INTRODUTION TO CLASSICAL MACHINE LEARNING AND NEURAL NETWORKS

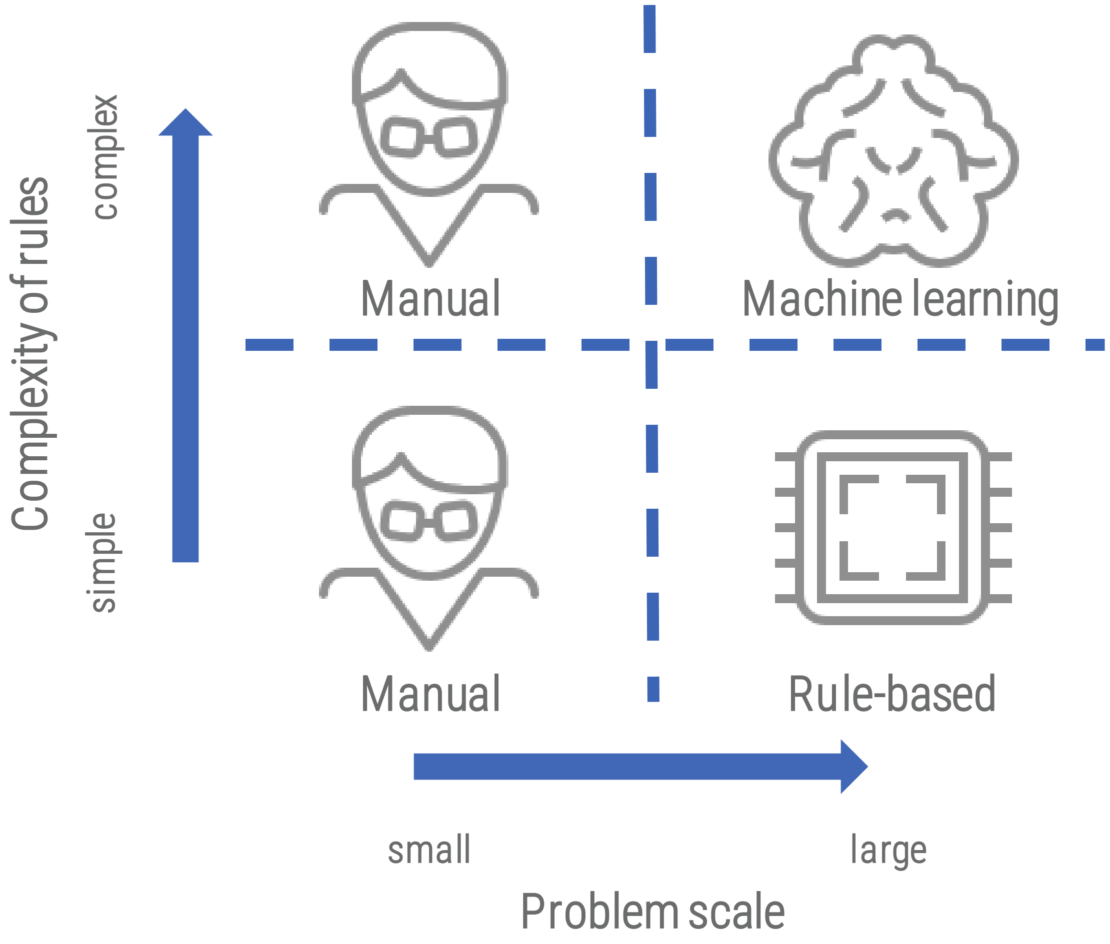
Marco Maronese – PhD student @ University of Bologna

CINECA QUANTUM COMPUTING LAB

INDEX

Machine Learning Introduction

Concept

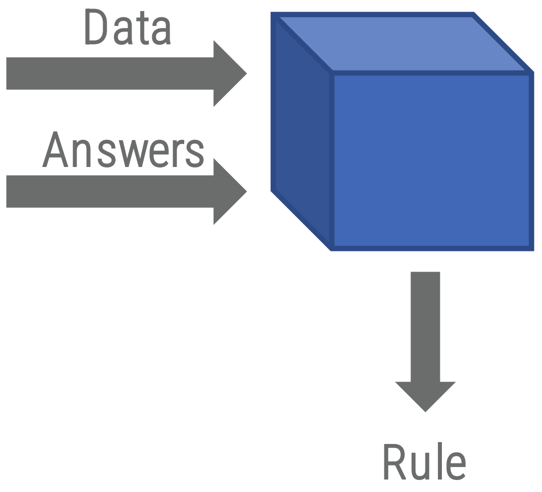


Complexity of rules
complex
simple
Manual
Machine learning
Manual
Rule-based
small
large
Problem scale

MACHINE LEARNING

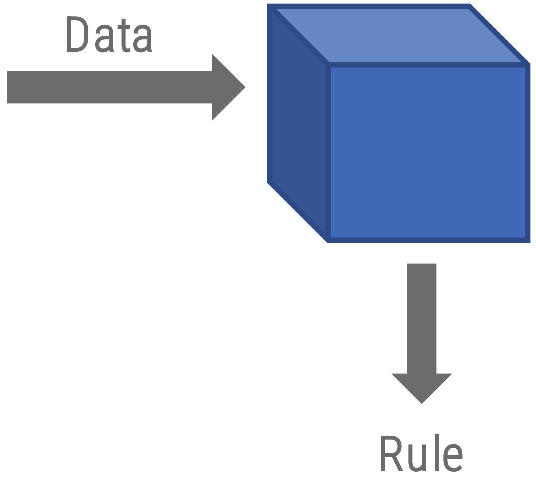
Branches of machine learning

Supervised



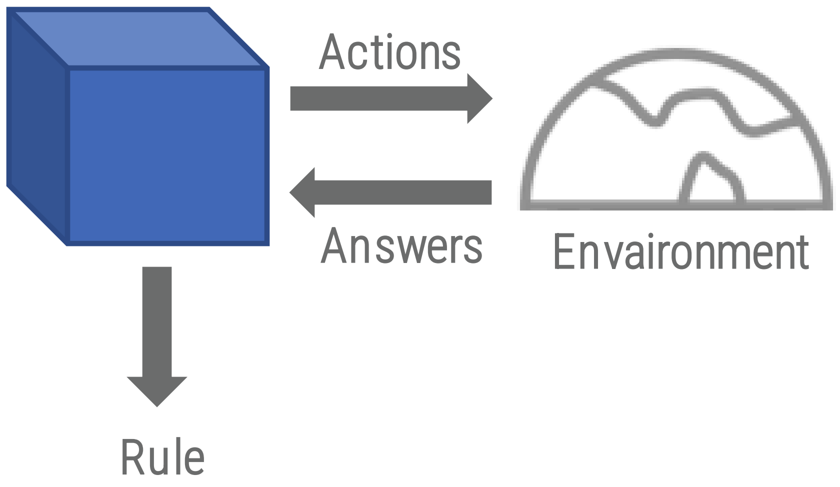
Labeled data

Unsupervised



Correlations

Reinforced



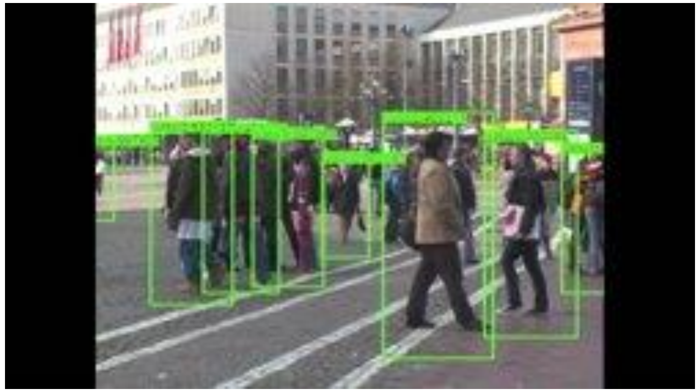
Strategy

MACHINE LEARNING

Branches of machine learning

Supervised

- Classification
- Regression

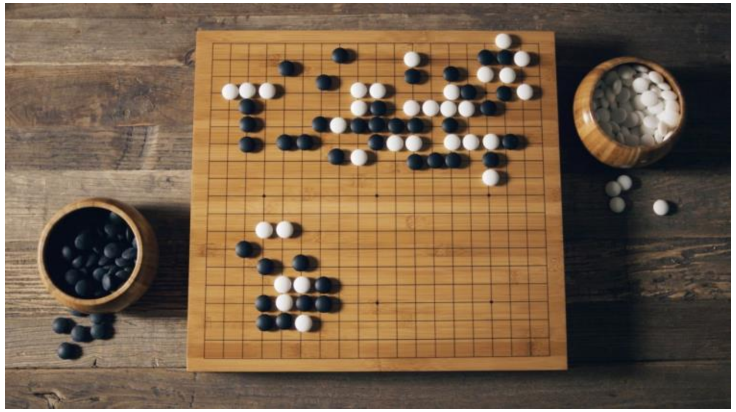


Unsupervised

- Clustering
- Data Generation

Reinforced

- Game theory
- Robotics



SUPERVISED LEARNING

APPLICATIONS

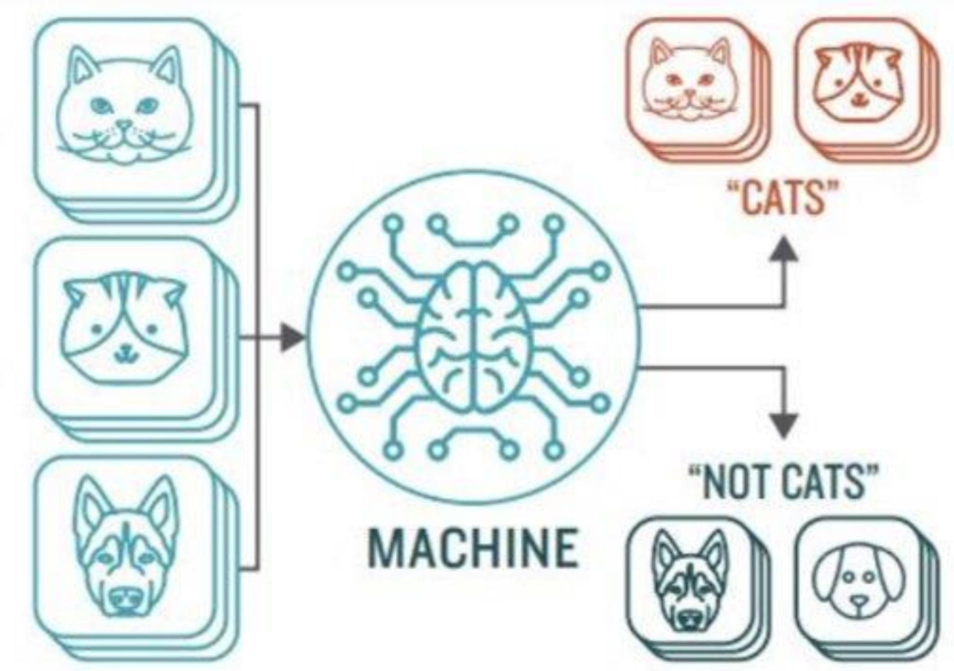
- Identification of faces in images
- Identification of pedestrian
- Classification of texts
- Bioinformatics research
- Classification of remotely sensed images

SUPERVISED LEARNING

Classification Problem

Classification Rule

$$y = \begin{cases} 1, & \text{if } f(x, \theta) > \text{threshold} \\ 0, & \text{if } f(x, \theta) \leq \text{threshold} \end{cases}$$



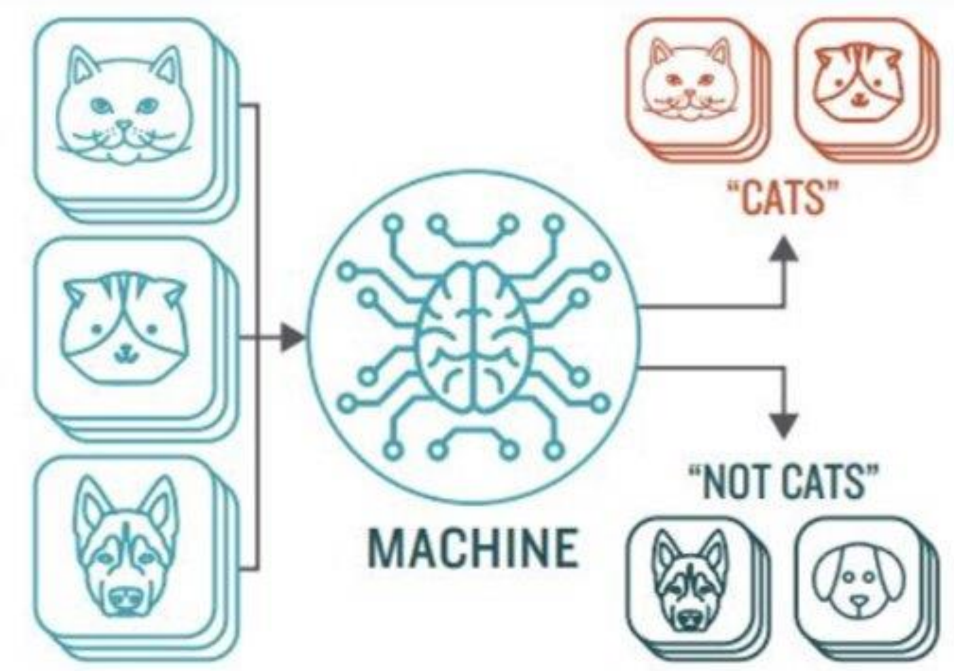
Classification Problem

Classification Rule

Learning parameters

$$y = \begin{cases} 1, & \text{if } f(x, \theta) > \text{threshold} \\ 0, & \text{if } \underline{f(x, \theta)} \leq \text{threshold} \end{cases}$$

Model



SUPERVISED LEARNING

Classification Problem

Classification Rule

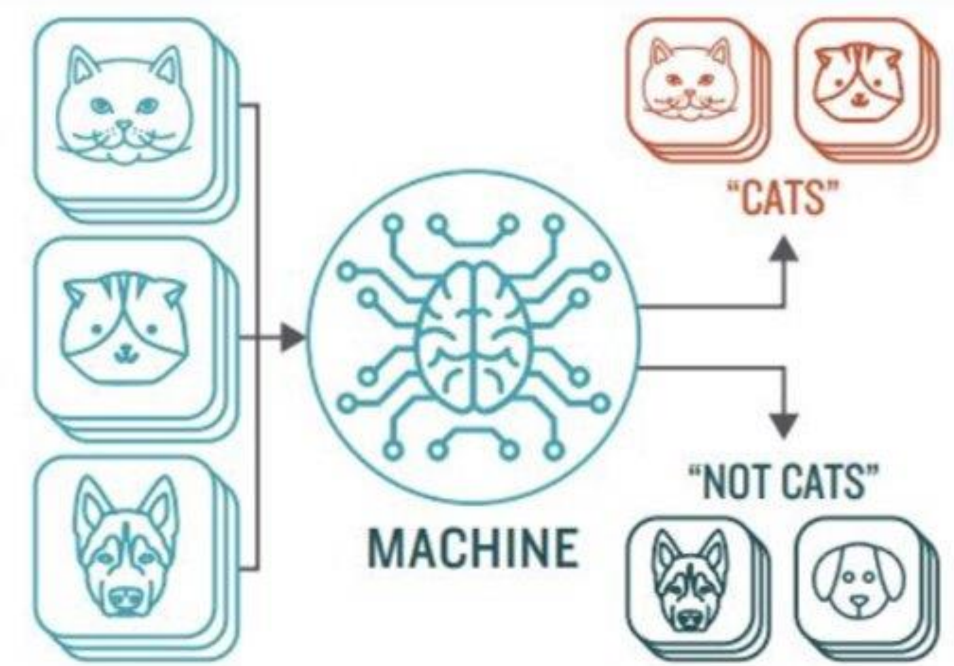
Learning parameters

$$y = \begin{cases} 1, & \text{if } f(x, \theta) > \text{threshold} \\ 0, & \text{if } f(x, \theta) \leq \text{threshold} \end{cases}$$

Model

Support Vector Machine Rule

$$y = \text{sign}\left(\vec{\varphi}(\vec{x}) \cdot \vec{w} + b\right)$$



Classification Problem

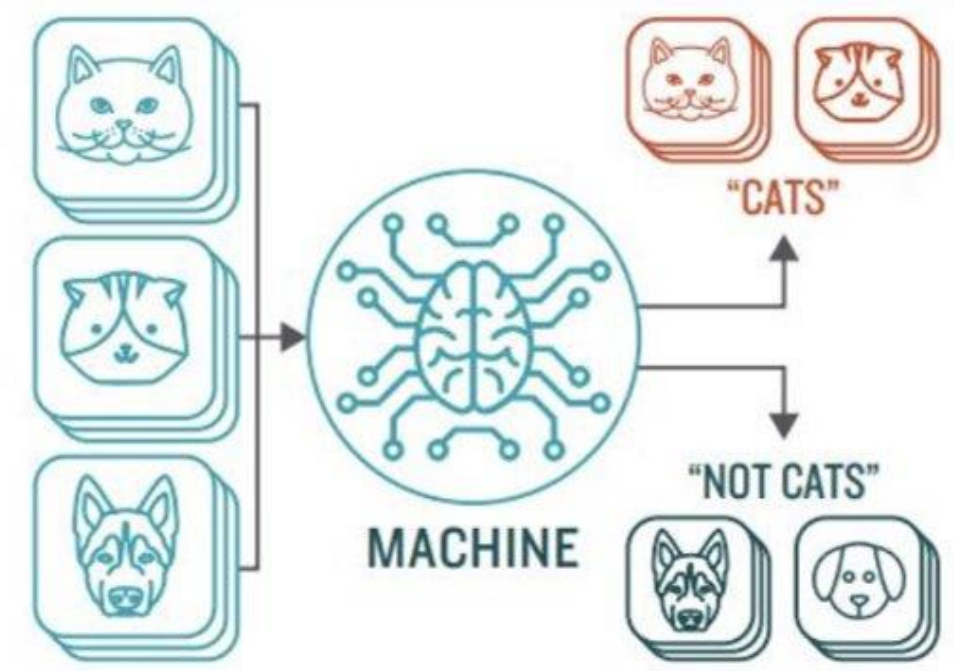
Classification Rule

Learning parameters

$$y = \begin{cases} 1, & \text{if } f(x, \theta) > \text{threshold} \\ 0, & \text{if } \underbrace{f(x, \theta)}_{\text{Model}} \leq \text{threshold} \end{cases}$$

Support Vector Machine Rule

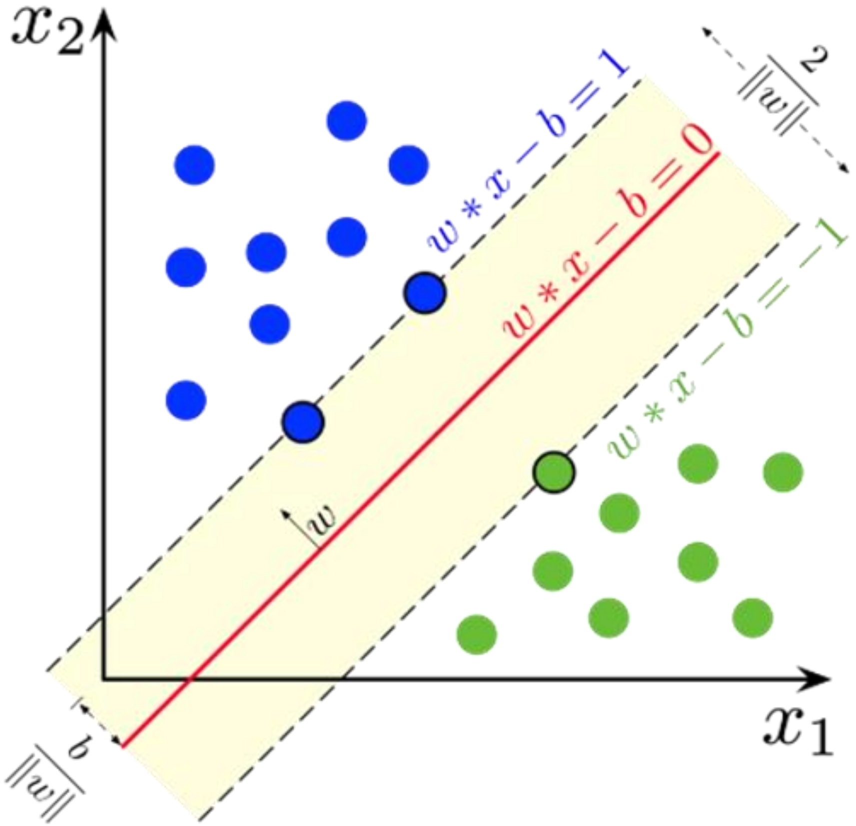
$$y = \operatorname{sign}\left(\underbrace{\vec{\varphi}(\vec{x})}_{\text{Non linear feature map}} \cdot \vec{w} + b \right)$$



Classical Support Vector Machine (SVM)

SUPPORT VECTOR MACHINE

Overview



Margin width

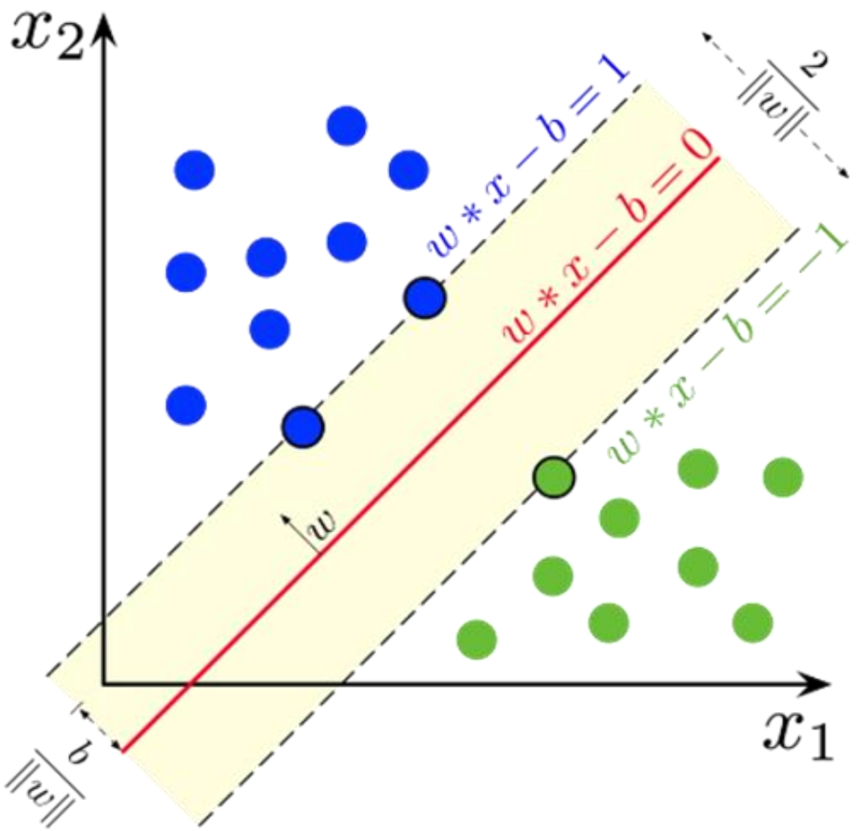
$$\frac{b+1}{\|\boldsymbol{w}\|} - \frac{b-1}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

Support vector

$\boldsymbol{x}_i$ Such that $y_i\left(\boldsymbol{x}_i \cdot \boldsymbol{w} - b\right) = 1$

SUPPORT VECTOR MACHINE

Overview



Objective:

minimize $\|\boldsymbol{w}\|$

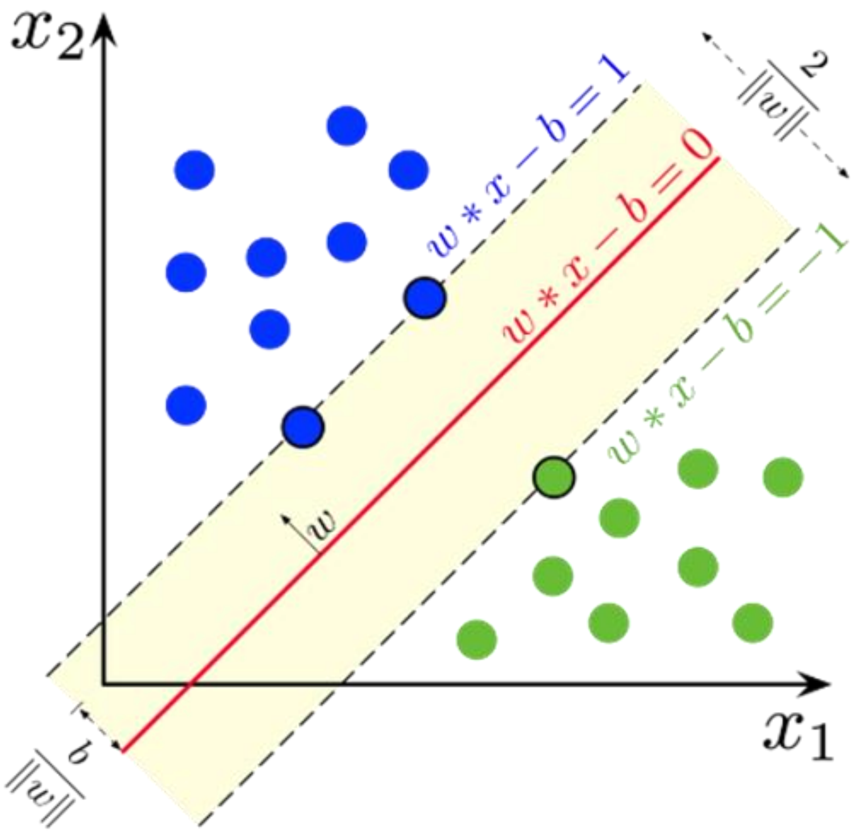
Constraints

$$\boldsymbol{x}_i \cdot \boldsymbol{w} - b \leq -1 \quad \text{for} \quad y_i = -1$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} - b \geq 1 \quad \text{for} \quad y_i = 1$$

SUPPORT VECTOR MACHINE

Overview



Objective:

minimize $\|\boldsymbol{w}\|$

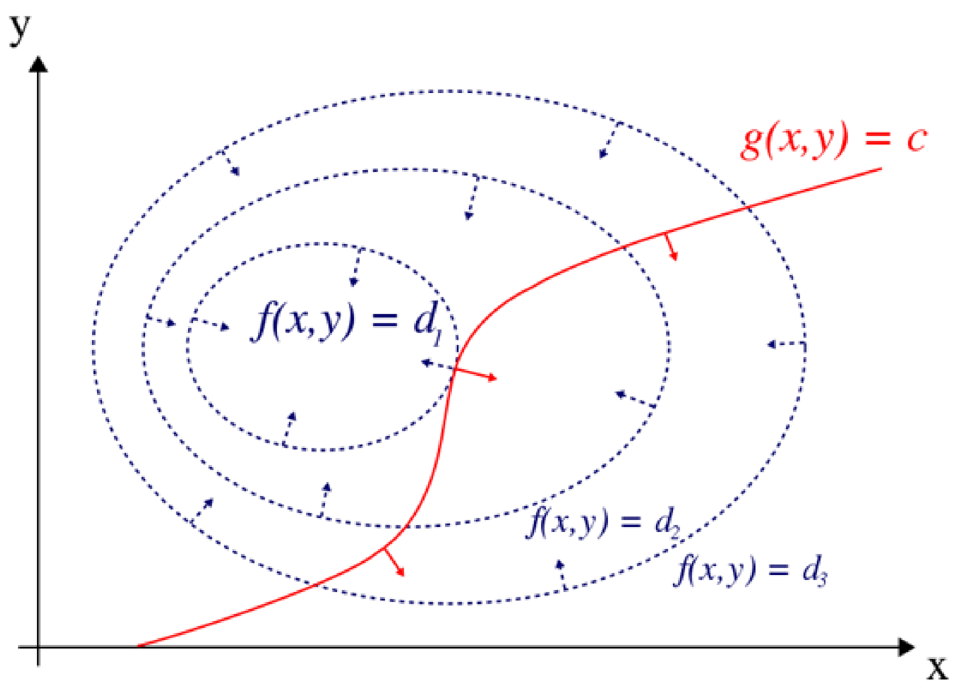
Constraints

$$\boxed{y_i\left(\boldsymbol{x}_i \cdot \boldsymbol{w} - b\right) - 1 \geq 0 \quad \forall i}$$

SUPPORT VECTOR MACHINE

Optimization

$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\boldsymbol{x}_i \cdot \boldsymbol{w} + b \right) - 1 \right]$$

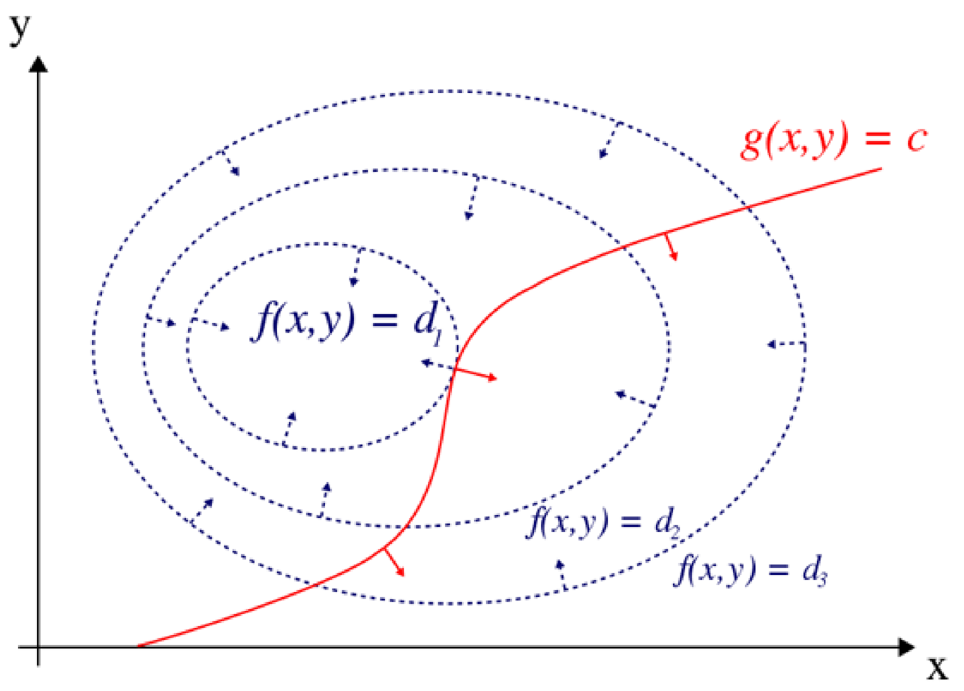


SUPPORT VECTOR MACHINE

Optimization

$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\boldsymbol{x}_i \cdot \boldsymbol{w} + b \right) - 1 \right]$$

Lagrange multipliers



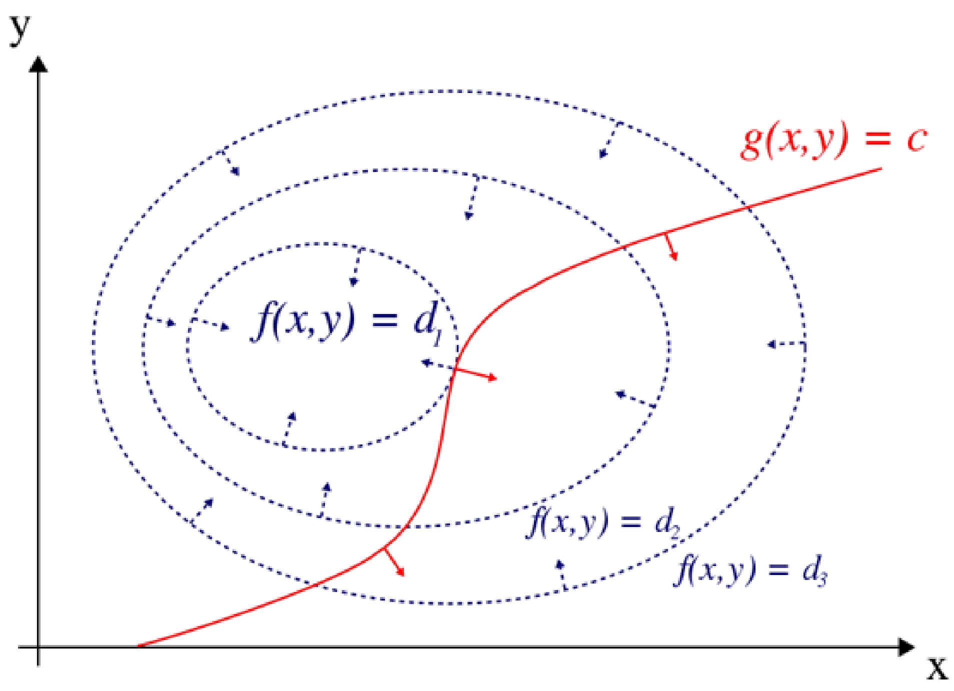
Optimization

$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\boldsymbol{x}_i \cdot \boldsymbol{w} + b \right) - 1 \right]$$

$$\frac{\partial L_P}{\partial \boldsymbol{w}} = 0 \qquad \Rightarrow \qquad \boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i$$

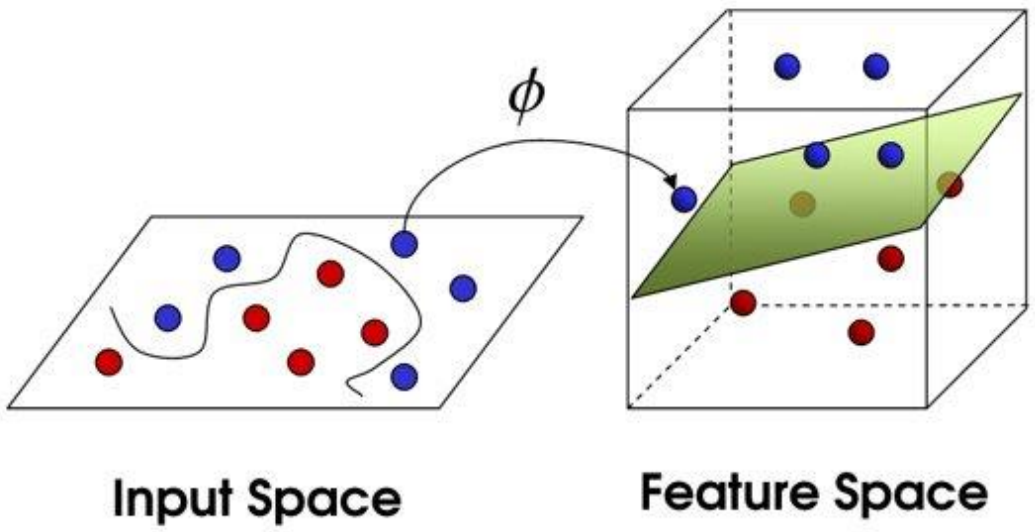
$$\frac{\partial L_P}{\partial b} = 0 \qquad \Rightarrow \qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$



SUPPORT VECTOR MACHINE

Non-linear case



Input space

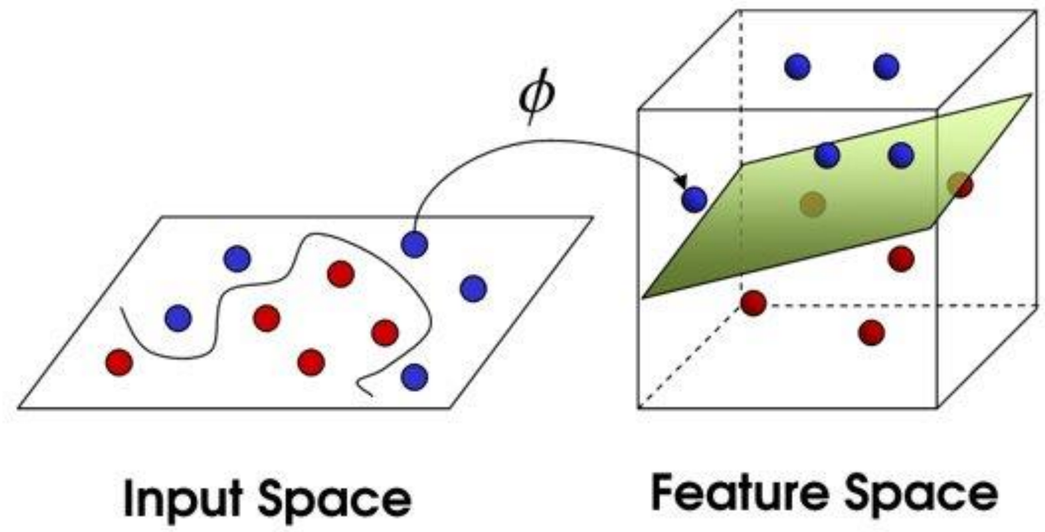
$$\boldsymbol{x}_i = \left(x_{0,i}, x_{1,i}, \ldots\right)$$

Feature space

$$\boldsymbol{\varphi}(\boldsymbol{x}_i) = \left(x_{0,i}, x_{1,i}, \ldots, \phi\left(x_{0,i}, x_{1,i}, \ldots\right)\right)$$

SUPPORT VECTOR MACHINE

Non-linear case



Input space

$$\boldsymbol{x}_i = \left(x_{0,i}, x_{1,i}, \ldots\right)$$

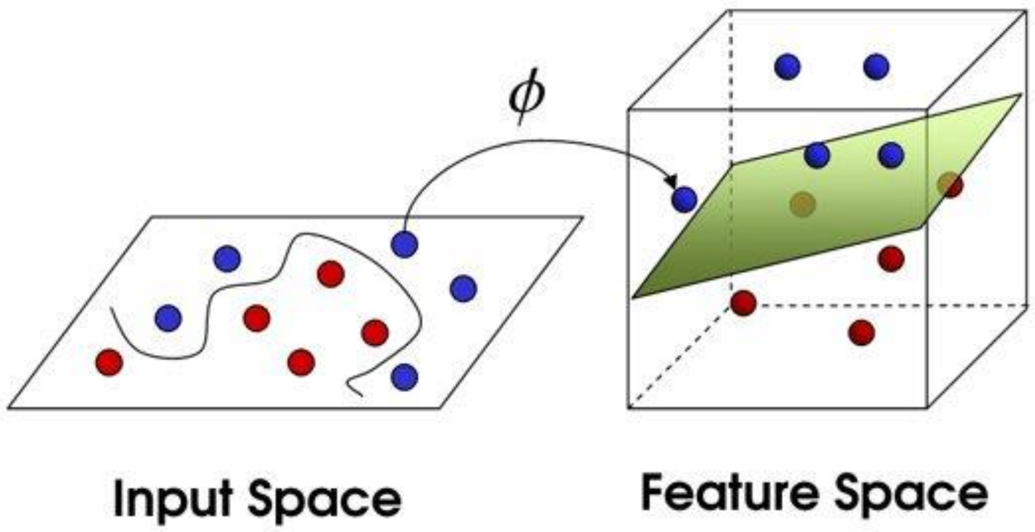
Feature space

$$\underline{\boldsymbol{\varphi}(\boldsymbol{x}_i)} = \Big(x_{0,i}, x_{1,i}, \ldots, \phi\left(x_{0,i}, x_{1,i}, \ldots\right)\Big)$$

Feature map

SUPPORT VECTOR MACHINE

Non-linear case



$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\boldsymbol{\varphi}(\boldsymbol{x}_i) \cdot \boldsymbol{w} + b \right) - 1 \right]$$

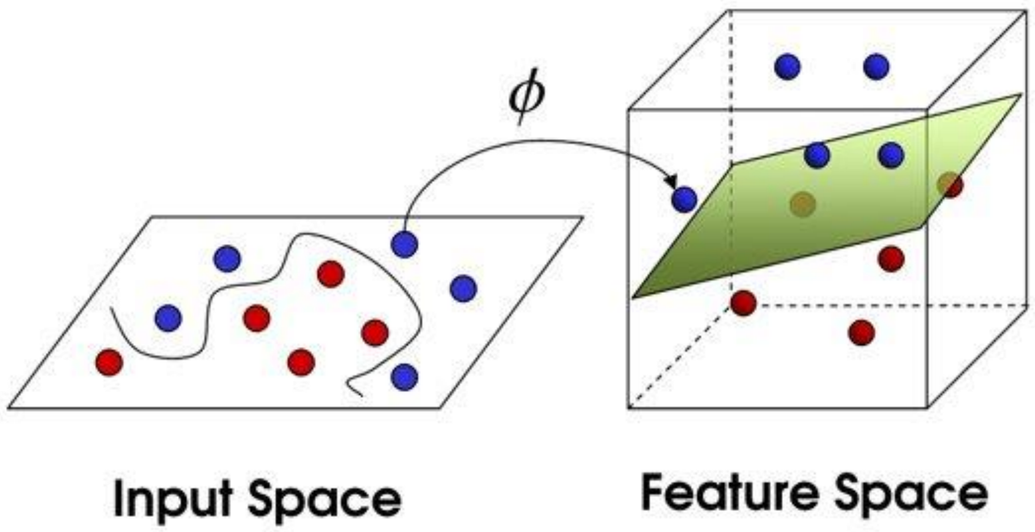
$$\frac{\partial L_P}{\partial \boldsymbol{w}} = 0 \qquad \Rightarrow \qquad \boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{\varphi}(\boldsymbol{x}_i)$$

$$\frac{\partial L_P}{\partial b} = 0 \qquad \Rightarrow \qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\varphi}(\boldsymbol{x}_i) \cdot \boldsymbol{\varphi}(\boldsymbol{x}_j)$$

SUPPORT VECTOR MACHINE

Non-linear case



$$L_P \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(\boldsymbol{\varphi}(\boldsymbol{x}_i) \cdot \boldsymbol{w} + b \right) - 1 \right]$$

$$\frac{\partial L_P}{\partial \boldsymbol{w}} = 0 \quad \Rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{\varphi}(\boldsymbol{x}_i)$$

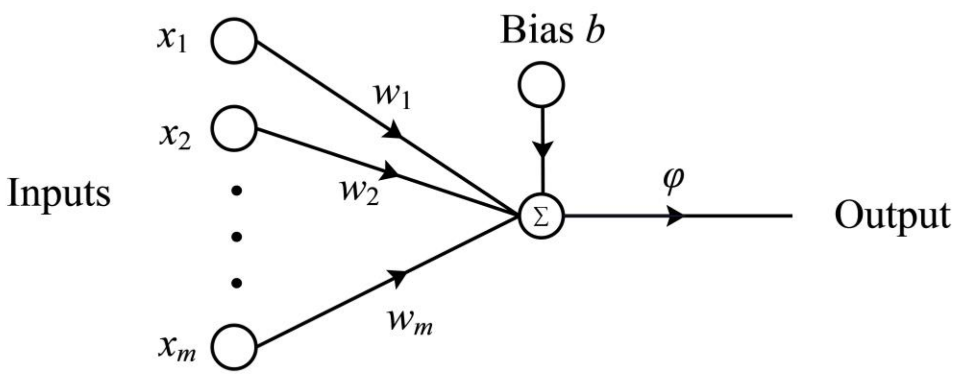
$$\frac{\partial L_P}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \underline{\boldsymbol{\varphi}(\boldsymbol{x}_i) \cdot \boldsymbol{\varphi}(\boldsymbol{x}_j)}$$

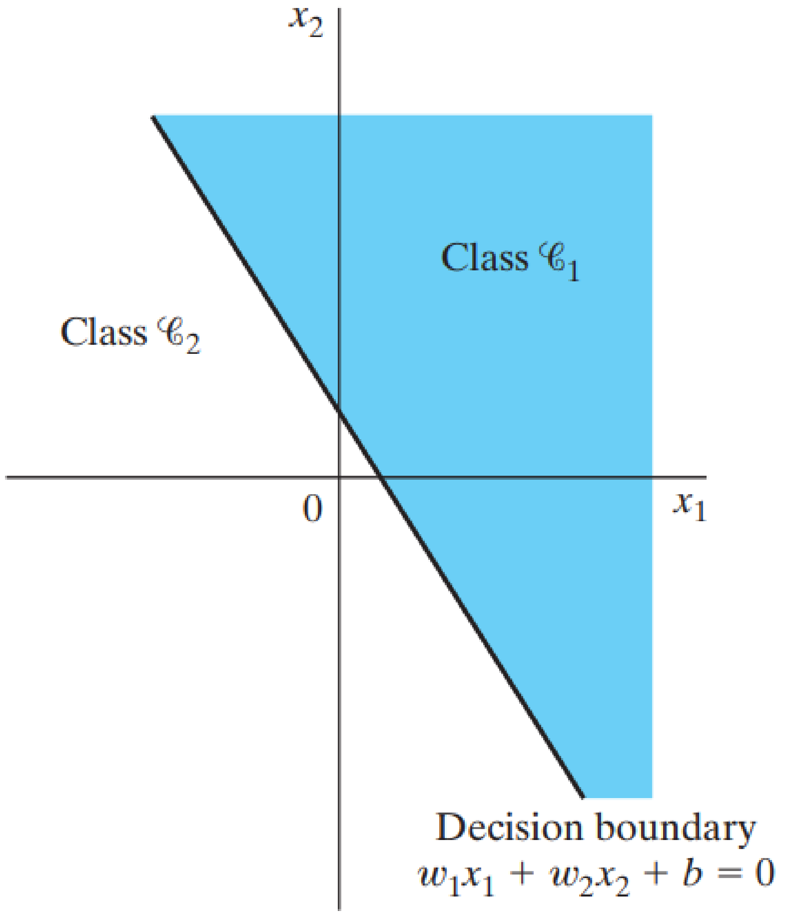
Kernel matrix $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\varphi}(\boldsymbol{x}_i) \cdot \boldsymbol{\varphi}(\boldsymbol{x}_j)$

Neural Networks

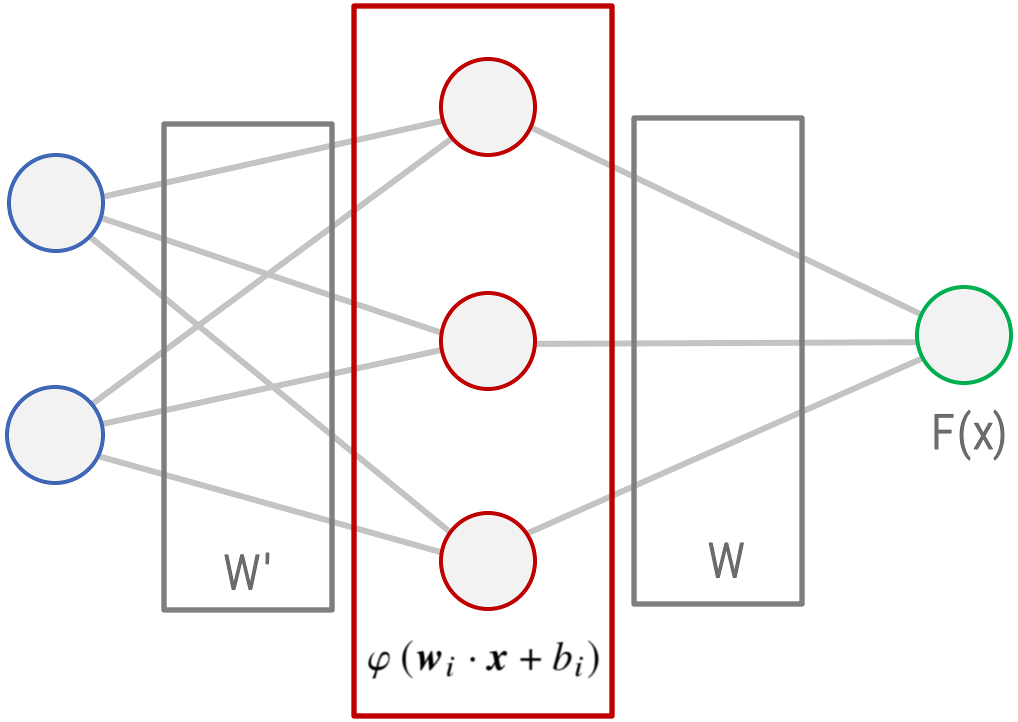
PERCEPTRON



$$F(\boldsymbol{x}) = \varphi\left(\sum_{i=0}^{m} x_i w_i + b\right) = \begin{cases} 1, & \text{if } \sum_{i=0}^{m} x_i w_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$



From Perceptron to Hornik theorem



Hornik theorem

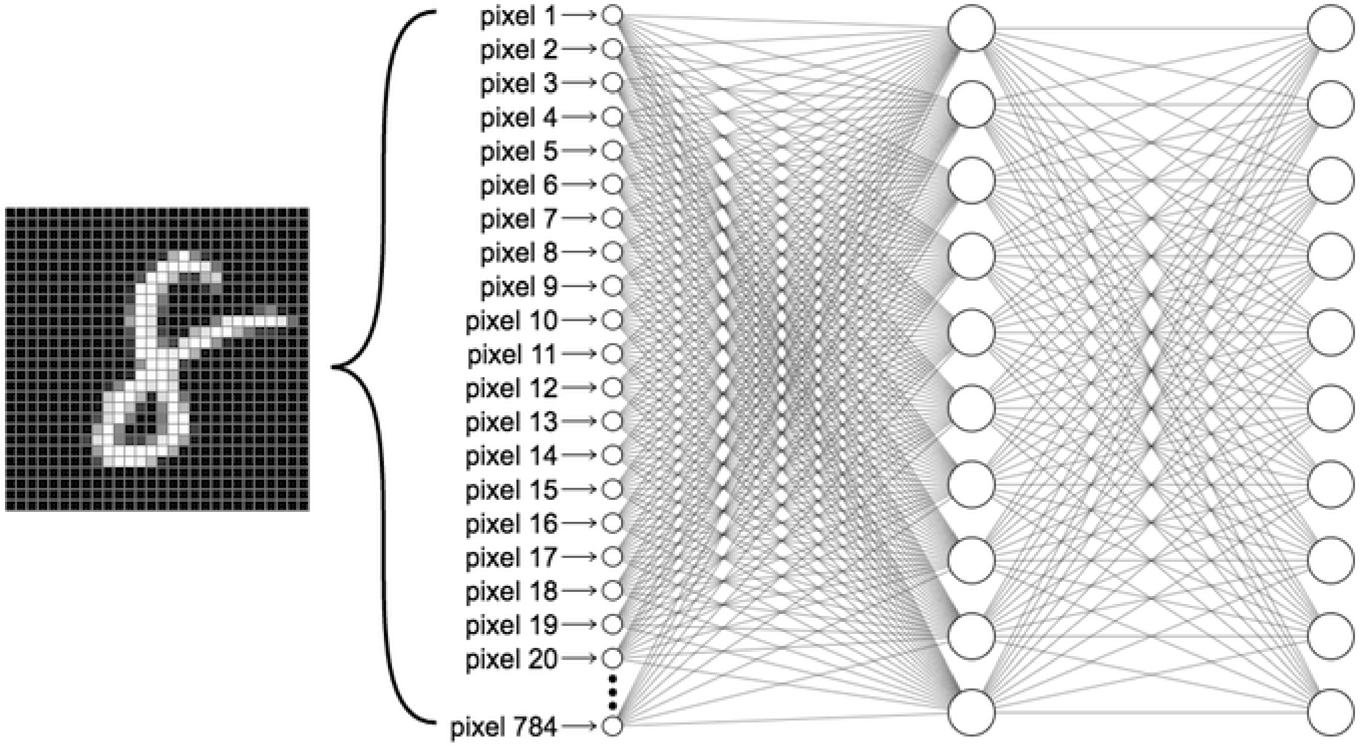
$$F(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{w}'_i \varphi\left(\boldsymbol{w}_i \cdot \boldsymbol{x} + b_i\right)$$

- 1 output
- 1 hidden layer
- N hidden neurons

NEURAL NETWORKS

Deep supervised learning

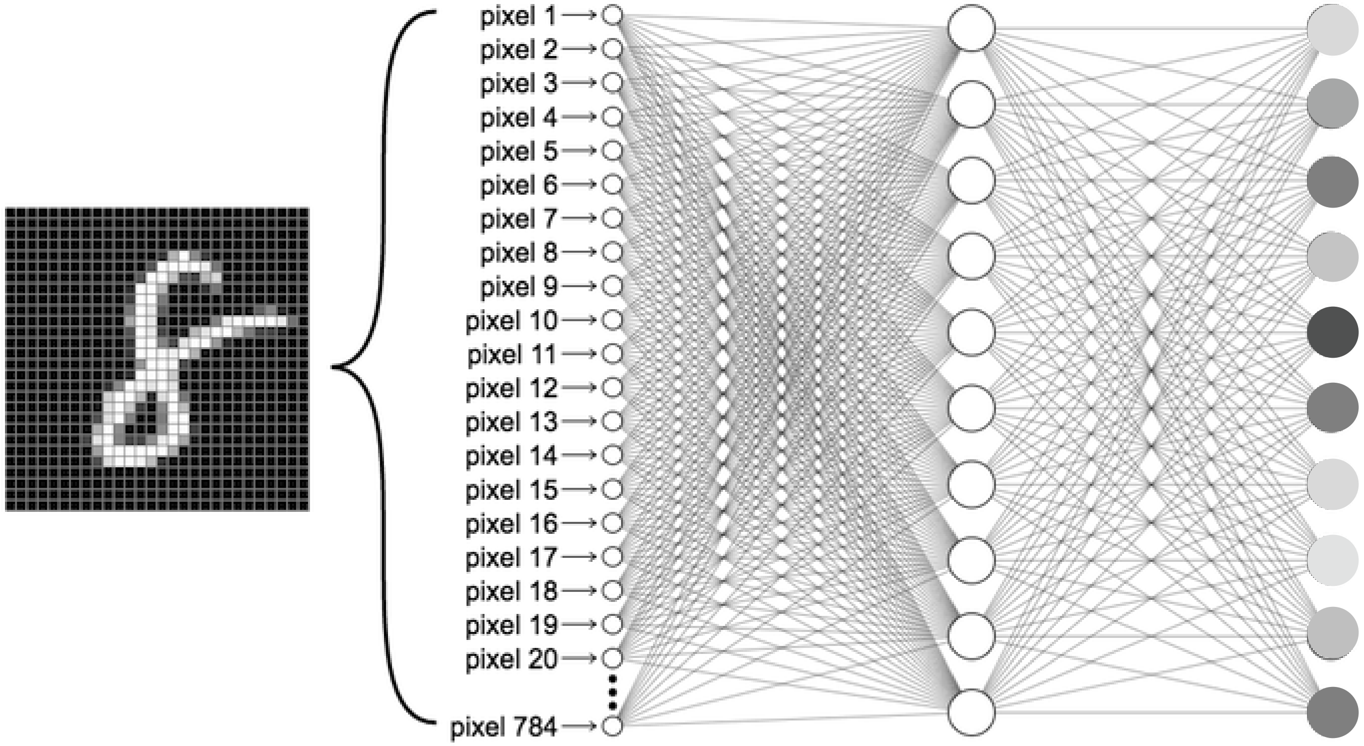
Before Training: Random parameters



NEURAL NETWORKS

Deep supervised learning

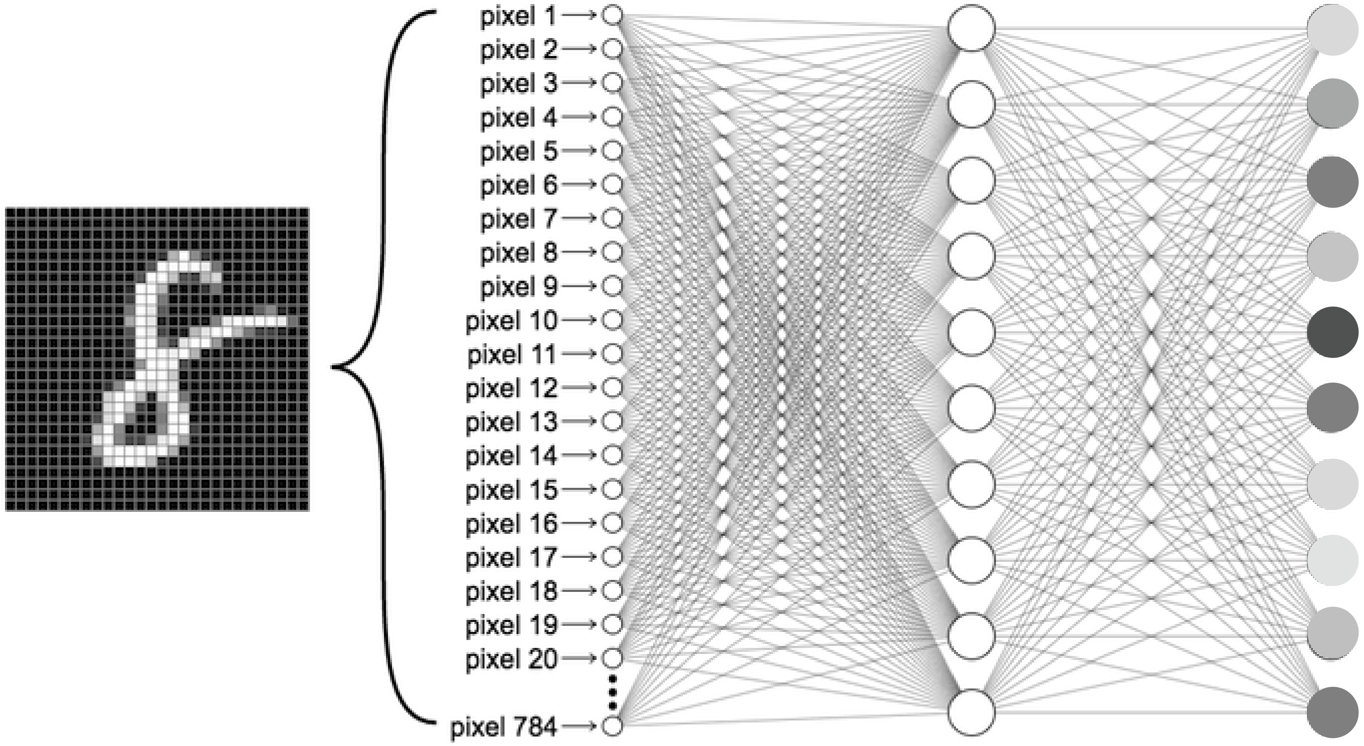
Before Training: Random parameters



NEURAL NETWORKS

Deep supervised learning

Before Training: Random parameters



LOSS function

The distance between the neural network predictions and the labels of the training set

- MSE

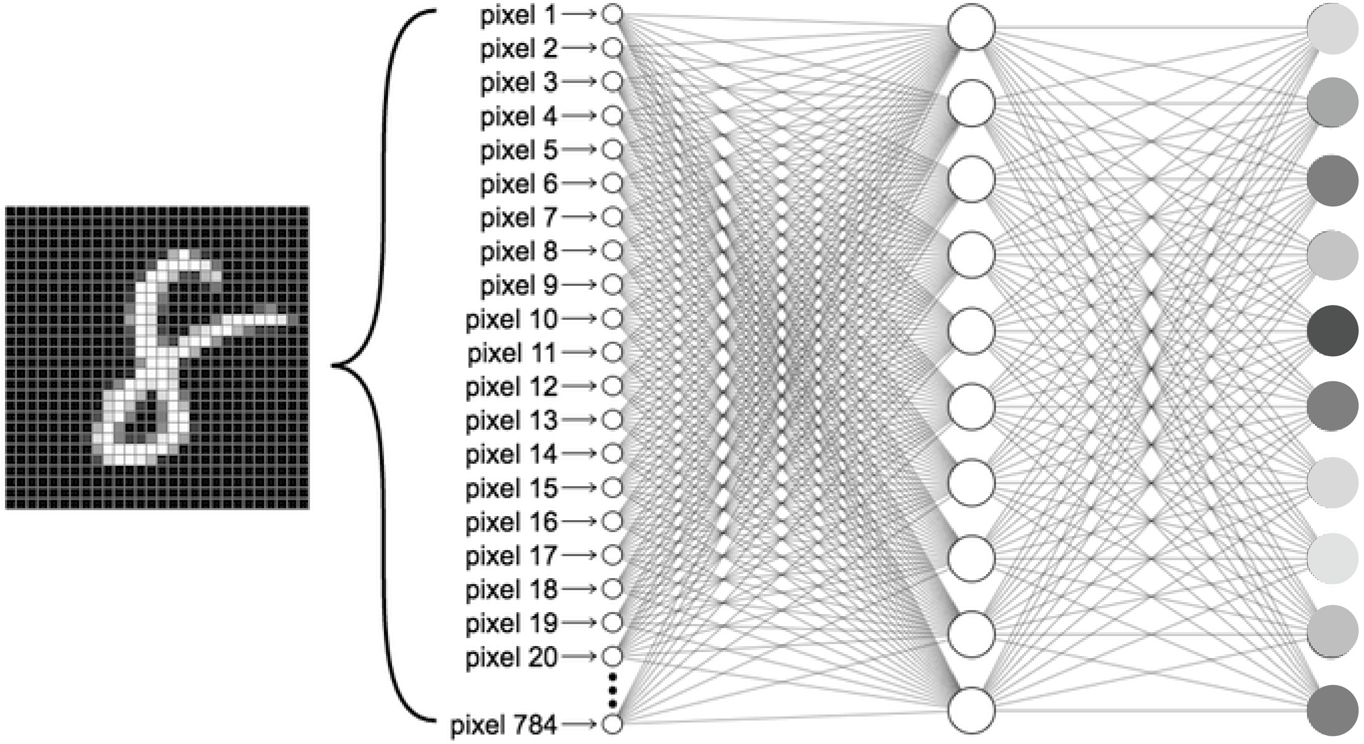
$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$$

- Cross-Entropy

$$\mathcal{L}(\theta) = -\sum_{i=0}^{N}\hat{y}_i \cdot log(y_i)$$

Deep supervised learning

Before Training: Random parameters



LOSS funciton

The distance between the neural network preditions and the labels of the training set

Truth label

- MSE

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=0}^{N} (y_i - \hat{y}_i)^2$$

Neural network prediction

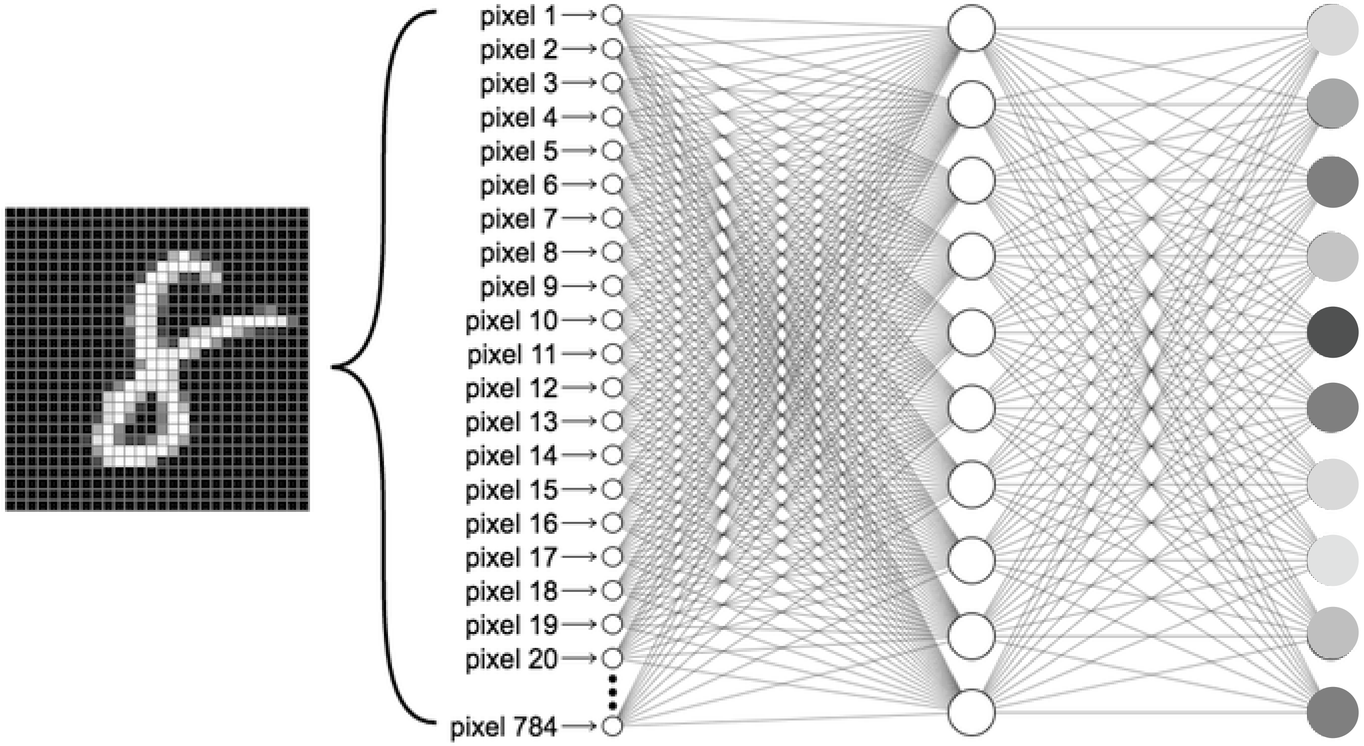
- Cross-Entropy

$$\mathcal{L}(\theta) = -\sum_{i=0}^{N} \hat{y}_i \cdot log(y_i)$$

NEURAL NETWORKS

Deep supervised learning

Before Training: Random parameters



Loss function optimization

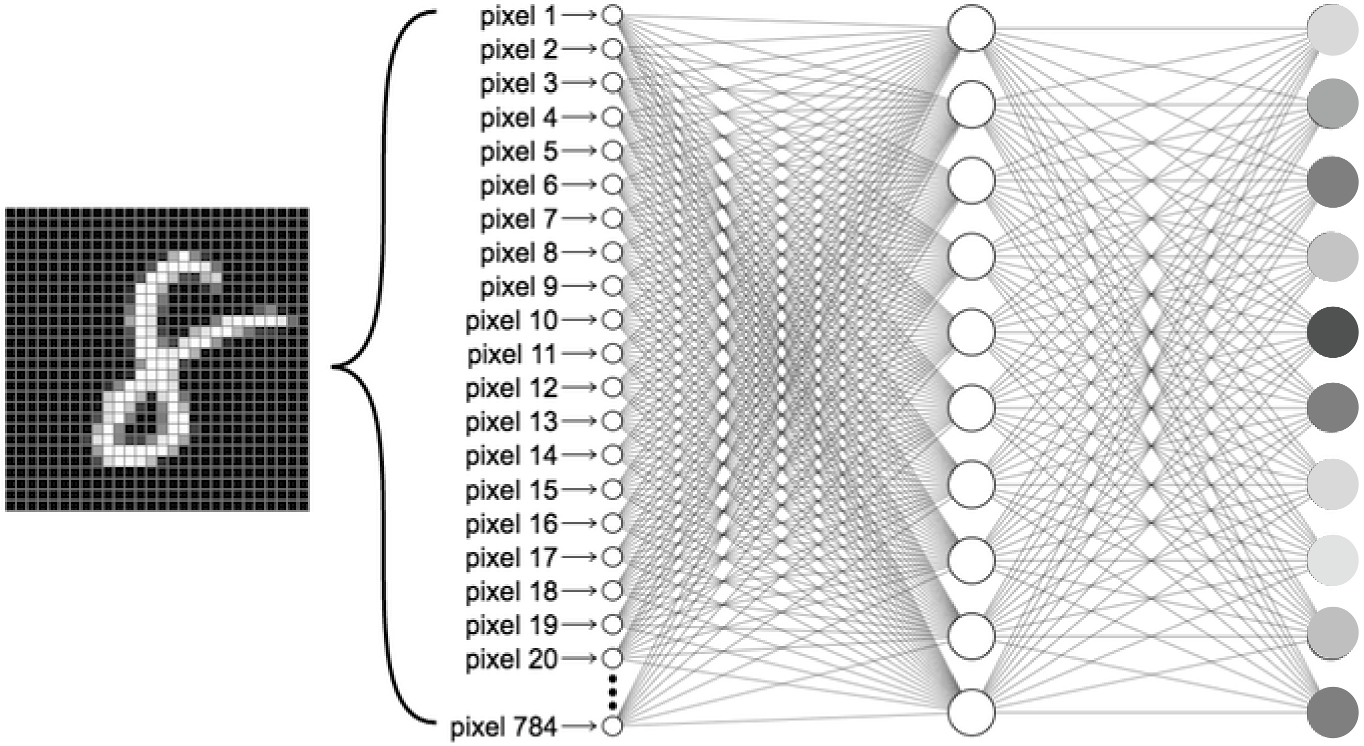
Gradient descent

$$\theta \leftarrow \theta - \alpha \nabla \mathcal{L}(\theta)$$

NEURAL NETWORKS

Deep supervised learning

Before Training: Random parameters



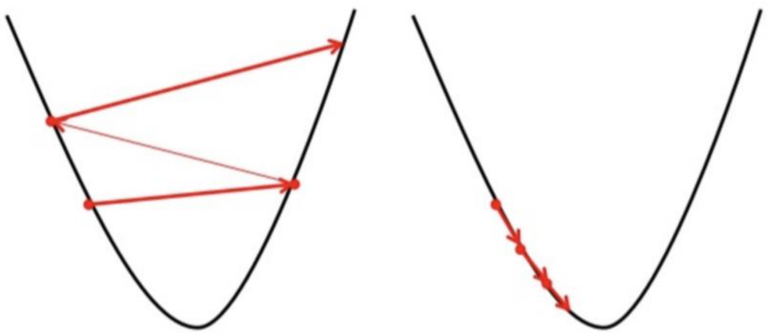
Loss function optimization

Gradient descent

$$\theta \leftarrow \theta - \alpha \nabla \mathcal{L}(\theta)$$

Learning rate

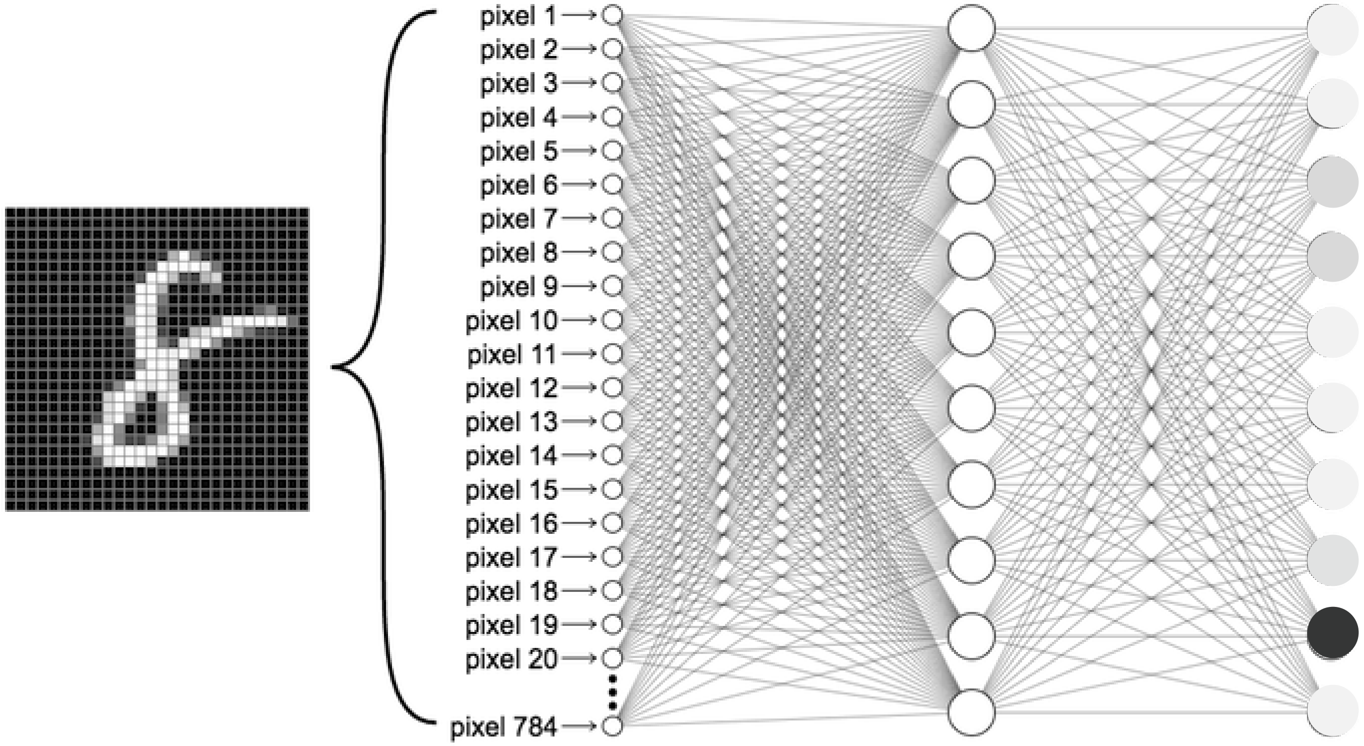
Backpropagatiorn:
Compute the gradient by chain rule



NEURAL NETWORKS

Deep supervised learning

After Training: Optimal parameters



https://playground.tensorflow.org/

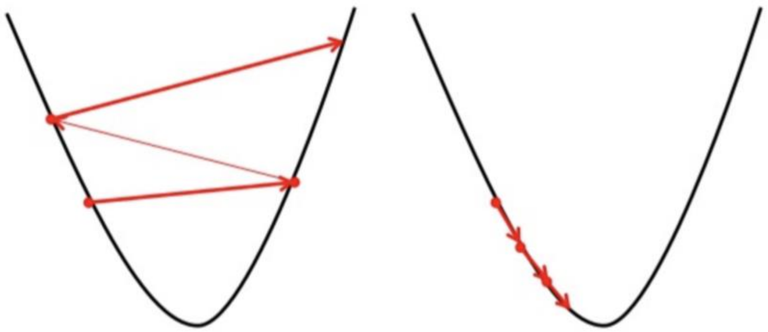
Loss function optimization

Gradient descent

$$\theta \leftarrow \theta - \alpha \nabla \mathcal{L}(\theta)$$

Learning rate

Backpropagatiorn:
Compute the gradient by chain rule



QUANTUM MACHINE LEARNING ON NISQ DEVICES

Marco Maronese – PhD student @ University of Bologna

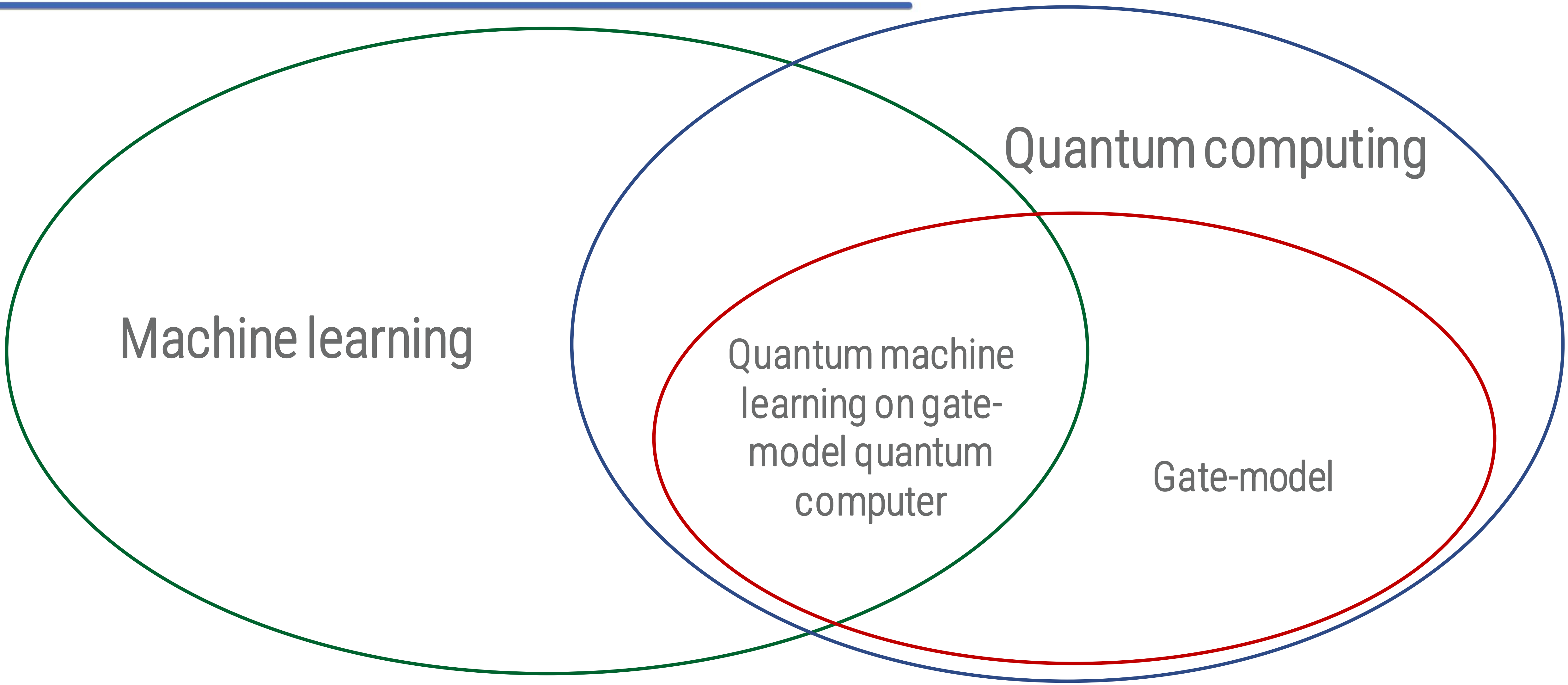
CINECA QUANTUM COMPUTING LAB

INDEX

- Quantum Machine Learning on NISQ devices
- Quantum Support Vector Machine (QSVM)
- Quantum Neural Networks (QNN)

Quantum Machine Learning on NISQ devices

QUANTUM MACHINE LEARNING



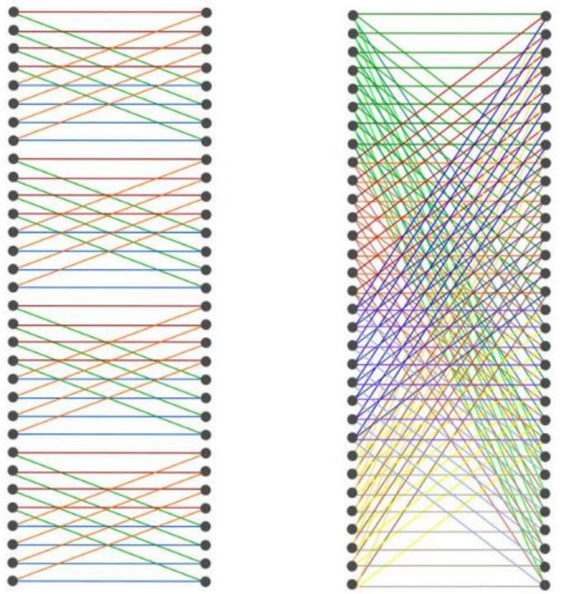
APPLICATIONS

- Drug discovery
- Finance
- Space
- Cybersecurety

QUANTUM MACHINE LEARNING

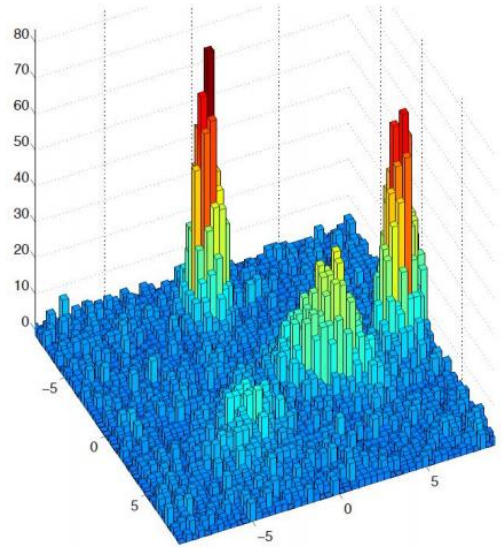
Possible advatage

Linear algebra



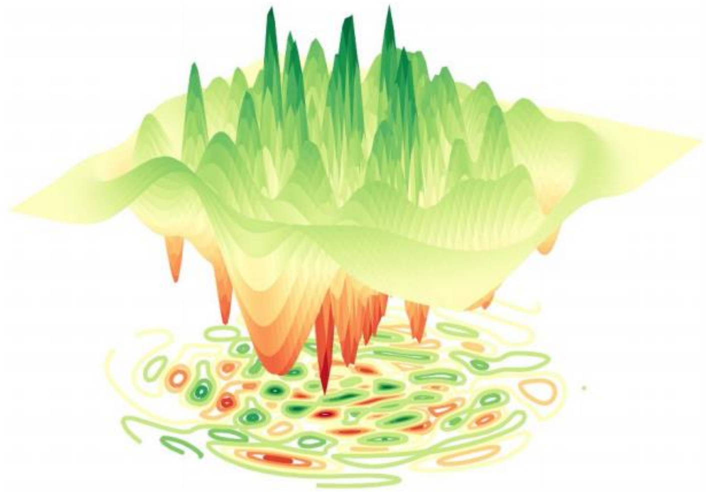
Operations on gate-model quantum computers follow the linear algebra rules

Sampling



Quantum mechanics is intrinsically probabilistic

Optimization



Explore more paths with quantum phenomena

Overview

- First generation of QML
  - Accelerated linear algebra on quantum computers
  - Only applies to fault-tolerant quantum computers

Overview

- First generation of QML
  - Accelerated linear algebra on quantum computers
  - Only applies to fault-tolerant quantum computers

- Second generation of QML
  - Low-depth quantum circuit learning (QNN)
  - Applicable to noisy-intermediate quantum (NISQ) devices

Overview

- First generation of QML
  - Accelerated linear algebra on quantum computers
  - Only applies to fault-tolerant quantum computers

- Second generation of QML
  - Low-depth quantum circuit learning (QNN)
  - Applicable to noisy-intermediate quantum (NISQ) devices

QUANTUM MACHINE LEARNING

Frameworks

QISKIT

Qiskit

Most used quantum SDK with access of quantum devices

Machine learning modules

PENNYLANE

PENNYLANE

For hybrid quantum-classical computation

Integration of pytorch and tensorflow with different quantum SDK

TF QUANTUM

= Cirq + TensorFlow

For hybrid quantum-classical computation

Equivalence between quantum and tensor operations

QUANTUM MACHINE LEARNING

NISQ application

Supervised learning with quantum enhanced feature spaces

Vojtech Havlicek[1],* Antonio D. Córcoles[1], Kristan Temme[1], Aram W. Harrow[2], Abhinav Kandala[1], Jerry M. Chow[1], and Jay M. Gambetta[1]
[1]*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA and*
[2]*Center for Theoretical Physics, Massachusetts Institute of Technology, USA*
(Dated: June 7, 2018)

Supervised learning:

Quantum Support Vector Machine (QSVM)
for regression and classification

Unsupervised learning:

Quantum Generative Adversal Network (QGAN)
hybrid quantum-classical generative model

Quantum Generative Adversarial Networks for Learning and Loading Random Distributions

Christa Zoufal,[1,2,*] Aurélien Lucchi,[2] and Stefan Woerner[1]
[1]*IBM Research – Zurich*
[2]*ETH Zurich*
(Dated: April 2, 2019)

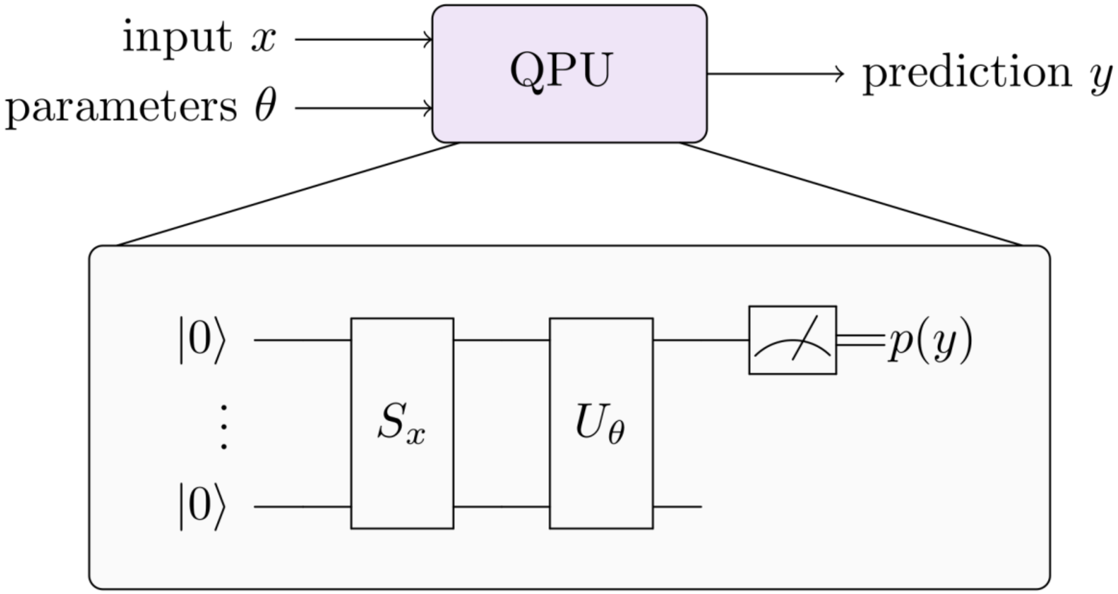
REINFORCEMENT LEARNING WITH QUANTUM VARIATIONAL CIRCUITS

Owen Lockwood[1*] and Mei Si[2]
[1]Department of Computer Science, Rensselaer Polytechnic Institute
[2]Department of Cognitive Science, Rensselaer Polytechnic Institute

Reinforcement learning:

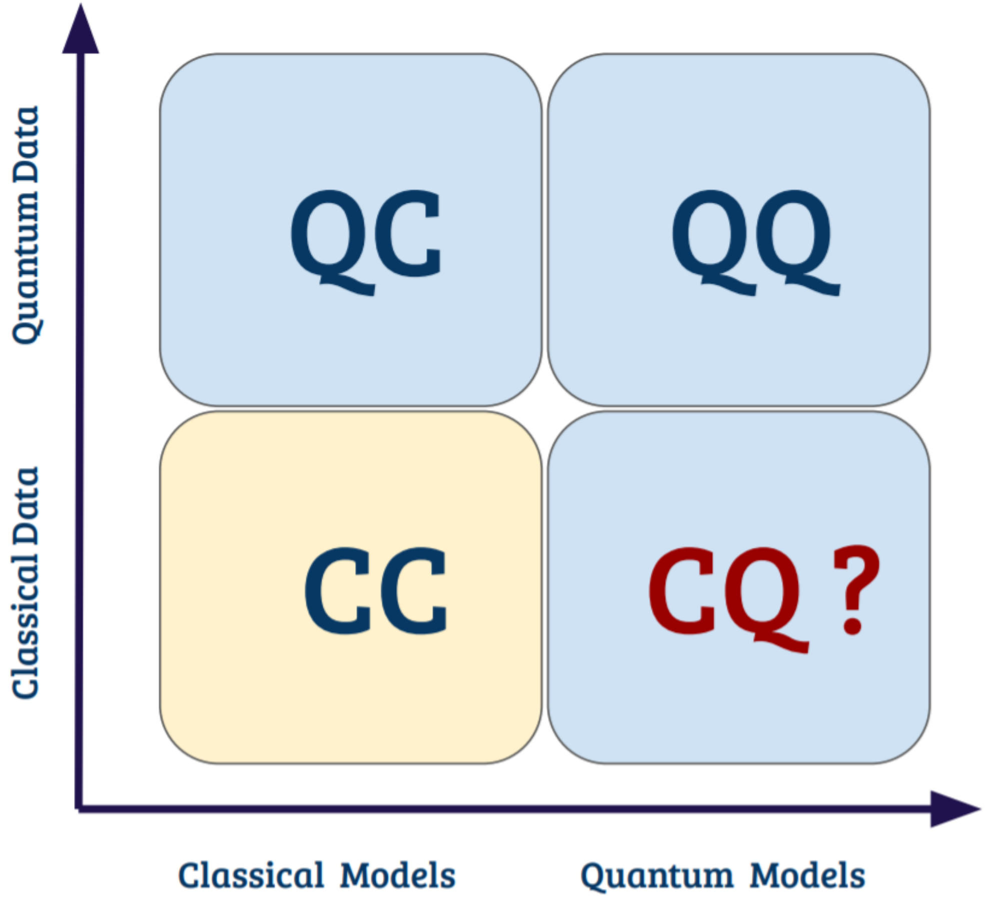
Reinforcement learning with QVC
hybrid reinforcement learning algorithm

Ideal blocks



- **Feature map**: Store the inputs in a quantum state
- **Variational circuit**: Learnable parameter circuit
- **Expectation value**: Measurements introducing non-linearity

Encoding problems



No QRAM to store data

↓

Encoding data generally hard

↓

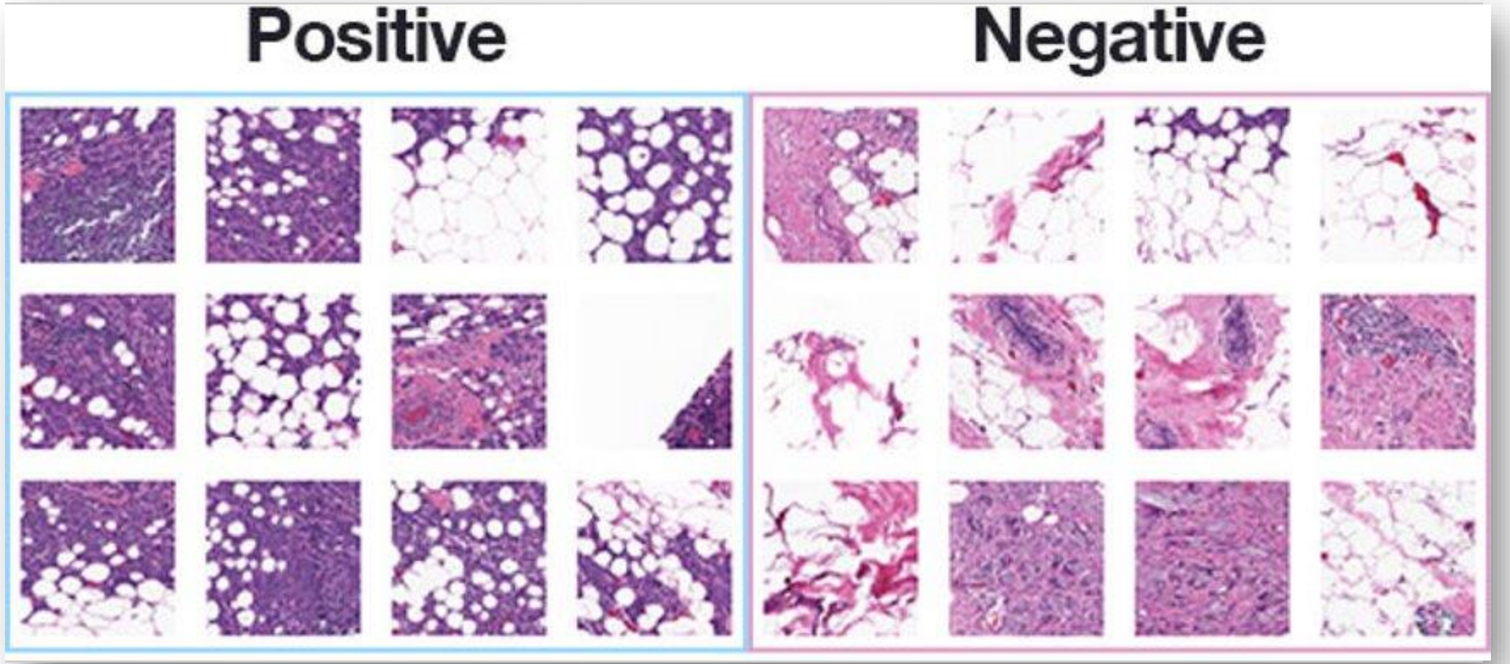
Better if are intrinsically quantum such as quantum states of a system

Quantum Support Vector Machine (QSVM)

QUANTUM SUPPORT VECTOR MACHINE

Use case

Breast cancer Wisconsin

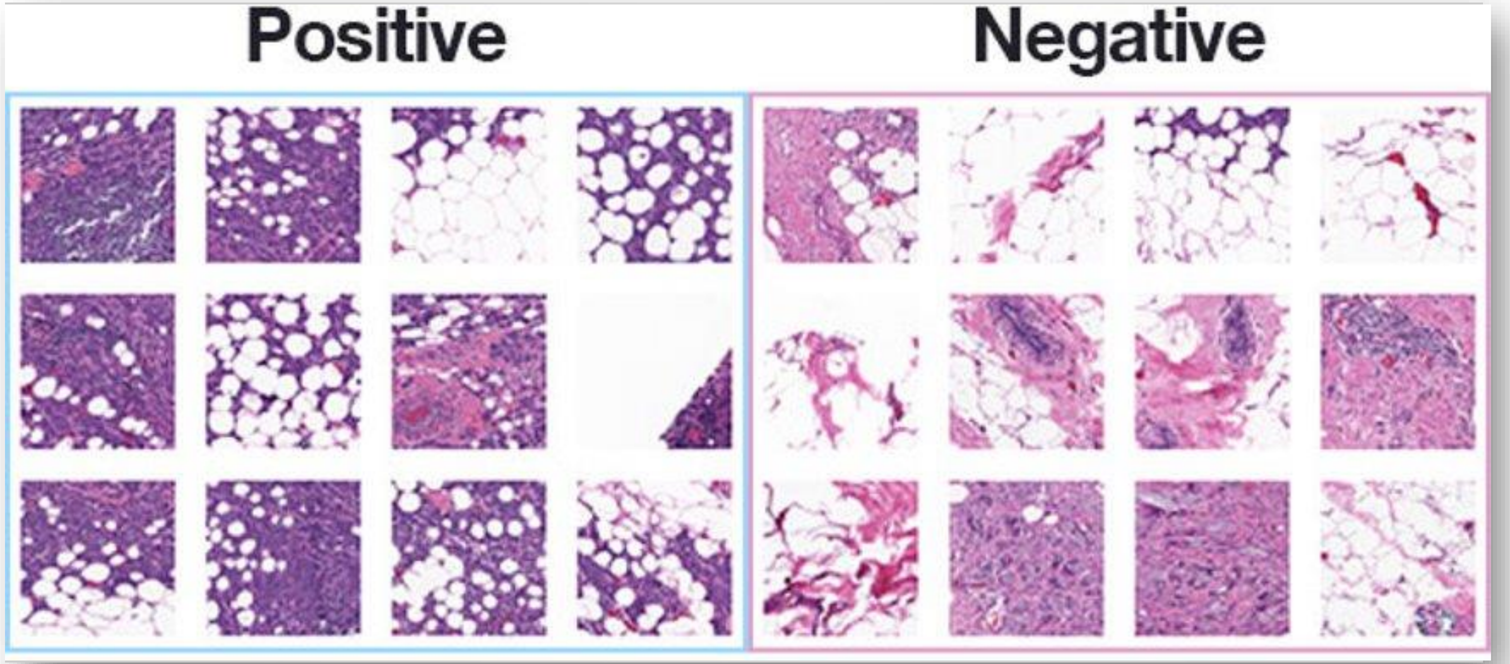


https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/version/2

QUANTUM SUPPORT VECTOR MACHINE

Use case

Breast cancer Wisconsin



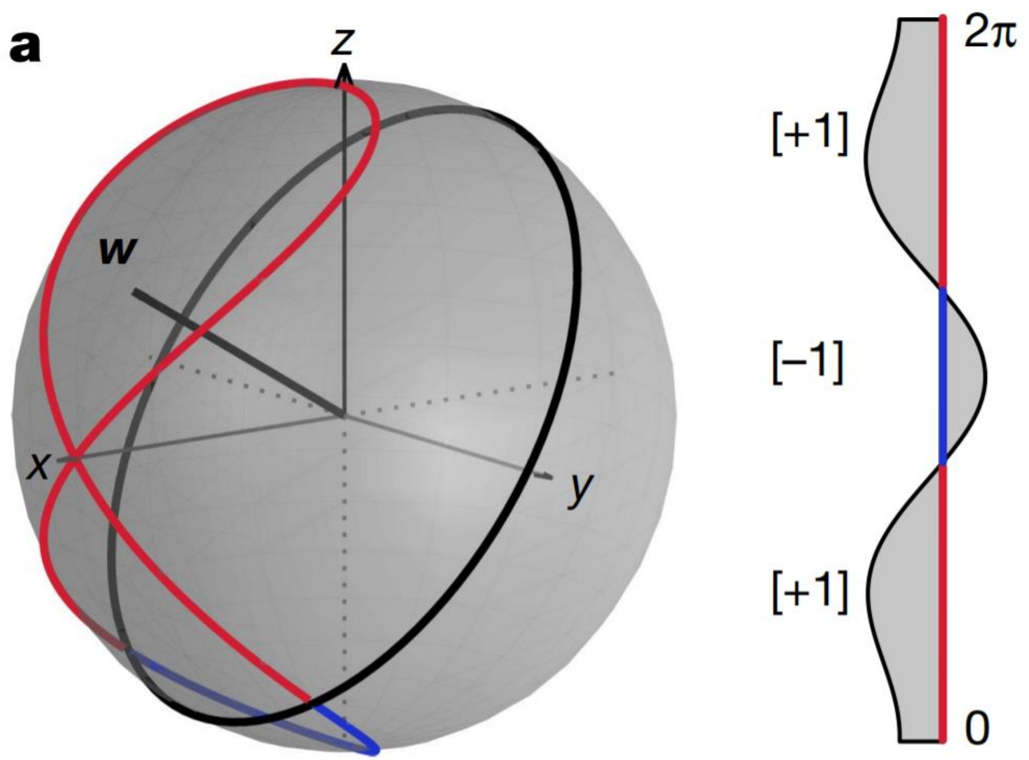
https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/version/2

Dataset with 2 feature

2 qubits to performe QSVM

QUANTUM SUPPORT VECTOR MACHINE

Intuition



Quantum advantage:

More complex feature map at low computational cost

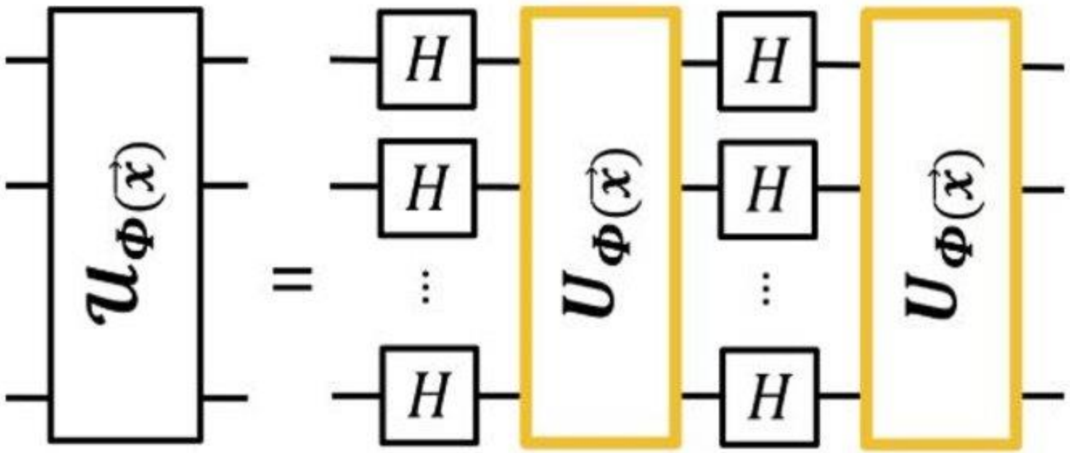
QUANTUM SUPPORT VECTOR MACHINE

Intuition

Non-linear feature map

$$\vec{x} \mapsto |\Phi(\vec{x})\rangle = \mathcal{U}_{\Phi(\vec{x})}|0\rangle^{\otimes n}$$

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S\subseteq[n]} \phi_S(\vec{x}) \prod_{j\in S} Z_j \right)$$



QUANTUM SUPPORT VECTOR MACHINE

Feature map

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{j \in S} Z_j \right)$$

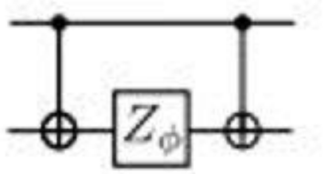
- First order expansion (1 or more qubits, no entanglement)

$$S \in \{0, 1, \ldots, n-1\} \qquad \phi_i(\boldsymbol{x}) = x_i$$

- Second order expansion (2 or more qubits, entanglement)

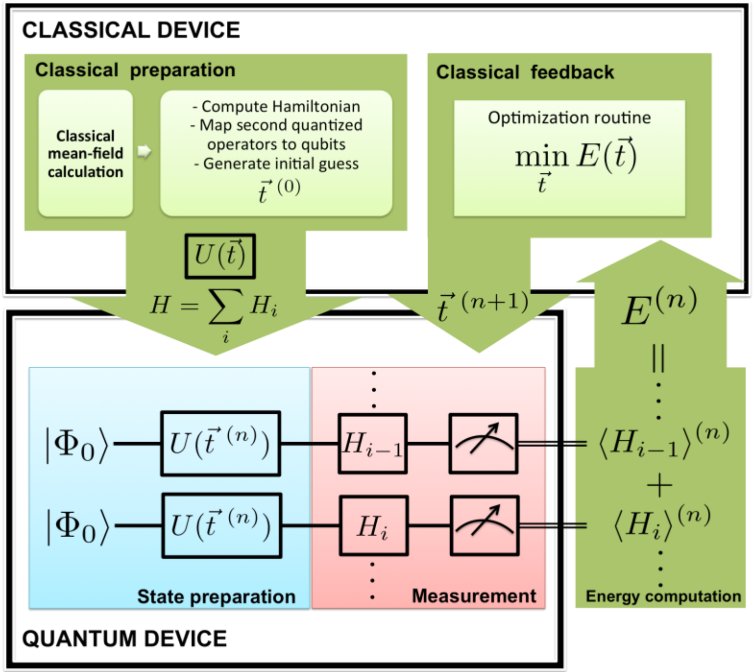
$$S \in \{0, 1, \ldots, n-1, (0,1), (1,2), \ldots, (n-2, n-1)\} \qquad \phi_{(i,j)}(\boldsymbol{x}) = (\pi - x_i)(\pi - x_j)$$

$$e^{i\phi_{\{l,m\}}(\vec{x}) Z_l Z_m} =$$



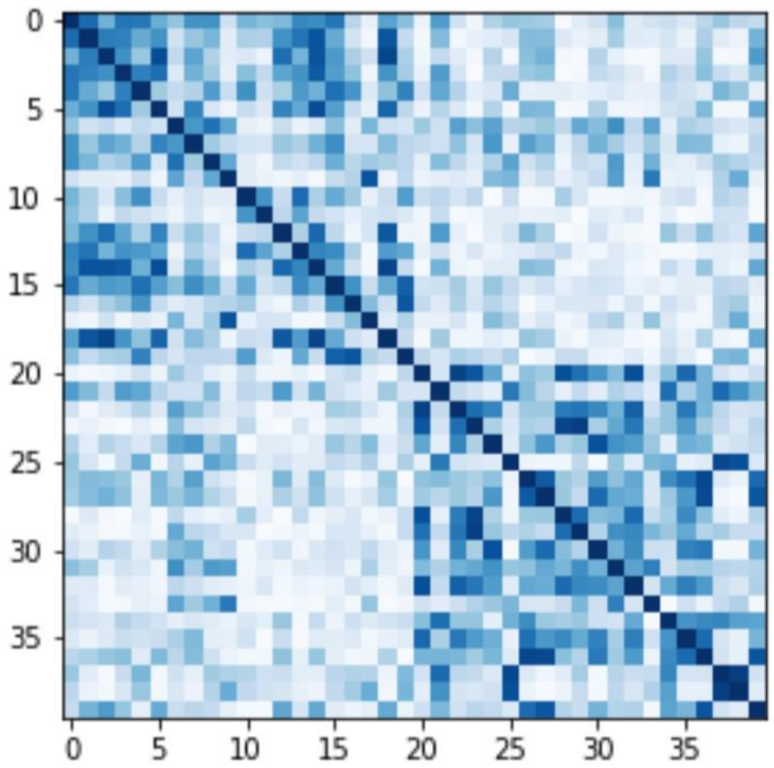
QUANTUM SUPPORT VECTOR MACHINE

- Variational quantum circuit



Using variational quantum eigensolver (VQE)

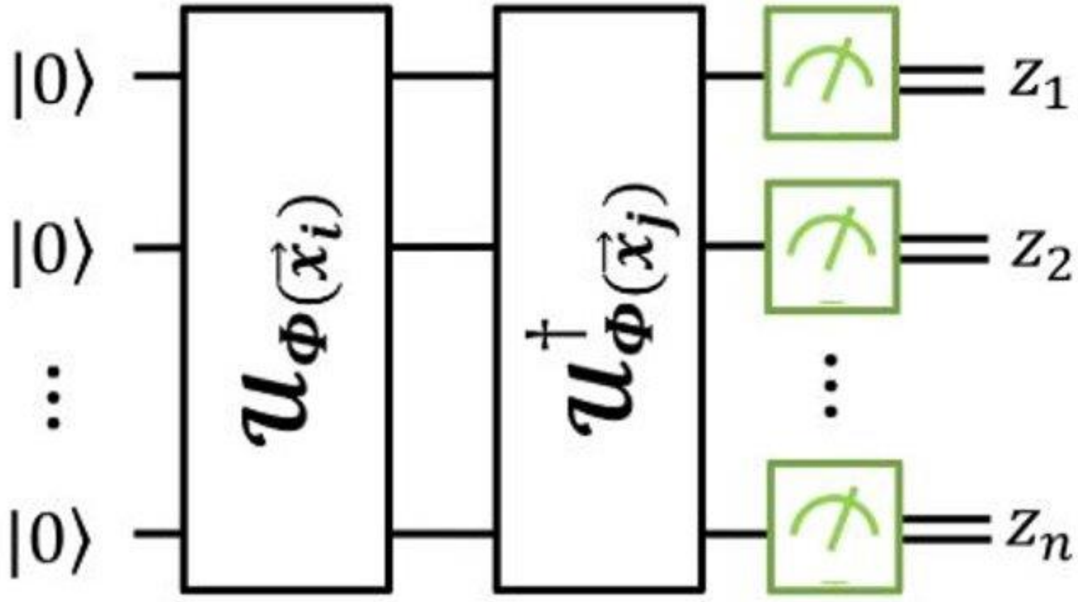
- Kernel matrix estimation



QUANTUM SUPPORT VECTOR MACHINE

Kernel estimation

$$K(\vec{x}_i, \vec{x}_j) = \left|\langle\Phi(\vec{x}_i)|\Phi(\vec{x}_j)\rangle\right|^2 = \left|\langle 0|\mathcal{U}^{\dagger}_{\Phi(\vec{x}_j)}\mathcal{U}_{\Phi(\vec{x}_i)}|0\rangle^{\otimes n}\right|^2$$



$$p_{0\ldots0} \sim K(\vec{x}_i, \vec{x}_j)$$

QUANTUM SUPPORT VECTOR MACHINE

Quantum variational circuit

$$p_y(\boldsymbol{x}) = \langle \Phi(\boldsymbol{x})| W^\dagger(\boldsymbol{\theta}) M_y W(\boldsymbol{\theta}) |\Phi(\boldsymbol{x})\rangle$$

If $\hat{p}_y(\boldsymbol{x}) > \hat{p}_{-y}(\boldsymbol{x}) - yb$

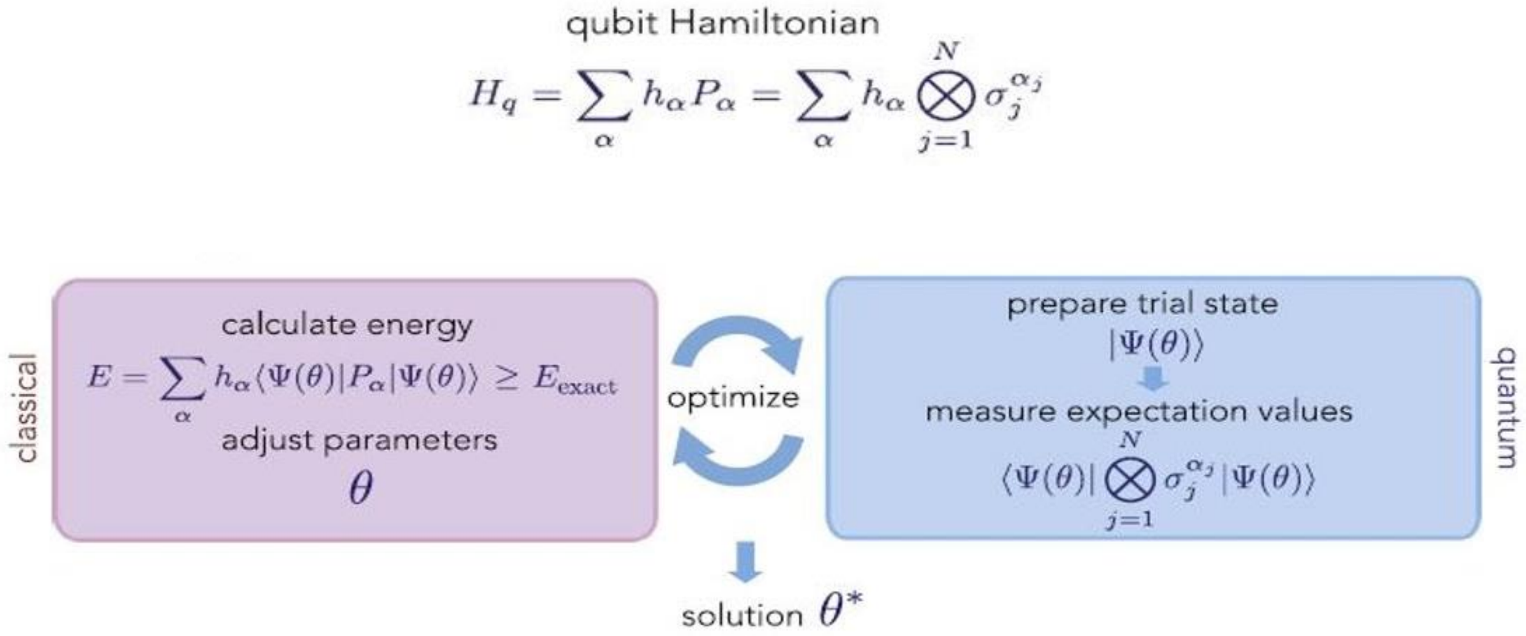
$$\boxed{\tilde{m}(\boldsymbol{x}) = y}$$

Cost function

$$\Pr(\tilde{m}(\boldsymbol{x}) \neq m(\boldsymbol{x}))$$

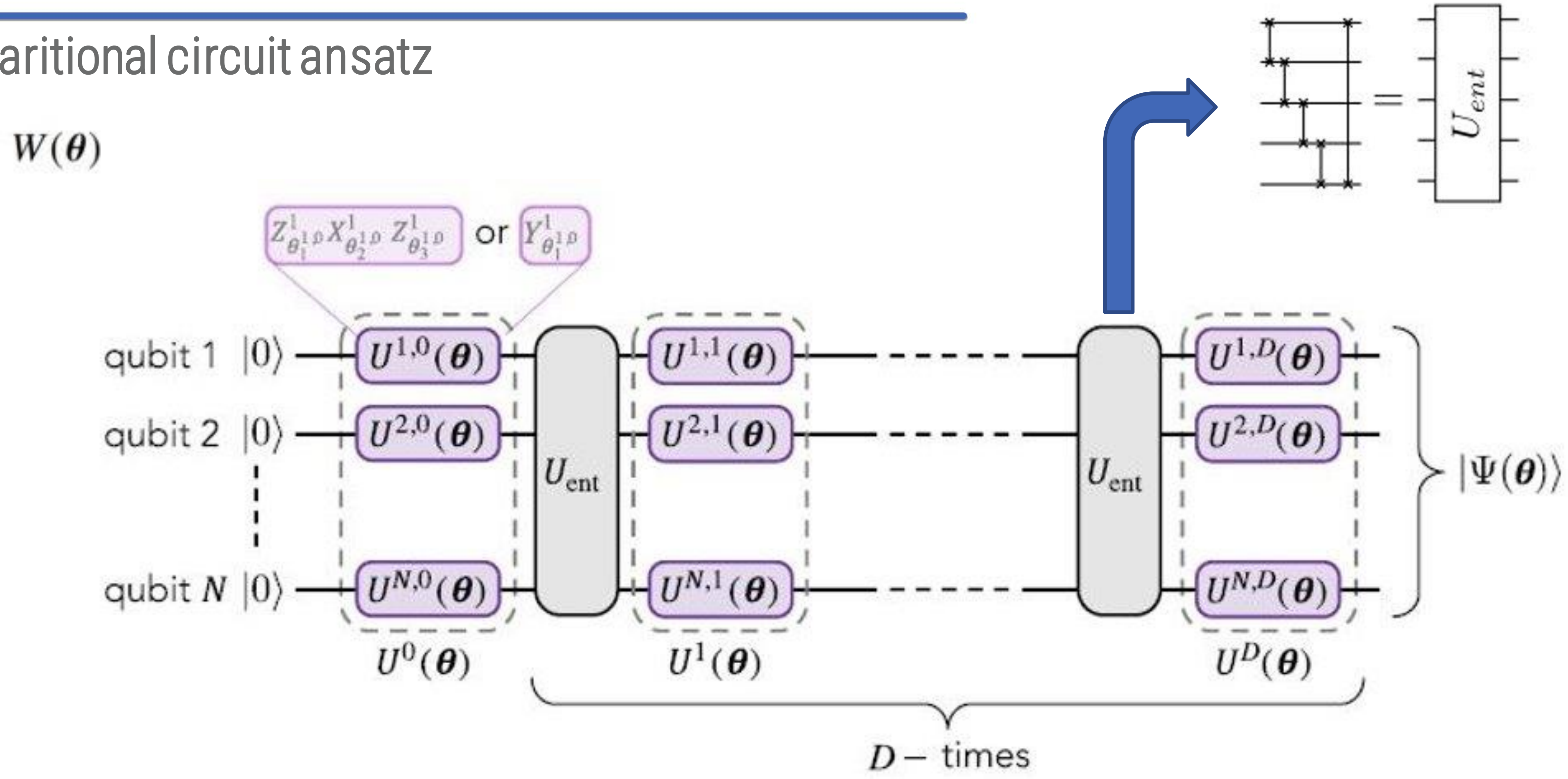
Learnable parameters

$$(\boldsymbol{\theta}, b)$$



QUANTUM SUPPORT VECTOR MACHINE

Varitional circuit ansatz



QUANTUM SUPPORT VECTOR MACHINE

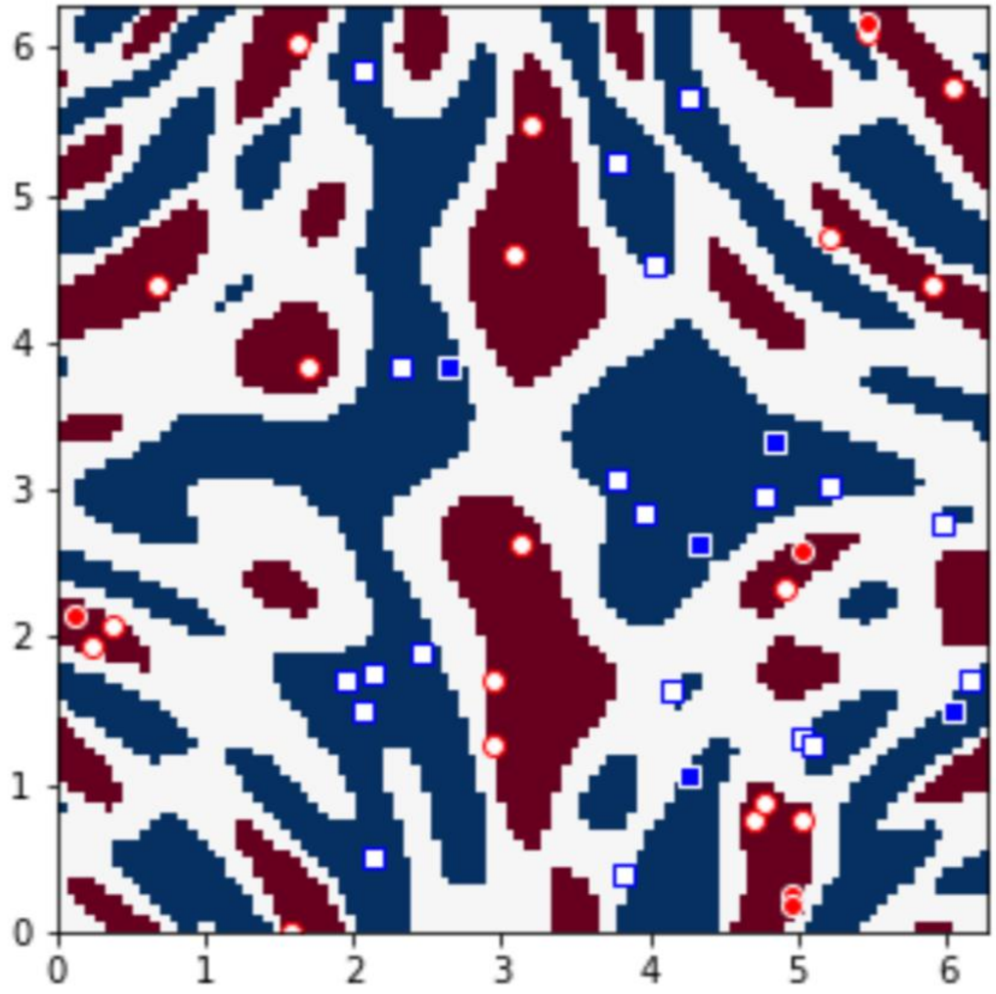
Quantum variational circuit

- Expectation value of an operator *f*

$$\langle \Phi(\vec{x}) | W^\dagger(\theta) \, \mathbf{f} \, W(\theta) | \Phi(\vec{x}) \rangle = \frac{1}{2^n} \sum_\alpha w_\alpha(\theta) \Phi_\alpha(\vec{x})$$

- Classification rule

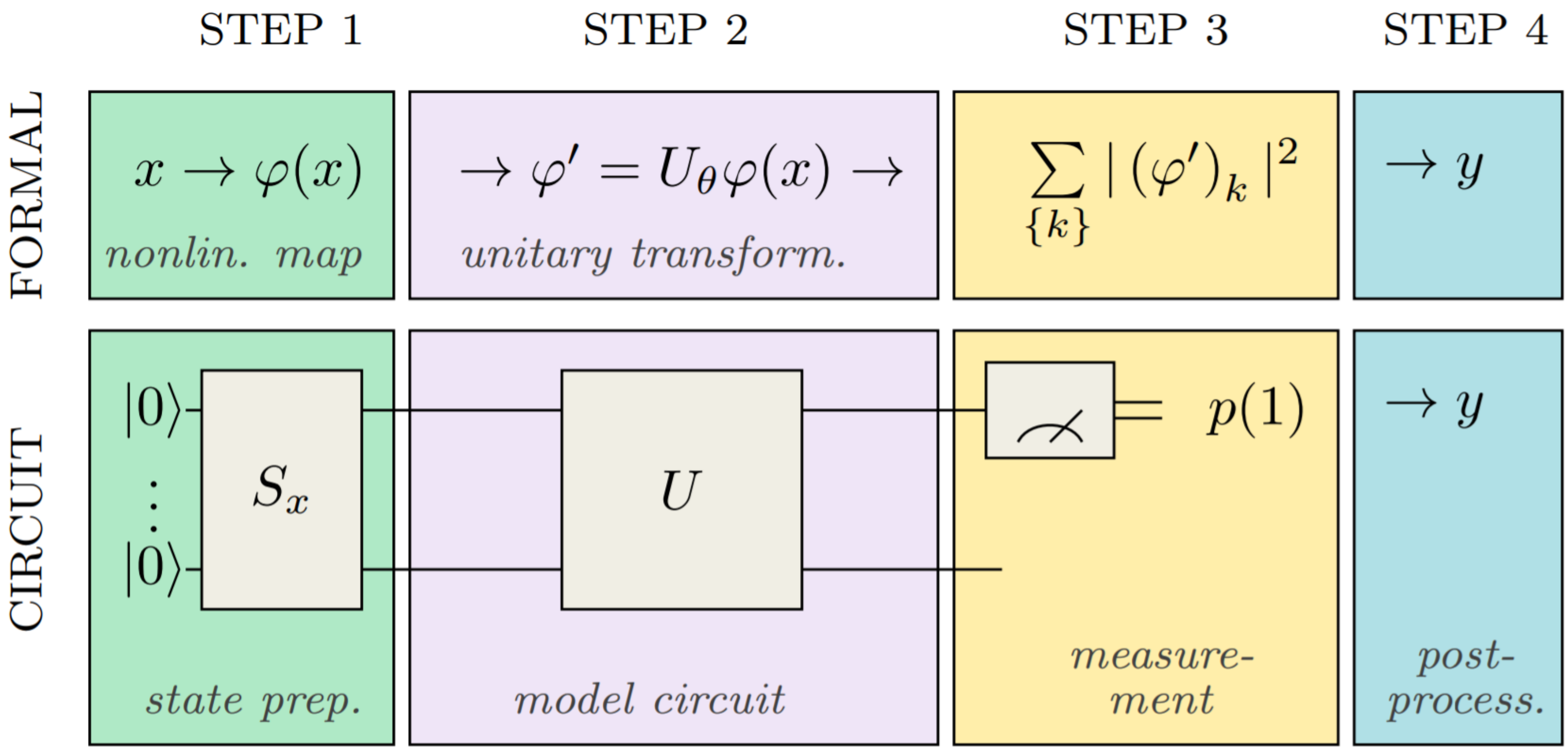
$$\tilde{m}(\vec{x}) = \text{sign}\left(\frac{1}{2^n} \sum_\alpha w_\alpha(\theta) \Phi_\alpha(\vec{x}) + b \right)$$



Quantum Neural Networks (QNN)

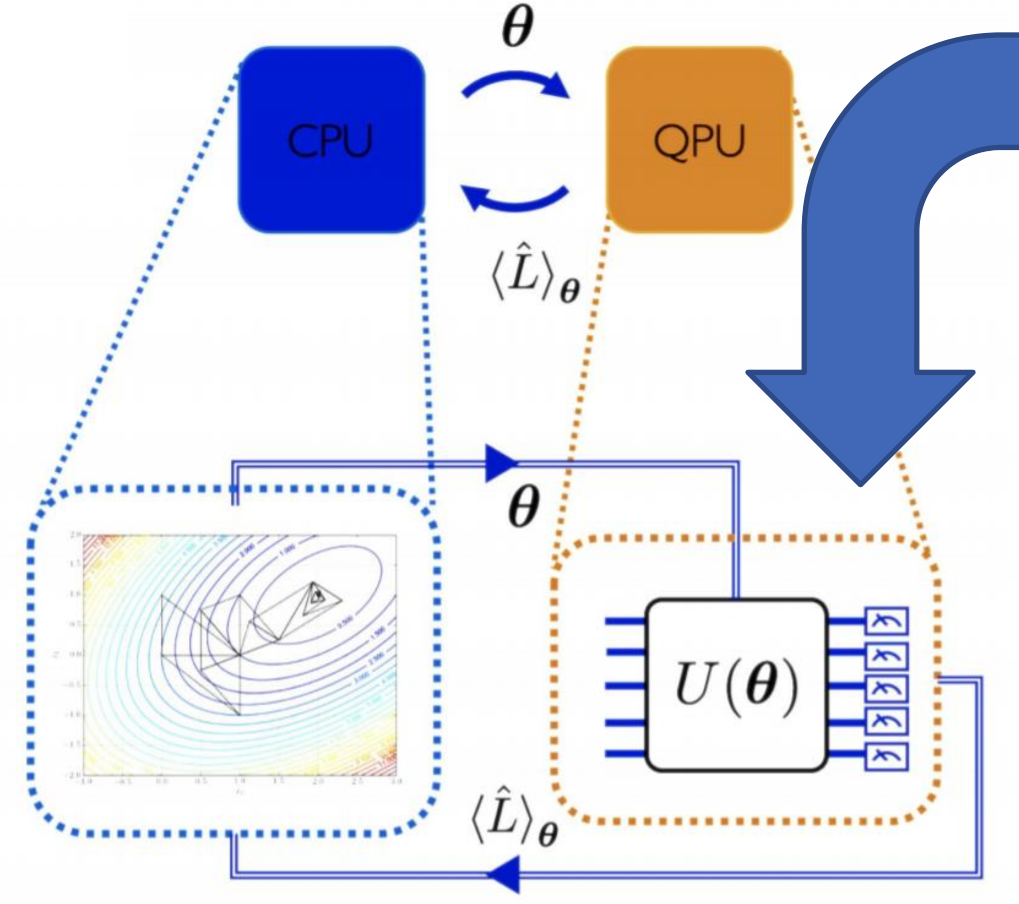
VARIATIONAL QUANTUM ALGORITHM

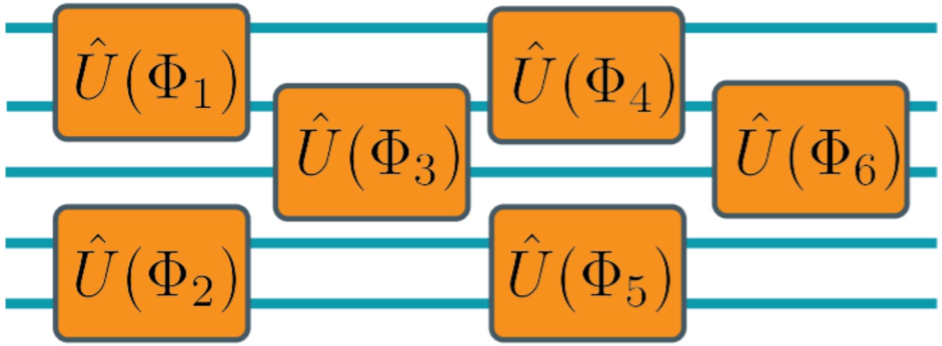
General idea



QUANTUM NEURAL NETWORK

Definition





- Sequence of continuously-parametrized rotations executed on QPU
- Measure observable expectation value (non-linearity)
- CPU optimization suggests new parameters to minimize the expectation value

https://qml.entropicalabs.io/

QUANTUM NEURAL NETWORK

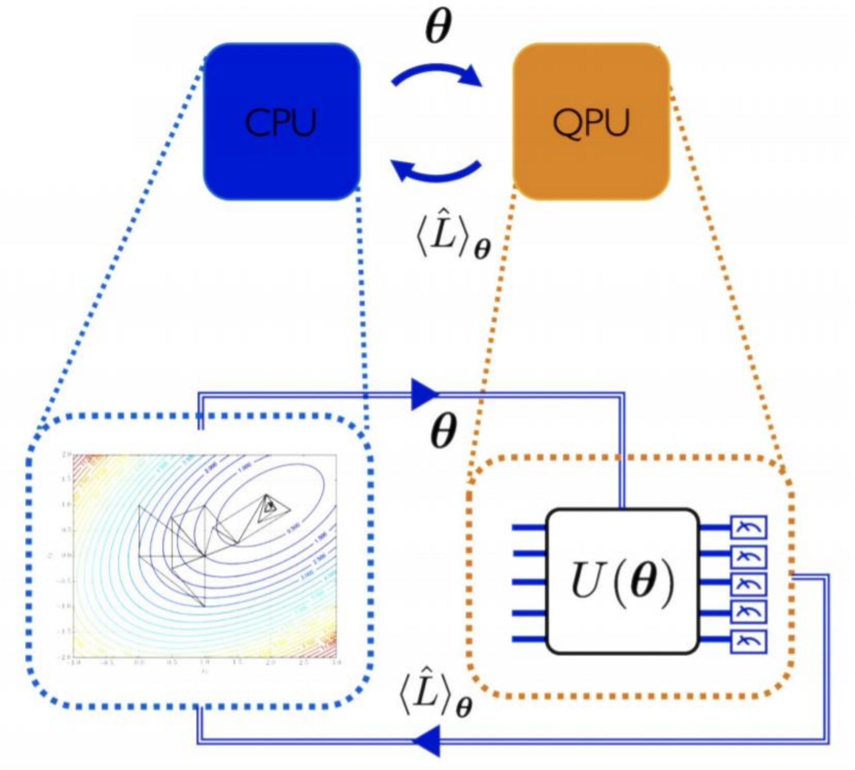
Problems

- Data must be prepared on the fly
- QPU needs full quantum program for each run (microseconds for each run)
- Relative high latency CPU-QPU (ms)

QUANTUM NEURAL NETWORK

Hybrid approach

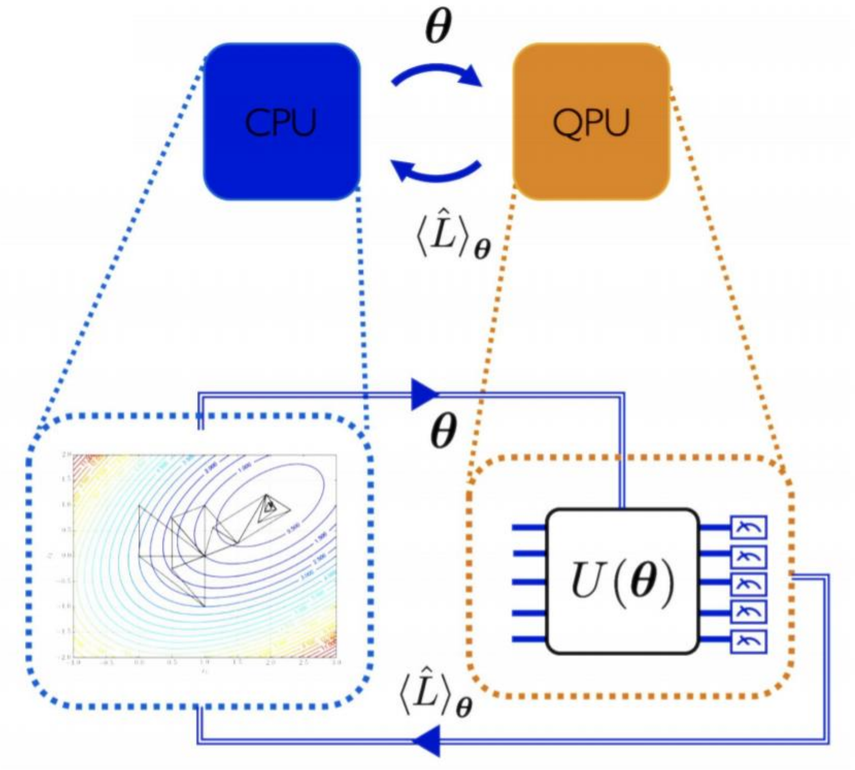
Classical optimization is used



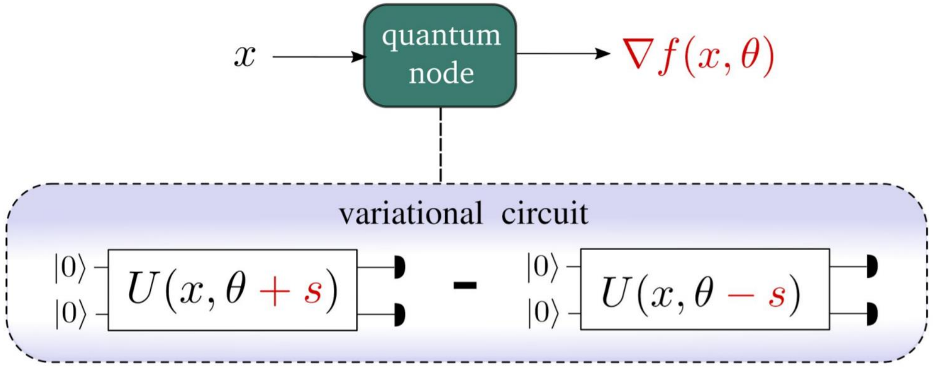
QUANTUM NEURAL NETWORK

Hybrid approach

Classical optimization is used



But the gradient is computed by QPU



Parameter shift

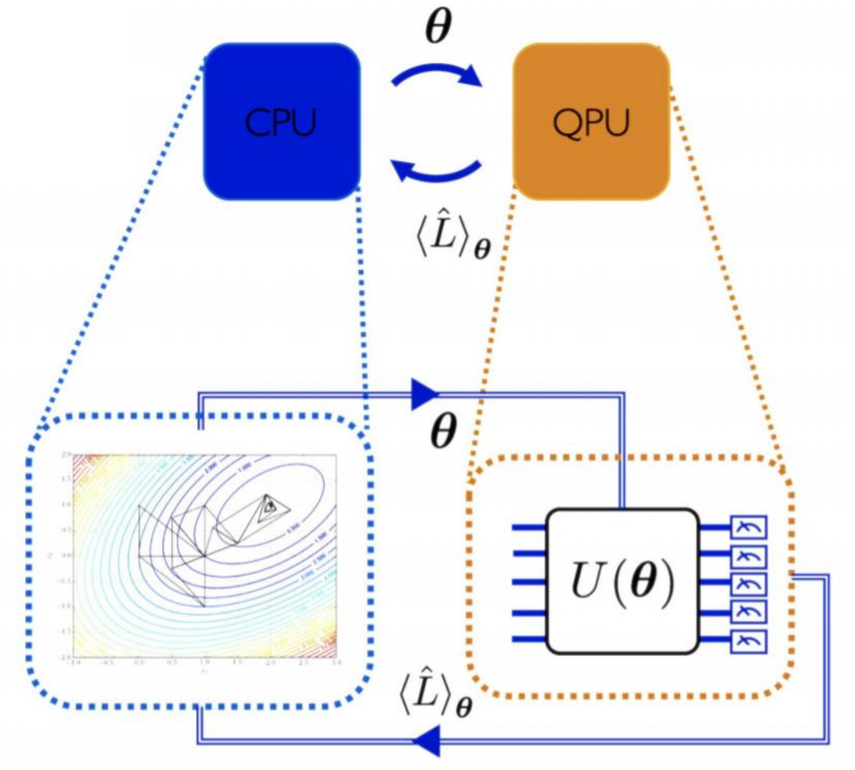
$$\partial_\theta f(\theta) = c[f(\theta + s) - f(\theta - s)]$$

CINECA QUANTUM COMPUTING LAB

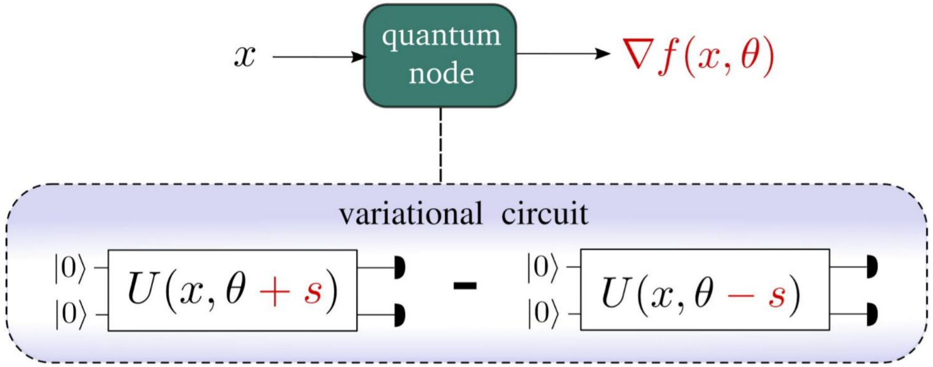
QUANTUM NEURAL NETWORK

Hybrid approach

Classical optimization is used



But the gradient is computed by QPU



Parameter shift

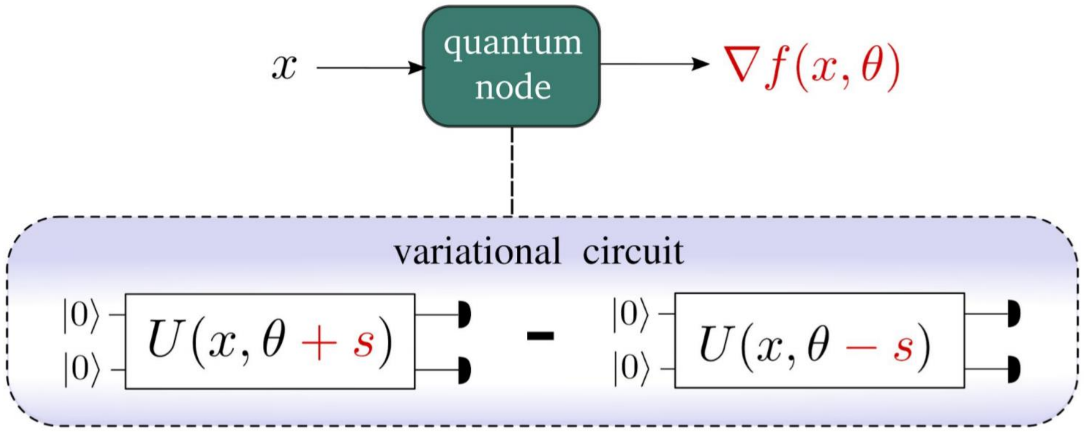
$$\partial_\theta f(\theta) = c[f(\theta + s) - f(\theta - s)]$$

Exact

CINECA QUANTUM COMPUTING LAB

QUANTUM NEURAL NETWORK

Parameter shift



State

$$|\psi_{i-1}\rangle = U_{i-1}(\theta_{i-1})\cdots U_1(\theta_1)U_0(x)|0\rangle$$

Operator (Observable)

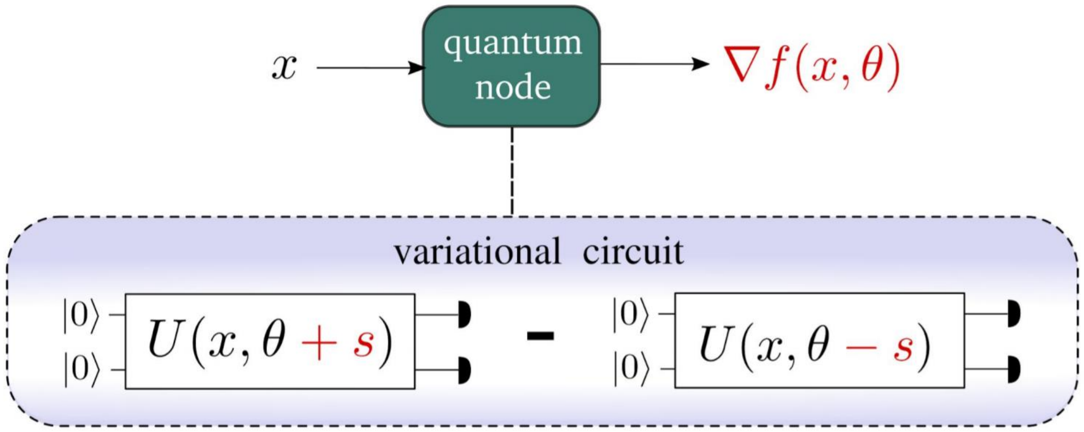
$$\hat{B}_{i+1} = U_N^\dagger(\theta_N)\cdots U_{i+1}^\dagger(\theta_{i+1})\hat{B}U_{i+1}(\theta_{i+1})\cdots U_N(\theta_N)$$

Expectation value

$$f(x;\theta) = \langle\psi_{i-1}|U_i^\dagger(\theta_i)\hat{B}_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle = \langle\psi_{i-1}|\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle$$

QUANTUM NEURAL NETWORK

Parameter shift



State

$$|\psi_{i-1}\rangle = U_{i-1}(\theta_{i-1})\cdots U_1(\theta_1)U_0(x)|0\rangle$$

Operator (Observable)

$$\hat{B}_{i+1} = U_N^\dagger(\theta_N)\cdots U_{i+1}^\dagger(\theta_{i+1})\hat{B}U_{i+1}(\theta_{i+1})\cdots U_N(\theta_N)$$

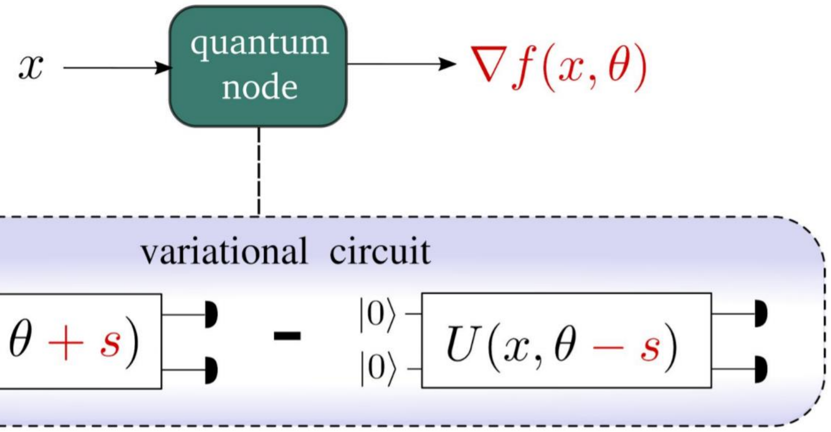
Expectation value

$$f(x;\theta) = \langle\psi_{i-1}|U_i^\dagger(\theta_i)\hat{B}_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle = \langle\psi_{i-1}|\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle$$

$$\nabla_{\theta_i} f(x;\theta) = \langle\psi_{i-1}|\nabla_{\theta_i}\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle \quad ?$$

QUANTUM NEURAL NETWORK

Parameter shift



State

$$|\psi_{i-1}\rangle = U_{i-1}(\theta_{i-1})\cdots U_1(\theta_1)U_0(x)|0\rangle$$

Operator (Observable)

$$\hat{B}_{i+1} = U_N^\dagger(\theta_N)\cdots U_{i+1}^\dagger(\theta_{i+1})\hat{B}U_{i+1}(\theta_{i+1})\cdots U_N(\theta_N)$$

Expectation value

$$f(x;\theta) = \langle\psi_{i-1}|U_i^\dagger(\theta_i)\hat{B}_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle = \langle\psi_{i-1}|\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle$$

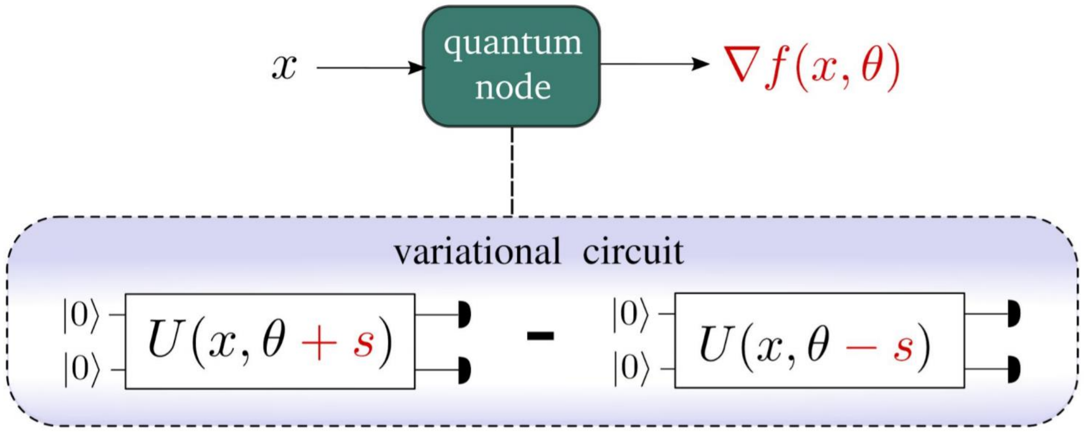
Pauli rotation

$$\boxed{U_i(\theta_i) = \exp\left(-i\frac{\theta_i}{2}\hat{P}_i\right)}$$

$$\nabla_{\theta_i}U_i(\theta_i) = -\frac{i}{2}\hat{P}_iU_i(\theta_i) = -\frac{i}{2}U_i(\theta_i)\hat{P}_i$$

QUANTUM NEURAL NETWORK

Parameter shift



$$\nabla_{\theta_i} f(x;\theta) = \frac{i}{2}\langle\psi_{i-1}|U_i^\dagger(\theta_i)\left(P_i\hat{B}_{i+1} - \hat{B}_{i+1}P_i\right)U_i(\theta_i)|\psi_{i-1}\rangle$$
$$= \frac{i}{2}\langle\psi_{i-1}|U_i^\dagger(\theta_i)\left[P_i, \hat{B}_{i+1}\right]U_i(\theta_i)|\psi_{i-1}\rangle,$$

State

$$|\psi_{i-1}\rangle = U_{i-1}(\theta_{i-1})\cdots U_1(\theta_1)U_0(x)|0\rangle$$

Operator (Observable)

$$\hat{B}_{i+1} = U_N^\dagger(\theta_N)\cdots U_{i+1}^\dagger(\theta_{i+1})\hat{B}U_{i+1}(\theta_{i+1})\cdots U_N(\theta_N)$$

Expectation value

$$f(x;\theta) = \langle\psi_{i-1}|U_i^\dagger(\theta_i)\hat{B}_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle = \langle\psi_{i-1}|\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle$$

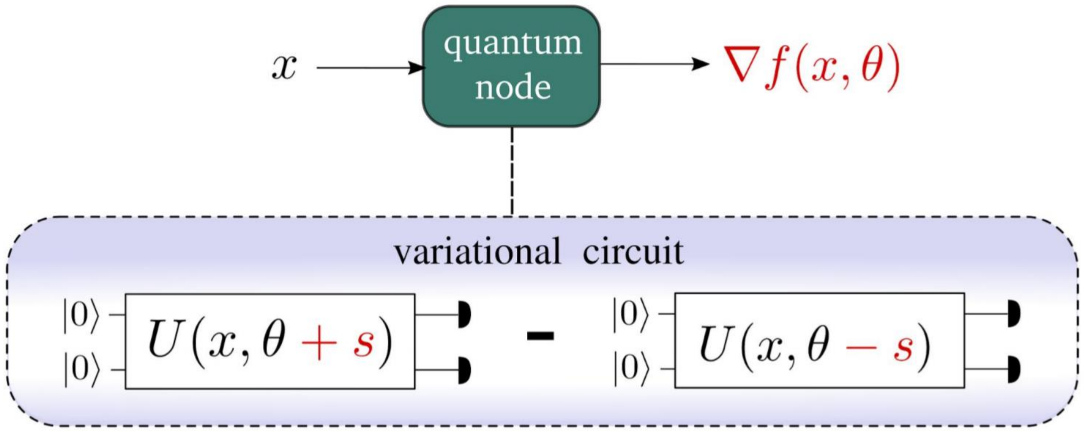
Pauli rotation

$$\boxed{U_i(\theta_i) = \exp\left(-i\frac{\theta_i}{2}\hat{P}_i\right)}$$

$$\nabla_{\theta_i}U_i(\theta_i) = -\frac{i}{2}\hat{P}_iU_i(\theta_i) = -\frac{i}{2}U_i(\theta_i)\hat{P}_i$$

QUANTUM NEURAL NETWORK

Parameter shift



$$\nabla_{\theta_i} f(x;\theta) = \frac{i}{2}\langle\psi_{i-1}|U_i^\dagger(\theta_i)\left(P_i\hat{B}_{i+1} - \hat{B}_{i+1}P_i\right)U_i(\theta_i)|\psi_{i-1}\rangle$$
$$= \frac{i}{2}\langle\psi_{i-1}|U_i^\dagger(\theta_i)\left[P_i, \hat{B}_{i+1}\right]U_i(\theta_i)|\psi_{i-1}\rangle,$$

Commutator $[X,Y] = XY - YX$

$$\left[\hat{P}_i, \hat{B}\right] = -i\left(U_i^\dagger\left(\frac{\pi}{2}\right)\hat{B}U_i\left(\frac{\pi}{2}\right) - U_i^\dagger\left(-\frac{\pi}{2}\right)\hat{B}U_i\left(-\frac{\pi}{2}\right)\right)$$

State

$$|\psi_{i-1}\rangle = U_{i-1}(\theta_{i-1})\cdots U_1(\theta_1)U_0(x)|0\rangle$$

Operator (Observable)

$$\hat{B}_{i+1} = U_N^\dagger(\theta_N)\cdots U_{i+1}^\dagger(\theta_{i+1})\hat{B}U_{i+1}(\theta_{i+1})\cdots U_N(\theta_N)$$

Expectation value

$$f(x;\theta) = \langle\psi_{i-1}|U_i^\dagger(\theta_i)\hat{B}_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle = \langle\psi_{i-1}|\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle$$

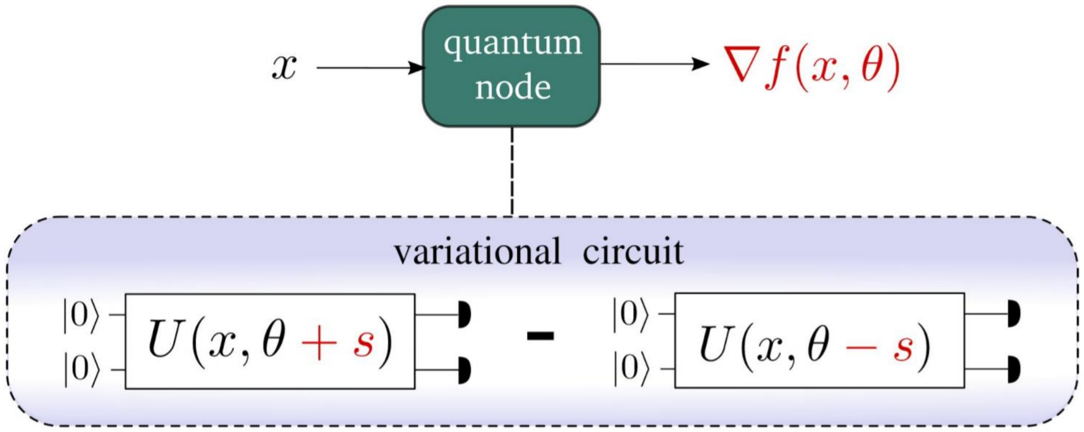
Pauli rotation

$$\boxed{U_i(\theta_i) = \exp\left(-i\frac{\theta_i}{2}\hat{P}_i\right)}$$

$$\nabla_{\theta_i} U_i(\theta_i) = -\frac{i}{2}\hat{P}_i U_i(\theta_i) = -\frac{i}{2}U_i(\theta_i)\hat{P}_i$$

QUANTUM NEURAL NETWORK

Parameter shift



State

$$|\psi_{i-1}\rangle = U_{i-1}(\theta_{i-1})\cdots U_1(\theta_1)U_0(x)|0\rangle$$

Operator (Observable)

$$\hat{B}_{i+1} = U_N^\dagger(\theta_N)\cdots U_{i+1}^\dagger(\theta_{i+1})\hat{B}U_{i+1}(\theta_{i+1})\cdots U_N(\theta_N)$$

Expectation value

$$f(x;\theta) = \langle\psi_{i-1}|U_i^\dagger(\theta_i)\hat{B}_{i+1}U_i(\theta_i)|\psi_{i-1}\rangle = \langle\psi_{i-1}|\mathcal{M}_{\theta_i}(\hat{B}_{i+1})|\psi_{i-1}\rangle$$

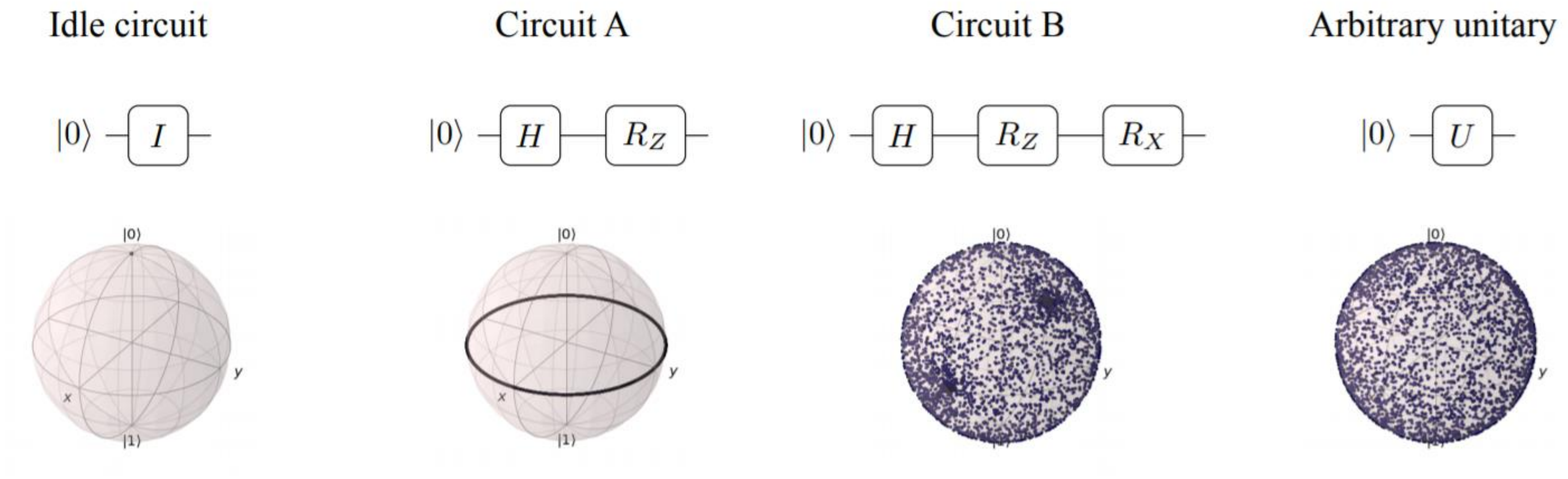
$$\nabla_\theta f(x;\theta) = \frac{1}{2}\left[f(x;\theta+\tfrac{\pi}{2}) - f(x;\theta-\tfrac{\pi}{2})\right]$$

QUANTUM NEURAL NETWORK

Variational cirucit ansatz

Choose a compromise between

- Computational cost (depth)
- Expressiveness of the unitary operator



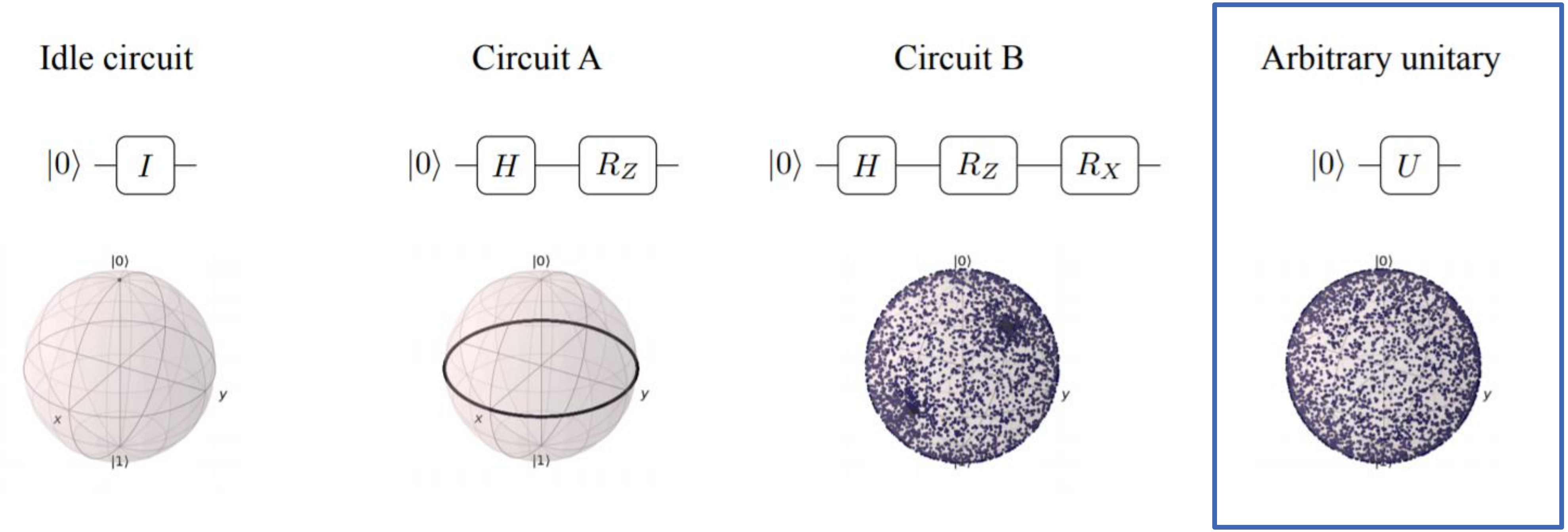
QUANTUM NEURAL NETWORK

Variational cirucit ansatz

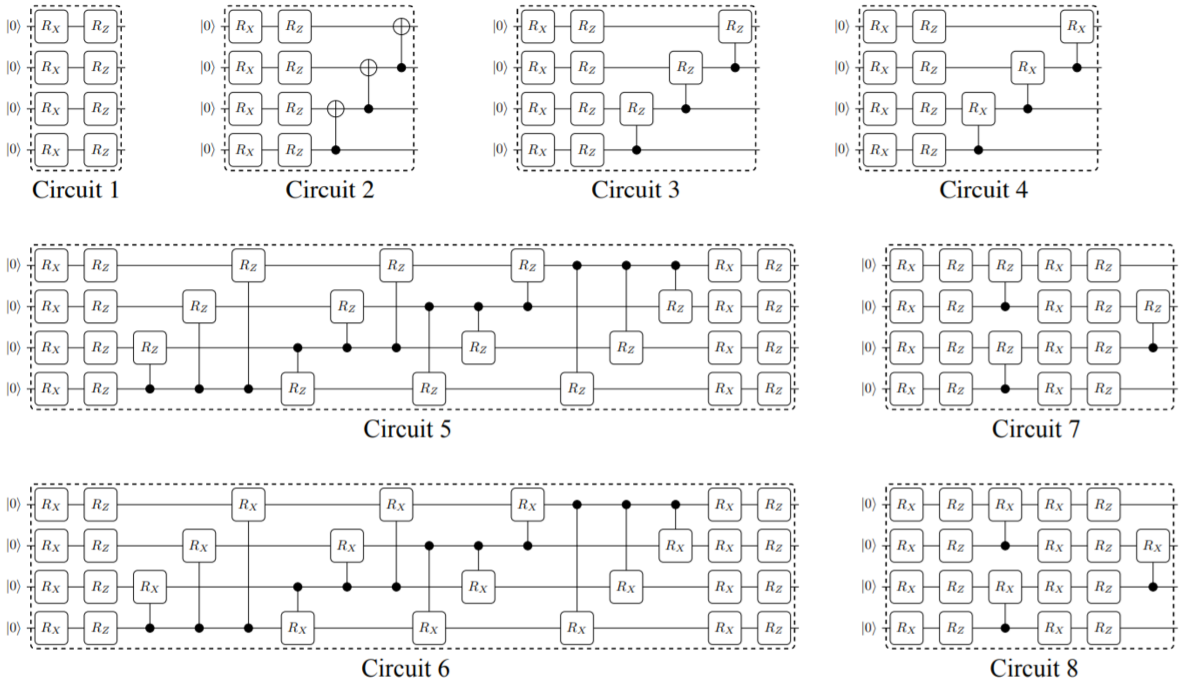
Choose a compromise between

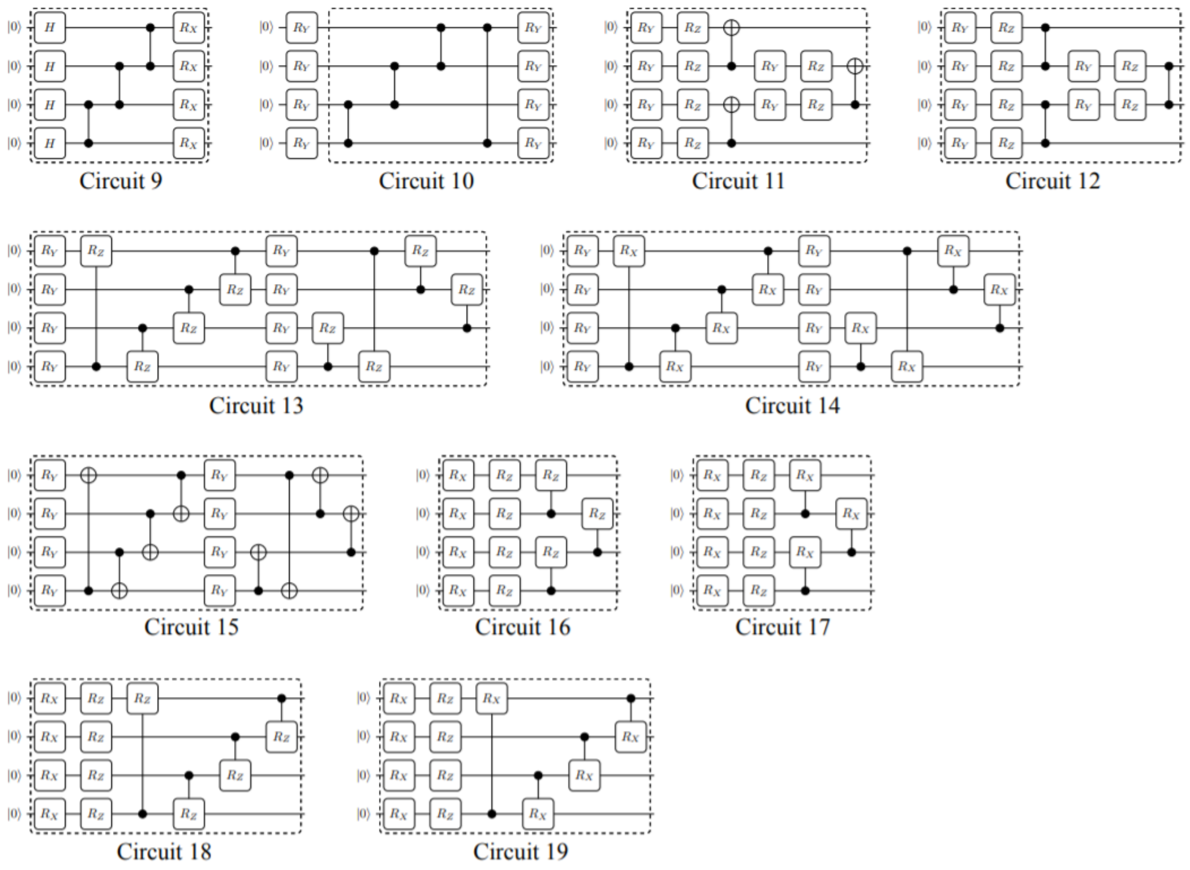
- Computational cost (depth)
- Expressiveness of the unitary operator

TARGET

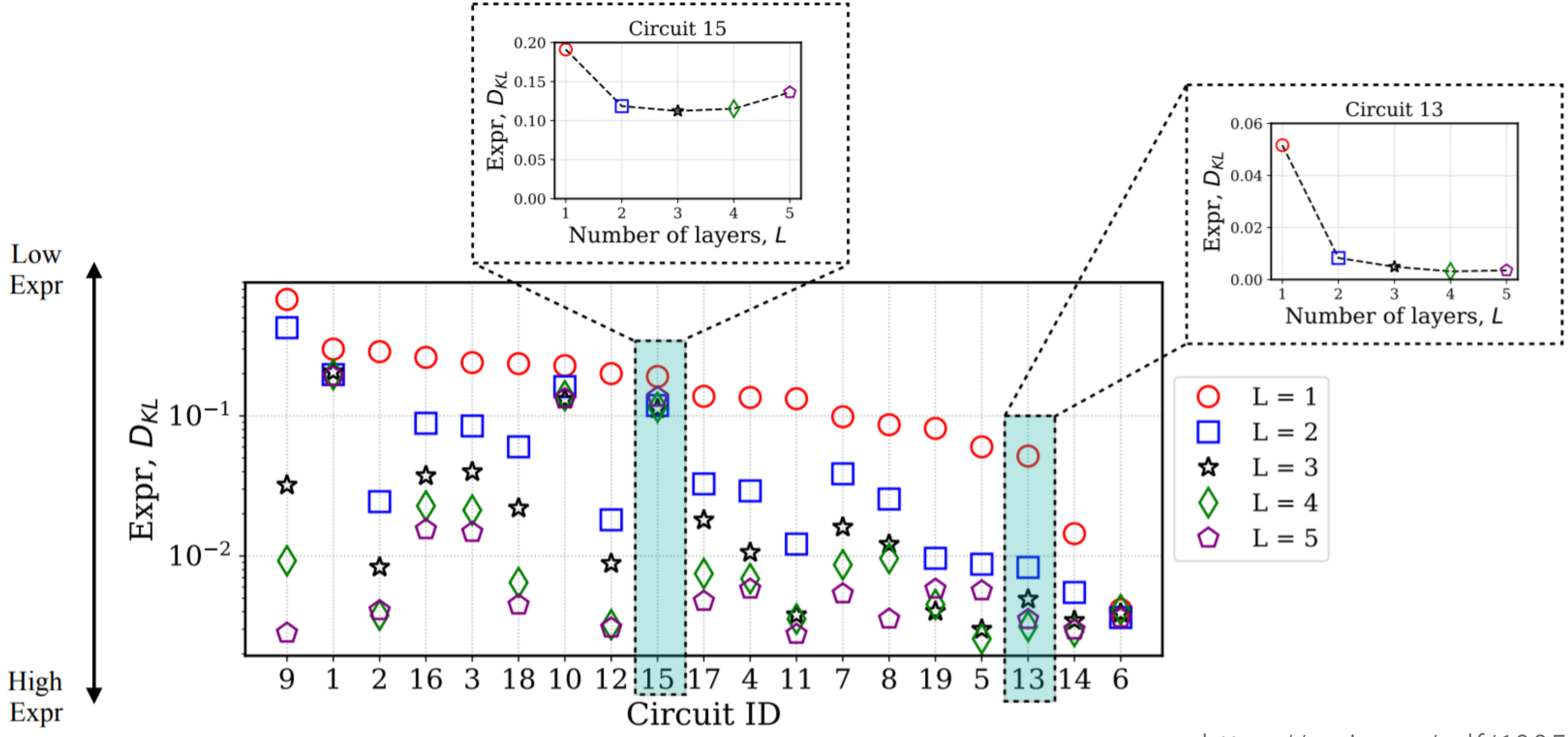


QUANTUM NEURAL NETWORK



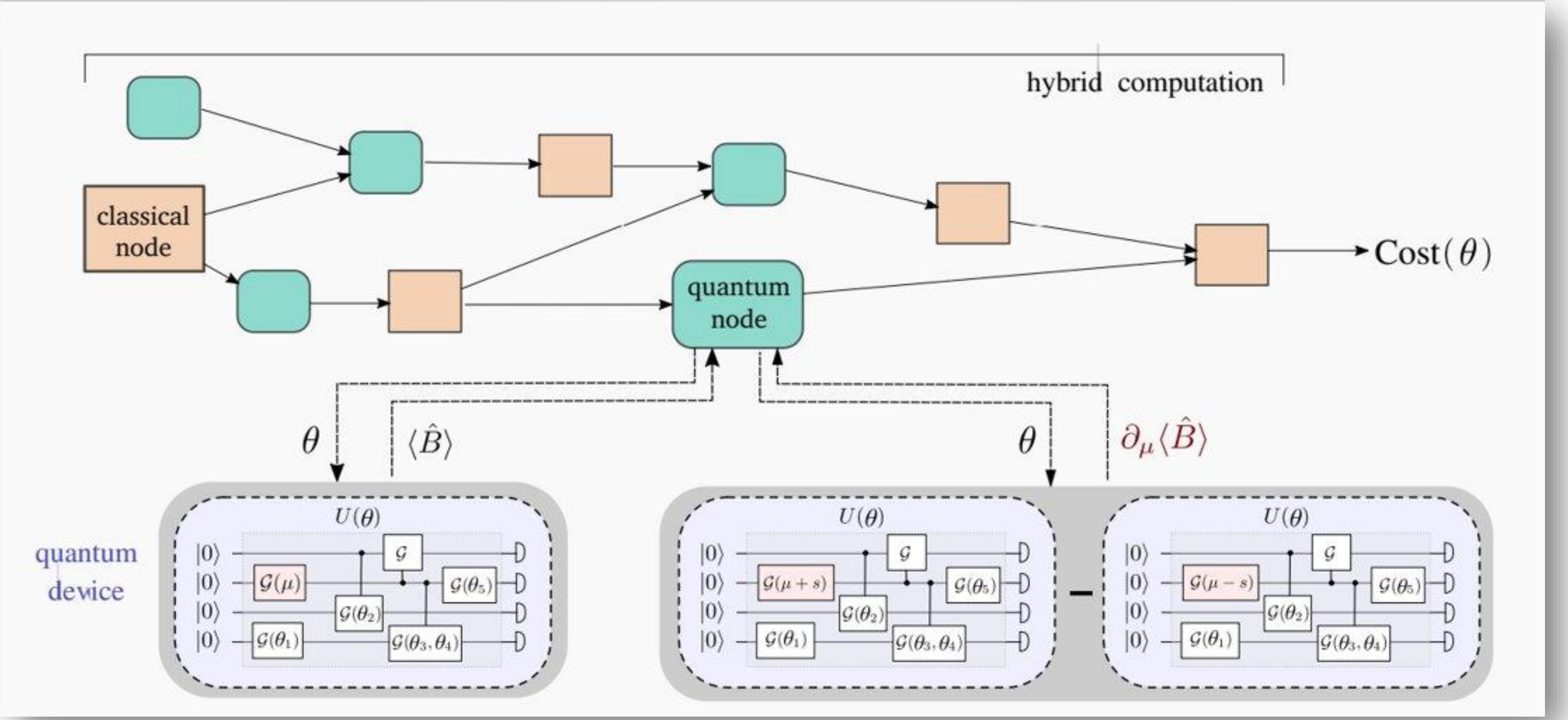


QUANTUM NEURAL NETWORK



https://arxiv.org/pdf/1905.10876

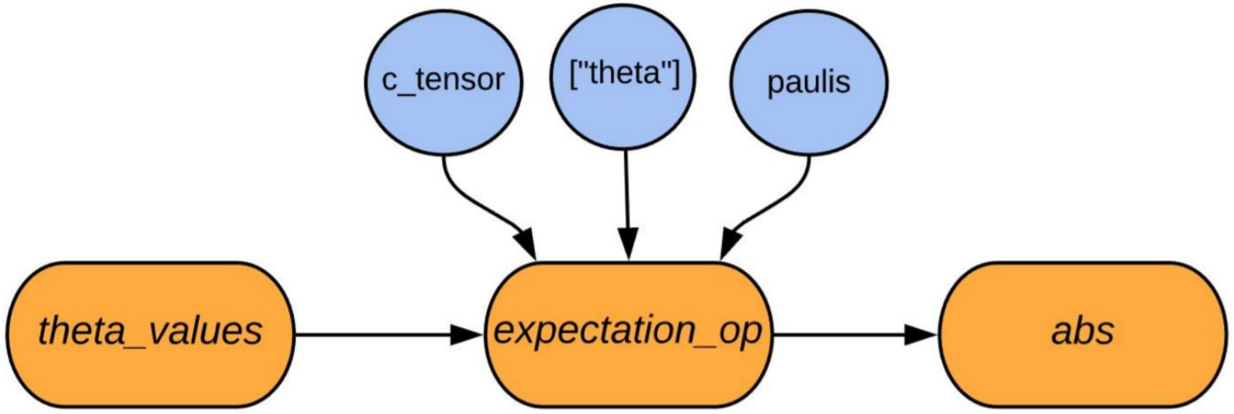
HYBRID QUANTUM-CLASSICAL COMPUTING



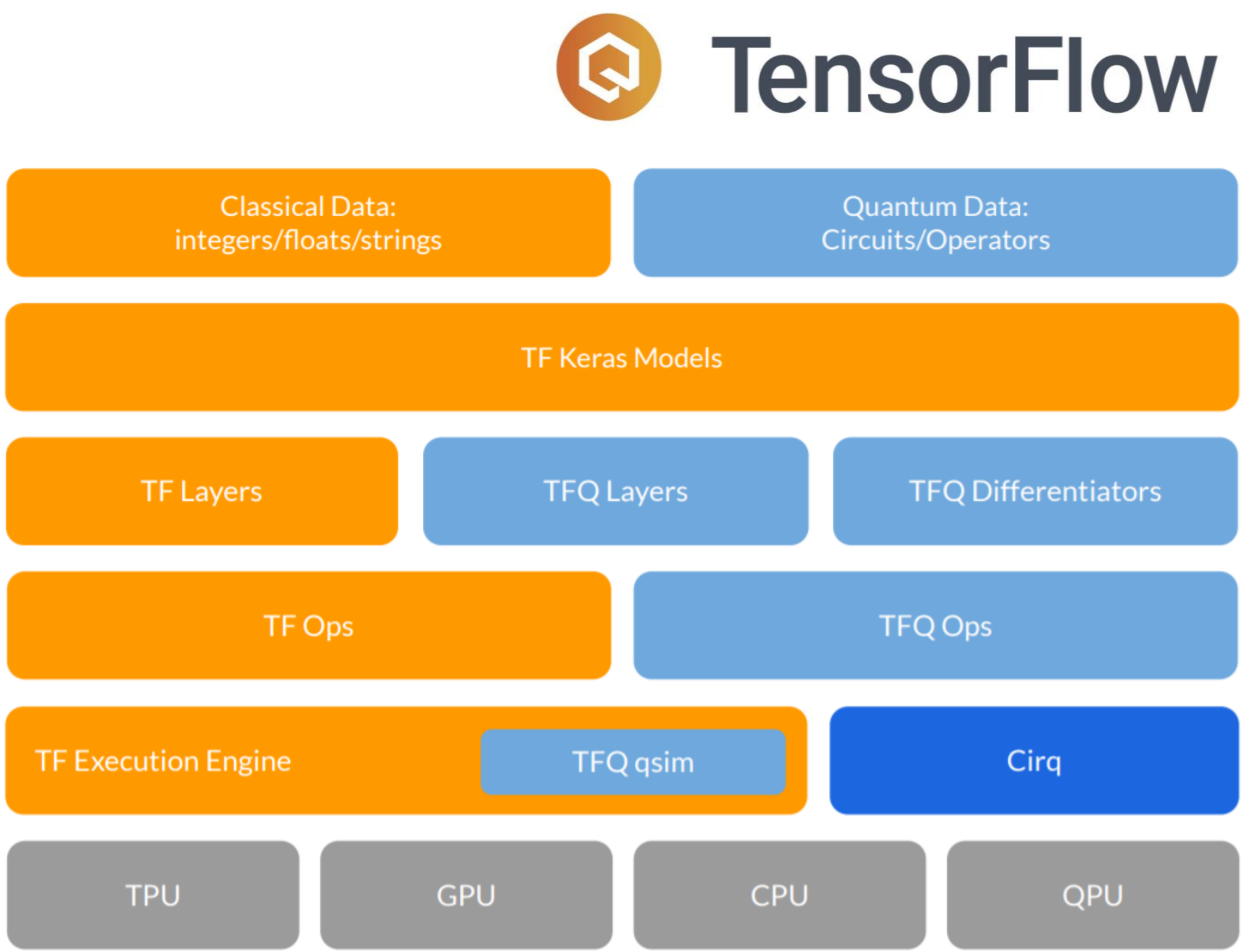
TENSORFLOW QUANTUM

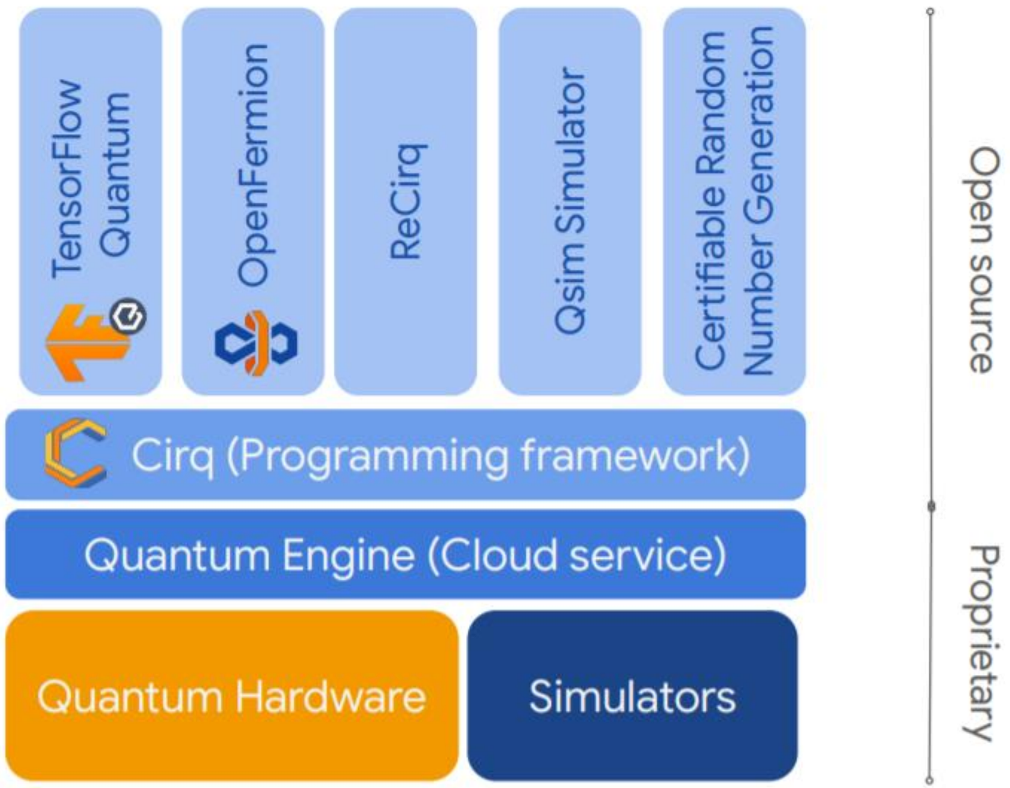
TensorFlow Quantum

- Circuits written with Cirq and converted into tensors
  - Expectation value of an operator (OPs)
- Backpropagation for parameter optimization



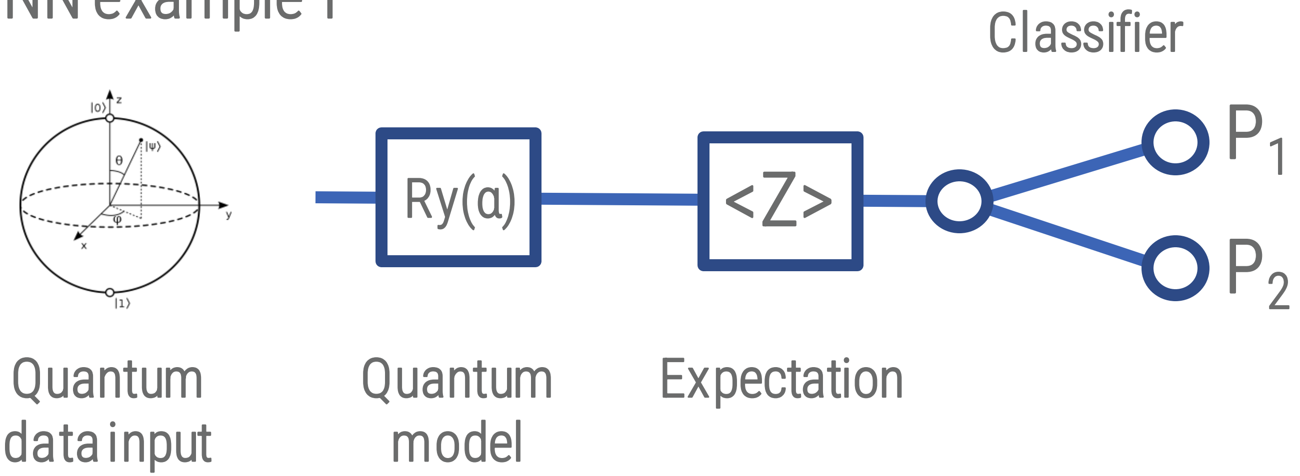
TENSORFLOW QUANTUM





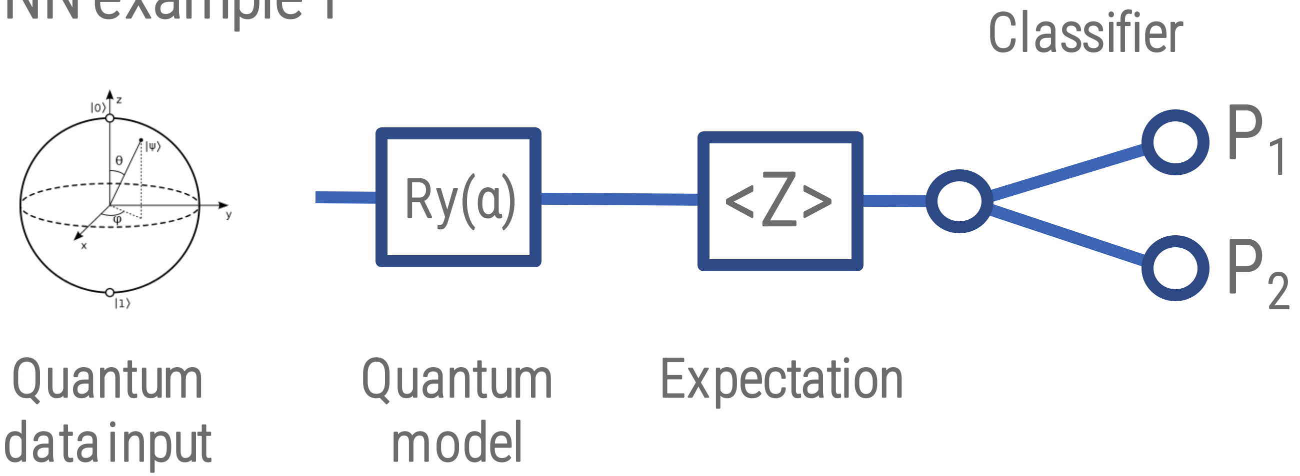
EXAMPLES

QNN example 1

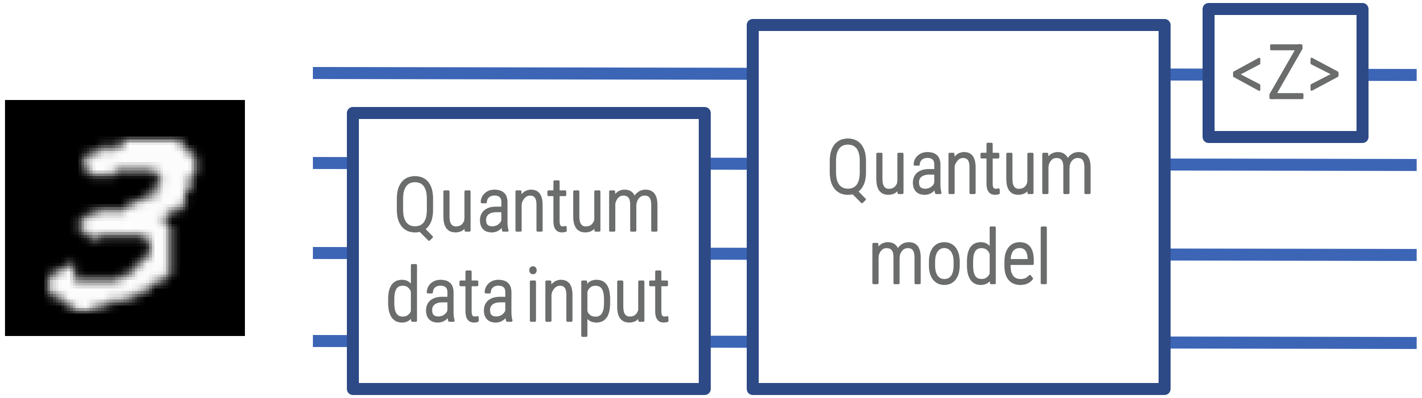


EXAMPLES

QNN example 1



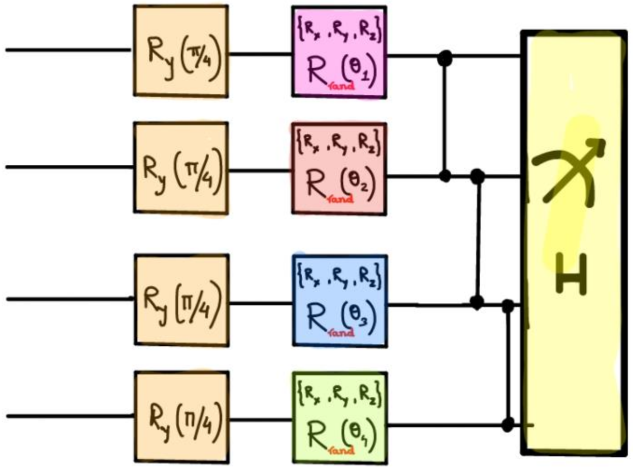
QNN example 2: mnist classification



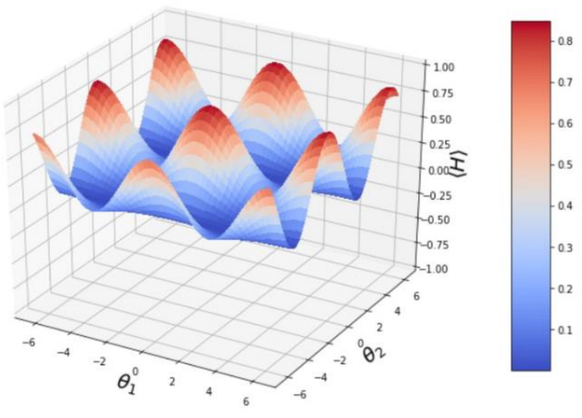
BARREN PLATEAUS

Pauli Rotation

$$\exp\left\{ i\boldsymbol{P}\frac{\theta}{2} \right\} = \boldsymbol{I}\cos\frac{\theta}{2} + i\boldsymbol{P}\sin\frac{\theta}{2}$$



A random variational circuit with gates chosen from the set {RX, RY, RZ}



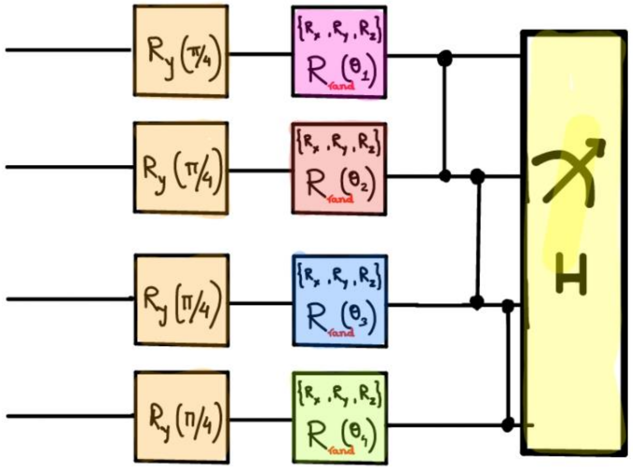
Expectation value of a Hermitian observable along a slice in the parameter space

BARREN PLATEAUS

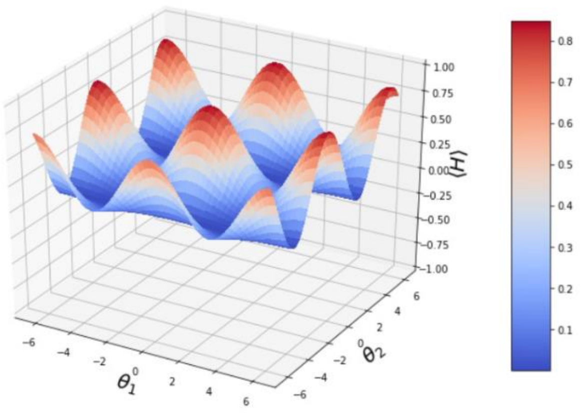
Pauli Rotation

$$\exp\left\{i\boldsymbol{P}\frac{\theta}{2}\right\} = \boldsymbol{I}\cos\frac{\theta}{2} + i\boldsymbol{P}\sin\frac{\theta}{2}$$

Periodic function → Saddle points



A random variational circuit with gates chosen from the set {RX, RY, RZ}

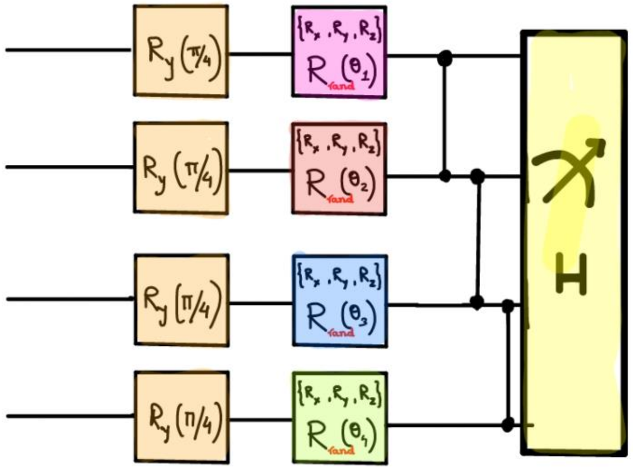


Expectation value of a Hermitian observable along a slice in the parameter space

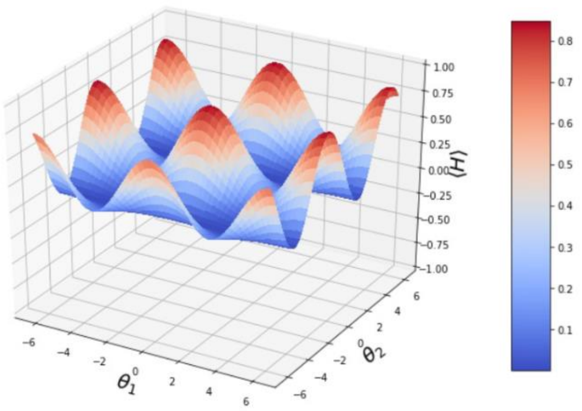
BARREN PLATEAUS

Pauli Rotation $$\exp\left\{i\boldsymbol{P}\frac{\theta}{2}\right\} = \boldsymbol{I}\cos\frac{\theta}{2} + i\boldsymbol{P}\sin\frac{\theta}{2}$$

Periodic function → Saddle points



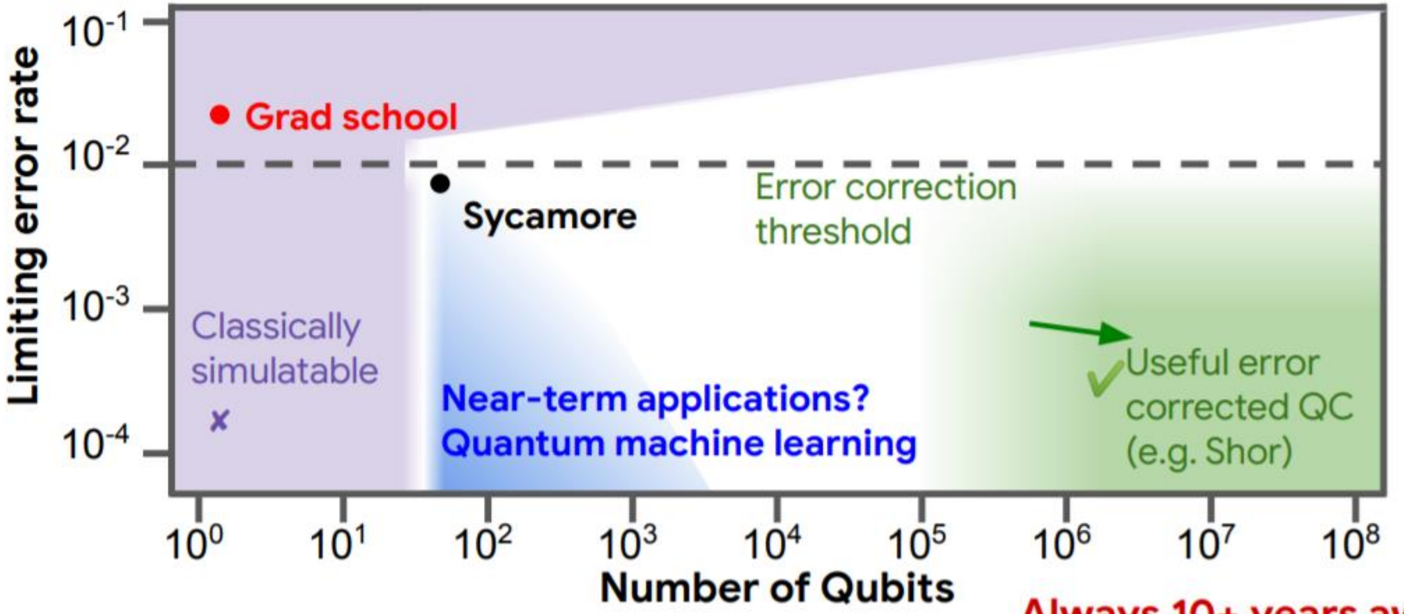
A random variational circuit with gates chosen from the set {RX, RY, RZ}



Expectation value of a Hermitian observable along a slice in the parameter space

- *In classical optimization, it is suggested that saddle points, not local minima, provide a fundamental impediment to rapid high-dimensional non-convex optimization.* (Dauphin et al., 2014)
- *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.* (McClean et al., 2018)

FUTURE PERSPECTIVES



Always 10+ years away!?

- QRAM for data encoding
- Fault tolerant QC for better variational circuit ansatz
- More qubits

Third generation of QML

- No barren plateaus
- No heuristic model