# Data encoding and quantum machine learning




**A lecture series by IQM and HS RM**

Authors: Stefan Seegerer and Nadia Milazzo

Last Updated 10/2025

www.meetiqm.com

# Machine learning

Given a task $T$ to a computer program, we say that it **learns** from experience $E$ if:

$$P(T) \propto E$$

Where $P$ is some performance metric of the program

*Mitchell, T. M. (1997), Machine learning (McGraw Hill)*

## SUPERVISED LEARNING
### Labeled data

Given examples from a **joint** probability distribution $P(X, Y)$, learn $P(Y = y|X = x)$

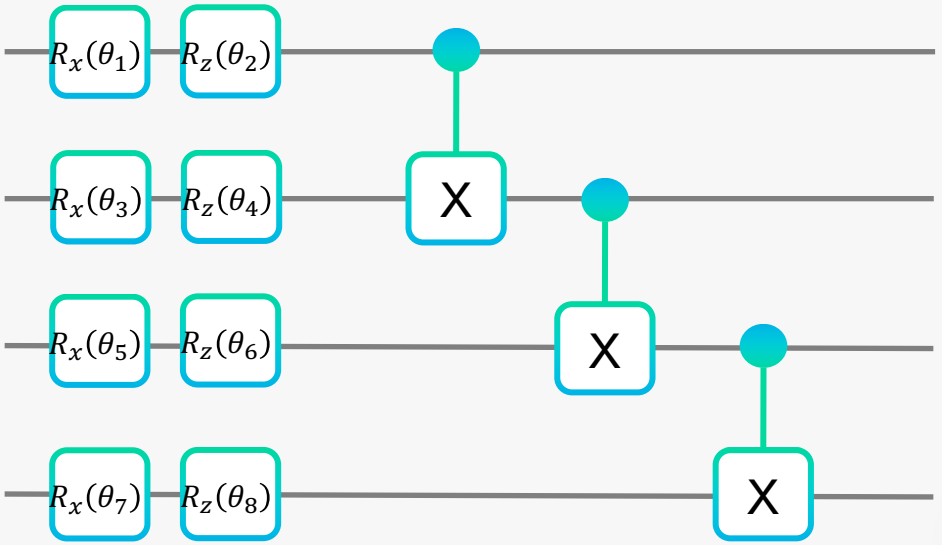
## UNSUPERVISED LEARNING
### Unlabeled data

Given examples $\mathrm{x} \in X$, learn the probability distribution $P(X)$, or some relevant property of $P(X)$

**Machine learning models are efficient in finding patterns in data**

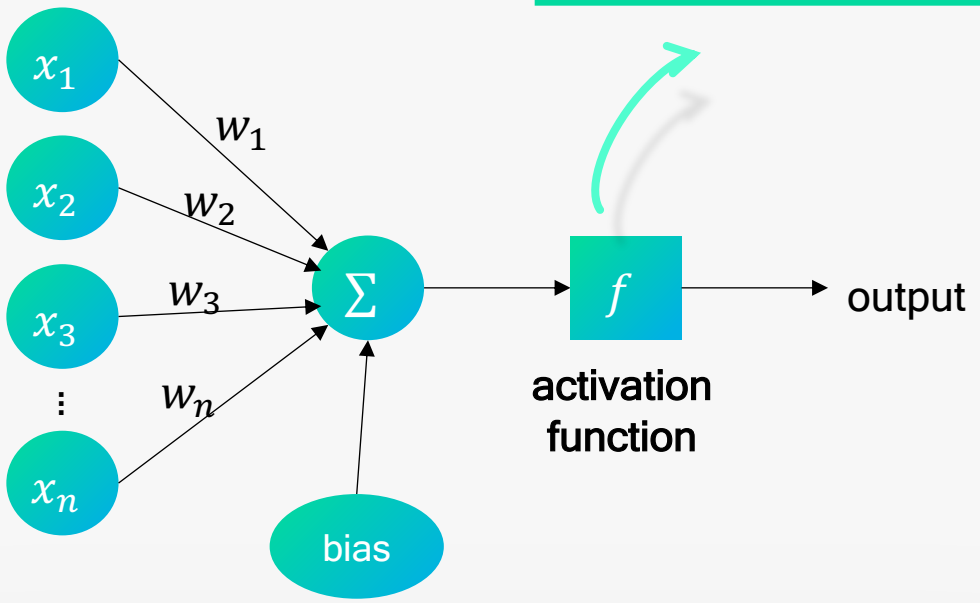
# Quantum computing

## Quantum computing programs
## Unitary circuits




LINEAR DYNAMICS

NON-LINEARITY

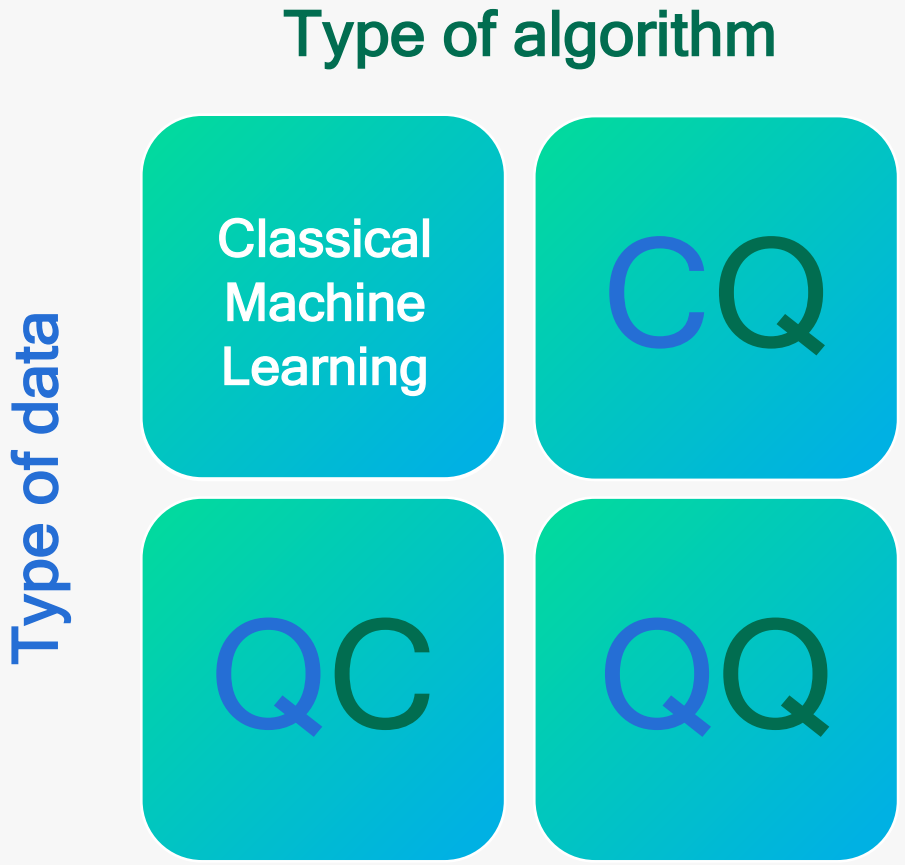



**Non-linearities in quantum physics can come from:**

- measurement
- post-selection
- coupling system-environment

# How to merge quantum computing and machine learning?

# Data and model




Can quantum computing enhance or innovate machine learning techniques for classical data?

- Quantum kernel methods
- Quantum neural networks

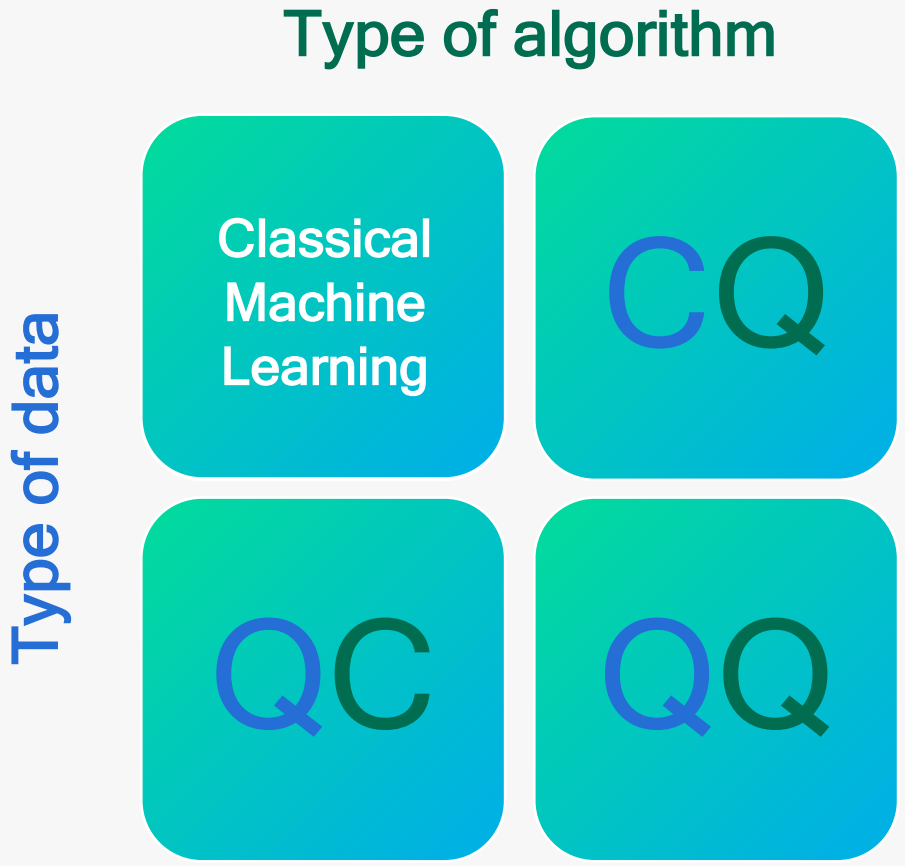
Can quantum-enhanced machine learning help with quantum data?

- Quantum phases of matter
- Many-body systems

Can machine learning help in quantum tasks?

- Quantum state tomography
- Controlling and preparing qubits

# Data and model




Can quantum computing enhance or innovate machine learning techniques for classical data?

- Quantum kernel methods
- Hybrid QNNs

Can quantum-enhanced machine learning help with quantum data?

- Quantum phases of matter
- Many-body systems

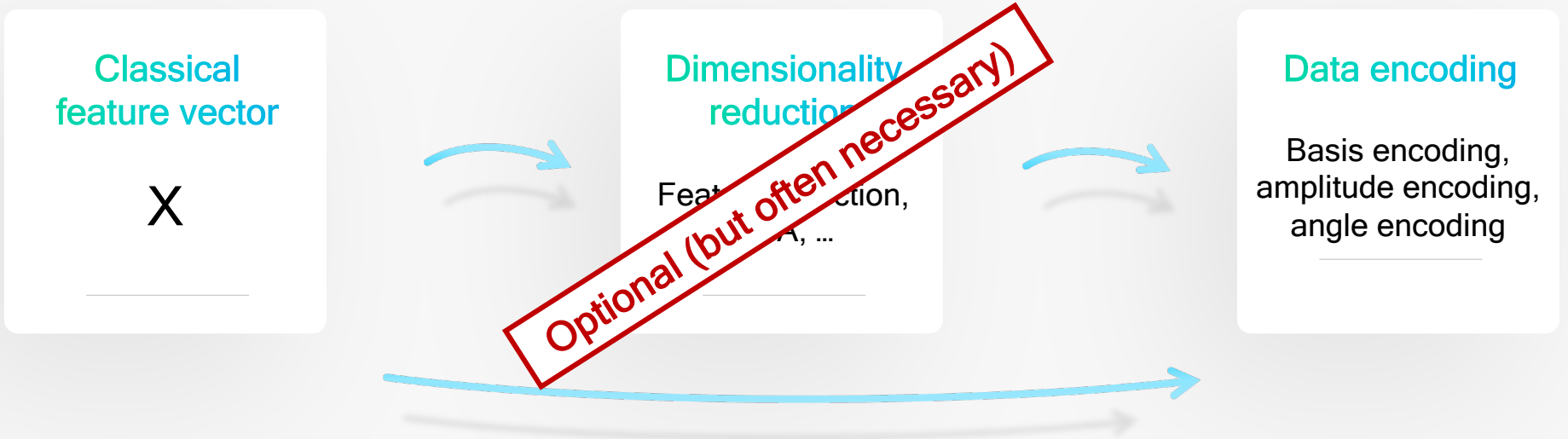
Can machine learning help in quantum tasks?

- Quantum state tomography
- Controlling and preparing qubits

# 1. Data encoding

# How to encode classical data into a quantum state?

The first step of a quantum machine learning algorithm is the **encoding** of classical data in a **quantum circuit***, i.e., in the so-prepared **quantum state.**




** For gate-based technologies*

The quantum circuit encoding the data is usually called a **feature map**

# Basis encoding

- **Idea**: Encode binary representation of data in computational **basis states**
- **Requires**: $\mathrm{N} = k \cdot n$ qubits for a vector of length $k$ with $n$ binary digits for each vector component

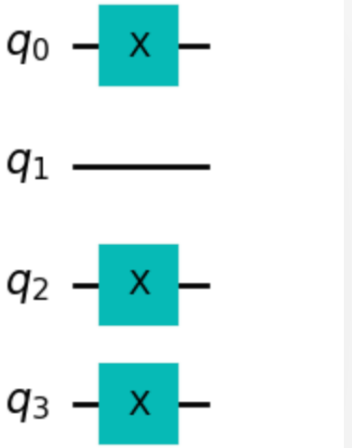
Classical feature vector

$$x = \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 10_2 \\ 11_2 \end{pmatrix}$$

Quantum encoding

$$|x\rangle = |\underbrace{10}\underbrace{11}\rangle$$






$$|\psi\rangle = \bigotimes_{i=0}^{N-1} X^{\boldsymbol{x}_i} |0\rangle^{\otimes N}$$

# Amplitude encoding

- **Idea**: Encode data in the amplitudes of a quantum state
- **Requires**: $\lceil \log_2(k) \rceil$ qubits for a vector of length $k$

Classical feature vector

$$x = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 3 \end{pmatrix}$$

Quantum encoding

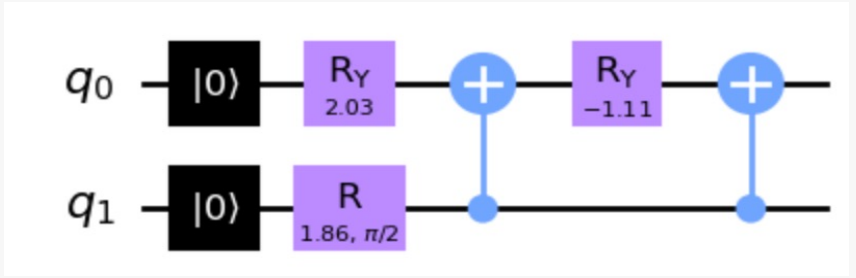
$$|x\rangle$$

$$|x\rangle = \frac{1}{\sqrt{2^2 + 1^2 + 0^2 + 3^2}} (2|00\rangle + 1|01\rangle + 0|10\rangle + 3|11\rangle)$$

We need the normalization constant!

$$\sum_{i=1}^{k} |x_i|^2 = |\alpha|^2$$

State preparation



# Angle encoding

- **Idea**: Rescale data and encode them in angle rotations
- **Requires**: $k$ qubits for a vector of length $k$

Classical feature vector

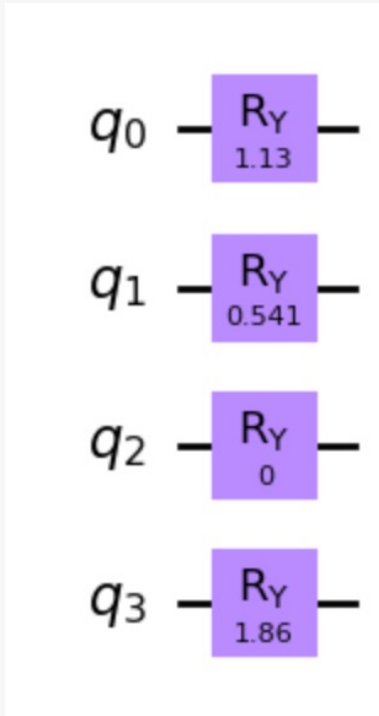
$$x = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 3 \end{pmatrix}$$

Data rescaling

$$x \in (0, 2\pi]$$

Quantum encoding

$$|x\rangle$$




**It could be any rotation!**

# Pros & cons of different encodings

## Basis encoding

✓ **Minimal** gate operations for state preparation

✗ **# of qubits** = # of binary digits to represent classical data

## Amplitude encoding

✓ **Good scaling** of # of qubits with classical data dimension

✗ **Inefficient** state preparation

## Angle encoding

✓ **Simple** state preparation - shallow circuit

✗ **No** representation of feature **dependencies** in classical data

# Note on non-linearity

We have seen that a feature map is an encoding $\boldsymbol{\Phi}$ such that:

$\boldsymbol{X}$: input space of classical data

$$\boldsymbol{\Phi}: \boldsymbol{X} \rightarrow \boldsymbol{H}$$

$\boldsymbol{H}$: Hilbert space of the quantum state encoding $x \in X$

- Consider the angle encoding of input $\boldsymbol{x}$ using the $\boldsymbol{R_y}$ gate:

$$x \rightarrow |\varphi(x)\rangle = R_y(x)|0\rangle = \cos\left(\frac{x}{2}\right)|0\rangle + \sin\left(\frac{x}{2}\right)|1\rangle$$



The amplitudes of the quantum state are **non-linear** in the input x

The needed non-linearities for a quantum machine learning model come from the data encoding step, aka the feature map

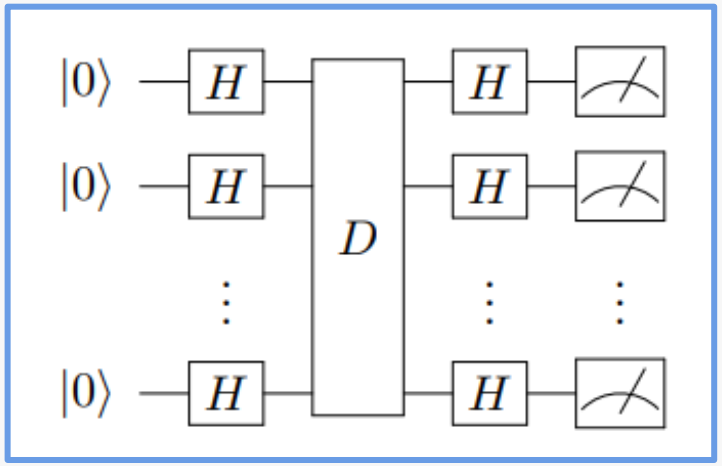
# 2. Examples of feature maps

# Parametrized quantum circuits

- Once the data are encoded in a quantum state, one can add further **unitary operations** to the state preparation circuit
- Common choices in the literature are **entangling operations** (both feature-dependent or not, e.g., ZZ or CNOT gates)
- In the search for quantum advantage, feature maps that are **classically hard** to simulate should be considered

Instantaneous Quantum Polynomial-Time **(IQP) circuits**

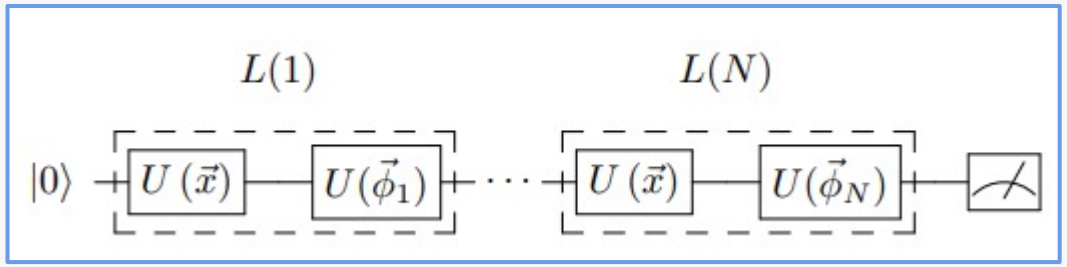
$$|x_i\rangle = \varphi(x_i) = U_z(x_i)\, H^{\otimes n}\, U_z(x_i)\, H^{\otimes n}\, |0\rangle^{\otimes n}$$




**It is hard for a classical sampler to produce samples from the output distribution of the IQP circuit**

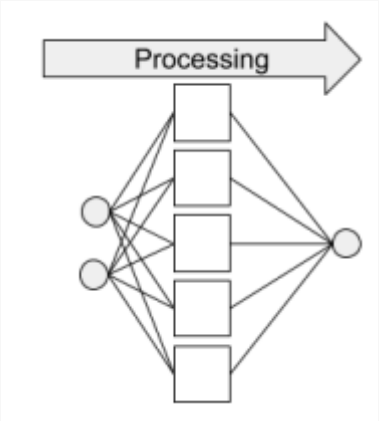
# Data re-uploading

**The idea**: alternate layers of data encoding and parametrized unitaries and repeat that N times




This was shown to enable the construction of a universal classifier with a **single qubit**!

Analogy




Classical NN take the same input many times

**Circumventing the limitations of no-cloning theorem**

| Advantages | Disadvantages |
|---|---|
| Potential enhancement of the performance and expressive power of QML models | Deeper circuits, harder to run on current quantum technologies due to hardware noise |

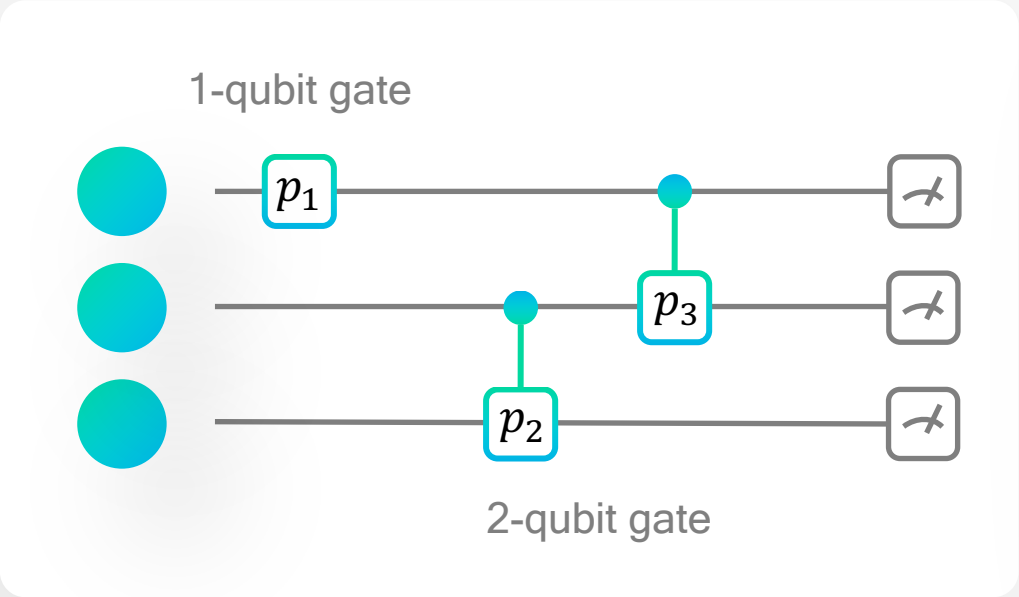


*Quantum 4, 226 (2020)*

# 3. Quantum routines in ML

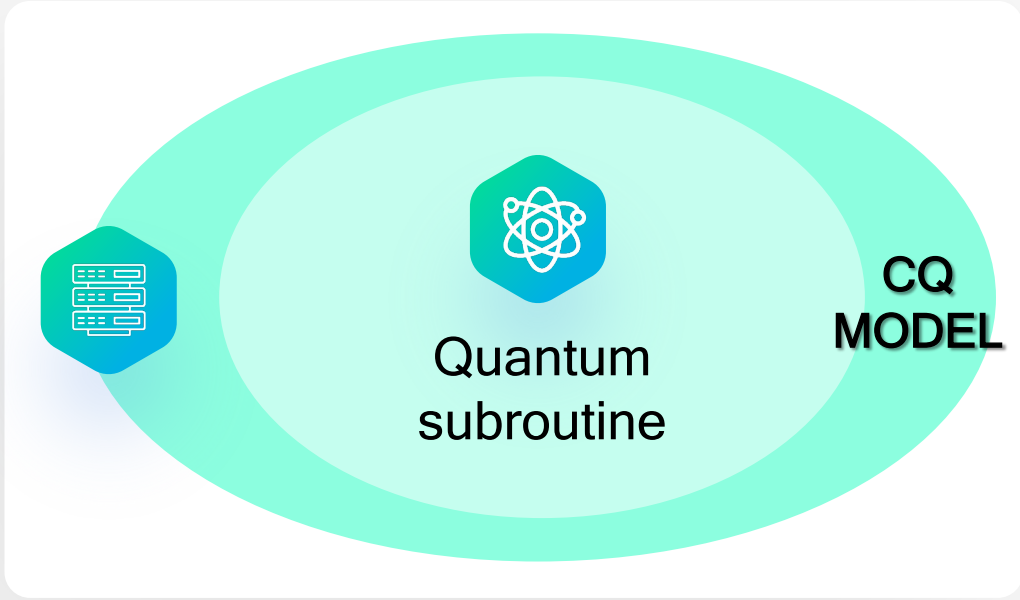
# Multiple forms of CQ

- Use a **variational parametrized quantum circuit** as the model




Optimize $p_1$, $p_2$ and $p_3$ based on measurement results

- Running only some parts of an ML pipeline on a quantum computer

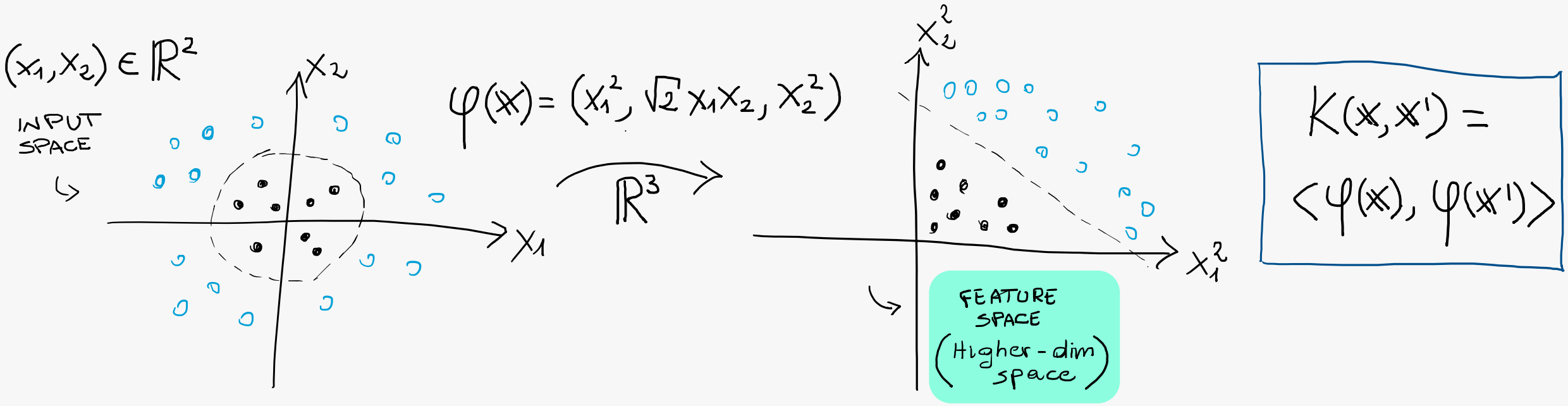



Example: quantum kernel estimation

# Quantum kernels

# The kernel method

**Example:** Polynomial kernel




The **kernel function** quantifies the distance between sample points in the Hilbert space

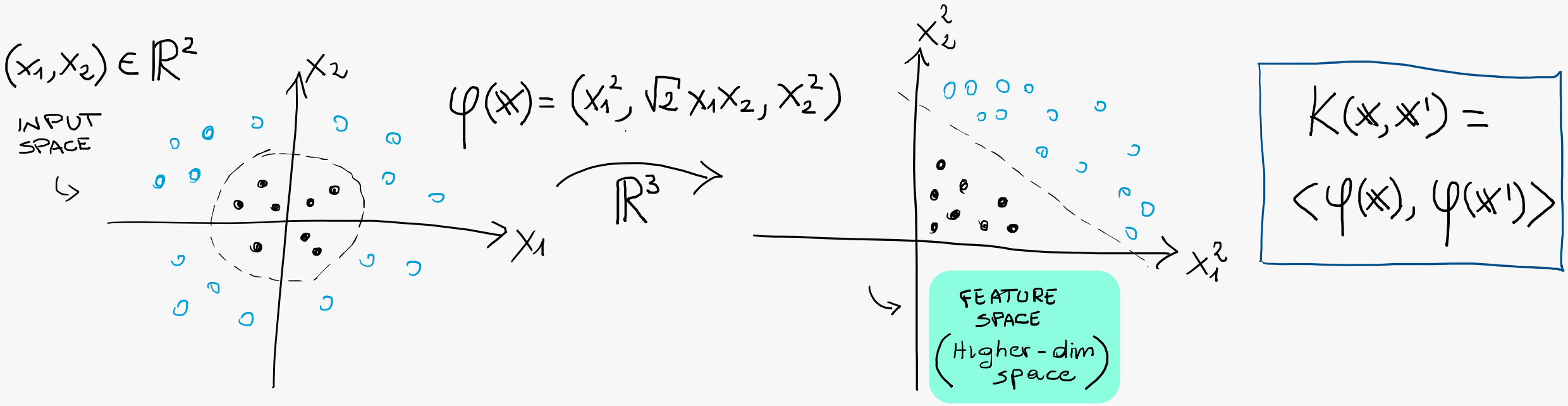
**Inner product**

$$k(x, x') = |\langle \varphi(x) | \varphi(x') \rangle|^2$$

φ is the feature map!

# The kernel method

**Example:** Polynomial kernel



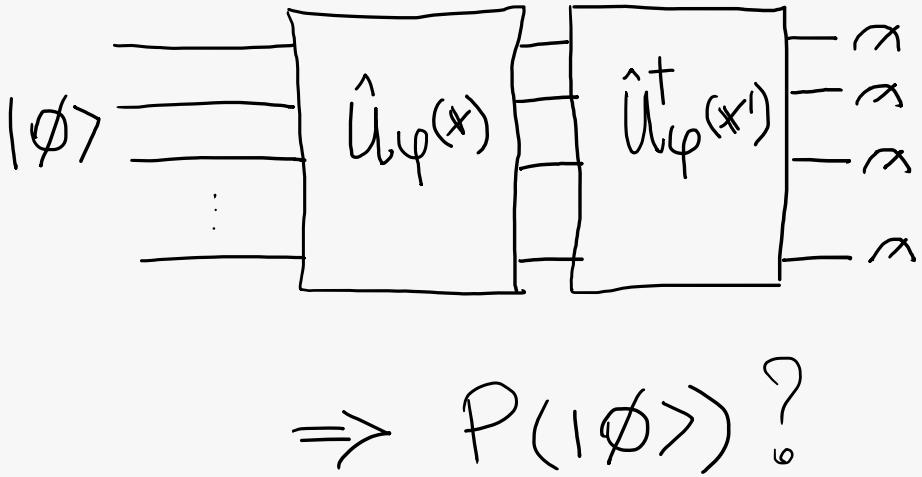

The **kernel function** quantifies the distance between sample points in the Hilbert space

**Kernel trick:** Avoiding to calculate the explicit mapping!

φ is the feature map!

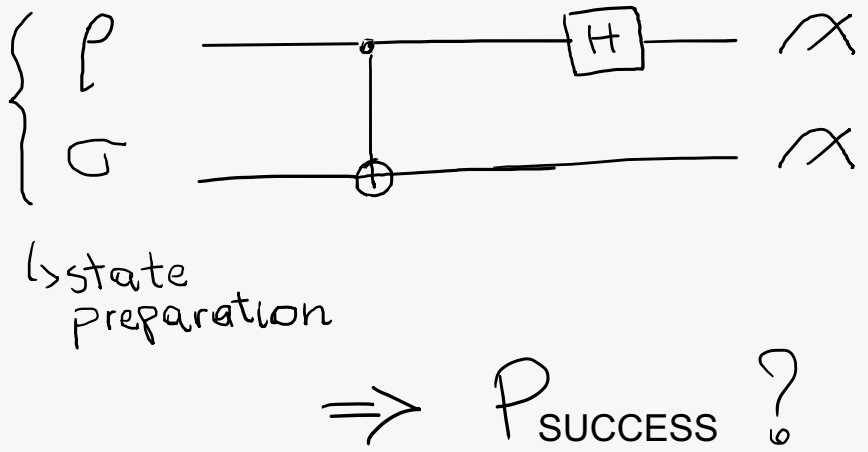
# How to calculate kernels on a QPU

## Return probabilities




It doubles the **depth** of the circuit

## Destructive SWAP




It doubles the **width** of the circuit

Repeating the measurement for **all pairs x,x'** in the classical dataset we obtain the KERNEL MATRIX

# A known example of quantum speedup

**Case study:** supervised classification with quantum kernels and support vector machines

**Result:** quantum classifier achieves high accuracy for a **classically hard** classification problem

DISCRETE LOGARITHM PROBLEM

- Consider a large prime number $p$ and a generator $g$ of $\mathbb{Z}_p^* = \{1,2,\dots,p-1\}$

"It is a widely-believed conjecture that **no classical algorithm** can compute $log_g x$ on input $x \in \mathbb{Z}_p^*$, **in time polynomial** in $n = [log_2 p]$, the number of bits needed to represent $p$"

- Define $C = \{f_s\}_{s \in \mathbb{Z}_p^*}$ over the data space $X = \mathbb{Z}_p^*$ as:

$$f_s(x) = \begin{cases} +1, & \text{if } \log_g x \in [s, s + \frac{p-3}{2}] \\ -1, & \text{else.} \end{cases}$$

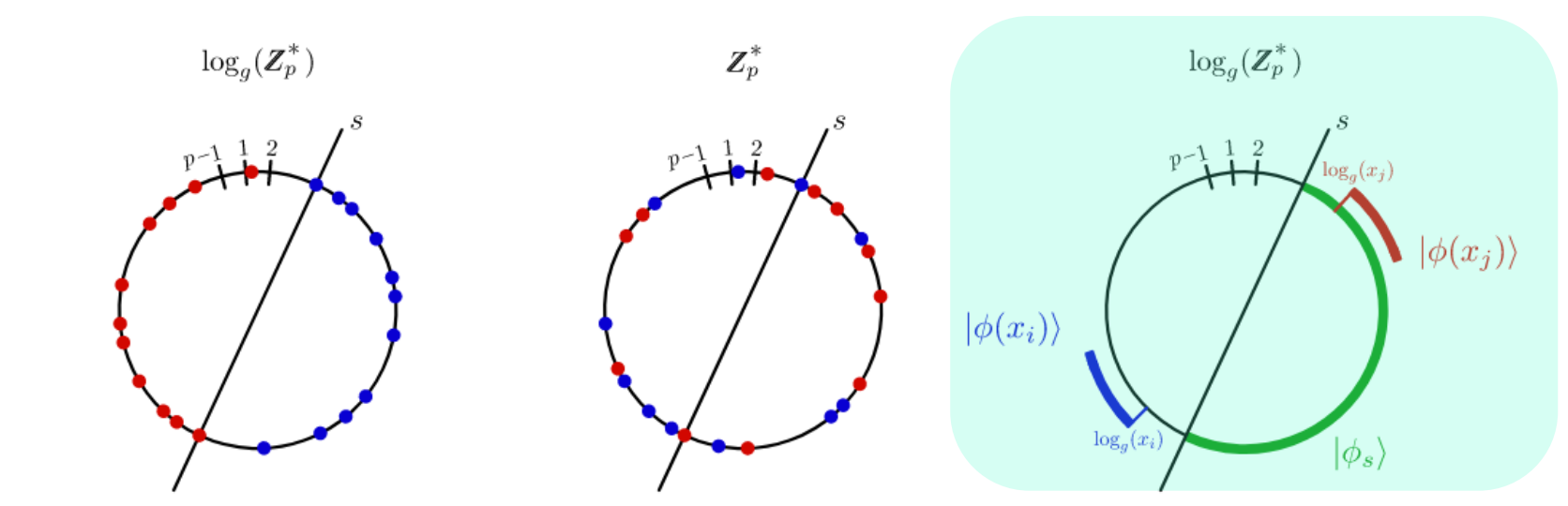
Liu, Y., Arunachalam, S. & Temme, K. **A rigorous and robust quantum speed-up in supervised machine learning**. *Nat. Phys.* **17**, 1013–1017 (2021).

# A known example of quantum speedup

**Case study:** supervised classification with quantum kernels and support vector machines

**Result:** quantum classifier achieves high accuracy for a **classically hard** classification problem

Learn C with a quantum feature map: exponential speed-up



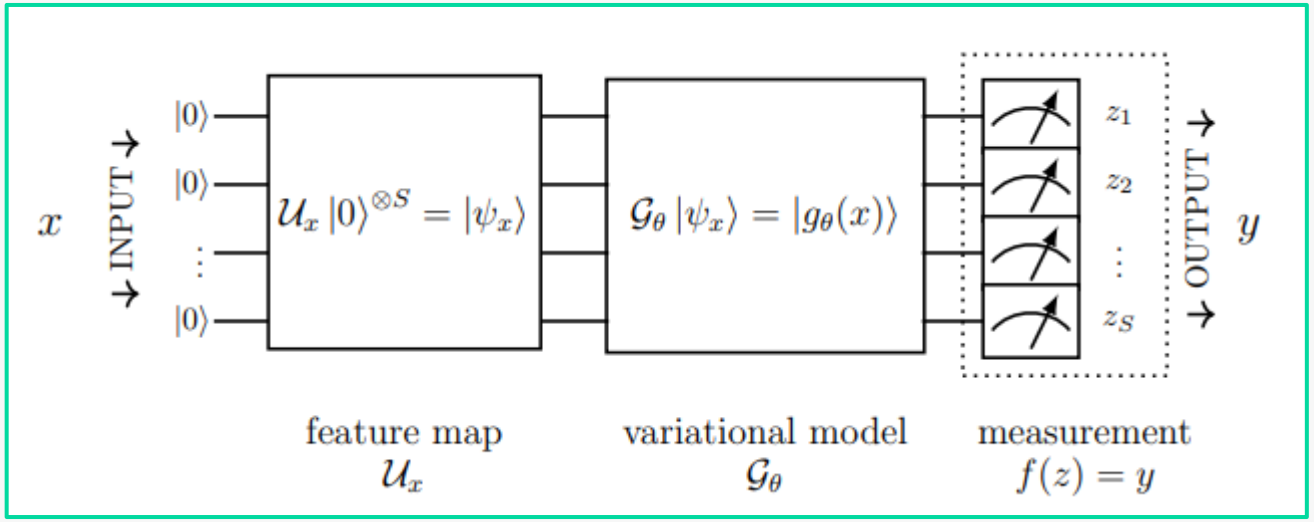

"This quantum feature map creates a **large margin**, as the **+1** labeled example (red interval) has **high overlap** with a separating hyperplane (green interval), while the **−1** labeled example (blue interval) has **zero overlap**"


Liu, Y., Arunachalam, S. & Temme, K. **A rigorous and robust quantum speed-up in supervised machine learning**. *Nat. Phys.* **17**, 1013–1017 (2021).

# Hybrid quantum-classical neural networks

# How does a QNN look like?

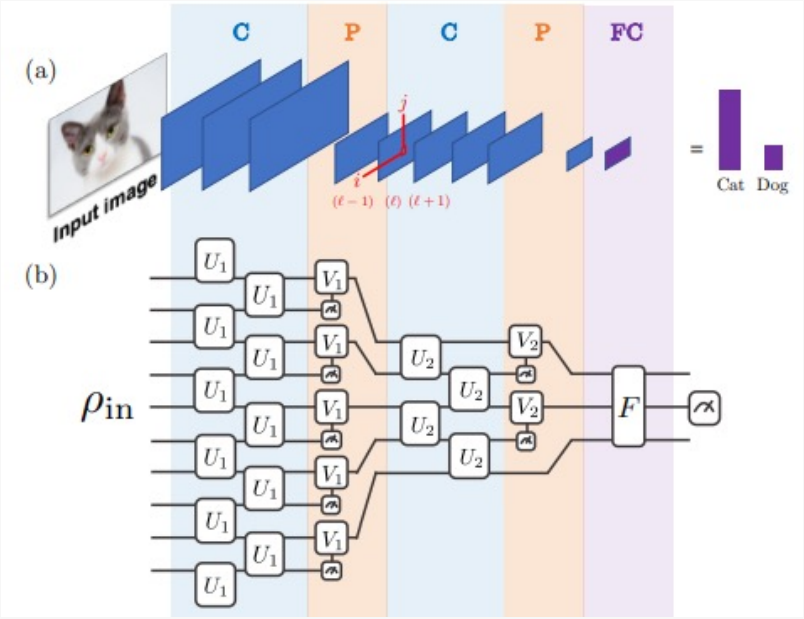
- Often what is referred to as Quantum Neural Network in QML literature is a more general **variational** model:




- A cost function is determined by the problem at hand and parameters are optimized to minimize it

TRAINING discussed in the next slide

There are models more closely recalling the structure of the classical neural network:



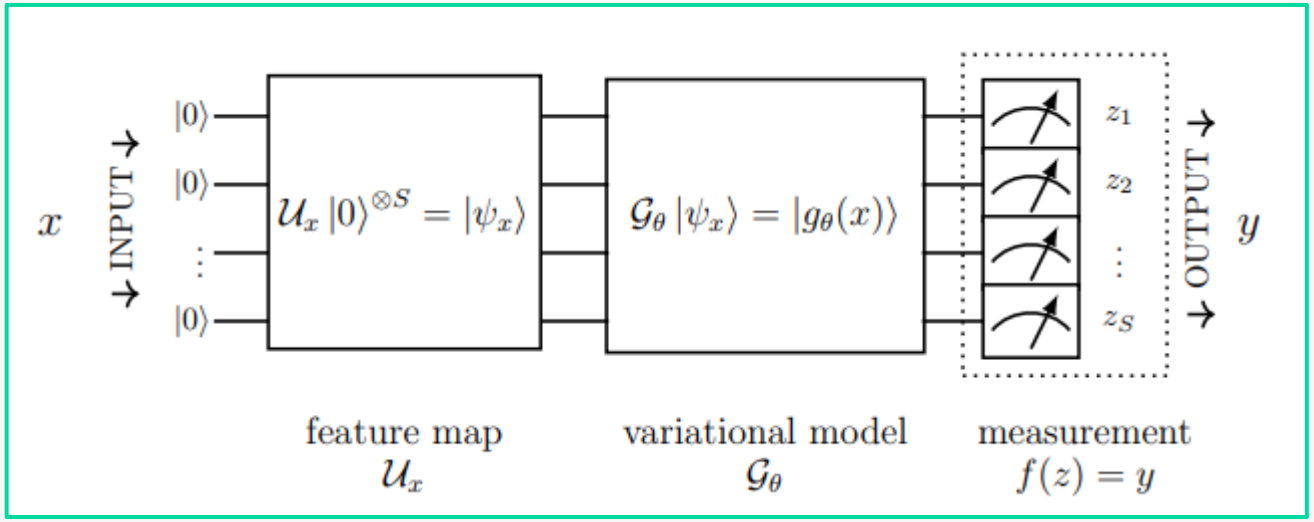

Quantum Convolutional Neural Networks



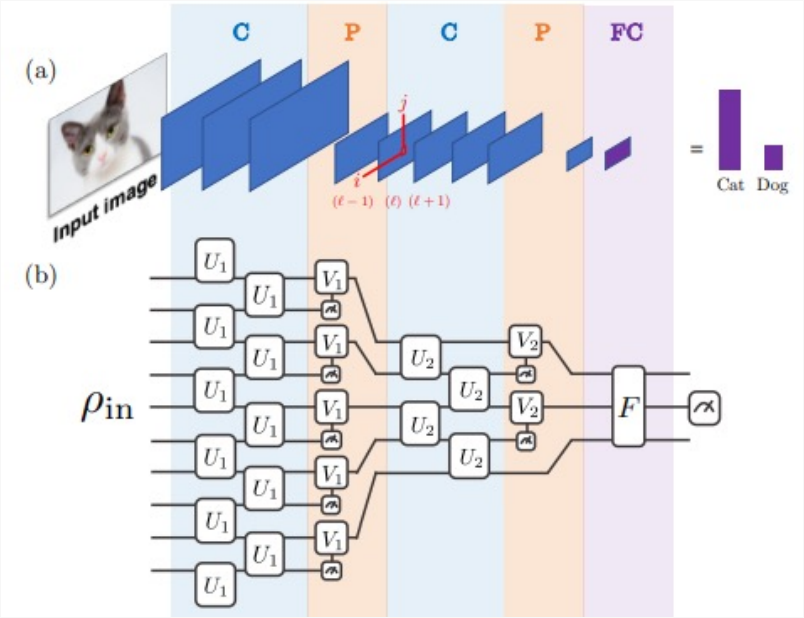
Cong, I., Choi, S. & Lukin, M.D. **Quantum convolutional neural networks**. *Nat. Phys.* **15**, 1273–1278 (2019)

# How does a QNN look like?

- Often what is referred to as Quantum Neural Network in QML literature is a more general **variational** model:




There are models more closely recalling the structure of the classical neural network:




Quantum Convolutional Neural Networks

QCNN use only $\boldsymbol{O(log(N))}$ variational parameters for input sizes of $\boldsymbol{N}$ qubits: good for implementation on near-term quantum devices



Cong, I., Choi, S. & Lukin, M.D. **Quantum convolutional neural networks**. *Nat. Phys.* **15**, 1273–1278 (2019)

# Getting better parameters through training

- Main idea in machine learning is to learn the parameters through data until we have a good fit for the given problem.

- In classical machine learning, this is typically acchieved using some gradient descent method: The parameters are gradually adjusted in a direction which reduces the loss.

- How can we compute the gradient of a quantum circuit when already simulating it is intractable with a classical computer? Backpropagation is not possible as intermediate measuring will destroy the quantum state.

# Calculating gradient using parameter shift rule

- The parameter shift rule calculates the gradient of a function by evaluating the function at two different points

- Example

$$f(\theta) = \sin(\theta)$$

$$\frac{\sin\left(\theta + \frac{\pi}{4}\right) - \sin(\theta - \frac{\pi}{4})}{\sqrt{2}} = \cos(\theta)$$

$$= \frac{1}{\sqrt{2}}\left(f\left(\theta + \frac{\pi}{4}\right) - f\left(\theta - \frac{\pi}{4}\right)\right) = \boldsymbol{\frac{df}{d\theta}}$$

- **Overhead** to calculate gradient this way is still two per parameter, expensive on current day quantum computers

**This is not finite difference!**

$$\partial_\theta f(\theta) \approx \frac{f(\theta + h) - f(\theta - h)}{2h}$$

Exact solution, not just an approximation

Points in parameter shift can be far apart, not infinitesimally close

Small differences could be more strongly influenced by noise



Maria Schuld et al. *Evaluating analytic gradients on quantum hardware*, Physical Review A 99, 032331 (2019)

# Calculating the gradient with SPSA

- Simultaneous Perturbation Stochastic Approximation (SPSA) is a noise-resistant stochastic gradient approximation, which requires only two measurements of the objective function, independent of the number of parameters $n$ (costs are constant!).

$$\nabla f = \frac{\delta f}{\delta \boldsymbol{\theta}} = \frac{f(\theta + c\Delta) - f(\theta - c\Delta)}{2c\ \Delta_i}$$

$\Delta = (\Delta_1, \ldots, \Delta_n)$ with $\Delta_i = \pm 1$ with a 50%/50% chance

$c$ is a step parameter depending on the iteration

- Good thing is: Engines do this for us (Pennylane, JAX, TensorFlow Quantum)
- (Note: Other optimizers like COBYLA exist with different (dis-)advantages (e.g. COBYLA only requires one circuit evaluation per iteration instead of two)

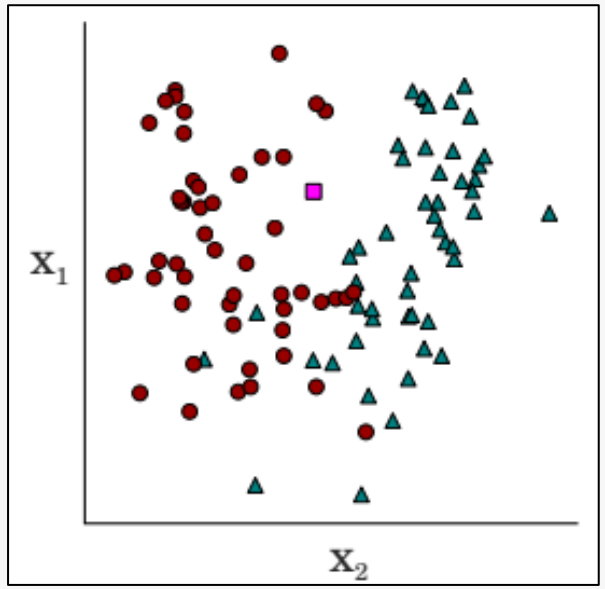
# 4. Application: a classification task

# Classification problems

- Classification tasks are one of the most common problems tackled with machine learning methods
- The approach can be supervised or unsupervised (e.g., anomaly detection), and the problem can be binary or multi-class
- **Applications explored in the literature range from high-energy physics problems, to medical ones (breast cancer) or finance related ones, like fraud detection**

Example: Supervised binary classification for two dimensional inputs. Given the red circle and blue triangle data points belonging to two different classes, guess the class of the new input (pink square).

*Schuld et al., Phys. Rev. A **101**, 032308 (2020)*




Now it's your turn!

# Hands-on exercises

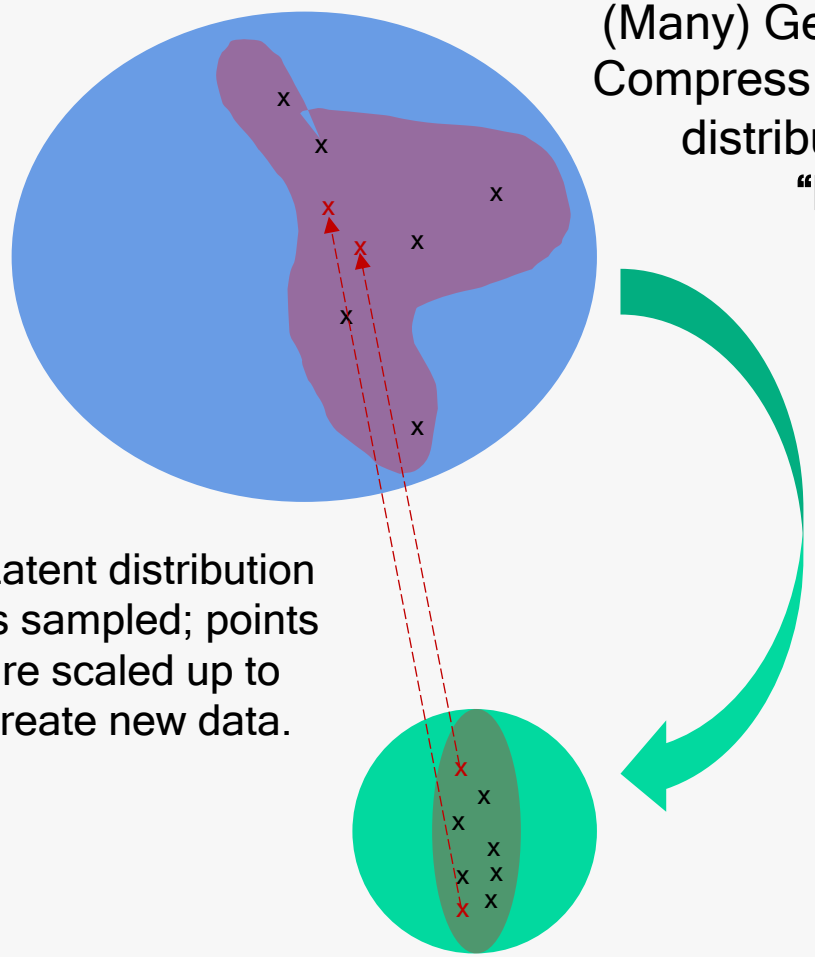
- STEP 1: Toy dataset https://bit.ly/iqm-4
  - Goal: UNDERSTAND
  - Execute the notebook
  - Interpret the results

- STEP 2: Iris dataset https://bit.ly/iqm-5
  - Goal: IMPROVE
  - Actively change the encoding, the ansatz etc.
  - Try to improve the performance of the model
  - Tip: Look at which rotations are used in the ansatz: do you recall what expressibility is?

# 5. What’s next?

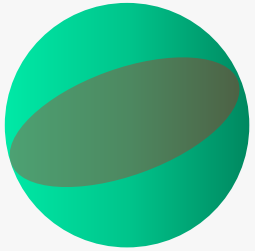
# Promising applications: generative QML

(Many) Generative Approaches:
Compress high dimensional data distribution onto smaller, **"latent space"**

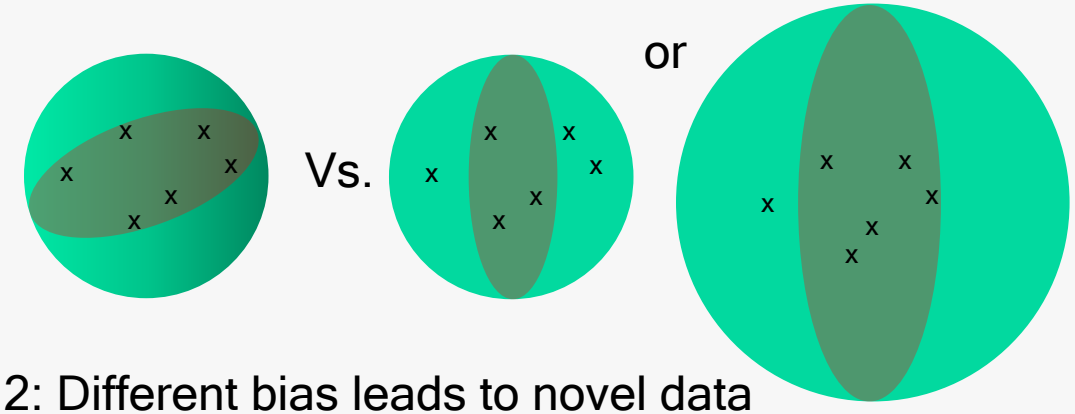


Latent distribution is sampled; points are scaled up to create new data.

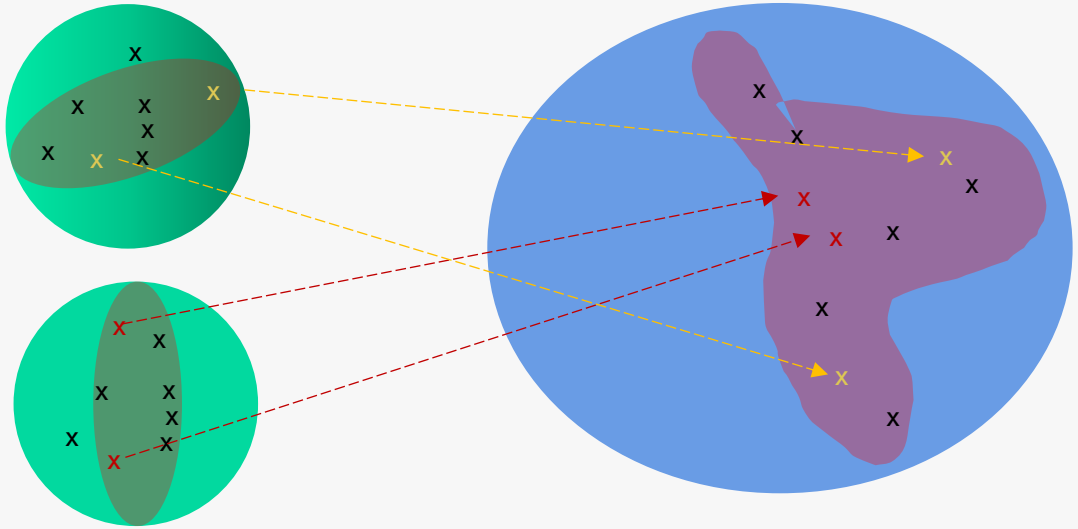
**Quantum circuits produce latent space distributions which are difficult to learn and sample classically.**



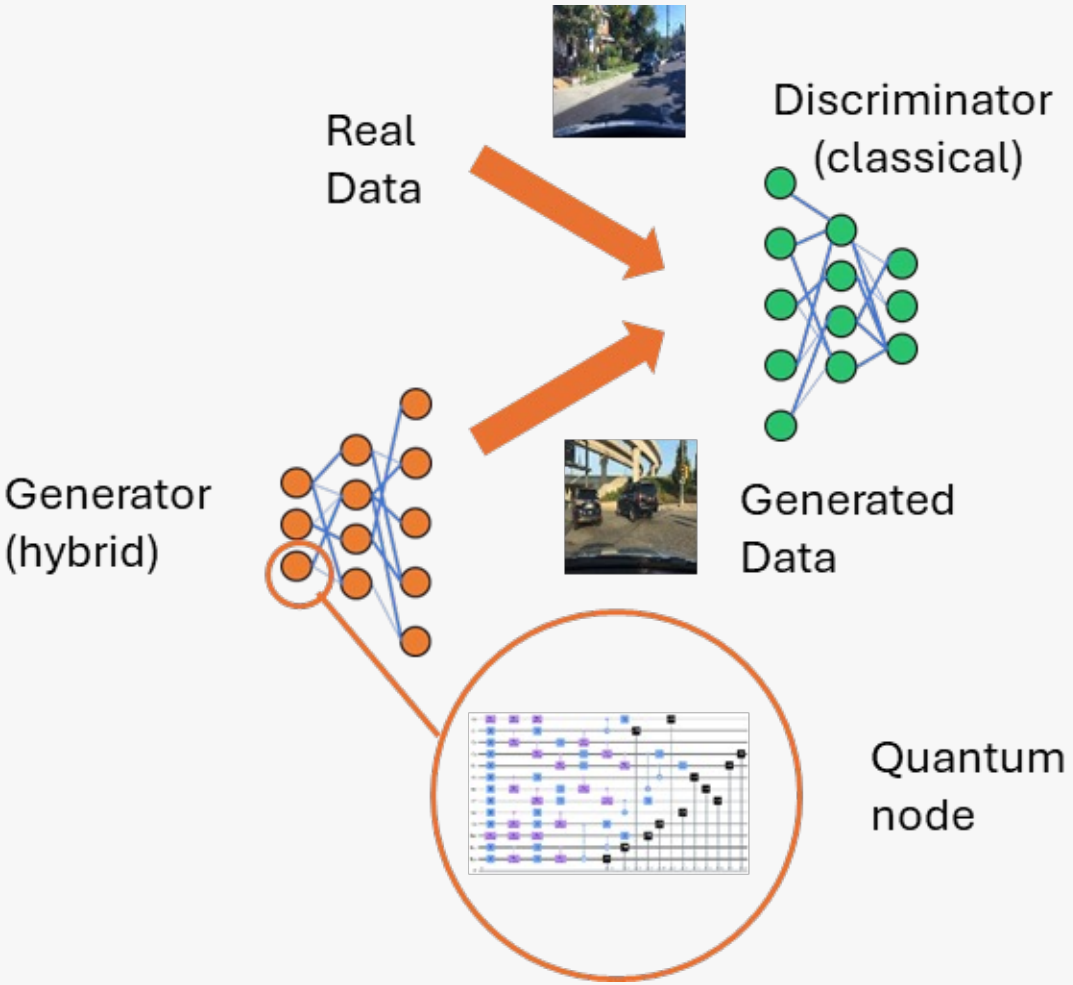
1: Better fit with less resources.




2: Different bias leads to novel data

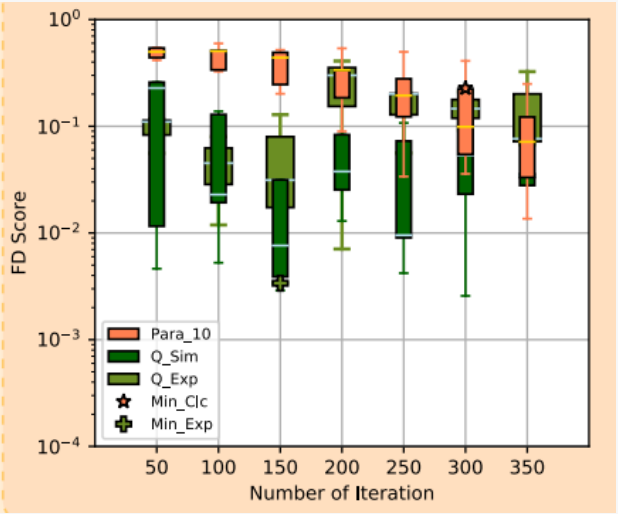


# Example: Quantum GAN










*"The results show that quantum GAN have a better performance than GAN-MLP and GAN-CNN with similar number of parameters."*

*Huang et al., Phys. Rev. Applied **16**, 024051 (2021)*

# The QQ section

**What are the possible QUANTUM DATA that can be interesting for quantum machine learning?**

## Hamiltonian

Applications:
Quantum chemistry
Quantum materials

## Quantum states

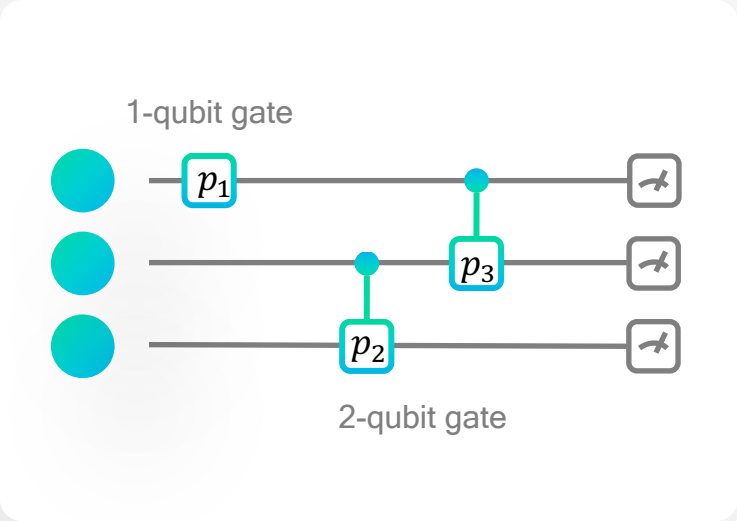
Applications:
Quantum communication networks

## Measurement operators

Applications:
Quantum sensing
Quantum imaging

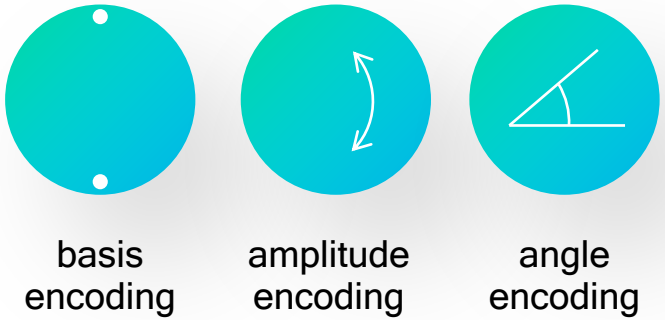
# Conclusions

- Quantum circuits can be used as a machine learning model (or part of it)
- The way we encode data matters (data sets are large, current systems are small)
- Proper feature encoding can make problems simpler in higher dimensional feature space
- Circuits can be trained similarly to neural networks, i.e. with gradient-based methods



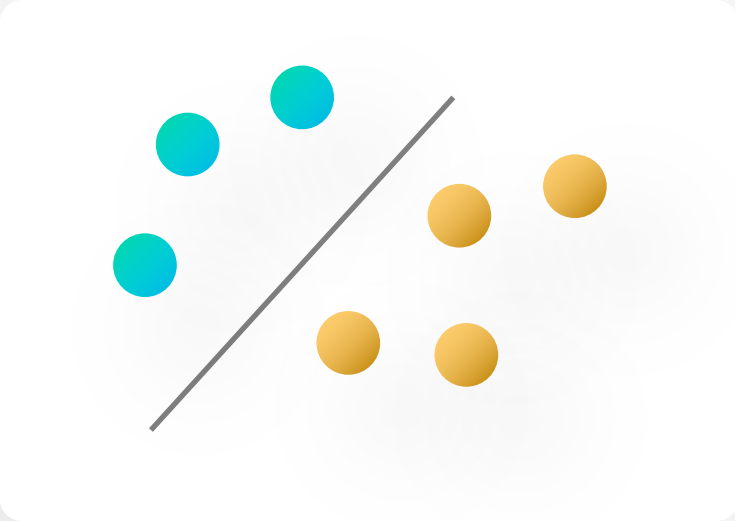
# Conclusions

- Quantum circuits can be used as a machine learning model (or part of it)
- The way we encode data matters (data sets are large, current systems are small)
- Proper feature encoding can make problems simpler in higher dimensional feature space
- Circuits can be trained similarly to neural networks, i.e. with gradient-based methods



# Conclusions

- Quantum circuits can be used as a machine learning model (or part of it)
- The way we encode data matters (data sets are large, current systems are small)
- Proper feature encoding can make problems simpler in higher dimensional feature space
- Circuits can be trained similarly to neural networks, i.e. with gradient-based methods



# Conclusions

- Quantum circuits can be used as a machine learning model (or part of it)
- The way we encode data matters (data sets are large, current systems are small)
- Proper feature encoding can make problems simpler in higher dimensional feature space
- Circuits can be trained similarly to neural networks, i.e. with gradient-based methods

