QSciTech - Quantum Machine Learning

Activity 2.1: Data encodings & Kernels

Christophe Pere

2024-01-24

Content

- Basis Encoding
- Amplitude Encoding
- Time-Evolution Encoding
- Hamiltonian Encoding
- Angle Encoding
- Quantum feature maps
- Quantum Metric Learning*
- Kernels

Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Me

Background in science

- Ph.D. in Astrophysics
- Lead Scientist at PINQ2
- Adjunct professor (AI) Université Laval
- Lecturer École Technologie Supérieure
  - Intro to QC
  - Intro to QML
- Author/Mentor

Experience

- More than 8 years in classical AI
- Learn quantum computing and QML during my free time
- Mentor for IBM (challenges and summer schools)
- Mentor in MILA
- Supervision of 17 students (Bac, MSc., Ph.D., Postdoc)
- Several partnerships in QC, QML, Math

Me

Background in science

- Ph.D. in Astrophysics
- Lead Scientist at PINQ2
- Adjunct professor (AI) Université Laval
- Lecturer École Technologie Supérieure
  - Intro to QC
  - Intro to QML
- Author/Mentor

Experience

- More than 8 years in classical AI
- Learn quantum computing and QML during my free time
- Mentor for IBM (challenges and summer schools)
- Mentor in MILA
- Supervision of 17 students (Bac, MSc., Ph.D., Postdoc)
- Several partnerships in QC, QML, Math

PINQ2

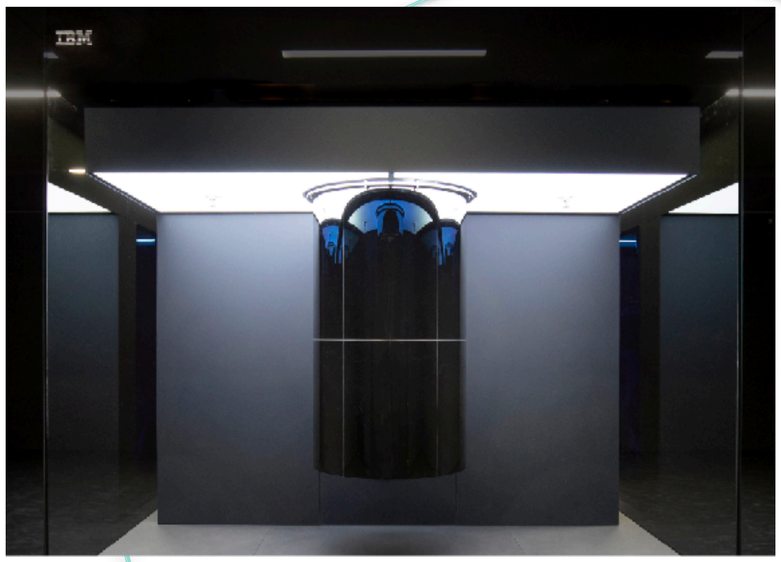
Plateforme d'innovation numérique et quantique

- Non-profit organization
  - De-risk innovation
- Classical HPC
  - CPUs, GPUs...

- Eagle 127 qubits

  ibm_quebec

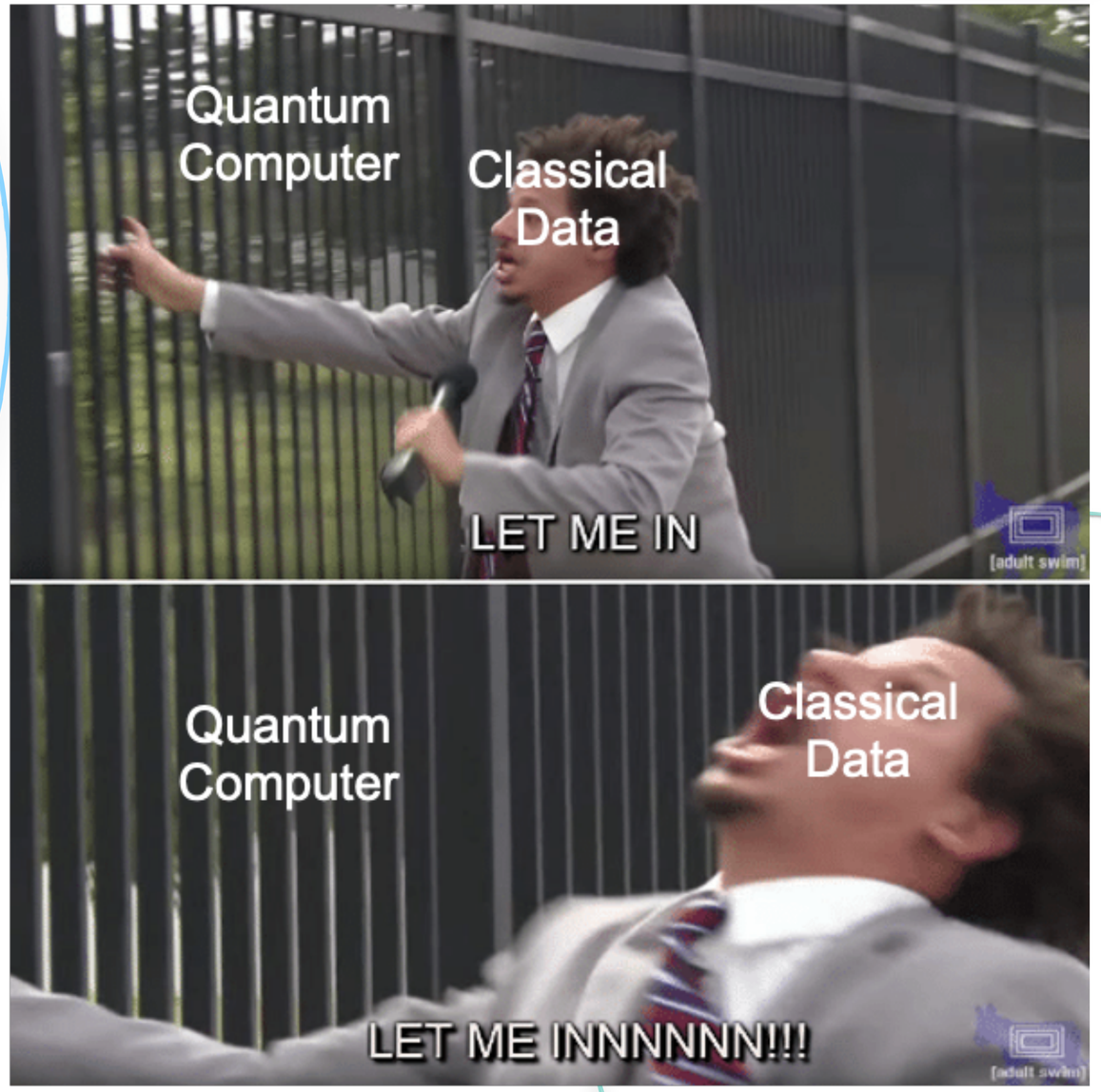
5th QC outside IBM lab



Introduction

Introduction

What is quantum data encoding?
Why is it important?



Introduction

What is quantum data encoding?
Why is it important?

Motivation

How to use classical data With a QC? How to encode them?

Introduction

What is quantum data encoding?
Why is it important?

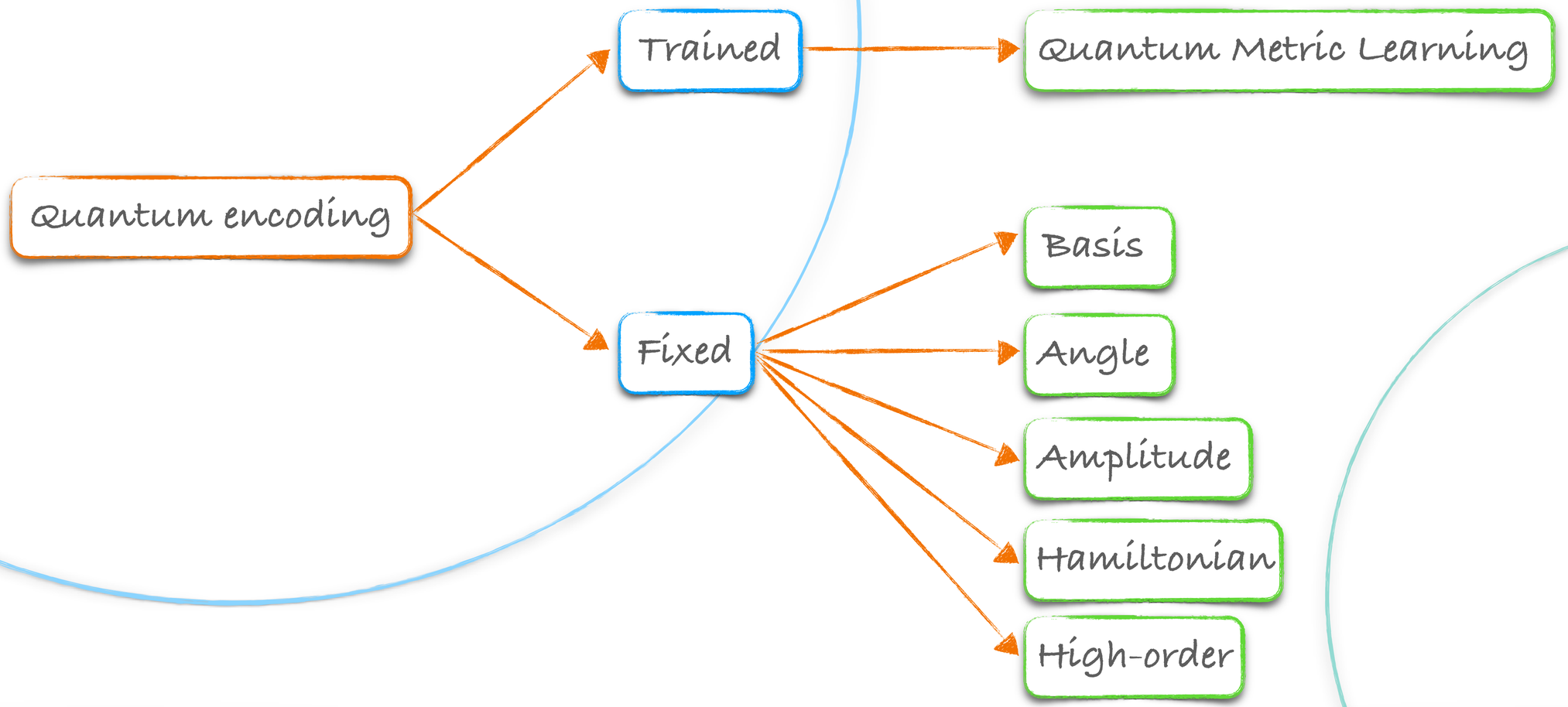
Motivation

How to use classical data With a QC? How to encode them?

Embeddings/ encodings

A series of algorithms and strategies for encoding classical data with quantum states

Introduction



Quantum Embeddings by Aroosa Ijaz - https://www.youtube.com/watch?v=mNR-7Omi1l0

Representation

Basis encoding of a binary string (1,0)
i.e. representation of integer 2

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$$

Amplitudes encoding of a complex vector
With unitary length
$(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$

Hamiltonian encoding with matrix A

$$e^{-iH_A t}$$

Temporal evolution encoding of a scalar t

Basis Encoding

Basis Encoding

Binary string representation

Real number to manipulate

Basis Encoding

Dataset directly represented

Binary string representation

Real number to manipulate

$x^{(m)} = (b_1, \ldots, b_N)$ with $b_i \in 0,1$ *for* $i = 1,...,N$

$x^{(m)}$ can be directly mapped into $|x^{(m)}\rangle$

Basis Encoding

Dataset directly represented

Binary string representation

Real number to manipulate

$x^{(m)} = (b_1, \ldots, b_N)$ with $b_i \in 0,1$ *for* $i = 1,...,N$

can be directly mapped into $|x^{(m)}\rangle$

Superposition

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle$$

Dataset

Basis Encoding

Dataset directly represented

Binary string representation

Real number to manipulate

$x^{(m)} = (b_1, \ldots, b_N)$ with $b_i \in 0,1$ *for* $i = 1,...,N$

can be directly mapped into $|x^{(m)}\rangle$

Superposition

$$|\mathscr{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle$$

Dataset

Example 1:

$$|x^{(1)}\rangle = |01\rangle$$

$$|x^{(2)}\rangle = |11\rangle$$

$$|\mathscr{D}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Basis Encoding

Dataset directly represented

Binary string representation

Real number to manipulate

$$x^{(m)} = (b_1, \ldots, b_N) \quad \text{with} \quad b_i \in 0,1 \; for \; i = 1,...,N$$

can be directly mapped into $|x^{(m)}\rangle$

Superposition

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle$$

Dataset

Example 1:

$$|x^{(1)}\rangle = |01\rangle$$

$$|x^{(2)}\rangle = |11\rangle$$

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Example 2:

4 qubits

$x = 1001 \rightarrow |1001\rangle$

$x = 1011 \rightarrow |1011\rangle$

$x = 0011 \rightarrow |0011\rangle$

Basis Encoding

Dataset directly represented

Binary string representation

Real number to manipulate

$$x^{(m)} = (b_1, \ldots, b_N) \quad \text{with} \quad b_i \in 0,1 \; for \; i = 1,...,N$$

can be directly mapped into $|x^{(m)}\rangle$

Superposition

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle$$

Dataset

Sign +/-
Add a supplementary bit
0 for + , 1 for -

Example 1:

$$|x^{(1)}\rangle = |01\rangle$$

$$|x^{(2)}\rangle = |11\rangle$$

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Example 2:

4 qubits

$x = 1001 \rightarrow |1001\rangle$

$x = 1011 \rightarrow |1011\rangle$

$x = 0011 \rightarrow |0011\rangle$

Basis Encoding

Dataset directly represented

Binary string representation

Real number to manipulate

$x^{(m)} = (b_1, \ldots, b_N)$ with $b_i \in 0,1$ *for* $i = 1,...,N$

Can be directly mapped into $|x^{(m)}\rangle$

Example 1:

$$|x^{(1)}\rangle = |01\rangle$$

$$|x^{(2)}\rangle = |11\rangle$$

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Superposition

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle$$

Dataset

Sign +/-
Add a supplementary bit
0 for + , 1 for -

Example 2:

4 qubits

$x = 1001 \rightarrow |1001\rangle$
$x = 1011 \rightarrow |1011\rangle$
$x = 0011 \rightarrow |0011\rangle$

Advantage	A dataset can be placed into superposition
Disadvantage	1 bit string is encoded into a qubit

N bits represent 2^N possible basis states

As $M \ll 2^N$ Encoding D will be sparse

Amplitude Encoding

Amplitude Encoding

Vector of real numbers

Amplitude Encoding

Vector of real numbers

Vector representation

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^n} \end{pmatrix}$$

Amplitude Encoding

Vector of real numbers

Vector representation

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^n} \end{pmatrix} \quad \longleftrightarrow \quad |\psi_x\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle$$

Amplitude Encoding

Vector of real numbers

Vector representation

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^n} \end{pmatrix} \longleftrightarrow |\psi_x\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle$$

Normalized and normed values
used as amplitudes

Complex

Normalized values

$$x \in \mathbb{C}^{2^n} \; ; \; \sum_k |x_k|^2 = 1$$

Amplitude Encoding

Vector of real numbers

Vector representation

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^n} \end{pmatrix} \longleftrightarrow |\psi_x\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle$$

Complex

Normalized values

$$x \in \mathbb{C}^{2^n} \; ; \; \sum_k |x_k|^2 = 1$$

Normalized and normed values used as amplitudes

Matrix representation

$$A \in \mathbb{C}^{2^n \times 2^n} \; ; \; \sum_{ij} |\alpha_{ij}|^2 = 1 \qquad |\psi_A\rangle = \sum_{i=0}^{2^m-1} \sum_{j=0}^{2^n-1} \alpha_{ij} |i\rangle |j\rangle$$

ith row

jth column

Amplitude Encoding

Vector of real numbers

Example

$x = (0.073, \ -0.438, \ 0.730, \ 0.000)$ Normalized vector

Amplitude Encoding

Vector of real numbers

Rappel $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\psi^*\rangle = \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|0\rangle + \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|1\rangle$$

Example

$x = (0.073, -0.438, 0.730, 0.000)$ Normalized vector

Amplitudes encoding

$$|\psi_x\rangle = 0.073|00\rangle - 0.438|01\rangle + 0.730|10\rangle + 0.000|11\rangle$$

Amplitude Encoding

Vector of real numbers

Rappel $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\psi^*\rangle = \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|0\rangle + \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|1\rangle$$

Example

$x = (0.073, -0.438, 0.730, 0.000)$ Normalized vector

Amplitudes encoding

$$|\psi_x\rangle = 0.073|00\rangle - 0.438|01\rangle + 0.730|10\rangle + 0.000|11\rangle$$

Corresponding matrix

$$A = \begin{array}{c} \begin{matrix} |0\rangle & |1\rangle \end{matrix} \\ \begin{pmatrix} 0.073 & -0.438 \\ 0.730 & 0.000 \end{pmatrix} \end{array} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix}$$

Amplitude Encoding

Vector of real numbers

Rappel $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\psi^*\rangle = \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|0\rangle + \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|1\rangle$$

Example

$x = (0.073, -0.438, 0.730, 0.000)$ Normalized vector

Amplitudes encoding

$$|\psi_x\rangle = 0.073|00\rangle - 0.438|01\rangle + 0.730|10\rangle + 0.000|11\rangle$$

2 qubits to encode 4 features

8 qubits for 256 features

Corresponding matrix

$$A = \begin{pmatrix} |0\rangle & |1\rangle \\ 0.073 & -0.438 \\ 0.730 & 0.000 \end{pmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix}$$

Amplitude Encoding

Vector of real numbers

Rappel $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\psi^*\rangle = \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|0\rangle + \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|1\rangle$$

Example

$x = (0.073, -0.438, 0.730, 0.000)$ Normalized vector

Amplitudes encoding

$$|\psi_x\rangle = 0.073|00\rangle - 0.438|01\rangle + 0.730|10\rangle + 0.000|11\rangle$$

2 qubits to encode 4 features

8 qubits for 256 features

Corresponding matrix

$$A = \begin{pmatrix} 0.073 & -0.438 \\ 0.730 & 0.000 \end{pmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix}$$

Gain in qubits compared to basis encoding

More compact representation, but higher cost in terms of calculation time

Amplitude Encoding

Vector of real numbers

Rappel $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\psi^*\rangle = \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|0\rangle + \frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|1\rangle$$

Example

$x = (0.073, -0.438, 0.730, 0.000)$ Normalized vector

Amplitudes encoding

$$|\psi_x\rangle = 0.073|00\rangle - 0.438|01\rangle + 0.730|10\rangle + 0.000|11\rangle$$

2 qubits to encode 4 features

8 qubits for 256 features

Corresponding matrix

$$A = \begin{pmatrix} |0\rangle & |1\rangle \\ 0.073 & -0.438 \\ 0.730 & 0.000 \end{pmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix}$$

How do you encode a dataset?

Amplitude Encoding

How do you encode a dataset?

Dataset: $\mathcal{D} = \{x^1, x^2, \ldots, x^M\}$

Quantum embedding - Pennylane blog - https://pennylane.ai/qml/glossary/quantum_embedding/

Amplitude Encoding

How do you encode a dataset?

Dataset: $\mathcal{D} = \left\{x^1, x^2, \ldots, x^M\right\}$

Amplitudes: $\alpha = C_{norm}\ x_1^{(1)}, \ldots, x_N^{(1)}, x_1^{(2)}, \ldots, x_N^{(2)}, \ldots, x_1^{(M)}, \ldots, x_N^{(M)}$

Quantum embedding - Pennylane blog - https://pennylane.ai/qml/glossary/quantum_embedding/

Amplitude Encoding

How do you encode a dataset?

Dataset: $\mathcal{D} = \{x^1, x^2, \ldots, x^M\}$

Amplitudes: $\alpha = C_{norm}\ x_1^{(1)}, \ldots, x_N^{(1)}, x_1^{(2)}, \ldots, x_N^{(2)}, \ldots, x_1^{(M)}, \ldots, x_N^{(M)}$

Normalization constant

Sample 1

Sample 2

Sample M

Amplitude Encoding

How do you encode a dataset?

Dataset: $\mathcal{D} = \{x^1, x^2, \ldots, x^M\}$

Amplitudes: $\alpha = C_{norm}\, x_1^{(1)}, \ldots, x_N^{(1)}, x_1^{(2)}, \ldots, x_N^{(2)}, \ldots, x_1^{(M)}, \ldots, x_N^{(M)}$

Normalization constant

Sample 1

Sample 2

Sample M

Normalization criteria

$$|\alpha|^2 = 1$$

$$|\mathcal{D}\rangle = \sum_{i=1}^{2^n} \alpha_i |i\rangle$$

Dataset

Element of amplitude vector

Basis state

Amplitude Encoding

How do you encode a dataset?

Dataset: $\mathscr{D} = \{x^1, x^2, \ldots, x^M\}$

Amplitudes: $\alpha = C_{norm}\, x_1^{(1)}, \ldots, x_N^{(1)}, x_1^{(2)}, \ldots, x_N^{(2)}, \ldots, x_1^{(M)}, \ldots, x_N^{(M)}$

Normalization constant

Sample 1

Sample 2

Sample M

We could store an exponentially higher volume of data

Normalization criteria

$$|\alpha|^2 = 1$$

$$|\mathscr{D}\rangle = \sum_{i=1}^{2^n} \alpha_i |i\rangle$$

Basis state

Dataset

Element of amplitude vector

Number of amplitudes NxM

Encoding needs

$n \geq \log_2 (N \times M)$ qubits

Quantum embedding - Pennylane blog - https://pennylane.ai/qml/glossary/quantum_embedding/

Amplitude Encoding

Amplitude-Efficient State Preparation

Rotations all branches in superposition

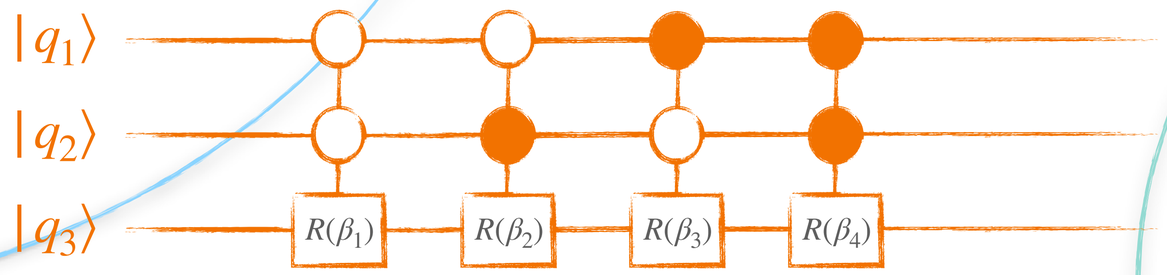
Each different on each branch

Multi-controlled rotations

A cascade of R_z

Equalizes the sign of the amplitude to make a global phase

$|0...0\rangle$ ← Inverse process ← $|1...1\rangle$



Followed by a cascade of R_y

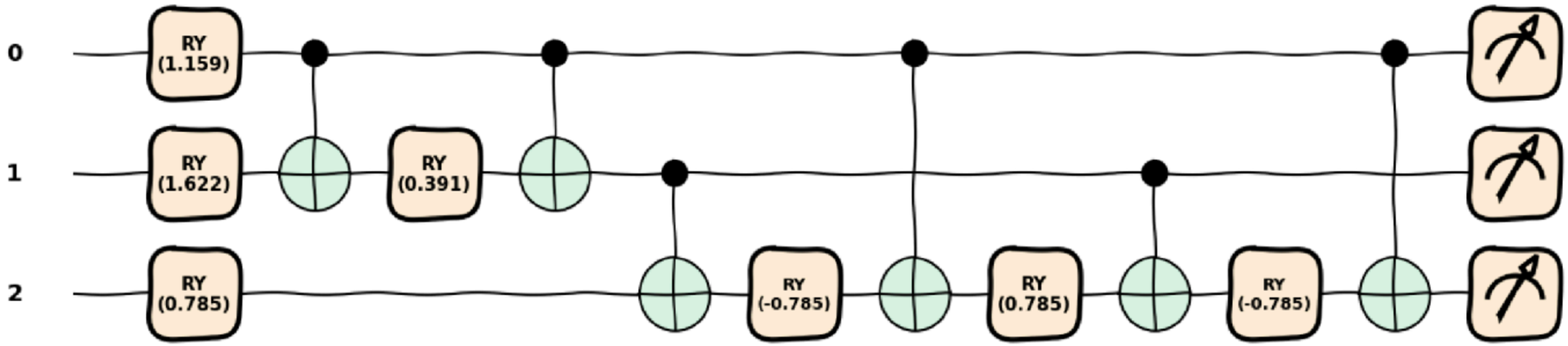
Equalize amplitudes to ground state

Highly time consuming

Amplitude Encoding

Amplitude-Efficient State Preparation

$$|\psi\rangle = \sqrt{0.2}\,|000\rangle + \sqrt{0.5}\,|010\rangle + \sqrt{0.2}\,|110\rangle + \sqrt{0.1}\,|111\rangle + 0\,|100\rangle + 0\,|001\rangle + 0\,|011\rangle + 0\,|101\rangle$$



Time-Evolution Encoding

Time-Evolution Encoding

Temporal evolution operator $U(x) = e^{-ixH}$

Time-Evolution Encoding

Temporal evolution operator

$$U(x) = e^{-ixH}$$

Hamiltonian

$$H = \frac{1}{2}\sigma_a \; ; \; a = \{x, y, z\}$$

Pauli operator

Applied to the initialization state 0

$$|\psi(x)\rangle = U(x)\,|\psi_0\rangle$$

Real value

Time-Evolution Encoding

Temporal evolution operator

$$U(x) = e^{-ixH}$$

$$H = \frac{1}{2}\sigma_a \; ; \; a = \{x, y, z\}$$

Pauli operator

Hamiltonian

Applied to the initialization state 0

$$|\psi(x)\rangle = U(x)\,|\psi_0\rangle$$

Real value

Example: $x = -0.438$ → $U(x) = e^{-ixH}$ With RY operator we have:

$$|\psi(-0.438)\rangle = \cos\left(\frac{-0.438}{2}\right)|0\rangle + \sin\left(\frac{-0.438}{2}\right)|1\rangle$$

$$\approx 0.976\,|0\rangle - 0.217\,|1\rangle$$

Time-Evolution Encoding

Temporal evolution operator $U(x) = e^{-ixH}$

$H = \frac{1}{2}\sigma_a \; ; \; a = \{x, y, z\}$

Pauli operator

Hamiltonian

Applied to the initialization state 0

$|\psi(x)\rangle = U(x)\,|\psi_0\rangle$

Real value

Example: $x = -0.438$ → $U(x) = e^{-ixH}$ With RY operator we have:

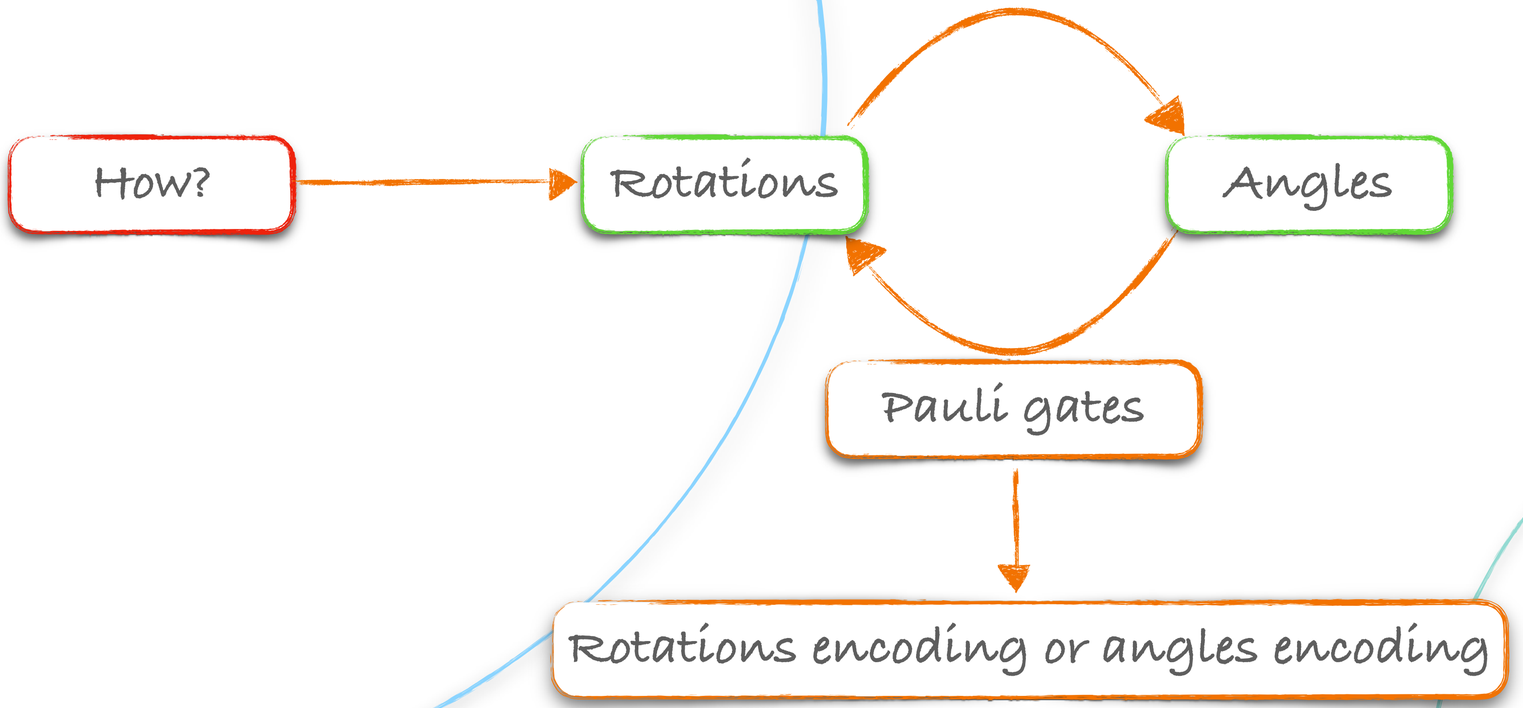
$$|\psi(-0.438)\rangle = \cos\left(\frac{-0.438}{2}\right)|0\rangle + \sin\left(\frac{-0.438}{2}\right)|1\rangle$$

$$\approx 0.976\,|0\rangle - 0.217\,|1\rangle$$

To encode a real value vector $\mathbf{x} \in \mathbb{R}^{\mathbf{N}}$ We must apply $U(x)$ N times for each value $x \in \mathbf{x}$

Time-Evolution Encoding

Code!!



Angle Encoding

And, what about phases?

Phase shift -> Rotation in the complex plan

Angles

Rotation (X, Y ou Z)

$$R_x(\theta) = e^{-i\theta X/2} = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -isin(\theta/2) & cos(\theta/2) \end{bmatrix}$$

$$R_y(\theta) = e^{-i\theta Y/2} = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ sin(\theta/2) & cos(\theta/2) \end{bmatrix}$$

$$R_z(\theta) = e^{-i\theta Y/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

1 feature / qubit

Angle Encoding

Angle Encoding

And, what about phases?

Phase shift -> Rotation in the complex plan

Angles

Rotation (X, Y ou Z)

$$data = \begin{bmatrix} 6 & -12.5 & 11.15 & 7 \\ 8 & 9.5 & -11 & -5 \\ 5 & 0.5 & 8 & -7 \end{bmatrix}$$

Angle Encoding

And, what about phases?

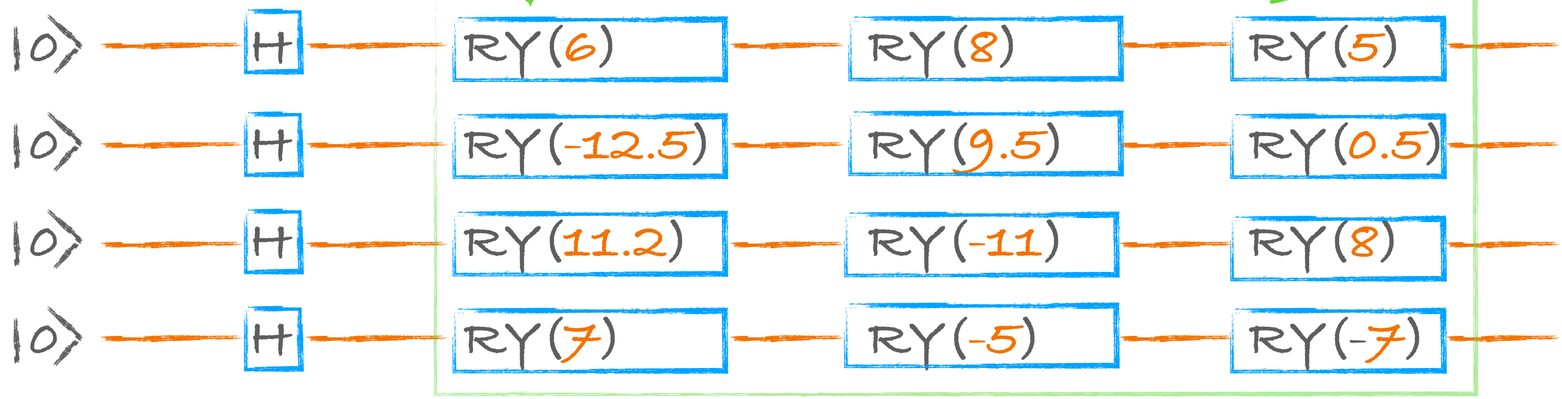
Phase shift -> Rotation in the complex plan

Angles

Rotation (X, Y ou Z)

$$data = \begin{bmatrix} 6 & -12.5 & 11.15 & 7 \\ 8 & 9.5 & -11 & -5 \\ 5 & 0.5 & 8 & -7 \end{bmatrix}$$

1 feature / qubit



Ansatz

VQA

Portes Multi-controle

Linear time per variable and qubit

Hamiltonian Encoding

Hamiltonian Encoding

To follow the "time-evolution"

For each unitary matrix, there exists a real vector that:

$$\mathbf{A} = \alpha + \beta\mathbf{X} + \delta\mathbf{Y} + \gamma\mathbf{Z}$$

$$\begin{aligned} e^{i\mathbf{A}t} &= e^{i(\alpha+\beta X+\delta Y+\gamma Z)t} \\ &= e^{i\alpha tI} e^{i\beta Xt} e^{i\delta Yt} e^{i\gamma Zt} \\ &= R_x(\beta t) R_y(\delta t) R_z(\gamma t) \end{aligned}$$

Pauli X

Pauli Y

Pauli Z

Hamiltonian Encoding

To follow the "time-evolution"

$$H_A = \begin{pmatrix} 0 & \mathbf{A} \\ \mathbf{A}^\dagger & 0 \end{pmatrix}$$

Hamiltonian

Amplitude-encoded vector

$$\mathbf{A} = \begin{pmatrix} 0.073 & -0.438 \\ 0.730 & 0.000 \end{pmatrix}$$

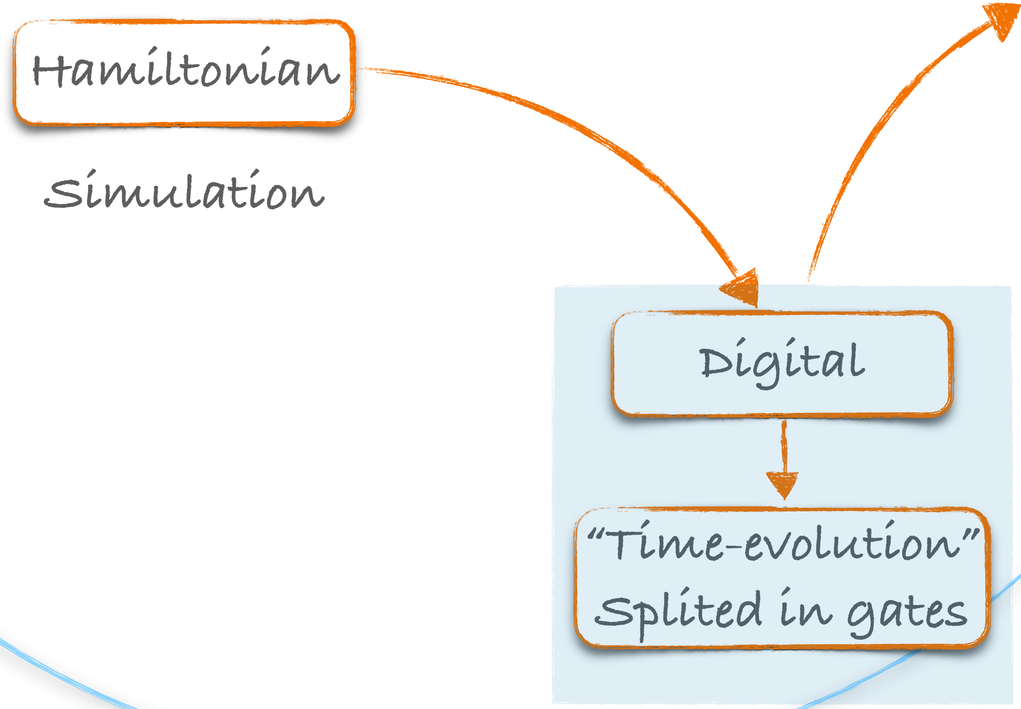
$$H_A = \begin{pmatrix} 0 & 0 & 0.073 & -0.438 \\ 0 & 0 & 0.730 & 0.000 \\ 0.073 & 0.730 & 0 & 0 \\ -0.438 & 0.000 & 0 & 0 \end{pmatrix}$$

Eigenvectors

Eigenvalues

Hamiltonian Encoding

To follow the "time-evolution"



Most of the time H is not diagonalizable

Decomposition into small Hamiltonians which are easily simulable

$$H = \sum_{j=1}^{J} H_j$$

Factorization

$$e^{-i\sum_j H_j t} \neq \prod_j e^{-iH_j t}$$

Trotterization

$$e^{-i\sum_j H_j t} \neq \prod_j e^{-iH_j t} + \mathcal{O}(t^2)$$

Chain of Pauli's operators

Hamiltonian Encoding

To follow the "time-evolution"

Most of the time H is not diagonalizable

Decomposition into small Hamiltonians which are easily simulable

$$H = \sum_{j=1}^{J} H_j$$

Factorization

$$e^{-i\sum_j H_j t} \neq \prod_j e^{-iH_j t}$$

Trotterization

$$e^{-i\sum_j H_j t} \neq \prod_j e^{-iH_j t} + \mathcal{O}(t^2)$$

Good approximation when t is small

$$H = XIIII + IXIII + IIXII + IIXZI + IIIIY$$

Pauli's operators

Equivalent to a local problem with few interactions

Autres Encodages?

- BasisEmbedding
- AmplitudeEmbedding
- AngleEmbedding
- DisplacementEmbedding
- IQPEmbedding
- QAOAEmbedding
- SqueezingEmbedding

Encoding

- CVNeuralNetworkLayers
- RandomLayers
- StronglyEntanglingLayers
- SimplifiedTwoDesign
- BasicEntanglerLayers

Layer

And so much more...

State preparation

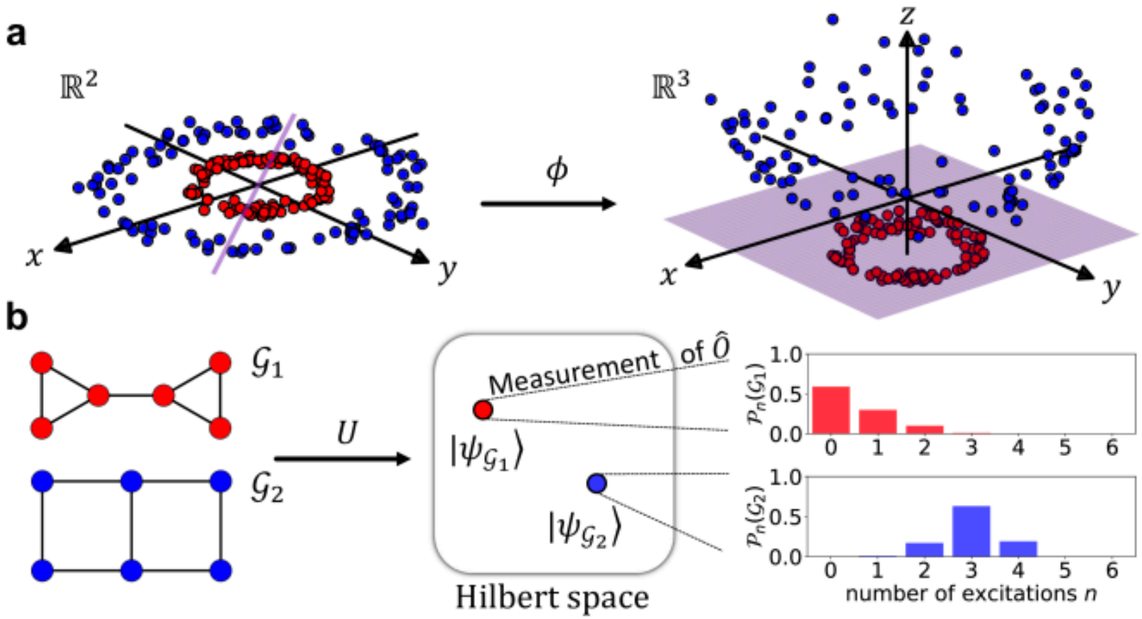
Tensor Network

Chemistry

Quantum Feature Maps

Quantum Feature Maps

Data in 2D space $\xrightarrow{\phi}$ Projection in 3D space



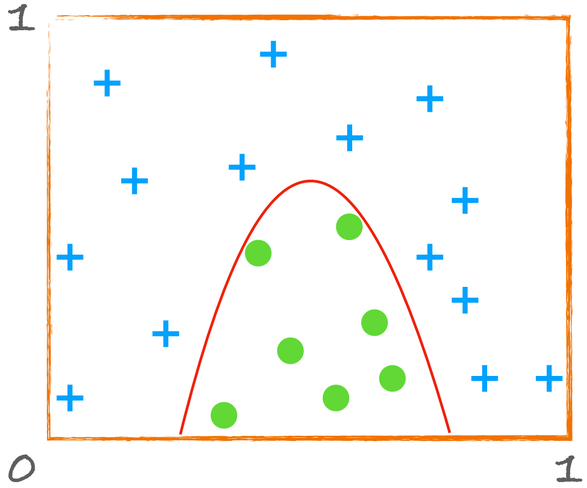
Way to project data into higher dimension

Albrecht et al. 2023, Quantum Feature Maps for Graph Machine Learning on a Neutral Atom Quantum Processor, https://www.arxiv-vanity.com/papers/2211.16337/

Quantum Feature Maps

How to use a linear method with a problem which is not linear?

2D



Feature Map

$x \in \Omega$ data space

$\mathcal{H} \in \mathbb{R}^N$ feature space

$\phi : \Omega \to \mathcal{H} \in \mathbb{R}^N$

$\phi : x \to \phi(x) \in \mathbb{R}^N$ non linear map $\phi(\lambda x + \mu y) \neq \lambda\phi(x) + \mu\phi(y)$

$\Omega \subseteq [0,1] \times [0,1]$

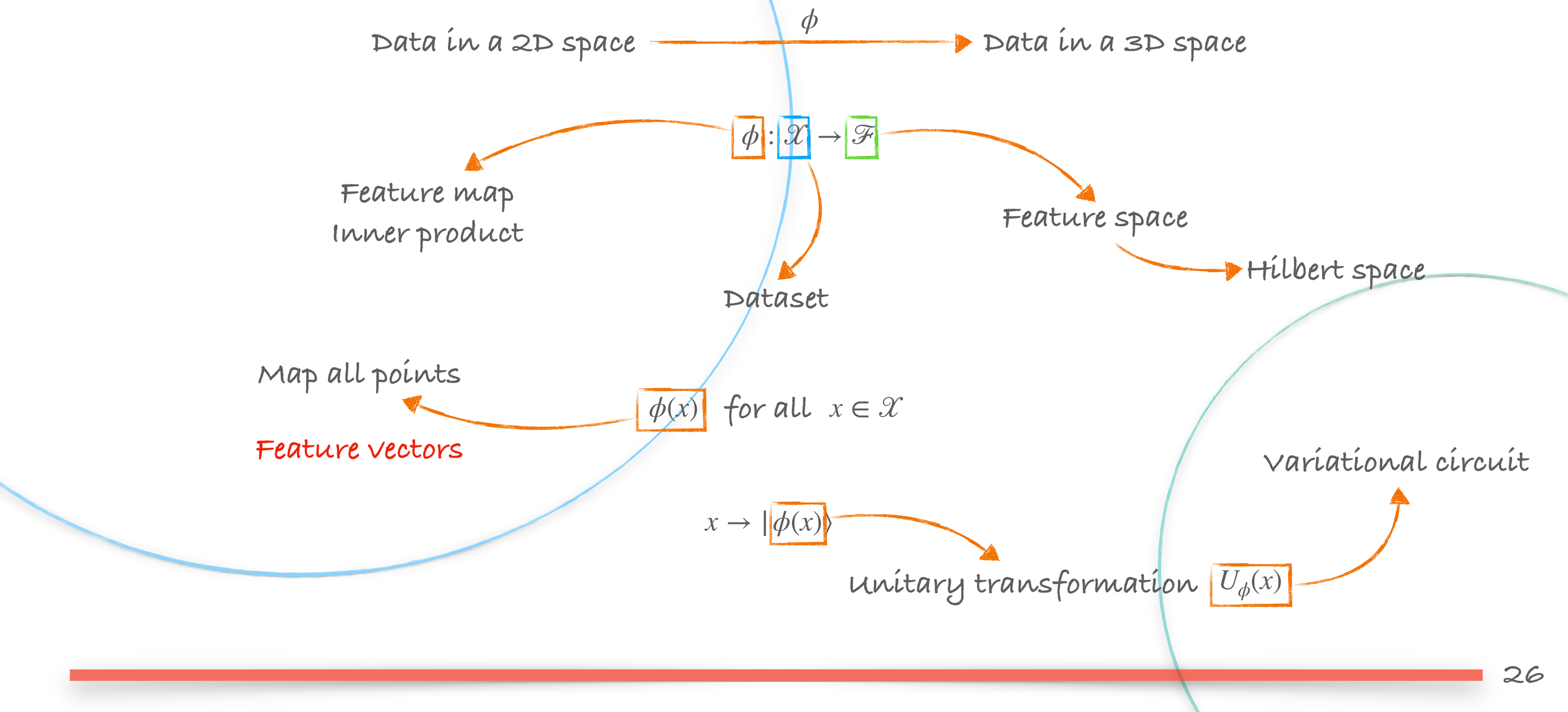
$x \subseteq \Omega(x_1, x_2)$

$$\phi_2(x_1, x_2) \to \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$ 6D

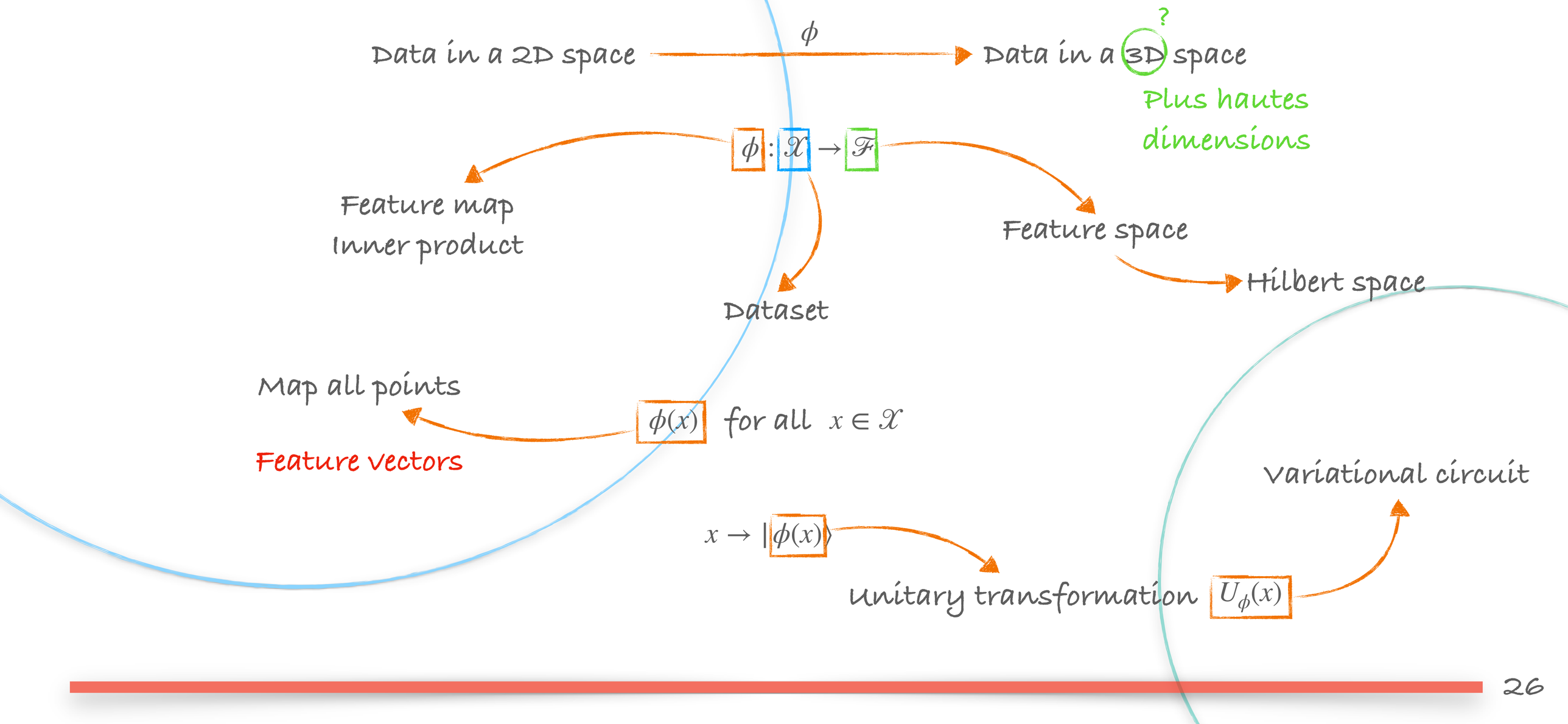
Bet that features will be linearly separable when projected into a higher dimension

QGSS, 2021, Lecture 6.2 Quantum Feature Spaces and Kernels, https://www.youtube.com/watch?v=zw3JYUrS-v8&list=PLOFEBzvs-VvqJwybFxkTiDzhf5E11p8BI&index=14

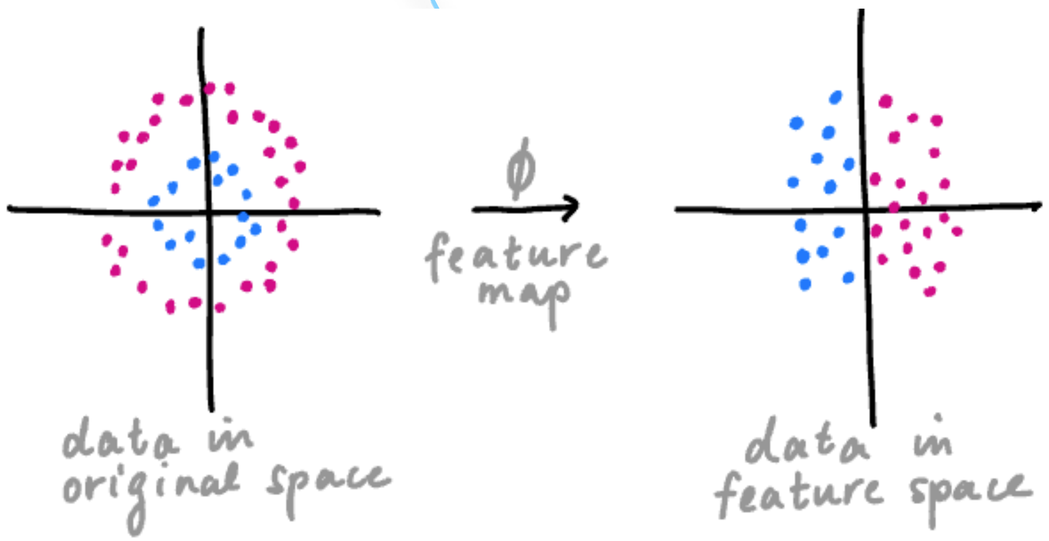
Quantum Feature Maps



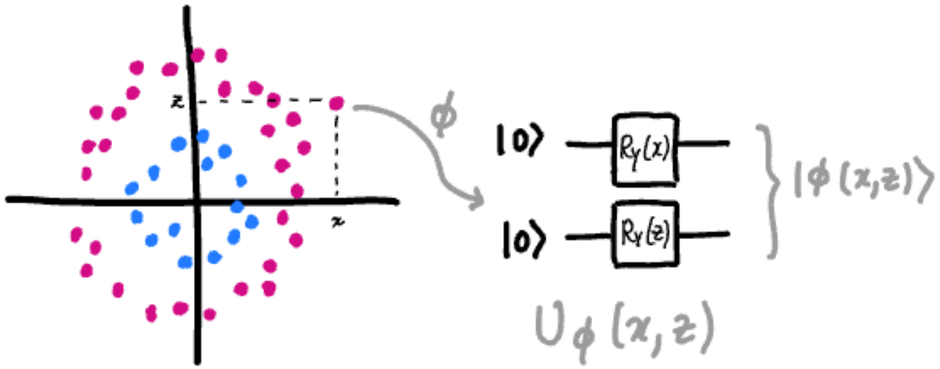
Quantum Feature Maps



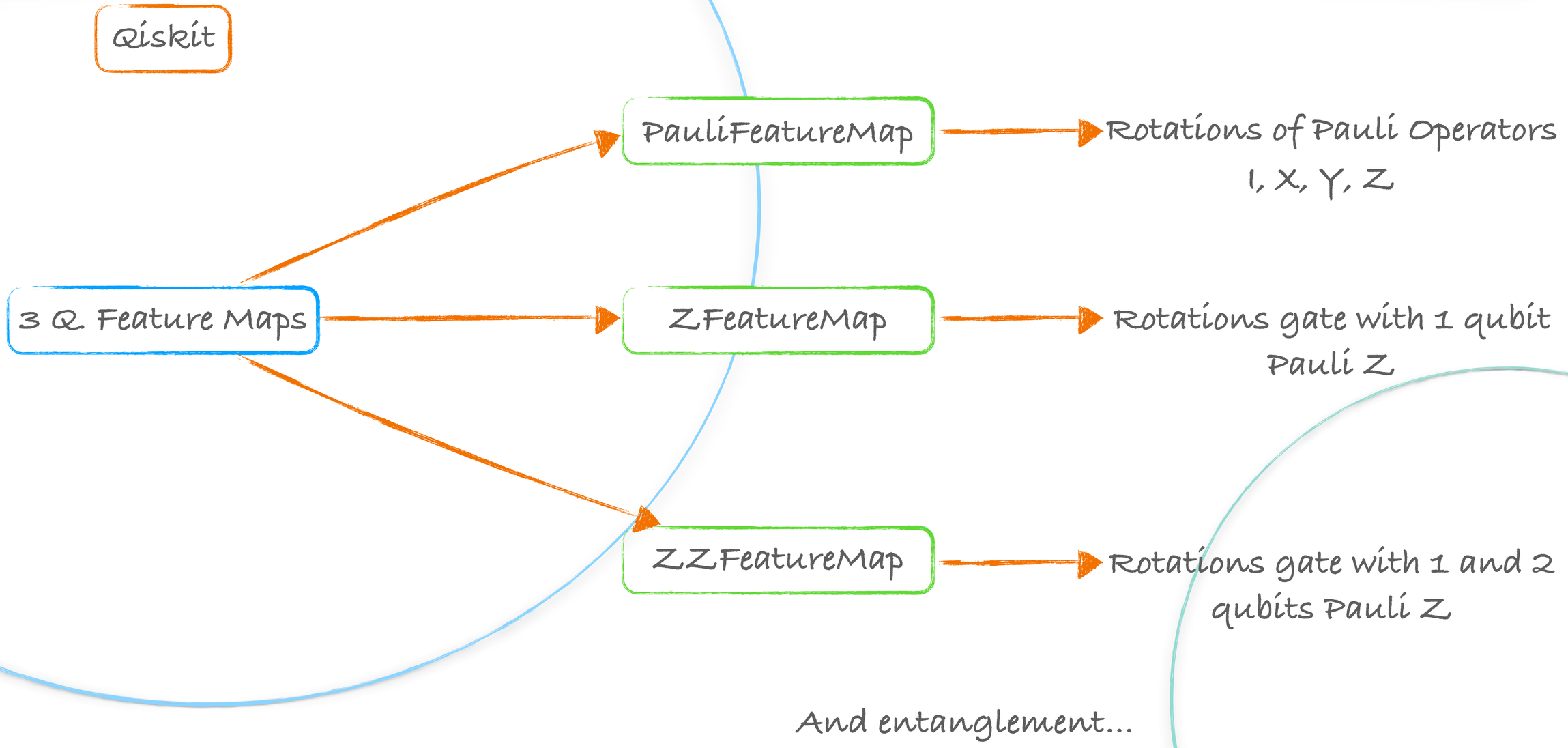
Quantum Feature Maps



A feature map can transform data into a space where it is easier to process.



Quantum Feature Maps



Albrecht et al. 2023, Quantum Feature Maps for Graph Machine Learning on a Neutral Atom Quantum Processor, https://www.arxiv-vanity.com/papers/2211.16337/

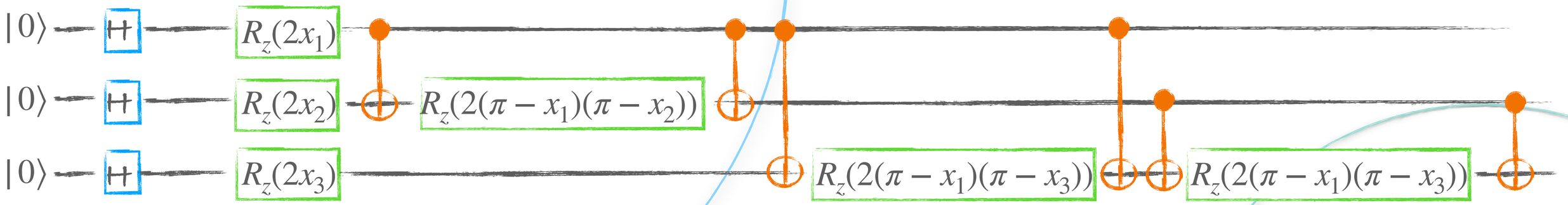
Quantum Feature Maps

ZZFeatureMap

3 qubits 3 features

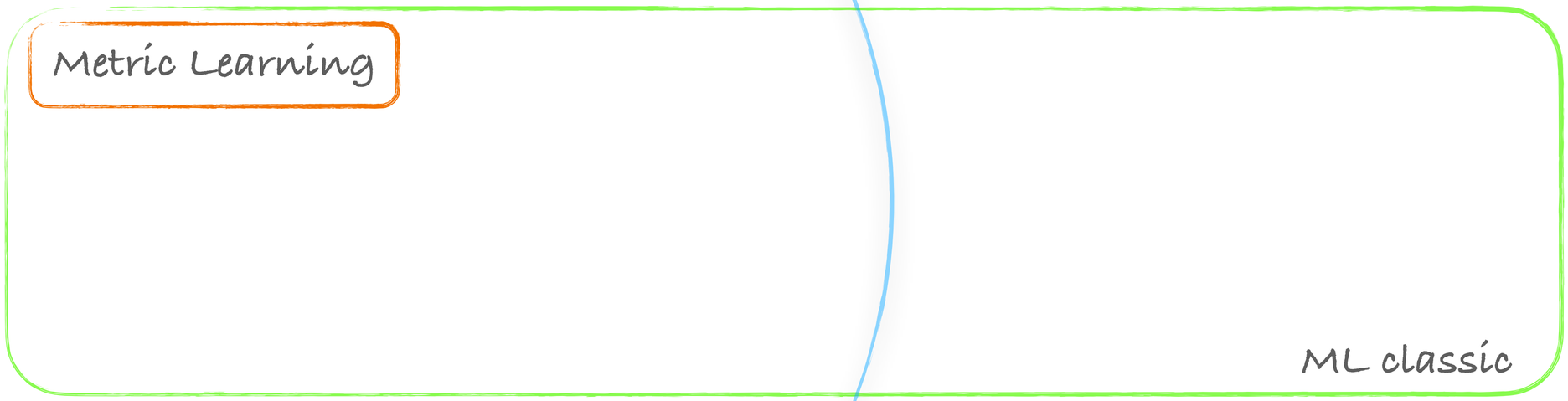
1 rep

Full entanglement



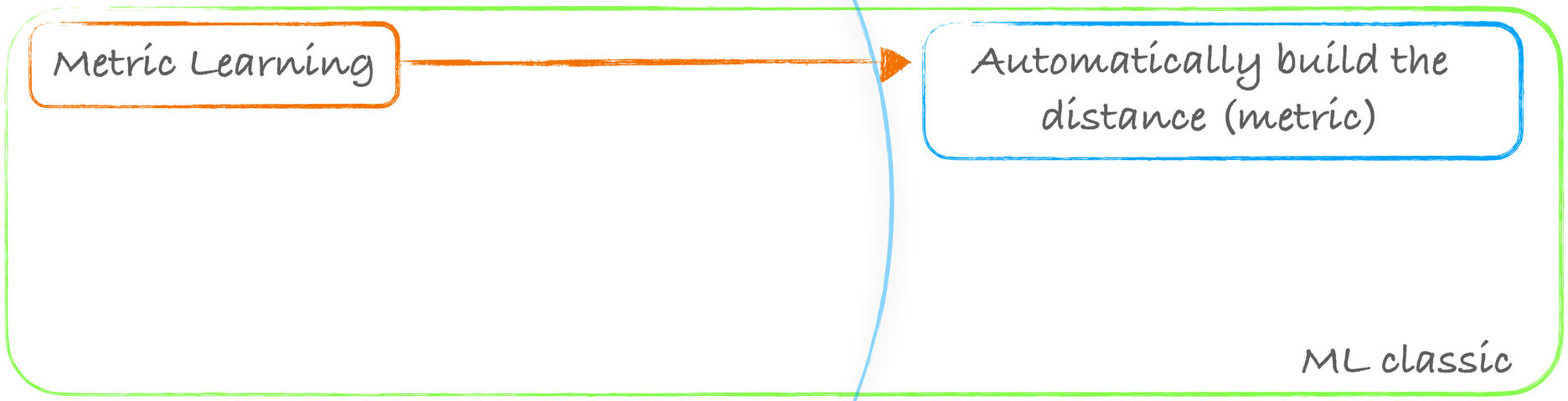
Quantum Metric Learning

Quantum Metric Learning



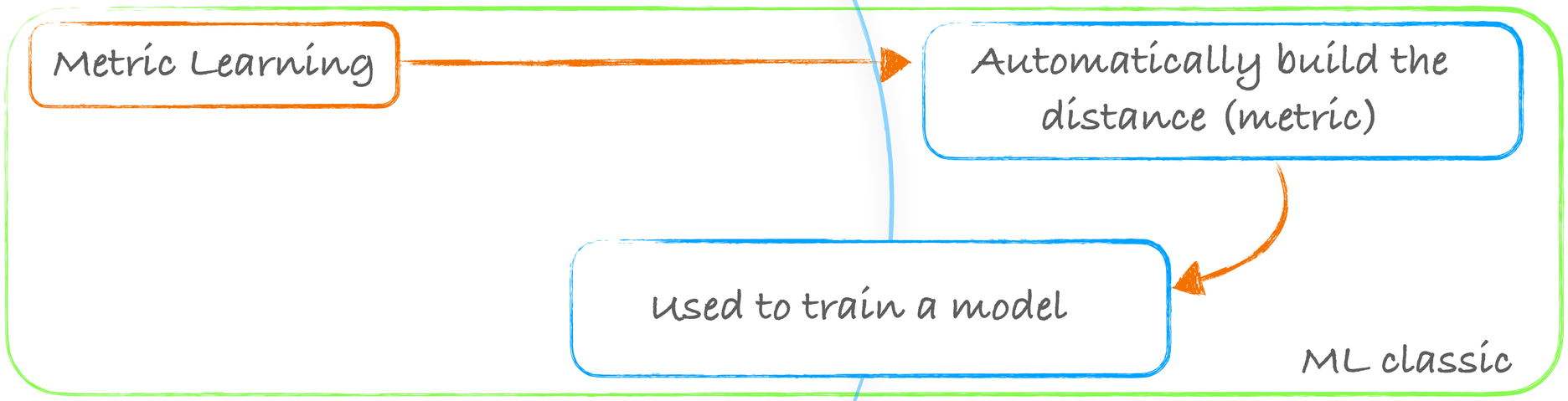
Quantum Embeddings by Aroosa Ijaz, 2021, https://www.youtube.com/watch?v=mNR-7Omil10

Quantum Metric Learning



Quantum Embeddings by Aroosa Ijaz, 2021, https://www.youtube.com/watch?v=mNR-7Omil1o

Quantum Metric Learning



Quantum Embeddings by Aroosa Ijaz, 2021, https://www.youtube.com/watch?v=mNR-7Omil1o

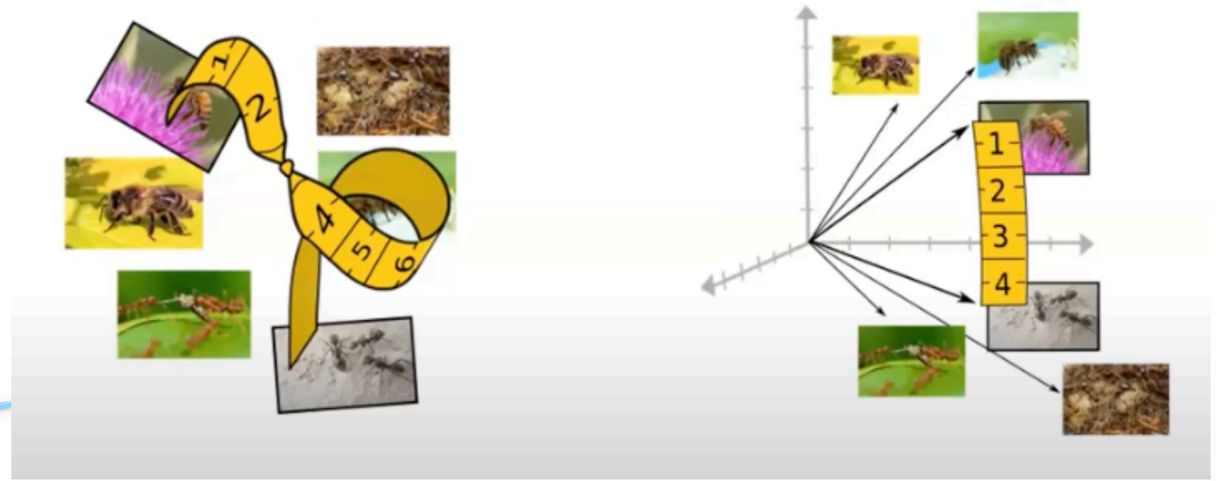
Quantum Metric Learning

Metric Learning

Automatically build the distance (metric)

Used to train a model

ML classic



Quantum Embeddings by Aroosa Ijaz, 2021, https://www.youtube.com/watch?v=mNR-7Omil1o

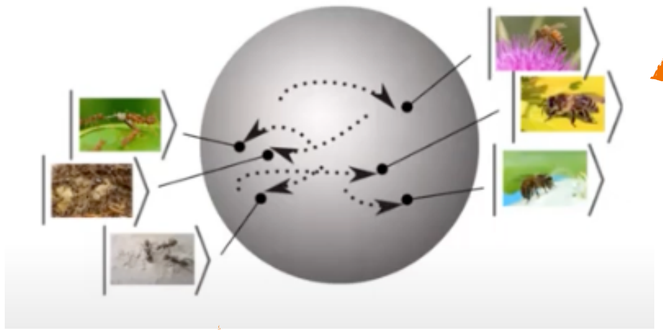
Quantum Metric Learning

Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Quantum Metric Learning

Quantum Metric Learning

Random states to encode data points



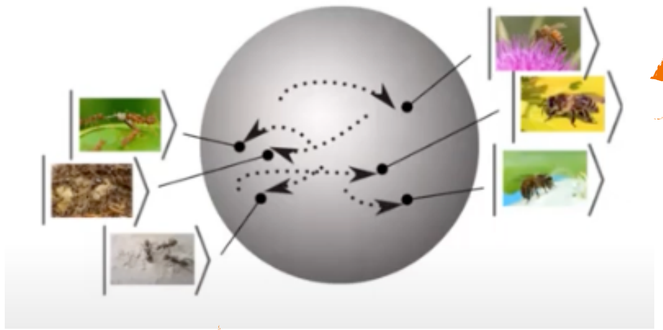
Patterns? Differences?

Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Quantum Metric Learning

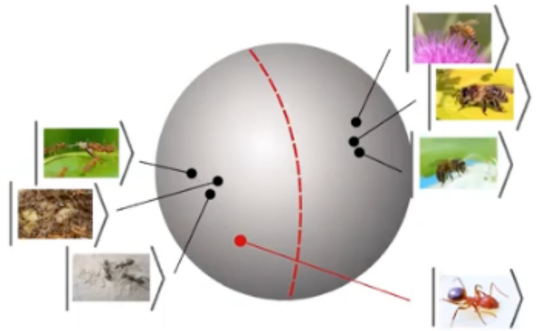
Quantum Metric Learning

Random states to encode data points



Patterns? Differences?

Variational approach



The algorithm learns how to separate the data based on their trained embeddings

Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

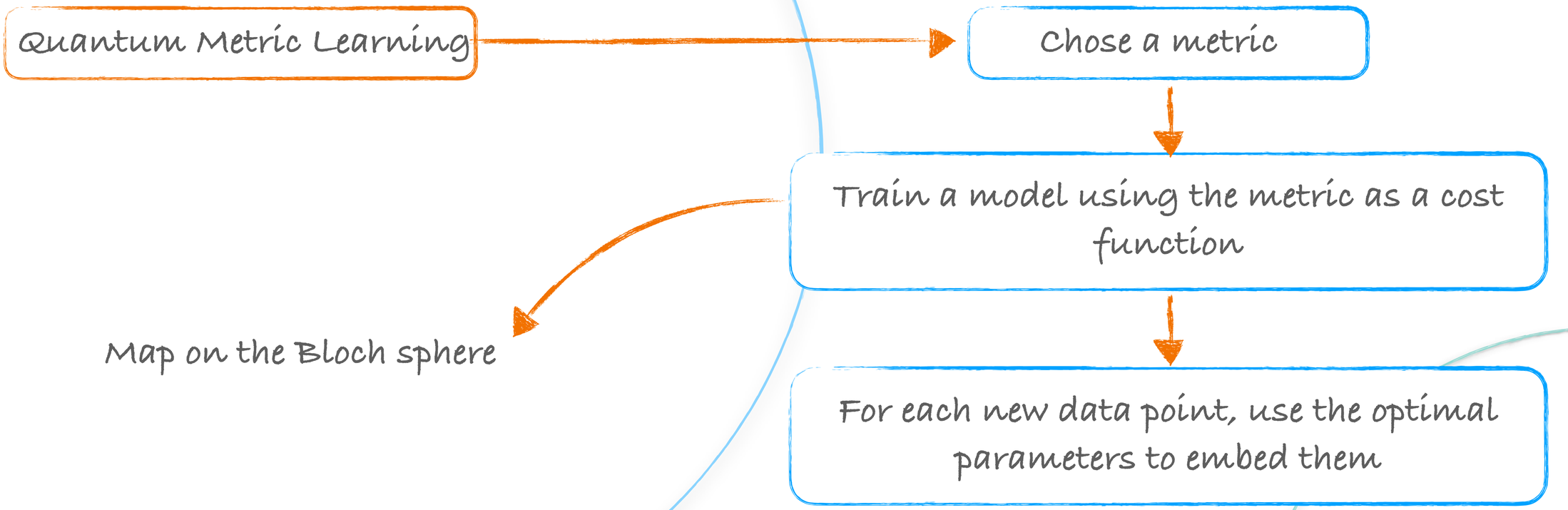
Quantum Metric Learning

Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Quantum Metric Learning

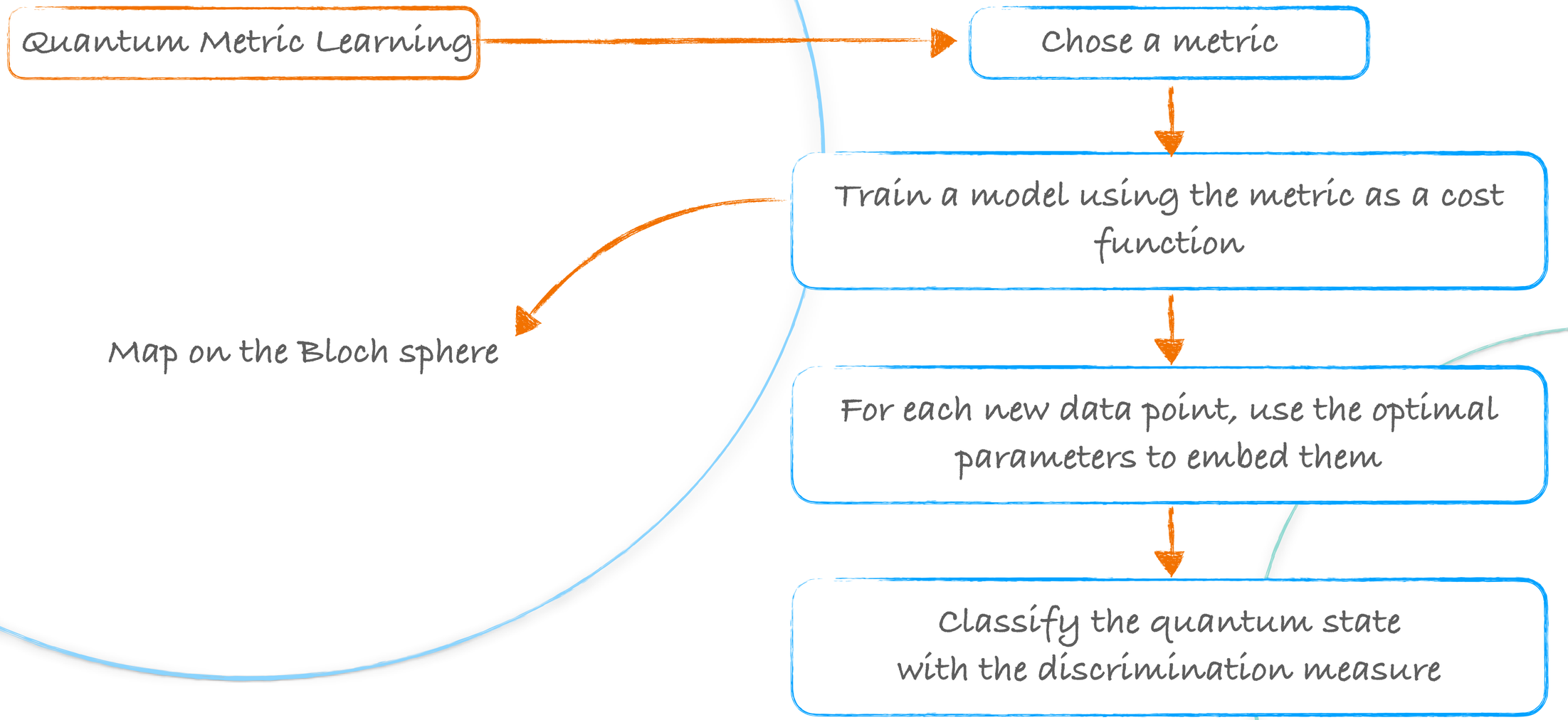
Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Quantum Metric Learning



Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Quantum Metric Learning

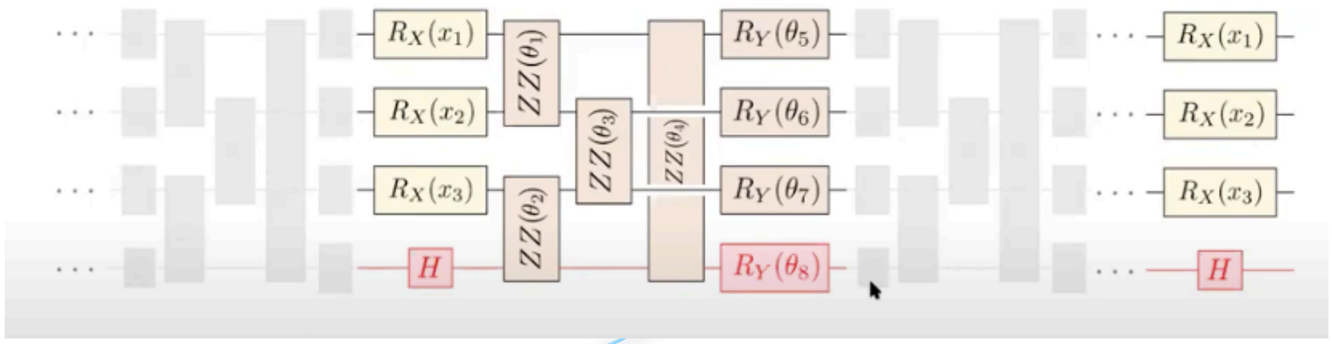


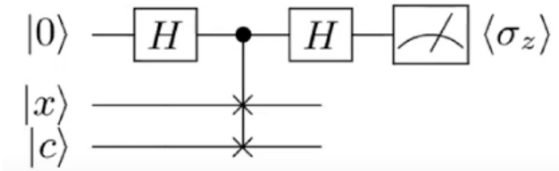
Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Quantum Metric Learning

Quantum Metric Learning

QAOA Embedding Ansatz





Fidelity measurement $F = |\langle c|x\rangle|^2$

Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

Kernels

Kernels

Classical Kernel

A kernel function lives the feature space

$K : \Omega \times \Omega \to \mathbb{R}$

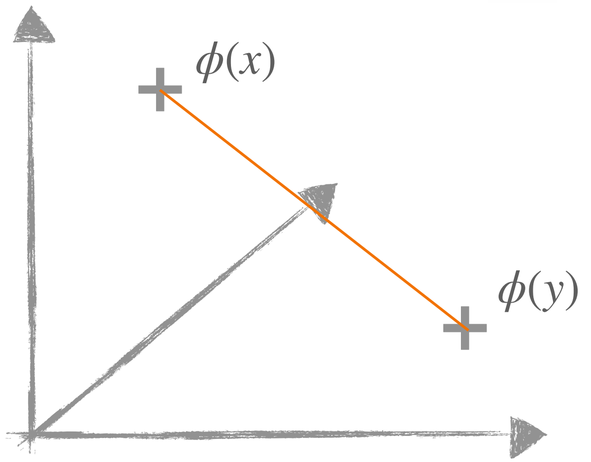
$K : (x, y) \to \langle \phi(x), \phi(y) \rangle$

→ Euclidean distance/product

Distance metric

Norm

$||.||_2^2$



Kernels

Classical Kernel

A kernel function lives the feature space

$K : \Omega \times \Omega \to \mathbb{R}$

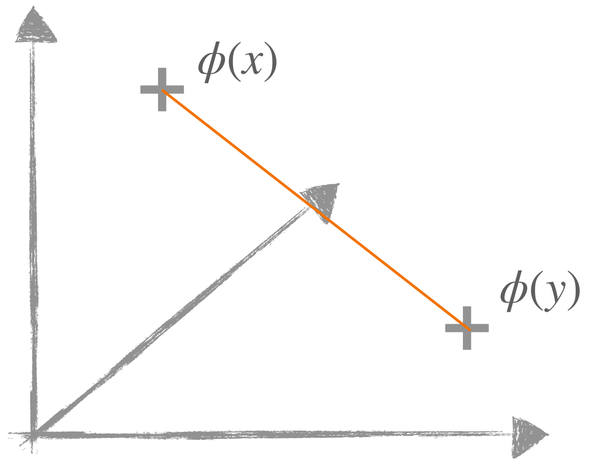
$K : (x, y) \to \langle \phi(x), \phi(y) \rangle$

→ Euclidean distance/product

Distance metric

Norm

$||\cdot||_2^2$



Quantum Kernel

Compute the inner product (Hilbert-Schmidt) between matrices

$\langle A, B \rangle_{HS} = \sum_{ij} A_{ij}^{\dagger} B_{ij}$

$|\phi(x)\rangle = U(x)|0\rangle$

$\Phi(x) = |\phi(x)\rangle\langle\phi(x)| = U(x)|0\rangle\langle 0|U^{\dagger}(x)$

Projector / density matrix

$K(x, y) = \langle \Phi(x), \Phi(y) \rangle_{HS}$

$= |\langle \Phi(x) | \Phi(y) \rangle|^2 \in \mathbb{R}^+$

$= |\langle 0 | U^{\dagger}(x) U(y) | 0 \rangle|^2 \in \mathbb{R}^+$

Transition amplitude

Kernels

Quantum Kernel

$$K(x,y) = \langle \Phi(x), \Phi(y) \rangle_{HS}$$
$$= |\langle \Phi(x) | \Phi(y) \rangle|^2 \in \mathbb{R}^+$$
$$= |\langle 0 | U^\dagger(x) U(y) | 0 \rangle|^2 \in \mathbb{R}^+$$

Overlap between 2 data points

Transition amplitude

What is the probability to obtain |0> after applying $U^\dagger(x)U(y)$?

Kernels

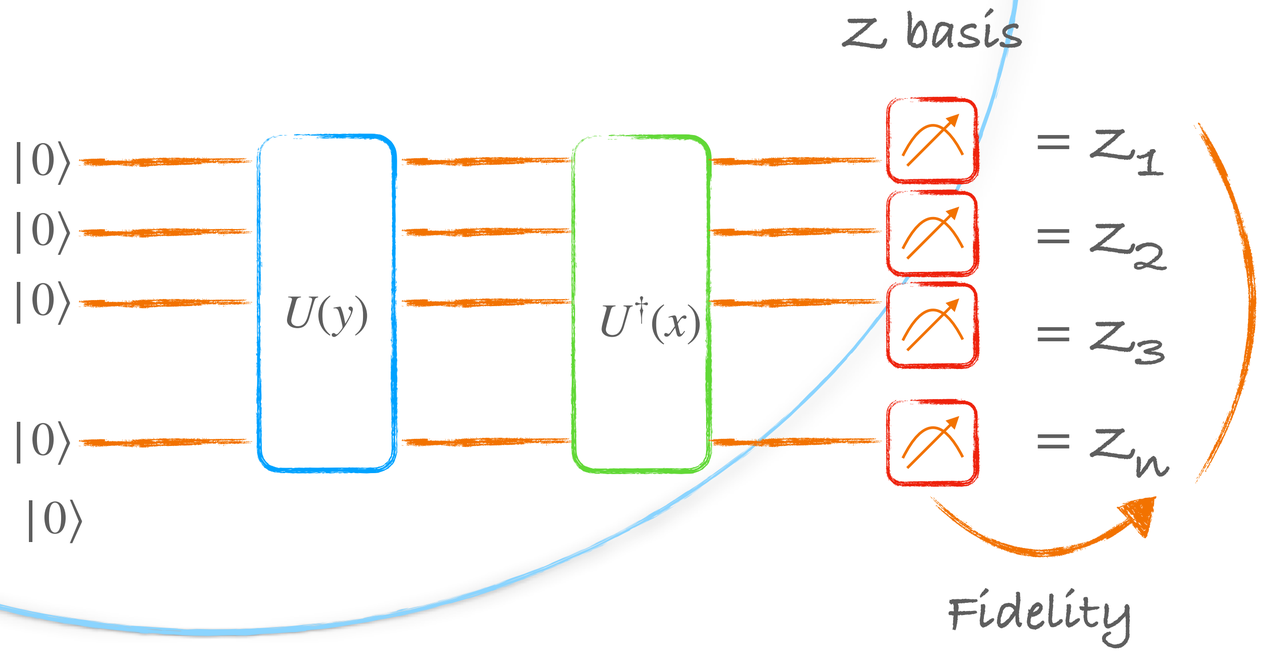
Quantum Kernel

$$K(x,y) = \langle \Phi(x), \Phi(y) \rangle_{HS}$$
$$= |\langle \Phi(x) | \Phi(y) \rangle|^2 \in \mathbb{R}^+$$
$$= |\langle 0 | U^\dagger(x) U(y) | 0 \rangle|^2 \in \mathbb{R}^+$$

Overlap between 2 data points

Transition amplitude

What is the probability to obtain |0> after applying $U^\dagger(x)U(y)$?



Kernels

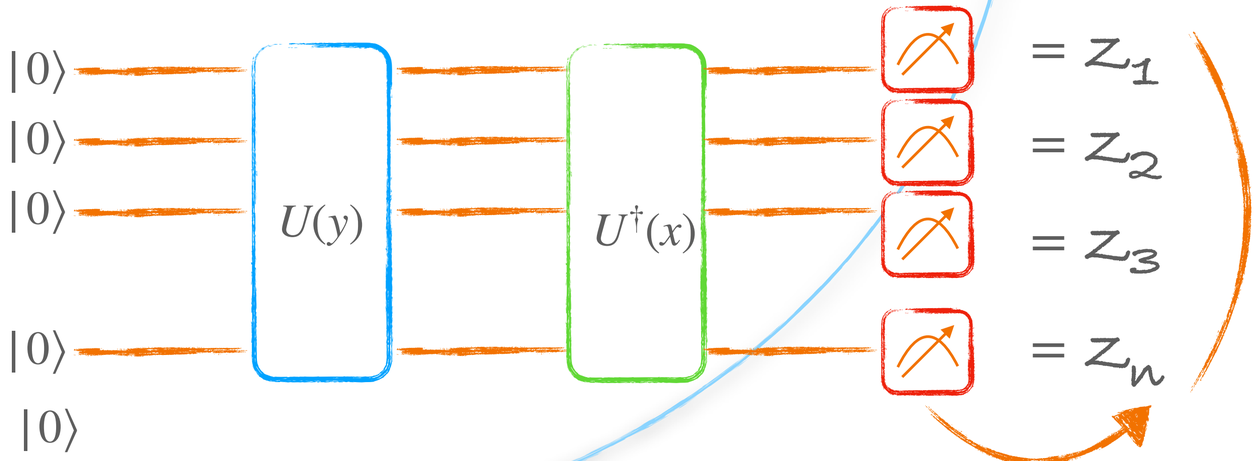
Quantum Kernel

$$K(x,y) = \langle \Phi(x), \Phi(y) \rangle_{HS}$$
$$= |\langle \Phi(x) | \Phi(y) \rangle|^2 \in \mathbb{R}^+$$
$$= |\langle 0 | U^\dagger(x) U(y) | 0 \rangle|^2 \in \mathbb{R}^+$$

Overlap between 2 data points

Transition amplitude

What is the probability to obtain |0> after applying $U^\dagger(x)U(y)$?



Kernel estimation

r times

Count how many times we got bitstring 0

Exact value of the kernel

Kernels

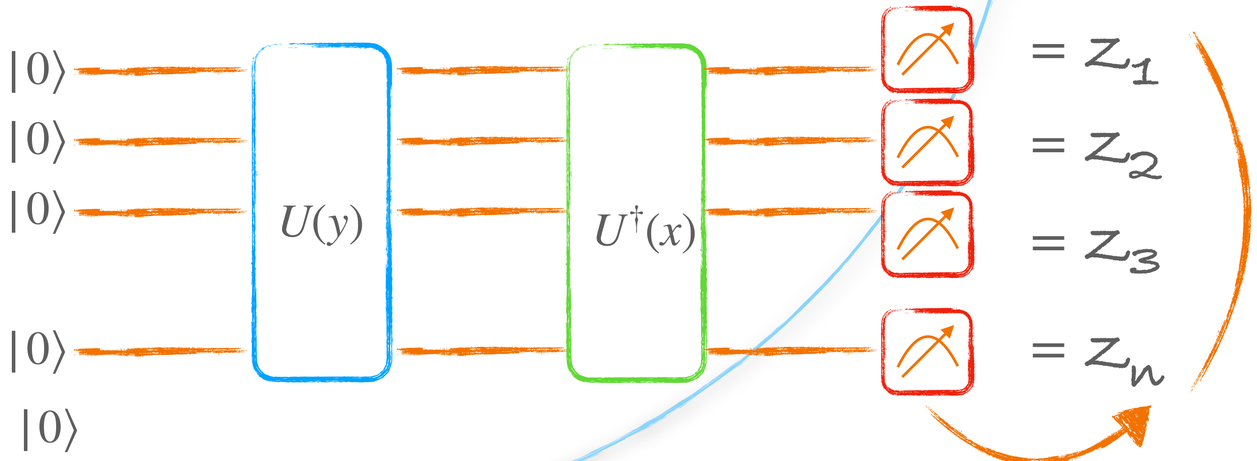
Quantum Kernel

$$K(x,y) = \langle \Phi(x), \Phi(y) \rangle_{HS}$$
$$= |\langle \Phi(x) | \Phi(y) \rangle|^2 \in \mathbb{R}^+$$
$$= |\langle 0 | U^\dagger(x) U(y) | 0 \rangle|^2 \in \mathbb{R}^+$$

Overlap between 2 data points

Transition amplitude

What is the probability to obtain |0> after applying $U^\dagger(x)U(y)$?



Fidelity

Count how many times we got bitstring 0

r times

Kernel estimation

Exact value of the kernel

Kernel estimation

$$\hat{K}(x,y) = K(x,y) + \frac{1}{\sqrt{R}}$$

Number of repetitions

$$\hat{K}(x,y) = \frac{r}{R} + \frac{1}{\sqrt{R}}$$

For m samples we have to perform the estimation m^2 times

References

[1] Llyod et al., 2020, Quantum embeddings for machine learning, https://arxiv.org/pdf/2001.03622.pdf

[2] Quantum Embeddings by Aroosa Ijaz - https://www.youtube.com/watch?v=mNR-7OmiI1o

[3] https://towardsdatascience.com/the-machine-learning-workflow-explained-557abf882079

[4] Albrecht et al. 2023, Quantum Feature Maps for Graph Machine Learning on a Neutral Atom Quantum Processor, https://www.arxiv-vanity.com/papers/2211.16337/

[5] Quantum embedding - Pennylane blog - https://pennylane.ai/qml/glossary/quantum_embedding/

[6] QGSS, 2021, Lecture 6.2 Quantum Feature Spaces and Kernels, https://www.youtube.com/watch?v=zw3JYUrS-v8&list=PLOFEBzvs-VVqJwybFxkTiDzhf5E11p8BI&index=14