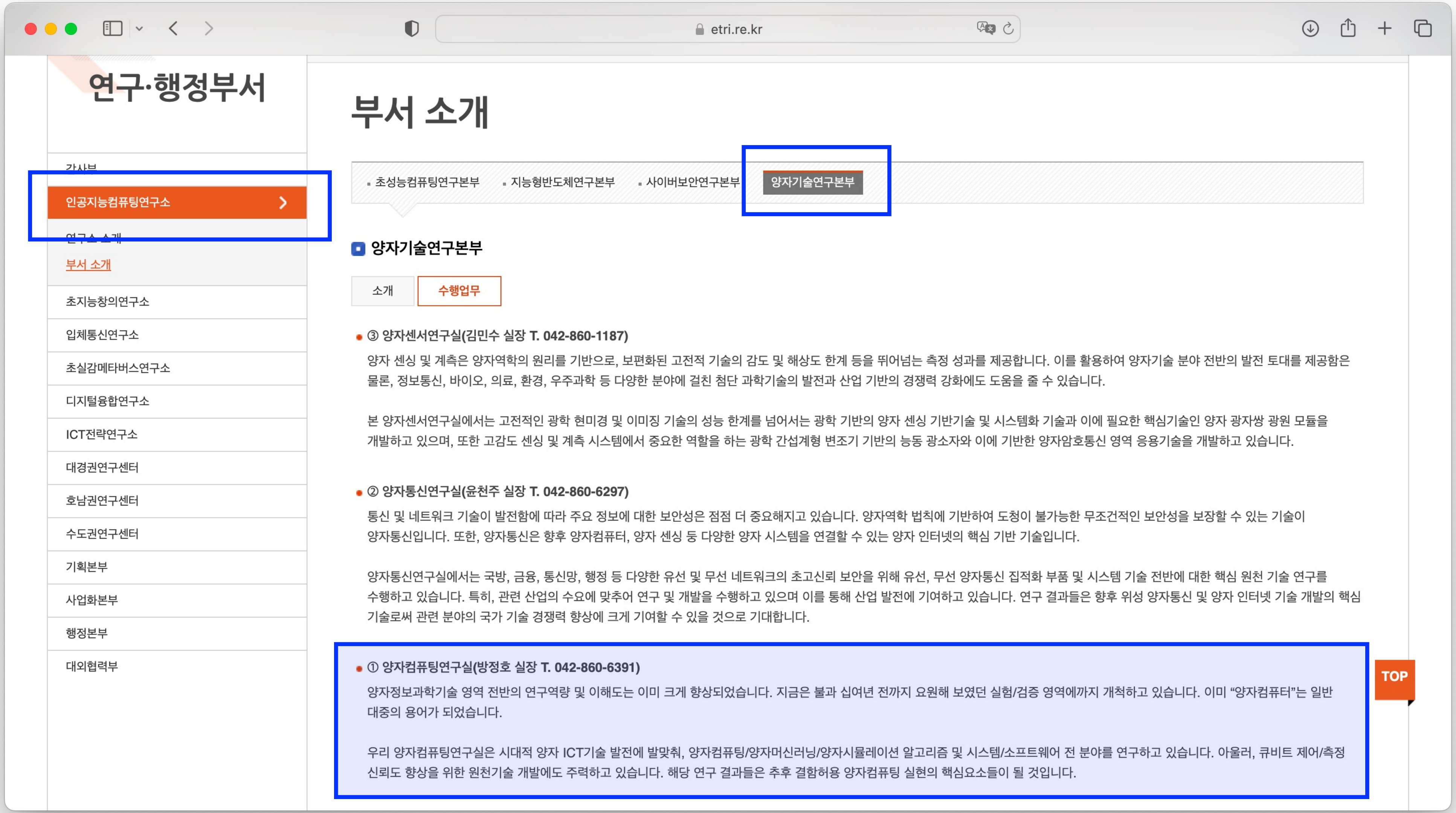
Embedding data into quantum circuits

황용수 @ ETRI

2023.07.17

Quantum Information Tech. @ ETRI

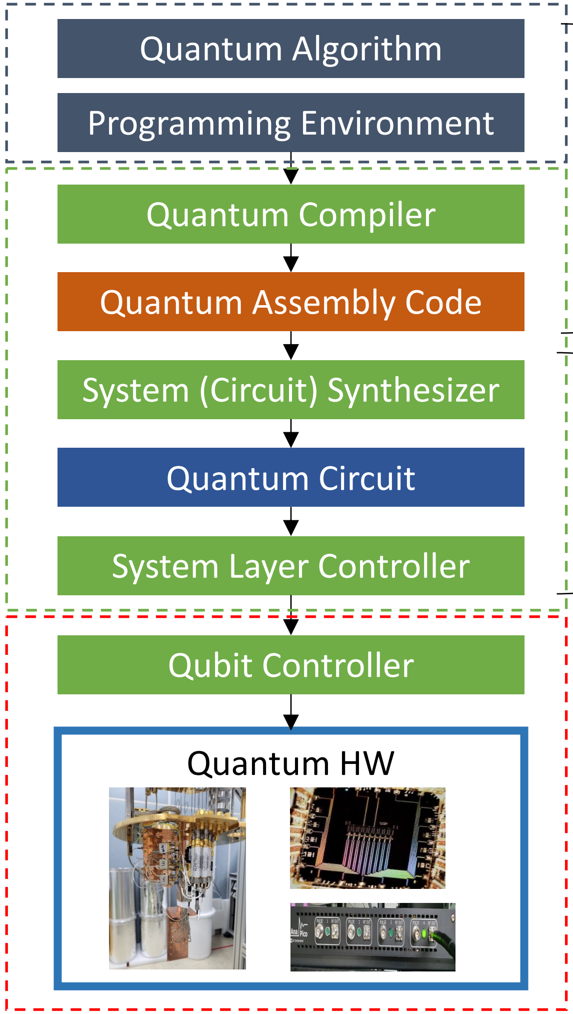


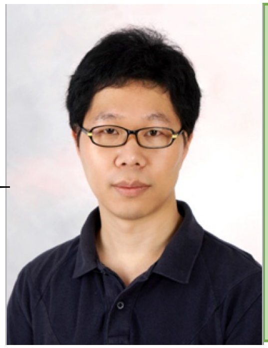
Quantum Information Tech. @ ETRI

Dr. Jeongho Bang
- Quantum information theory
- Quantum algorithm
- Quantum machine learning

Dr. Kyunghyun Baek
- Quantum information theory
- Quantum algorithm
- Quantum machine learning

Dr. Seungjin Lee
- Quantum information theory
- Quantum algorithm
- Quantum machine learning





Dr. Taewan Kim
- Quantum information theory
- Quantum compile
- Quantum circuit optimization

Dr. Yongsoo Hwang
- Quantum error correction
- Quantum system (circuit) synthesis

Dr. Sungun Cho
- Superconducting Qubit
- Cryostat
- Qubit Control

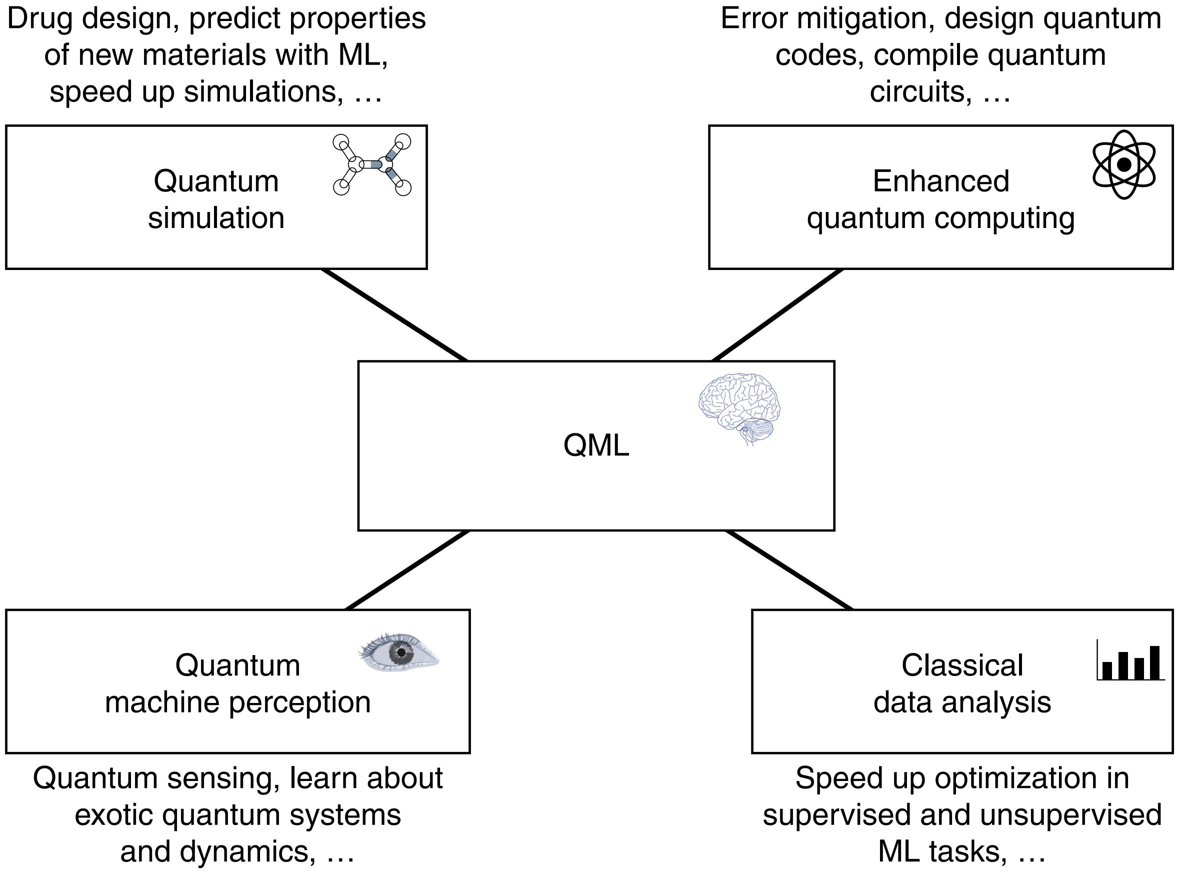
Outline

- Quantum Data Embedding
- Ansatz, depth, layers
- How to embed data into a quantum state

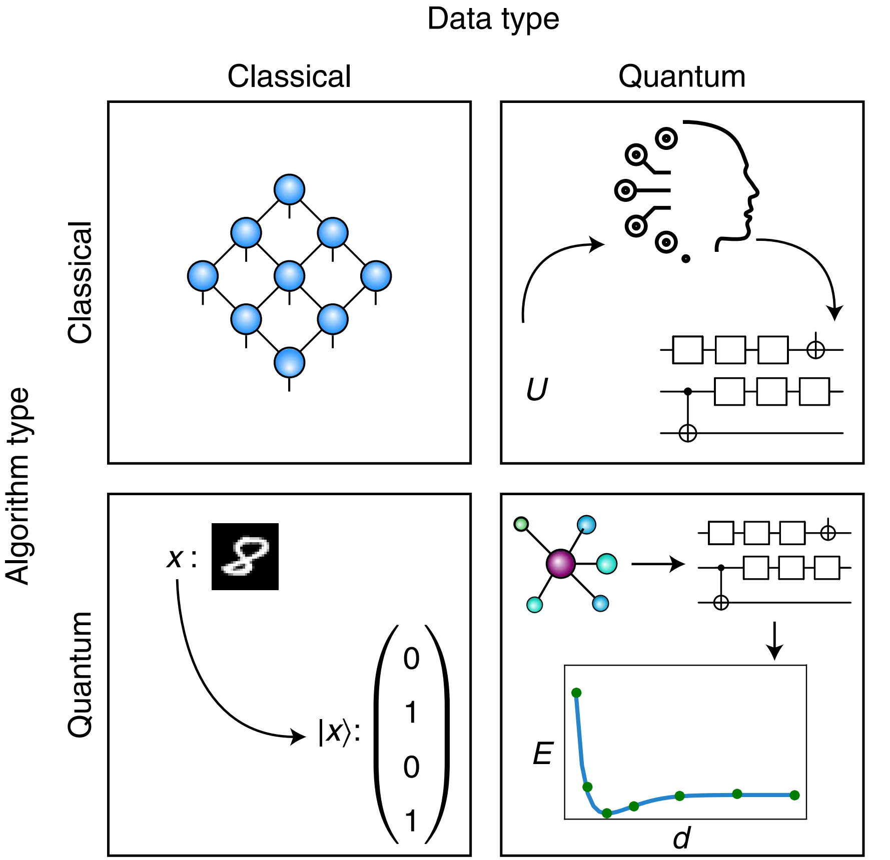
Reference

- Jacob Biamonte et al., Quantum machine learning, nature 549, 195—202 (2017)
- M. Cerezo et al., Challenges and opportunities in quantum machine learning, nature computational science 2, 567—576 (2022)
- Maria Schuld and Nathan Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. 122, 040504 (2019)
- Seth Lloyd et al., Quantum embeddings for machine learning, arXiv:2001.03622 (2020)
- Manuela Weigold et al., Encoding patterns for quantum algorithms, IET Quantum Communication, Vol. 2, Issue 4, pp.141—152 (2021)
- https://pennylane.ai

Quantum Machine Learning



<Key applications for QML>



<QML Tasks : Types of algorithm and data>

Nature Computational Science 2, 567-576

Quantum Machine Learning

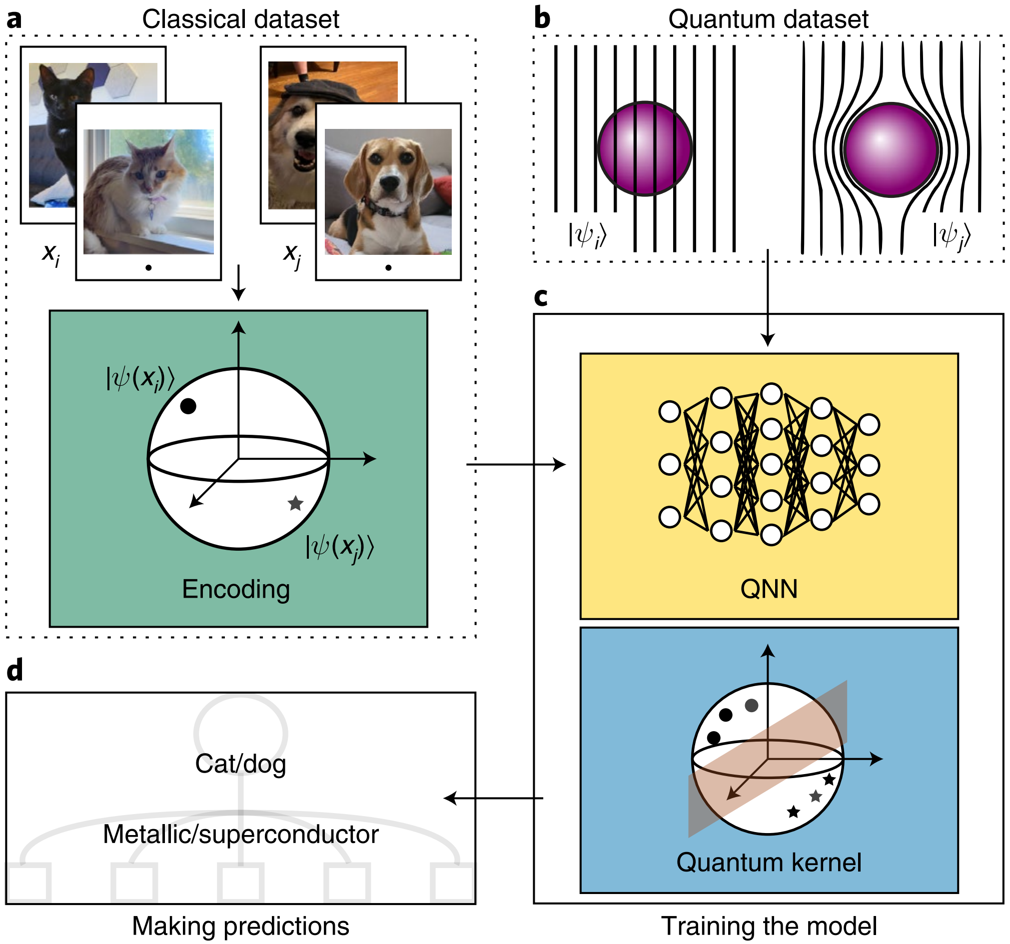


Fig. 3 | Classification with QML. a, The classical data x, that is, images of cats and images of dogs, is encoded into a Hilbert space via some map $x \rightarrow |\psi(x)\rangle$. Ideally, data from different classes (here represented by dots and stars) are mapped to different regions of the Hilbert space. **b**, Quantum data $|\psi\rangle$ can be directly analyzed on a quantum device. Here the dataset is composed of states representing metallic or superconducting systems. **c**, The dataset is used to train a QML model. Two common paradigms in QML are QNNs and quantum kernels, both of which allow for classification of either classical or quantum data. In kernel methods we fit a decision hyperplane that separates the classes. **d**, Once the model is trained, it can be used to make predictions.

Quantum Machine Learning

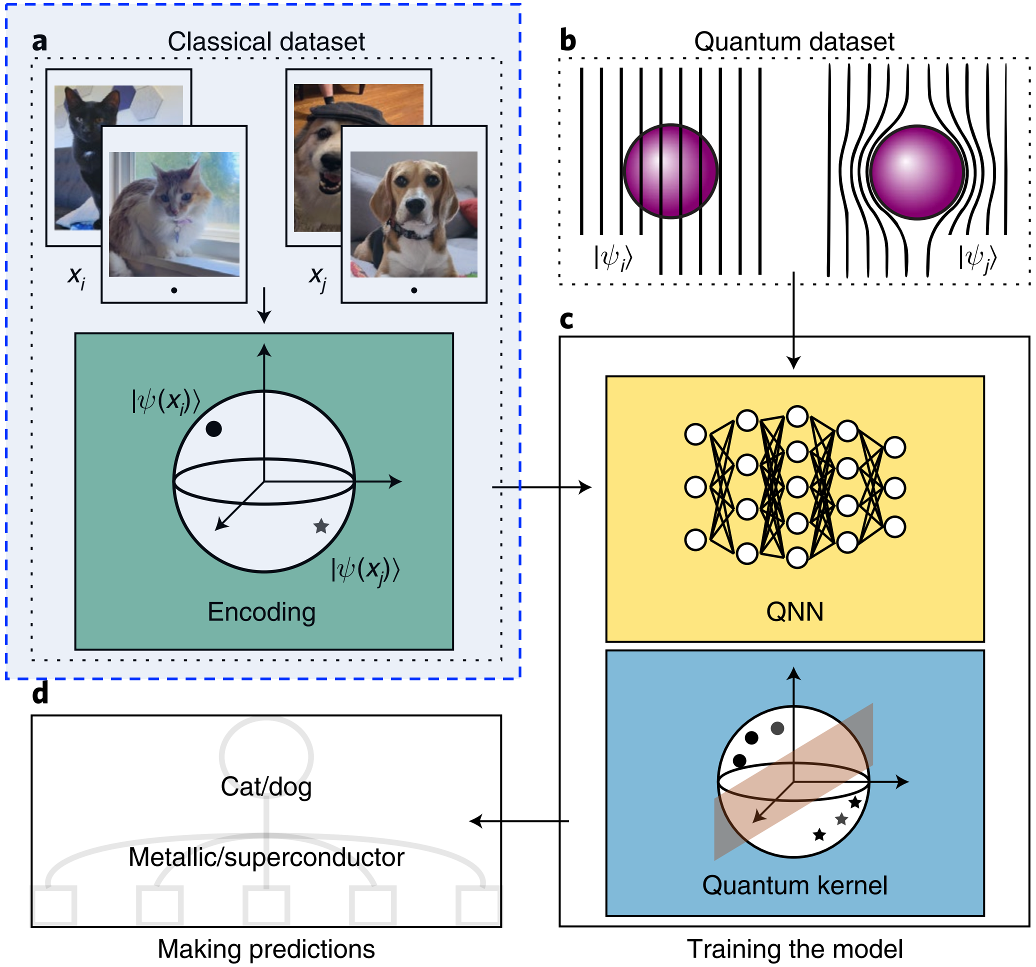
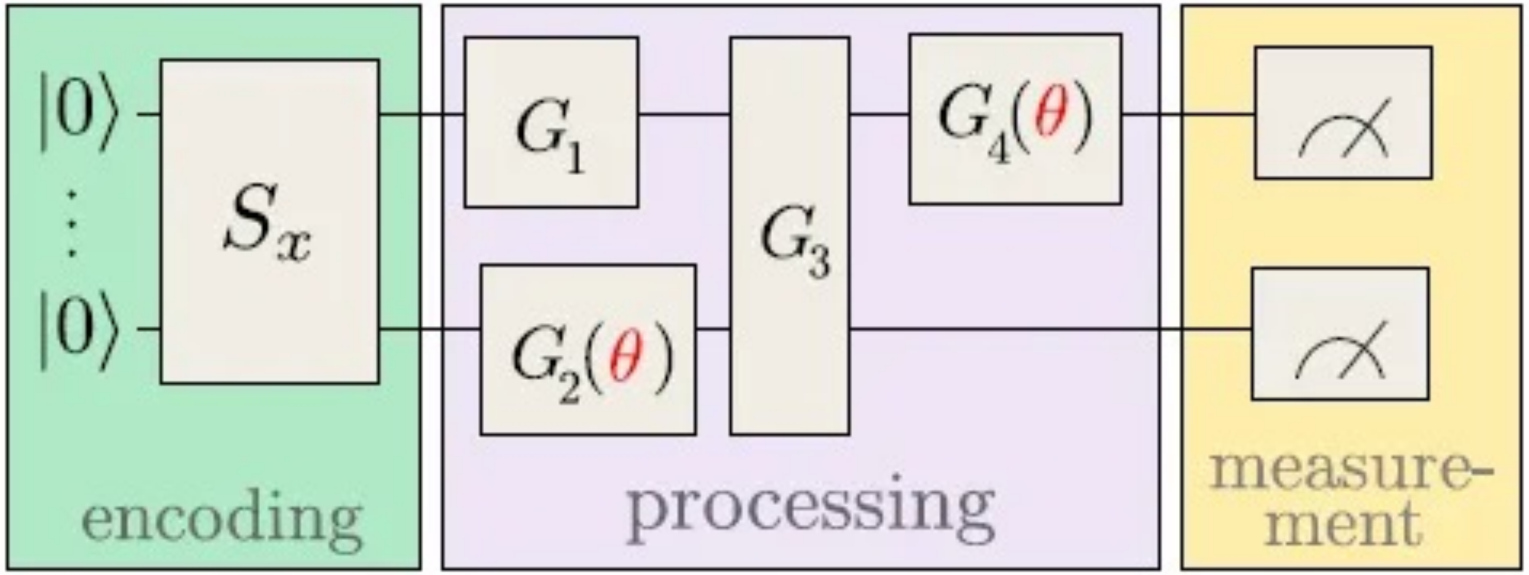


Fig. 3 | Classification with QML. a, The classical data x, that is, images of cats and images of dogs, is encoded into a Hilbert space via some map $x \to |\psi(x)\rangle$. Ideally, data from different classes (here represented by dots and stars) are mapped to different regions of the Hilbert space. **b**, Quantum data $|\psi\rangle$ can be directly analyzed on a quantum device. Here the dataset is composed of states representing metallic or superconducting systems. **c**, The dataset is used to train a QML model. Two common paradigms in QML are QNNs and quantum kernels, both of which allow for classification of either classical or quantum data. In kernel methods we fit a decision hyperplane that separates the classes. **d**, Once the model is trained, it can be used to make predictions.

3 Steps of QML



Quantum Data Encoding (or Embedding)

- Quantum Data Encoding (Embedding)
  - Process to load classical data onto a quantum system

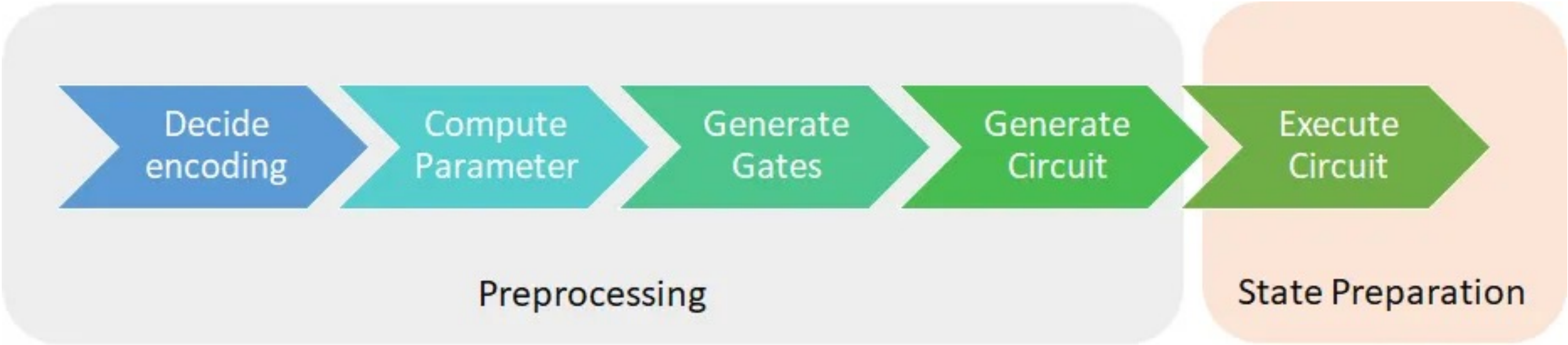
$$x \mapsto |\psi(x)\rangle$$

  - Two categories:
    - ***Digital*** : data → qubit strings, $\frac{1}{\sqrt{M}}\sum_{m=1}^{M}|x(m)\rangle$
    - ***Analogue*** : data → amplitude of a quantum state, $\sum_{i=1}^{N} x_i|i\rangle$ with $\sum|x_i|^2 = 1$

Quantum Data Encoding (or Embedding)

- Preference
  - #Qubits should be minimal
  - #Parallel Op. should be minimal to minimize the width of the circuit
  - Data must be represented appropriately for further calculations
    - Digital for arithmetic computation
    - Analogue as mapping data into large Hilbert space for QML

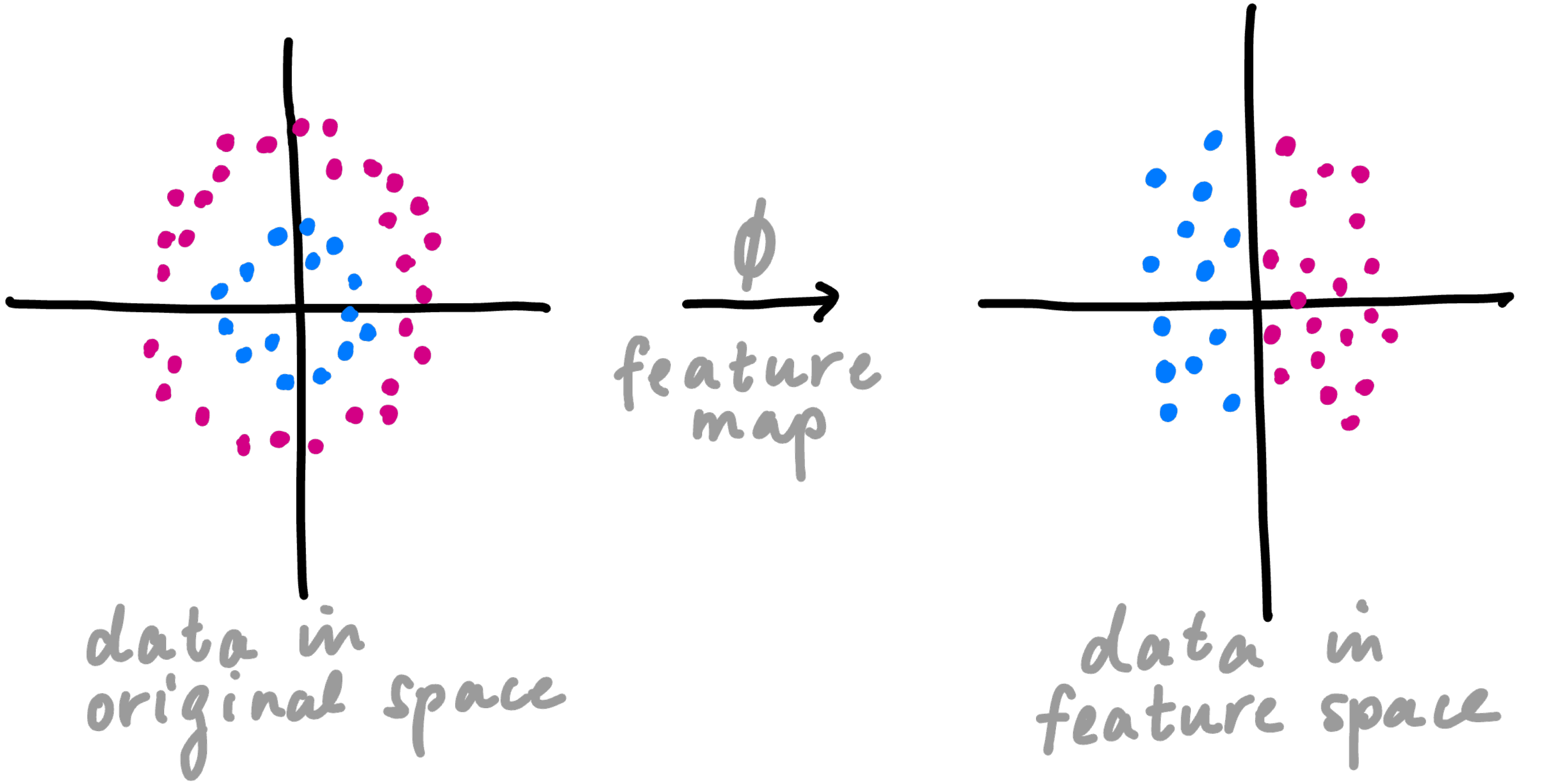
Quantum Data Encoding (or Embedding)



Feature Map

- Feature Mapping
  - Technique used in data analysis and machine learning to transform input data *from a lower-dimensional space* to *a higher-dimensional space*, where it can be more easily analyzed or classified (https://www.geeksforgeeks.org/feature-mapping/)

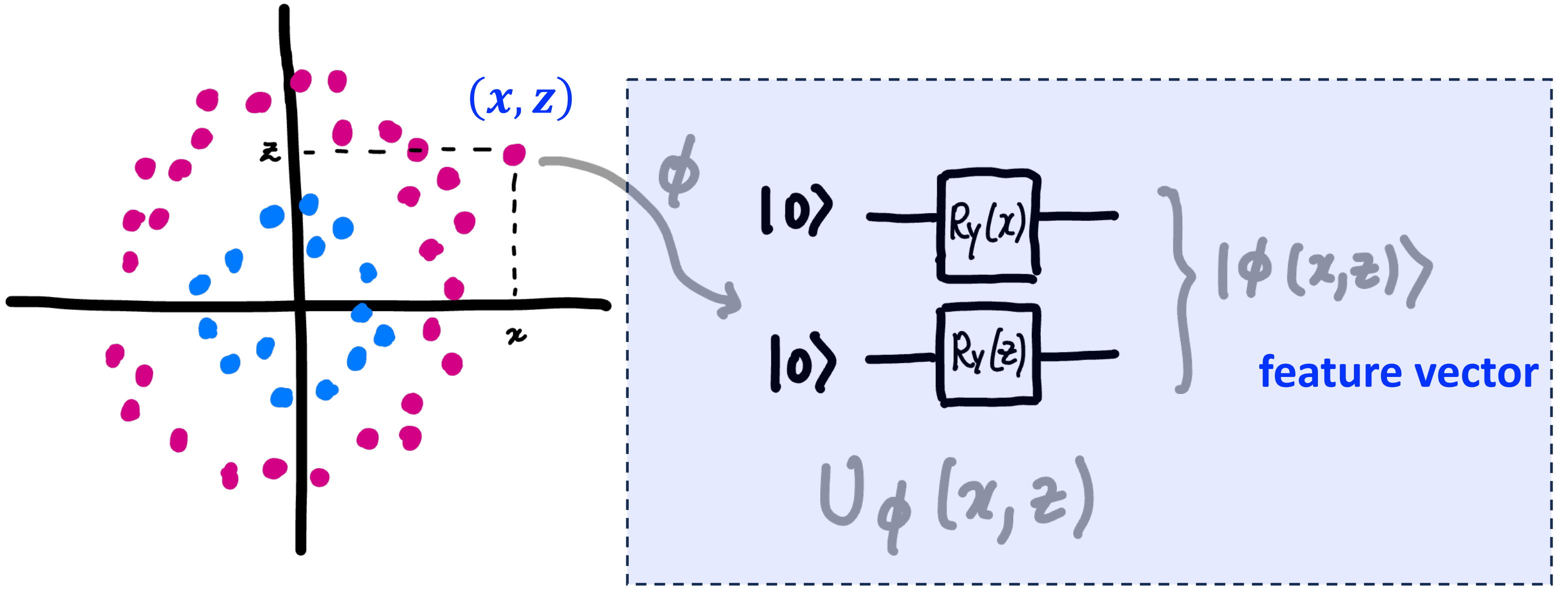
Feature Map



$\mathcal{X}$ be a set of input data. A **feature map** $\phi: \mathcal{X} \mapsto \mathcal{F}$ where $\mathcal{F}$ is the **feature space**. The output of the map $\phi(x)$ for all $x \in \mathcal{X}$ are called **feature vectors**.

Quantum Feature Map

A **quantum feature map** $\phi: \mathcal{X} \mapsto \mathcal{F}$ is a feature map where the vector space $\mathcal{F}$ is a Hilbert space and the feature vectors are quantum states. The map transforms $x \mapsto |\phi(x)\rangle$ by way of a unitary transformation $U_\phi(x)$, which is typically a **variational circuit** whose parameters depend on the input data.

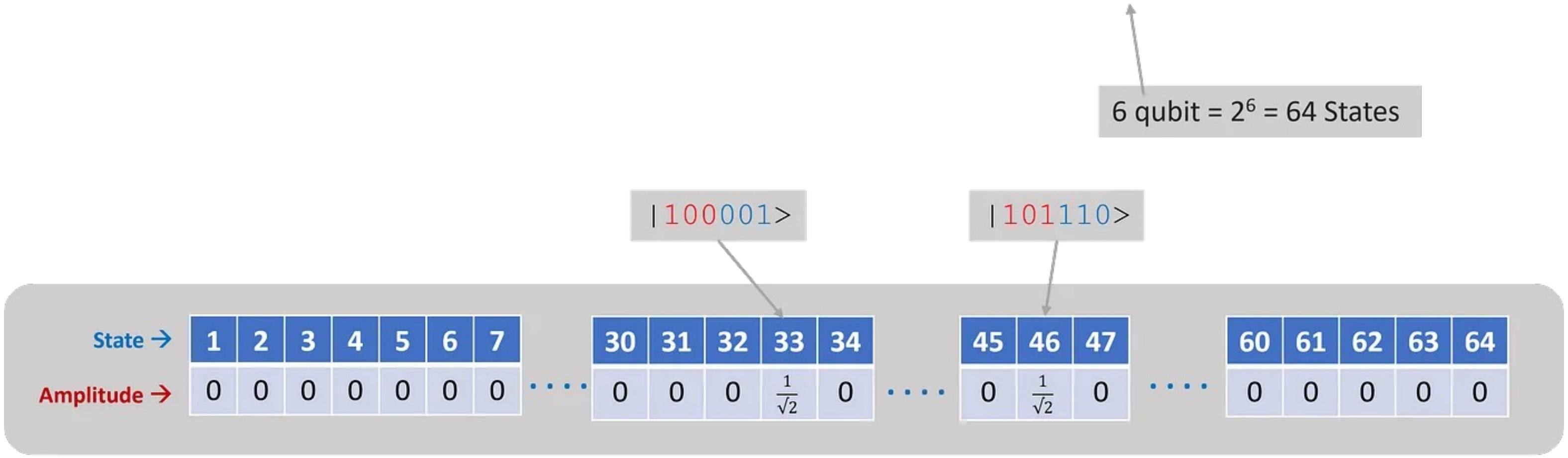


Variational circuit depends on *parameters* and *encoding*

Basis Encoding

- Basis encoding is primarily used when real numbers have to be arithmetically manipulated in a quantum algorithm. Such an encoding represents **real numbers** as **binary numbers** and then transforms them into a **quantum state** on a computational basis.

Input Data			**Pre-processing to convert to Binary form**		**Basis encoded state**	
Input Sample	Feature x1	Feature x2	Binary(x1)	Binary(x2)	Basis encoded Quantum state	Amplitude vector
X^1	5	6	101	110	|101110>	$\frac{1}{\sqrt{2}}$|101110>
X^2	4	1	100	001	|100001>	$\frac{1}{\sqrt{2}}$|100001>



Amplitude vector is very sparse , only 2 state has non-zero amplitude out of 64 states

Real number as bit string (Binary fraction representation)

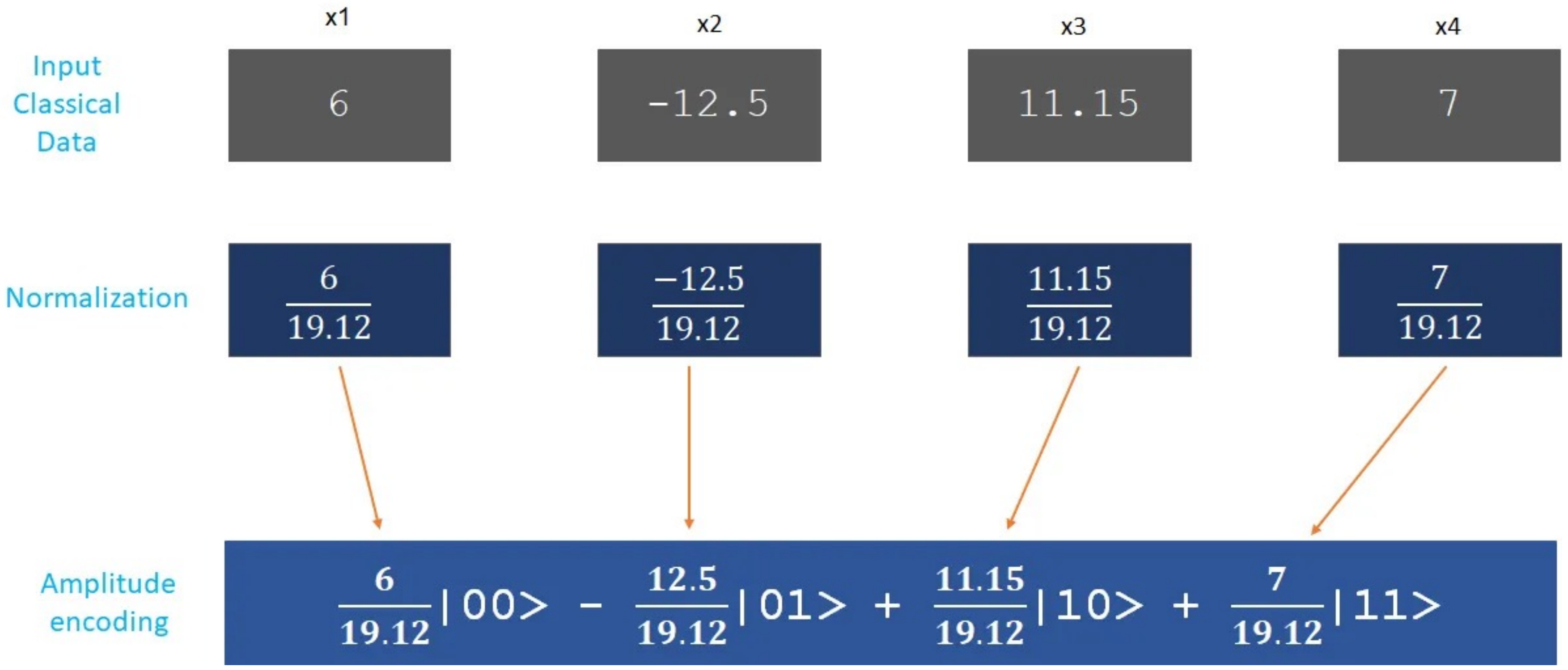
Definition 3.1 (Fixed-point encoding of real numbers (Rebentrost et al., 2021))**.** Let c_1, c_2 be positive integers, and $a \in \{0,1\}^{c_1}$, $b \in \{0,1\}^{c_2}$, and $s \in \{0,1\}$ be bit strings. Define the rational number as:

$$\mathcal{Q}(a,b,s) := (-1)^s \left(2^{c_1-1}a_{c_1} + \cdots + 2a_2 + a_1 + \frac{1}{2}b_1 + \cdots + \frac{1}{2^{c_2}}b_{c_2}\right) \in [-R, R], \tag{3.2}$$

where $R := 2^{c_1} - 2^{-c_2}$. If c_1, c_2 are clear from the context, we can use the shorthand notation for a number $z := (a,b,s)$ and write $\mathcal{Q}(z)$ instead of $\mathcal{Q}(a,b,s)$. Given an n-dimensional vector $v \in (\{0,1\}^{c_1} \times \{0,1\}^{c_2} \times \{0,1\})^n$ the notation $\mathcal{Q}(v)$ means an n-dimensional vector whose j-th component is $\mathcal{Q}(v_j)$, for $j \in [n]$.

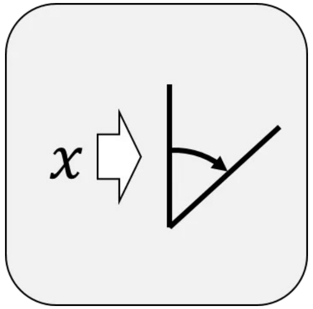
Amplitude Encoding

- Data is encoded into the ***amplitudes*** of a quantum state. This encoding requires **$\log_2$(n) qubits** to represent an ***n*-dimensional data points**.

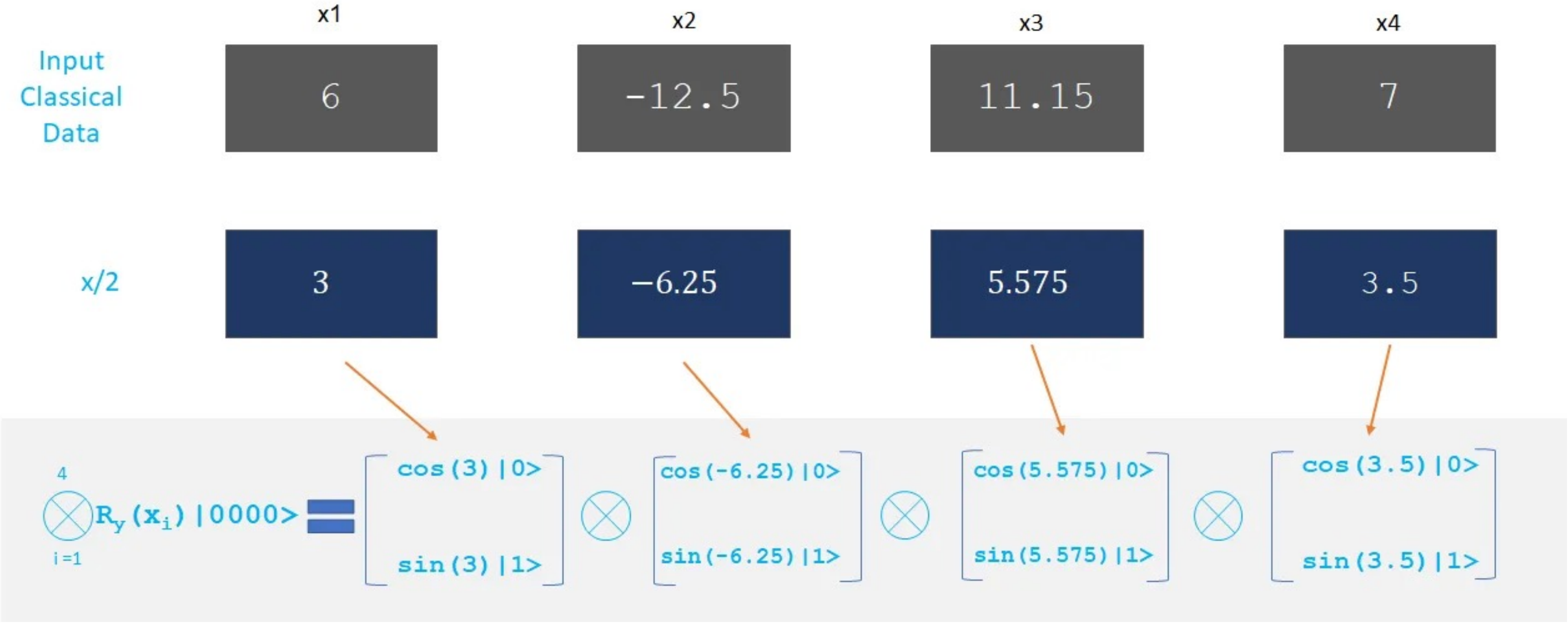


norm factor: $\sqrt{6^2 + (-12.5)^2 + 11.15^2 + 7^2}$

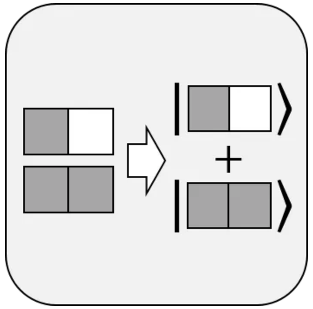
Angle Encoding (a.k.a tensor product encoding)



- Angle encoding is essentially the most basic form of encoding classical data into a quantum state. The *n* classical features are encoded into the rotation angle of the *n* qubits. This encoding requires *n* qubits to represent *n*-dimensional data but is chaper to prepare in complexity (***constant*** circuit depth): it requires one rotation on each qubit, $R_x(v)$ or $R_y(v)$ for the value *v* to encode.



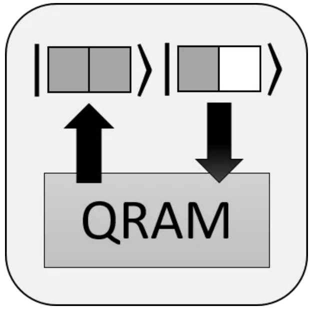
QuAM (Q- Associative Memory)



- This encoding is based on superposition to encode a set of data points in a qubit register of the same length. This requires a binary representation of all equally long values, or we need to pad with zeros. We need to use a quantum associative memory (QuAM) to prepare a superposition of basis encoded values in the same qubit register format. Note that the quantum register is an equally weighted superposition of the basis encoded values.

Input variable	Input Classical Data	Binary Number	Basis encoded Quantum Data	QUAM encoded value
X1	10	1010	|1010>	$\frac{1}{\sqrt{3}}\|1010> + \frac{1}{\sqrt{3}}\|1111> + \frac{1}{\sqrt{3}}\|1000>$
X2	15	1111	|1111>	
X3	8	1000	|1000>	

QRAM (Q-Random Access Memory)



- QRAM is used to access a superposition of data values at once. A **classical RAM** that receives an address with a memory index loads the data stored at the address into an output register. **QRAM** provides the same funcitonality, but the address and the output register are quantum register. Both the address and the output register can be the superposition of multiple values. For this encoding, l qubits are needed to encode the data values using Basis encoding. The address register requires **log(*n*)** additional qubits for a maximum of *n* addresses.

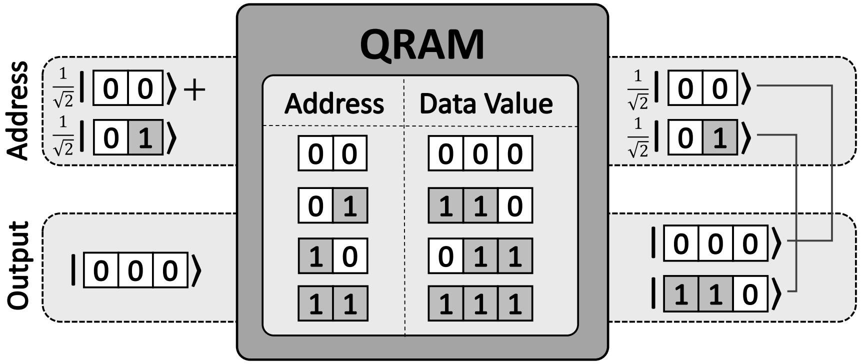


FIGURE 6 Basic functionality of a quantum random access memory (QRAM) is based on [8]. Given an address register that is in a superposition of addresses ($|00\rangle$ and $|01\rangle$), QRAM creates a superposition of addresses and their data values: $\frac{1}{\sqrt{2}}|00\rangle|000\rangle + \frac{1}{\sqrt{2}}|01\rangle|110\rangle$

address register

$$\frac{1}{\sqrt{m}}\sum_{i=0}^{m-1}|a_i\rangle|0\rangle \mapsto \frac{1}{\sqrt{m}}\sum_{i=0}^{m-1}|a_i\rangle|x_{a_i}\rangle$$

data register

Qsample encoding

- Qsample encoding is a hybrid case of basis and amplitude encoding. Qsample associates a real amplitude vector with classical discrete probability distributions. We use amplitude, but at the same time, all features are encoded in the qubit.
- Assume there is a pmf random variable X as Pr(X=i). Any discrete random variable could be represented like it just by indexing the events. For them, we can define qubit state $|\psi\rangle = \sum_{i=1}^{N} p_i |i\rangle$.
- For states of two random variables $|x\rangle = \sum_{i=1}^{2^n} \sqrt{Pr(x=i)}|i\rangle$ and $|y\rangle = \sum_{j=1}^{2^n} \sqrt{Pr(y=j)}|j\rangle$. Then, the joint state of both is $|x, y\rangle = \sum_{i,j=1}^{2^n} \sqrt{Pr(x=i)Pr(y=j)}|i\rangle|j\rangle$

Hamiltonian encoding

- Hamiltonian encoding method encodes the data into the operator. To make some matrix A to Hamiltonian, we have to make it Hermitian first. Because the definition of Hermitian is $H = H^*$, we can make it by $H = A^*A$. Then to make it as a unitary matrix, we can use a matrix exponential as e^{-iAt}. Then, we can develop a unitary evolution as follows,

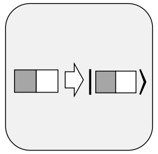
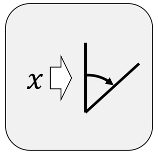
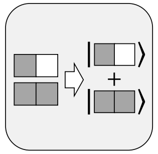
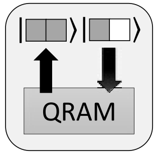
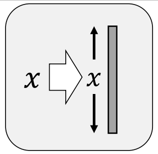
$$e^{iAt} = e^{i(\alpha+\beta X+\delta Y+\gamma Z)t} = e^{i\alpha t}e^{i\beta tX}e^{i\delta tY}e^{i\gamma tZ} = R_x(\beta t)R_y(\delta t)R_z(\gamma t).$$

- Note that for any unitary matrix A, there is a real vector $(\alpha, \beta, \delta, \gamma)$ such that $A = (\alpha + \beta X + \delta Y + \gamma Z)$.
- However, in reality, the above relation is not estabilished because the commutative condition for matrices does not hold $AB \neq BA$ $(e^{A+B} \neq e^A e^B)$.
- We can overcome this by using the **Trotter Suzuki formula**

  For large enough r, the following equation holds $e^{-i(H_1+H_2)} \approx \left(e^{-iH_A t/r} e^{-iH_B t/r}\right)^r$
- That is, we can implement some hamiltonian unitary matrices by finding the rotating magnitudes and rotation the state with them gradually.

Summary

TABLE 1 Comparison of data encoding patterns. For the QUANTUM RANDOM ACCESS MEMORY (QRAM) ENCODING, we assume that all n data points are loaded

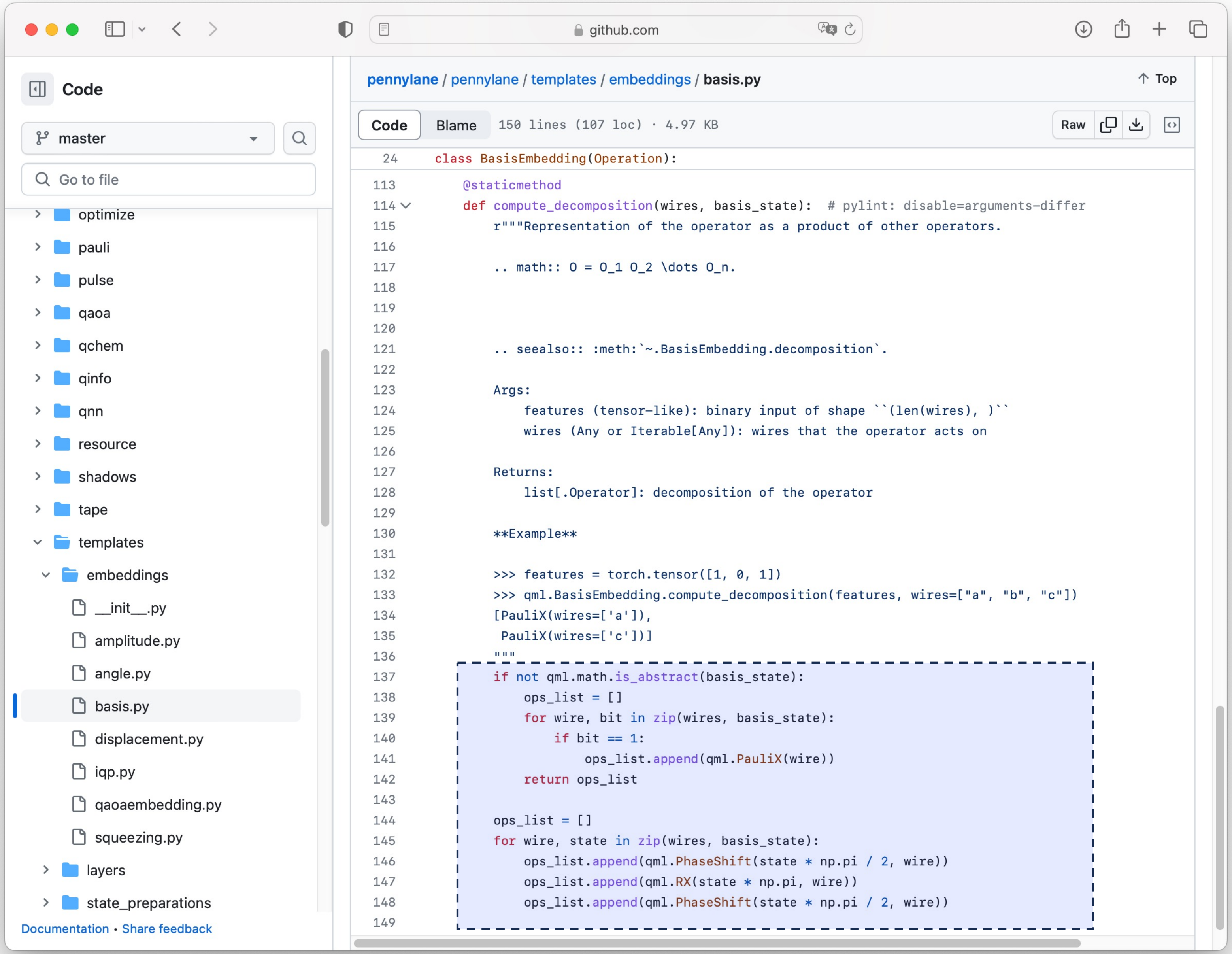
	Encoding pattern	Encoding	Req. qubits
	BASIS ENCODING [13]	$x_i \approx \sum_{i=-k}^{m} b_i 2^i \mapsto \lvert b_m \ldots b_{-k}\rangle$	$l = k+m$ per data-point
	**ANGLE ENCODING**	$x_i \mapsto cos(x_i)\lvert 0\rangle + sin(x_i)\lvert 1\rangle$	1 per data-point
	QUAM ENCODING [13]	$X \mapsto \sum_{i=0}^{n-1} \frac{1}{\sqrt{n}} \lvert x_i\rangle$	l
	**QRAM ENCODING**	$X \mapsto \sum_{n=0}^{n-1} \frac{1}{\sqrt{n}} \lvert i\rangle \lvert x_i\rangle$	$\lceil \log n \rceil + l$
	AMPLITUDE ENCODING [13]	$X \mapsto \sum_{i=0}^{n-1} x_i \lvert i\rangle$	$\lceil \log n \rceil$

Manuela Weigold et al., Encoding patterns for quantum algorithms, IET Quantum communication. Vol. 2, Issue 4 (2021)

Appendix : Circuit for encoding (Pennylane API)

Basis encoding

Apply **pauli-X** for “1”



Basis encoding

https://medium.datadriveninvestor.com/all-about-da

Pennylane sample code as below:

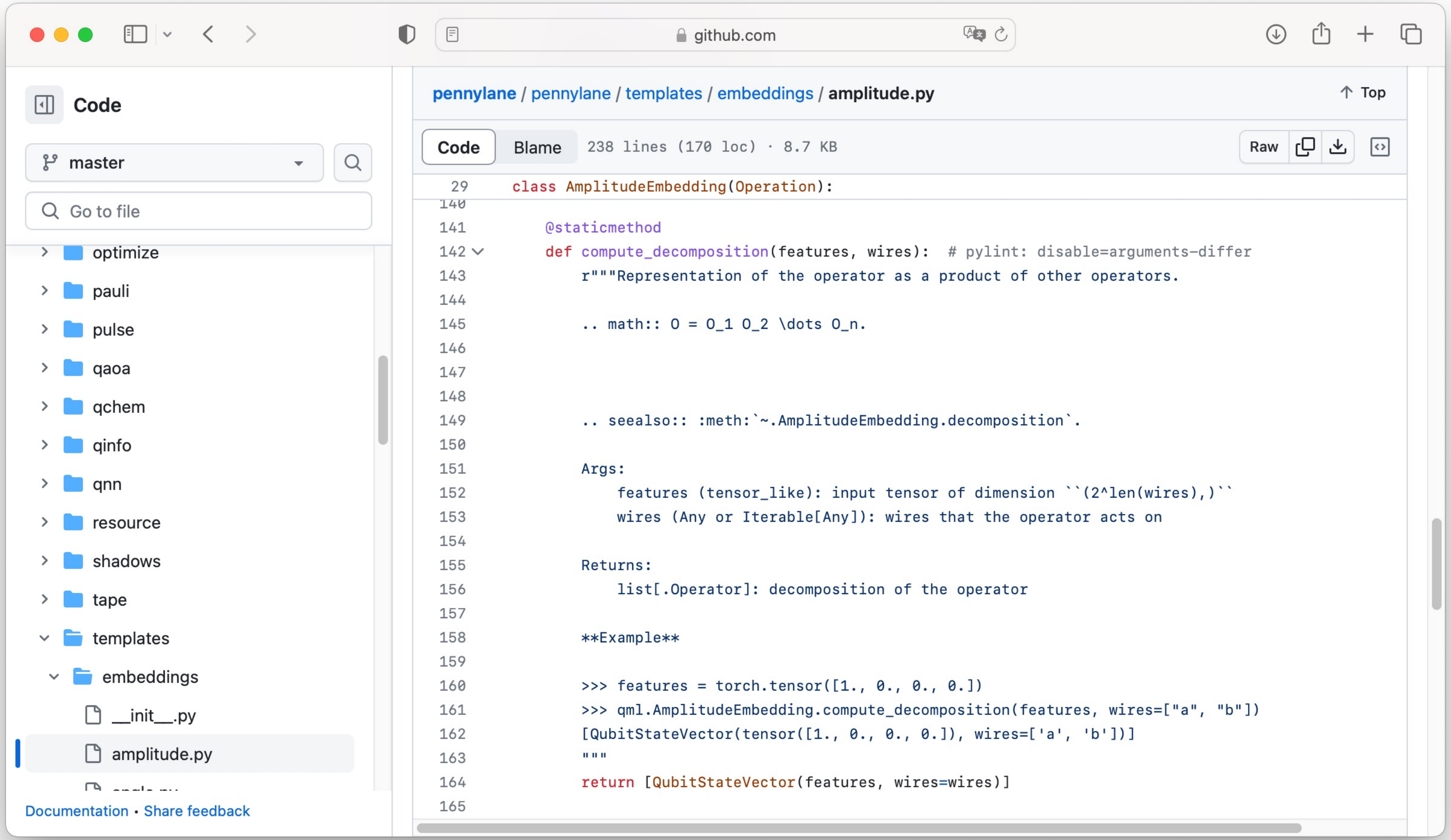
```python
import pennylane as qml
from pennylane import numpy as np

# import the template
from pennylane.templates.embeddings import BasisEmbedding

# quantum device where you want to run and how many Qubits
dev = qml.device('default.qubit', wires=6)

@qml.qnode(dev)
def circuit(data):
    for i in range(6):
        qml.Hadamard(i)
    for i in range(len(data)):
        BasisEmbedding(features=data[i], wires=range(6),do_queue=True)
    return  qml.state()

data=[[1,0,1,1,1,0],
      [1,0,0,0,0,1]]

circuit(data)

print(circuit.draw(show_all_wires=True))

#print output

 0: ──H──X──X──╭┤ State
 1: ──H────────├┤ State
 2: ──H──X─────├┤ State
 3: ──H──X─────├┤ State
 4: ──H──X─────├┤ State
 5: ──H──X─────╰┤ State
```

pl-basisencoding.py hosted with ❤ by GitHub   view raw

DataEmbedding — -zsh — 97×18

```
(base) yongsoo@yongsooui-MacBookPro DataEmbedding % python pennylane_embedding.py
/Users/yongsoo/anaconda3/lib/python3.10/site-packages/pennylane/operation.py:1034: UserWarning: The do_queue keyword argument is deprecated. Instead of setting it to False, use qml.queuing.QueuingManager.stop_recording()
  warnings.warn(do_queue_deprecation_warning, UserWarning)
ic| basis_state: [1, 0, 1, 1, 1, 0]
ic| ops_list: [PauliX(wires=[0]), PauliX(wires=[2]), PauliX(wires=[3]), PauliX(wires=[4])]
ic| basis_state: [1, 0, 0, 0, 0, 1]
ic| ops_list: [PauliX(wires=[0]), PauliX(wires=[5])]
(base) yongsoo@yongsooui-MacBookPro DataEmbedding %
```

Amplitude encoding



??