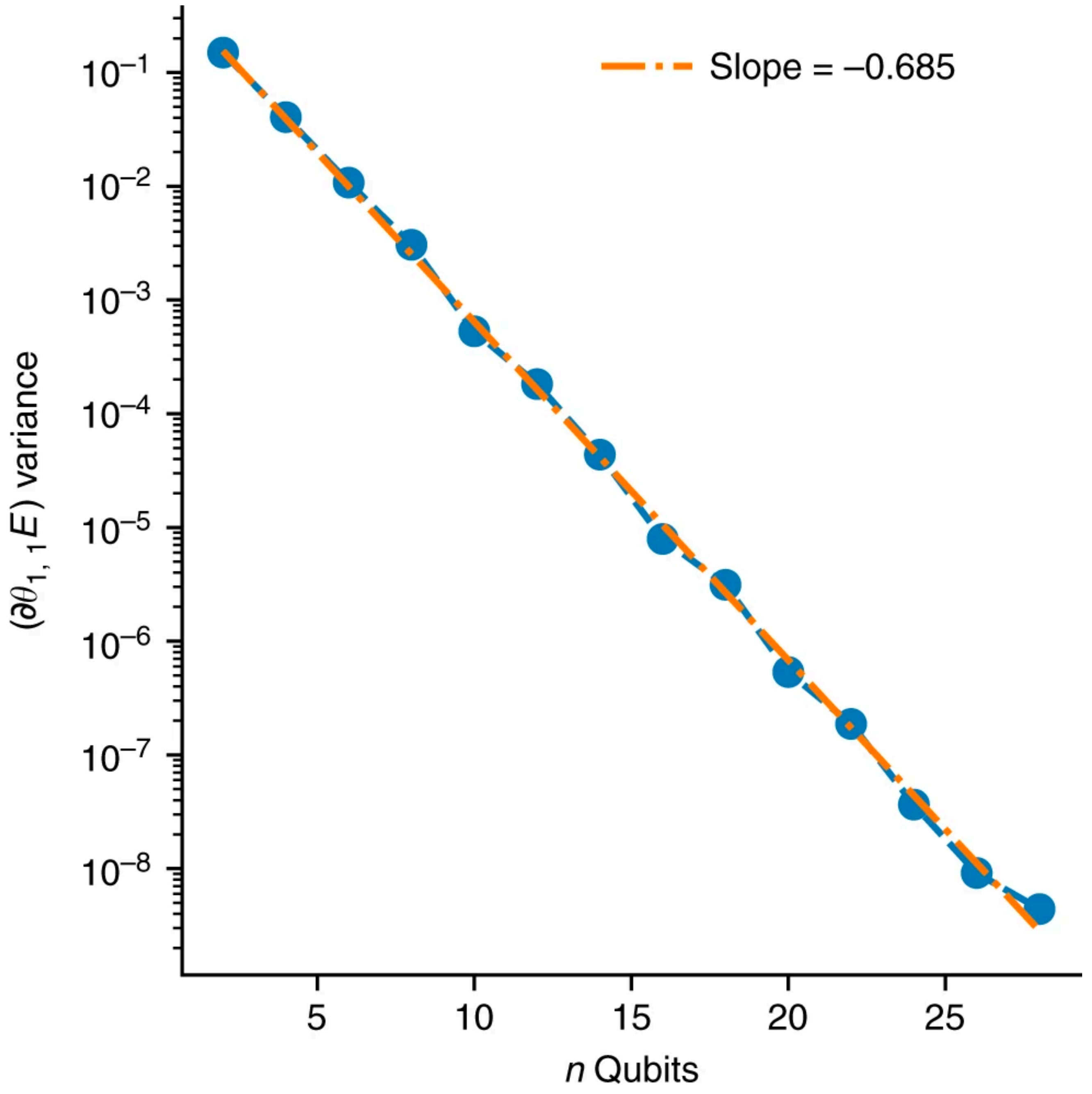
$$H = Z_1 Z_2$$

The number on layers is:

$$L \sim 10n$$



$|0\rangle$
$R_Y\left(\frac{\pi}{4}\right)$
$R_{P_{1,1}}(\theta_{1,1})$
$R_{P_{1,2}}(\theta_{1,2})$
$R_{P_{1,3}}(\theta_{1,3})$
$R_{P_{1,4}}(\theta_{1,4})$
$R_{P_{1,5}}(\theta_{1,5})$



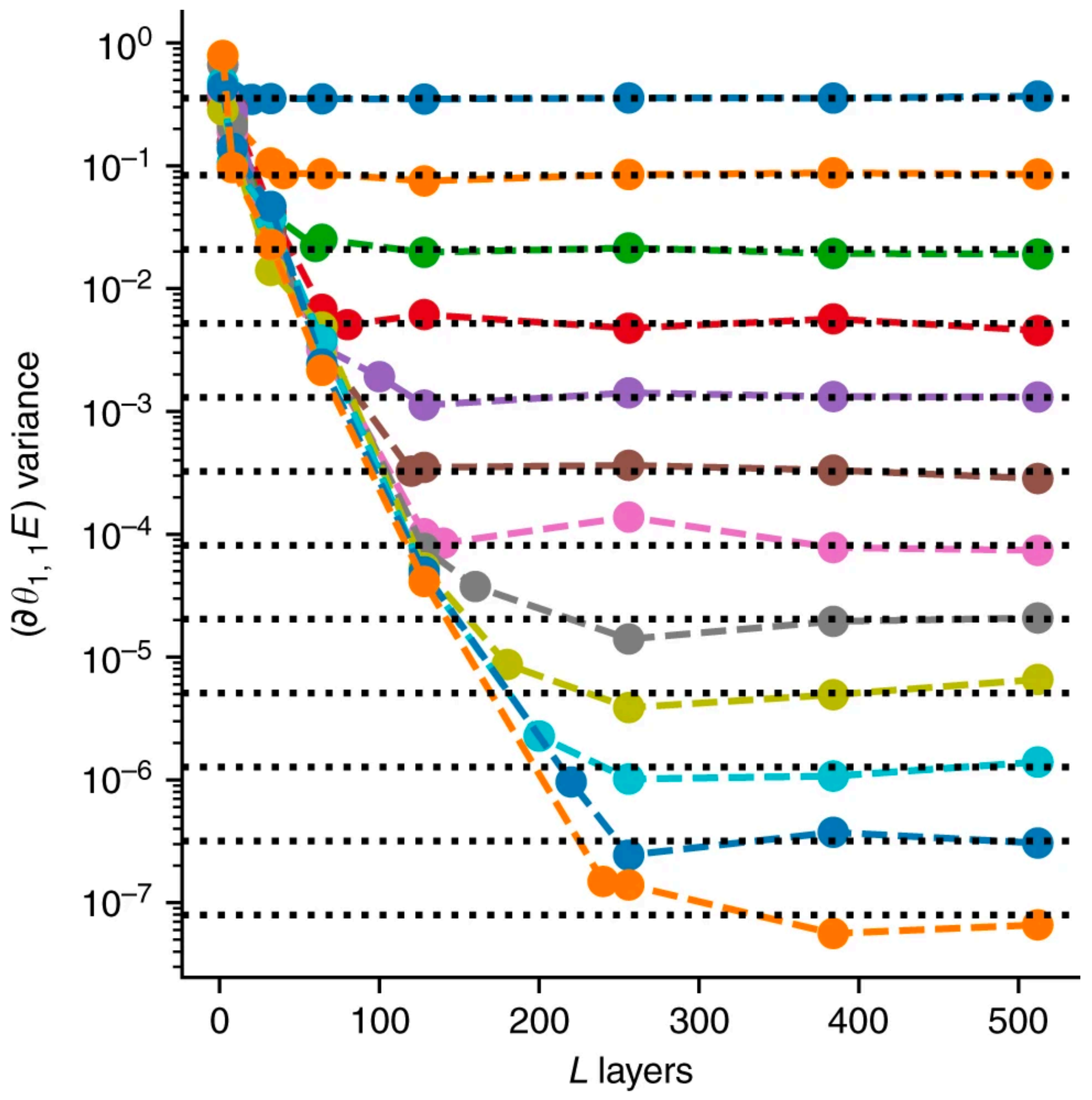
*Nature Communications volume **9**, Article number: 4812 (2018)*

Training a quantum neural network

- When we increase the number of qubits from 2 to 24 the problem become persist

- Large number of layers to approach unitary two design

- Issue in trying deep quantum neural network



Nature Communications volume 9, Article number: 4812 (2018)

References

Quantum machine learning (review) arXiv

Noisy intermediate-scale quantum (NISQ) algorithms

Recent advances for quantum classifiers arXiv

Variational quantum algorithms arXiv

Qiskit tutorial on quantum machine learning

Book on Machine learning for quantum physics arxiv.org/2204.04198

Comparison with classical NN

- The gradient in a classical deep neural network can vanish exponentially in the number of layers
- in a quantum circuit the gradient may vanish exponentially in the number of qubits
- The gradient saturates to an exponential in the number of qubits because the output state is normalized.
- For QNN, the problem is not solved but mitigated in some cases

Nature Communications volume **9**, Article number: 4812 (2018)