Lecture 6: Amplitude Encoding and ML

Andreas Wichert

Department of Computer Science and Engineering

Técnico Lisboa

Overview

- Amplitude Encoding
- Estimating Scalar Product
- qRAM
- Quantum Kernels
- Variational Classifier

Amplitude Encoding

- Amplitude encoding encodes a real or complexed value vector of the **length one** into the amplitudes of a quantum state $|q_2q_1q_0\rangle$
- We build a top-down binary tree that divides the probability of observing $|q_2\rangle$ on the first level,
- The probability of observing $|q_2q_1\rangle$ on the second level
- The probability of observing $|q_2q_1q_0\rangle$ on the third level representing the required superposition $|\psi\rangle$
- The binary tree is represented by multi control rotation gates and requires $log_2\, n$ qubits to represent a vector of dimension n

Amplitude encoding encodes data into the amplitudes ω_i of a quantum state.

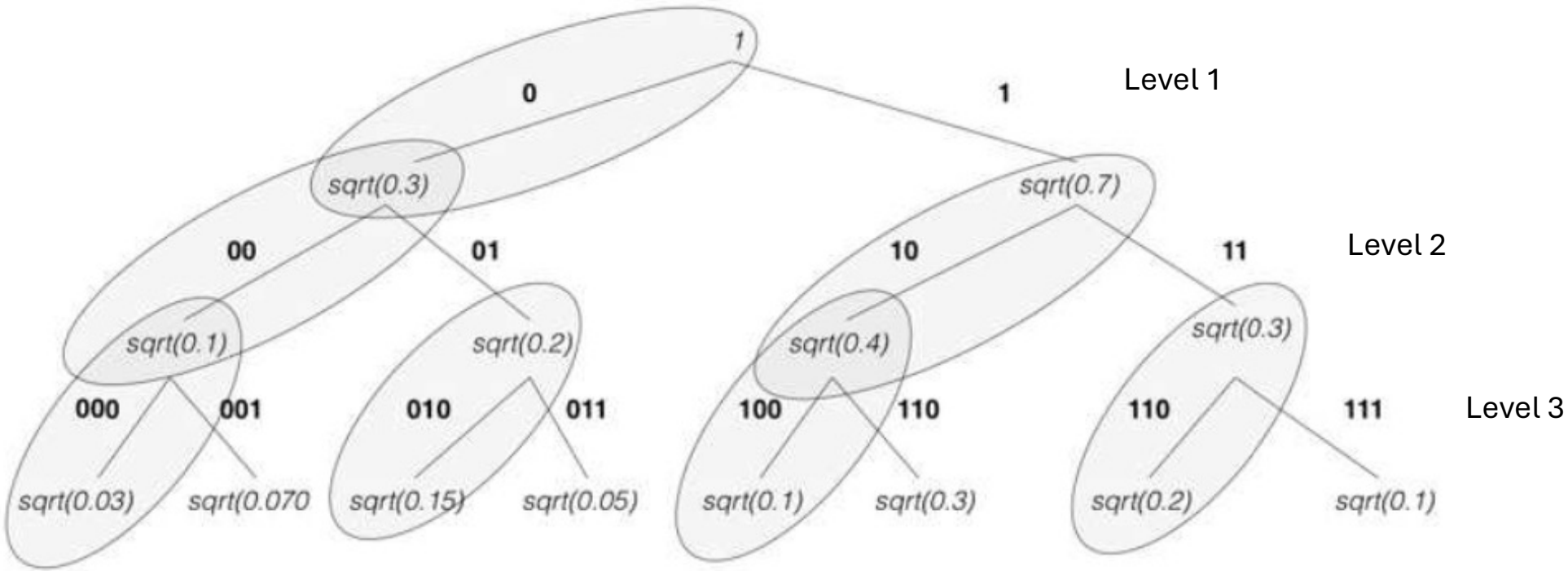
$$|\psi\rangle = \sum_{i=1}^{N} \omega_i \cdot |x\rangle$$

A complex normalized vector $\mathbf{x}$ (length one), for example

$$\mathbf{x} = \begin{pmatrix} \sqrt{0.03} \\ \sqrt{0.07} \\ \sqrt{0.15} \\ \sqrt{0.05} \\ \sqrt{0.1} \\ \sqrt{0.3} \\ \sqrt{0.2} \\ \sqrt{0.1} \end{pmatrix}.$$

with *qiskit* little endian ordering $|q_2 q_1 q_0\rangle$

$$|\psi\rangle = \sqrt{0.03} \cdot |000\rangle + \sqrt{0.07} \cdot |001\rangle + \sqrt{0.15} \cdot |010\rangle + \sqrt{0.05} \cdot |011\rangle +$$
$$+\sqrt{0.1} \cdot |100\rangle + \sqrt{0.3} \cdot |101\rangle + \sqrt{0.2} \cdot |110\rangle + \sqrt{0.1} \cdot |111\rangle$$



$$|\psi\rangle = \sqrt{0.03} \cdot |000\rangle + \sqrt{0.07} \cdot |001\rangle + \sqrt{0.15} \cdot |010\rangle + \sqrt{0.05} \cdot |011\rangle +$$

$$+\sqrt{0.1} \cdot |100\rangle + \sqrt{0.3} \cdot |101\rangle + \sqrt{0.2} \cdot |110\rangle + \sqrt{0.1} \cdot |111\rangle$$

Level 1

$$|\psi\rangle = \sqrt{0.03} \cdot |000\rangle + \sqrt{0.07} \cdot |001\rangle + \sqrt{0.15} \cdot |010\rangle + \sqrt{0.05} \cdot |011\rangle +$$
$$+\sqrt{0.1} \cdot |100\rangle + \sqrt{0.3} \cdot |101\rangle + \sqrt{0.2} \cdot |110\rangle + \sqrt{0.1} \cdot |111\rangle$$

The probability of observing $q_2 = 0$ is $\sqrt{0.3}$;

$$\sqrt{0.03} \cdot |000\rangle + \sqrt{0.07} \cdot |001\rangle + \sqrt{0.15} \cdot |010\rangle + \sqrt{0.05} \cdot |011\rangle$$

and probability of observing $q_2 = 1$ is $\sqrt{0.7}$.

We use a parameterized RY gate

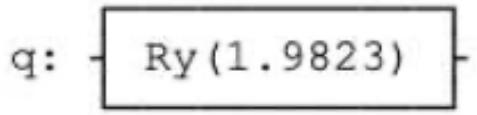
$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

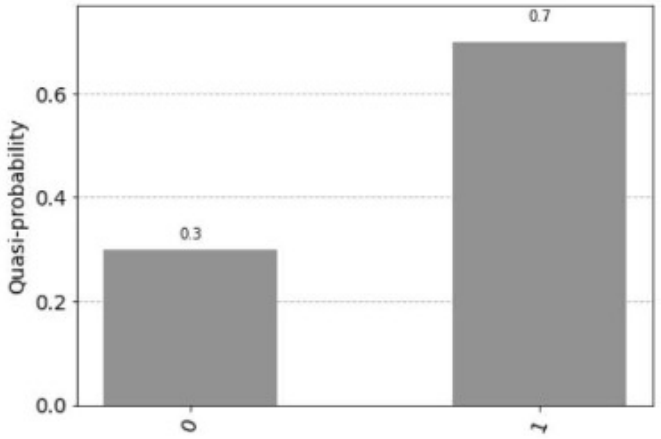
to performs a rotation of one qubit along the y-axis by the rotation angle θ (in radiants)

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{0.3};$$

$$\theta_0 = 1.98231 = 2 \cdot \arccos(\sqrt{0.3})$$

$$RY_{\theta_0}|0\rangle = \sqrt{0.3} \cdot |0\rangle + \sqrt{0.7} \cdot |1\rangle$$





Level 2

$$|\psi\rangle = \sqrt{0.03}\cdot|000\rangle + \sqrt{0.07}\cdot|001\rangle + \sqrt{0.15}\cdot|010\rangle + \sqrt{0.05}\cdot|011\rangle +$$
$$+\sqrt{0.1}\cdot|100\rangle + \sqrt{0.3}\cdot|101\rangle + \sqrt{0.2}\cdot|110\rangle + \sqrt{0.1}\cdot|111\rangle$$

The probability of observing $q_2q_1 = 00$ is $\sqrt{0.1}$

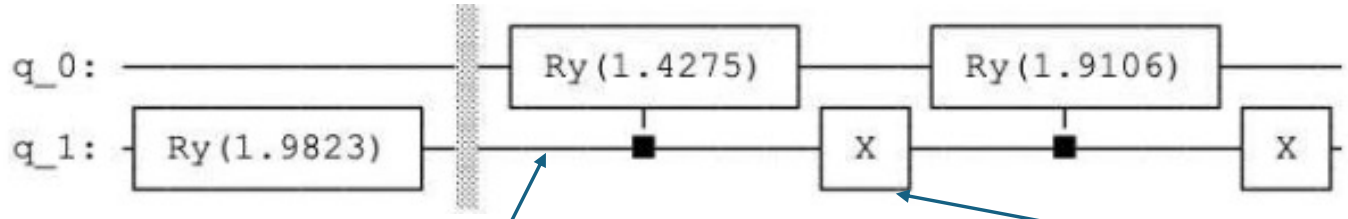
$$\theta_{00} = 1.91063 = 2\cdot\arccos\left(\frac{\sqrt{0.1}}{\sqrt{0.3}}\right). \qquad RY_{\theta_{00}}\sqrt{0.3}\cdot|0\rangle = \sqrt{0.1}\cdot|00\rangle + \sqrt{0.2}\cdot|01\rangle.$$

The probability of observing $q_2q_1 = 10$ is $\sqrt{0.4}$;

$$\theta_{10} = 1.42745 = 2\cdot\arccos\left(\frac{\sqrt{0.4}}{\sqrt{0.7}}\right). \qquad RY_{\theta_{10}}\sqrt{0.7}\cdot|1\rangle = \sqrt{0.4}\cdot|10\rangle + \sqrt{0.3}\cdot|11\rangle.$$

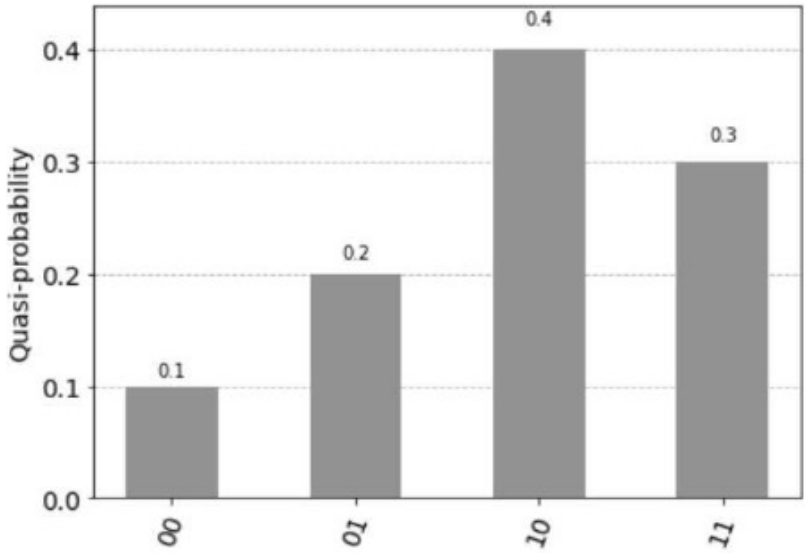
Level 2

$$RY_{\theta_{00}}\sqrt{0.3}\cdot|0\rangle+RY_{\theta_{10}}\sqrt{0.7}\cdot|1\rangle = \sqrt{0.1}\cdot|00\rangle+\sqrt{0.2}\cdot|01\rangle+\sqrt{0.4}\cdot|10\rangle+\sqrt{0.3}\cdot|11\rangle.$$



The probability of observing $q_2q_1 = 00$ is $\sqrt{0.1}$

The probability of observing $q_2q_1 = 10$ is $\sqrt{0.4}$;



Level 3

$$|\psi\rangle = \sqrt{0.03}\cdot|000\rangle + \sqrt{0.07}\cdot|001\rangle + \sqrt{0.15}\cdot|010\rangle + \sqrt{0.05}\cdot|011\rangle +$$
$$+\sqrt{0.1}\cdot|100\rangle + \sqrt{0.3}\cdot|101\rangle + \sqrt{0.2}\cdot|110\rangle + \sqrt{0.1}\cdot|111\rangle$$

$$\theta_{110} = 1.23096 = 2\cdot\arccos\left(\frac{\sqrt{0.2}}{\sqrt{0.3}}\right),$$

$$RY_{\theta_{110}}\sqrt{0.3}\cdot|11\rangle = \sqrt{0.2}\cdot|110\rangle + \sqrt{0.1}\cdot|111\rangle.$$

$$\theta_{100} = 2.0944 = 2\cdot\arccos\left(\frac{\sqrt{0.1}}{\sqrt{0.4}}\right),$$

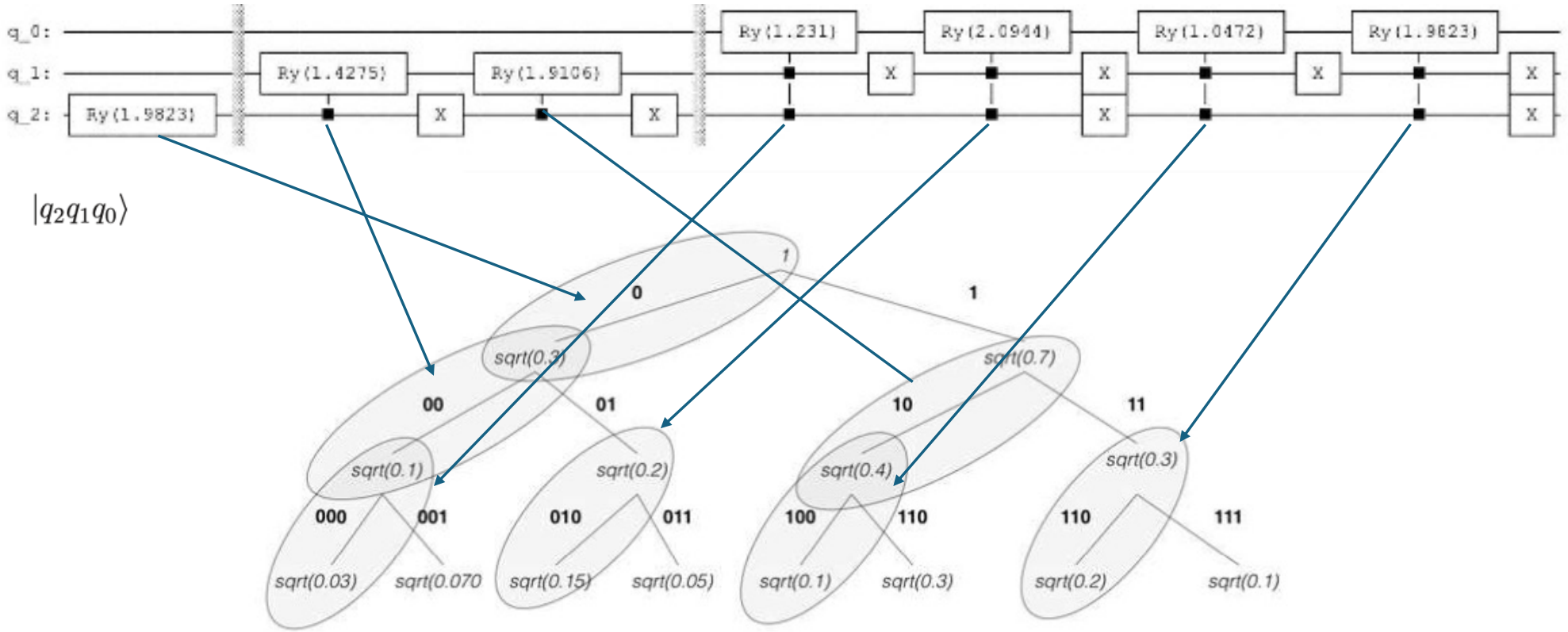
$$RY_{\theta_{100}}\sqrt{0.4}\cdot|10\rangle = \sqrt{0.1}\cdot|100\rangle + \sqrt{0.3}\cdot|101\rangle.$$

$$\theta_{010} = 1.0472 = 2\cdot\arccos\left(\frac{\sqrt{0.15}}{\sqrt{0.2}}\right),$$

$$RY_{\theta_{010}}\sqrt{0.2}\cdot|01\rangle = \sqrt{0.15}\cdot|010\rangle + \sqrt{0.05}\cdot|011\rangle.$$

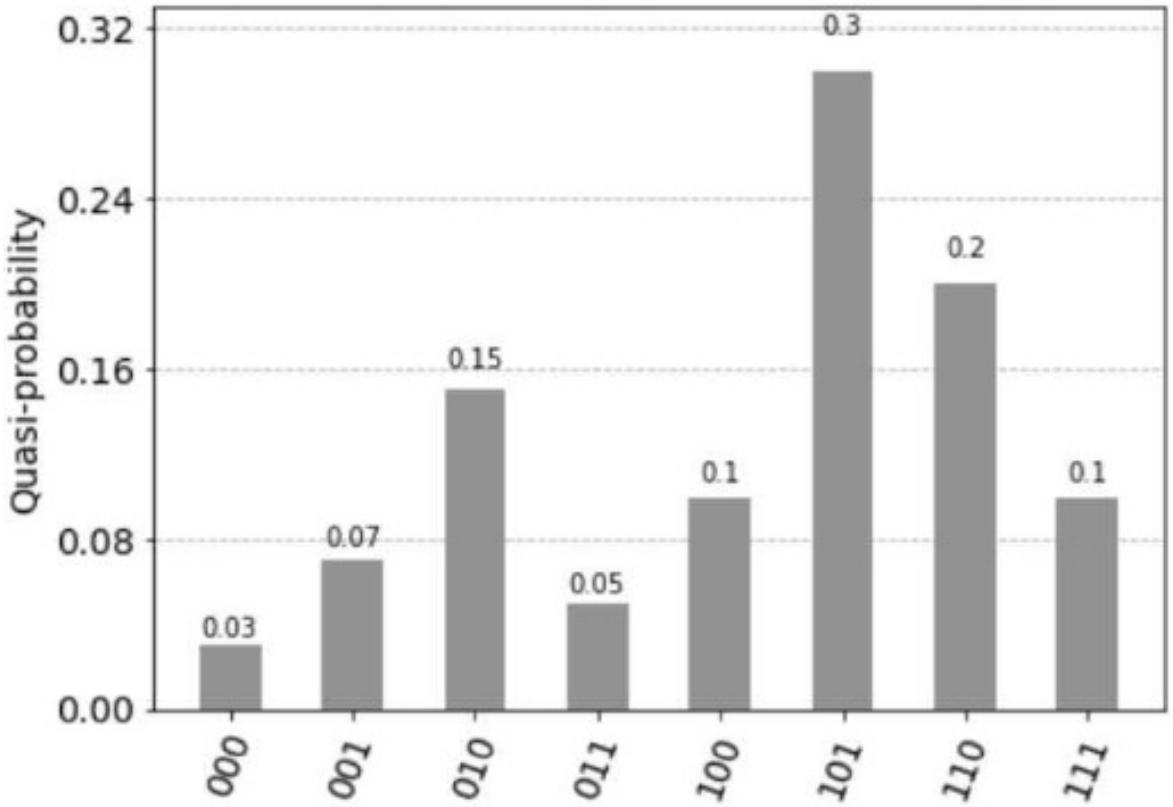
$$\theta_{000} = 1.98231 = 2\cdot\arccos\left(\frac{\sqrt{0.03}}{\sqrt{0.1}}\right),$$

$$RY_{\theta_{000}}\sqrt{0.1}\cdot|00\rangle = \sqrt{0.03}\cdot|000\rangle + \sqrt{0.07}\cdot|001\rangle.$$



$$|\psi\rangle = \sqrt{0.03} \cdot |000\rangle + \sqrt{0.07} \cdot |001\rangle + \sqrt{0.15} \cdot |010\rangle + \sqrt{0.05} \cdot |011\rangle +$$

$$+\sqrt{0.1} \cdot |100\rangle + \sqrt{0.3} \cdot |101\rangle + \sqrt{0.2} \cdot |110\rangle + \sqrt{0.1} \cdot |111\rangle$$



$$|\psi\rangle = \sqrt{0.03} \cdot |000\rangle + \sqrt{0.07} \cdot |001\rangle + \sqrt{0.15} \cdot |010\rangle + \sqrt{0.05} \cdot |011\rangle +$$

$$+\sqrt{0.1} \cdot |100\rangle + \sqrt{0.3} \cdot |101\rangle + \sqrt{0.2} \cdot |110\rangle + \sqrt{0.1} \cdot |111\rangle$$

Estimating Scalar Product

- **We cannot access the amplitudes** that represent vectors, but we estimate ⟨x|y⟩ using the swap test.
- Note that the quantum |x⟩ and |y⟩ are of **length one** in the l_2 norm
- With an additional auxiliary qubit |0⟩ generate the state

$$|0\rangle \otimes |x\rangle \otimes |y\rangle = |0, x, y\rangle$$

- Apply Hadamard gate on the control qubit |0⟩

$$(H \otimes I_m \otimes I_m) \cdot |0\rangle|x, y\rangle = \frac{1}{\sqrt{2}} \cdot (|0, x, y\rangle + |1, x, y\rangle)$$

Estimating Scalar Product

- Apply **controlled swap operator** on |x⟩ and |y⟩ states which swaps |x⟩ and |y⟩

$$|x\rangle|y\rangle \rightarrow |y\rangle|x\rangle \qquad \frac{1}{\sqrt{2}} \cdot (|0,x,y\rangle + |1,x,y\rangle) \rightarrow \frac{1}{\sqrt{2}} \cdot (|0,x,y\rangle + |1,y,x\rangle)$$

- We apply another **Hadamard gate** on the control qubit

$$(H \otimes I_m \otimes I_m) \cdot \left(\frac{1}{\sqrt{2}} \cdot (|0,x,y\rangle + |1,y,x\rangle) \right) =$$

$$\frac{1}{2} \cdot |0\rangle \cdot (|x,y\rangle + |y,x\rangle) + \frac{1}{2} \cdot |1\rangle \cdot (|x,y\rangle - |y,x\rangle)$$

The probability of measuring the control qubit in state $|0\rangle$ is given by

$$p(|0\rangle) = \left| \frac{1}{2} \cdot \langle 0|0\rangle \cdot (|x,y\rangle + |y,x\rangle) \quad + \frac{1}{2} \cdot \langle 0|1\rangle \cdot (|x,y\rangle - |y,x\rangle) \right|^2$$

$$p(|0\rangle) = \frac{1}{4} \cdot |(|x\rangle|y\rangle + |y\rangle|x\rangle)|^2$$

$$p(|0\rangle) = \frac{1}{4} \cdot (\langle y|y\rangle\langle x|x\rangle + \langle y|x\rangle\langle x|y\rangle + \langle x|y\rangle\langle y|x\rangle + \langle x|x\rangle\langle y|y\rangle)$$

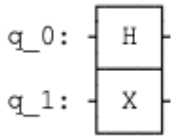
$$p(|0\rangle) = \frac{1}{4} \cdot (1 + \langle y|x\rangle\langle x|y\rangle + \langle x|y\rangle\langle y|x\rangle + 1)$$

$$p(|0\rangle) = \frac{1}{2} + \frac{1}{4} \cdot (\langle y|x\rangle\langle x|y\rangle + \langle x|y\rangle\langle y|x\rangle)$$

$$p(|0\rangle) = \frac{1}{2} + \frac{1}{2}|\langle x|y\rangle|^2$$

$$p(|1\rangle) = \frac{1}{2} - \frac{1}{2}|\langle x|y\rangle|^2 \qquad |\langle x|y\rangle| \approx \sqrt{2 \cdot p(|0\rangle) - 1} = \sqrt{1 - 2 \cdot p(|1\rangle)}.$$

$$\mathbf{x} = \begin{pmatrix} \sqrt{0.1} \\ \sqrt{0.2} \\ \sqrt{0.4} \\ \sqrt{0.3} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 0 \\ 0 \\ \sqrt{0.5} \\ \sqrt{0.5} \end{pmatrix}.$$



$$y = \frac{\sqrt{2}}{2}|10\rangle + \frac{\sqrt{2}}{2}|11\rangle$$

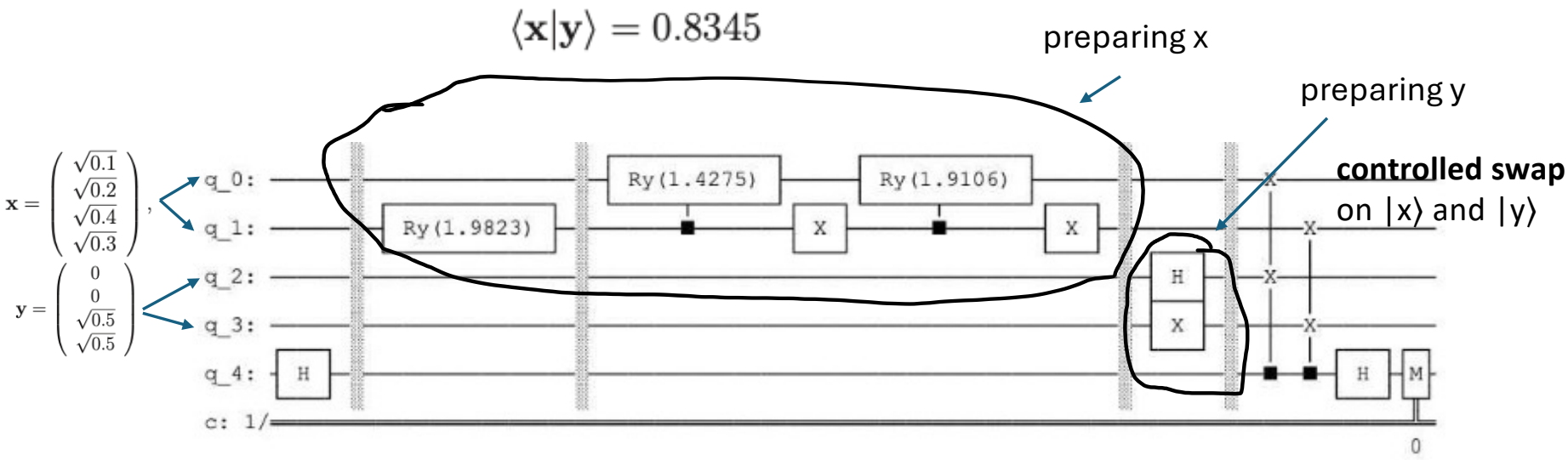
$$\text{Statevector} = \begin{bmatrix} 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

with

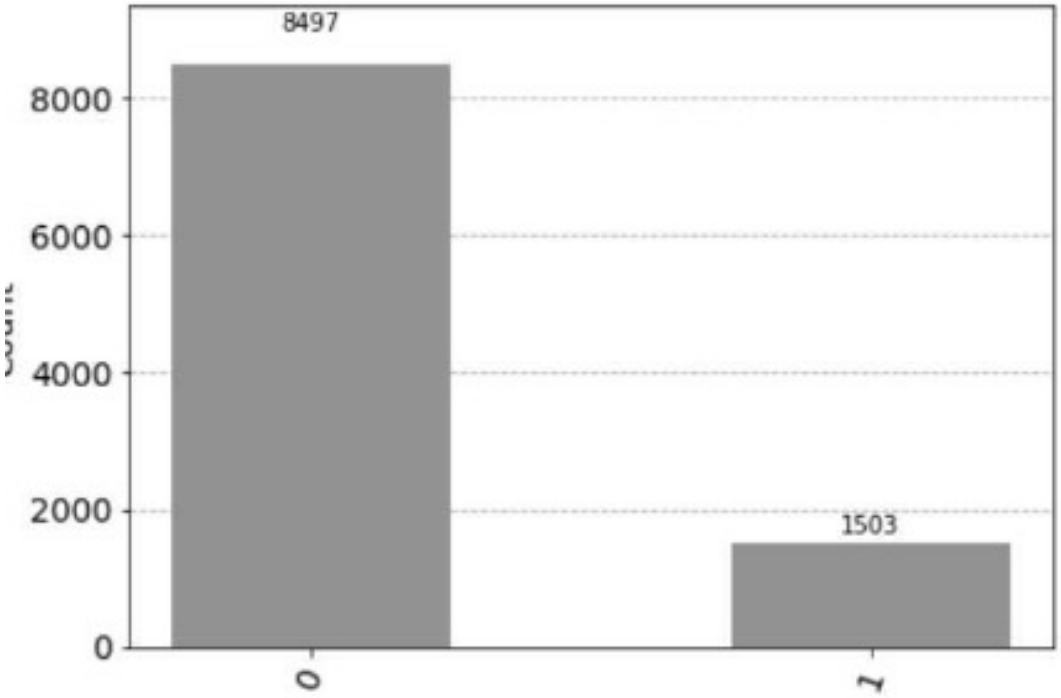
$$\langle \mathbf{x}|\mathbf{x}\rangle = \||\mathbf{x}\rangle\|_2 = 1, \quad \langle \mathbf{y}|\mathbf{y}\rangle = \||\mathbf{y}\rangle\|_2 = 1.$$

and

$$\langle \mathbf{x}|\mathbf{y}\rangle = 0.8345$$



$$|\langle \mathbf{x}|\mathbf{y}\rangle| \approx \sqrt{2 \cdot 0.8479 - 1} = 0.8345.$$



After **10000** shots we measure p(|0⟩) = 0.8479

$$|\langle \mathbf{x}|\mathbf{y}\rangle| \approx \sqrt{2 \cdot 0.8479 - 1} = 0.8345.$$

Constraint:
Note that the quantum |x⟩ and |y⟩ are of **length one** in the l_2 norm

We estimate the cos similarity!

$\langle x|y\rangle = \cos\theta$

$$\langle \mathbf{x}|\mathbf{y}\rangle = 0.8345$$

```
In[1]:= Clear["Global`*"]

        x = {√0.1, √0.2, √0.4, √0.3};

In[3]:= y = {0, 0, √0.5, √0.5};

In[4]:=
        x.y

Out[4]= 0.834512
```

qRAM

- Quantum memory is proposed as an analogue to classical computer memory, like the random-access memory (RAM)
- Random-access memory (RAM) is a form of computer memory that can be read and changed in any order, typically used to store working data
- In quantum machine learning domain, the usage of quantum random access memory (qRAM) is proposed to avoid (**?**) input destruction problem
  - However, qRAM **cannot avoid** the problems due to **entanglement…**
  - In literature is often used as a future (*metaphysical)* device
    - *(often not explained)*

qRAM

- A qRAM queries a register $|i\rangle$ and load the i^{th} binary patter into the second register

$$|i\rangle|0\rangle \rightarrow |i\rangle|x^i\rangle,$$

  with $|x^i\rangle$ being a basis state representing a binary vector

- Such an operation can be executed in parallel with

$$\frac{1}{\sqrt{m}}\sum_{i=1}^{m}|i\rangle|0\rangle \rightarrow \frac{1}{\sqrt{n}}\sum_{i=1}^{m}|i\rangle|x^i\rangle,$$

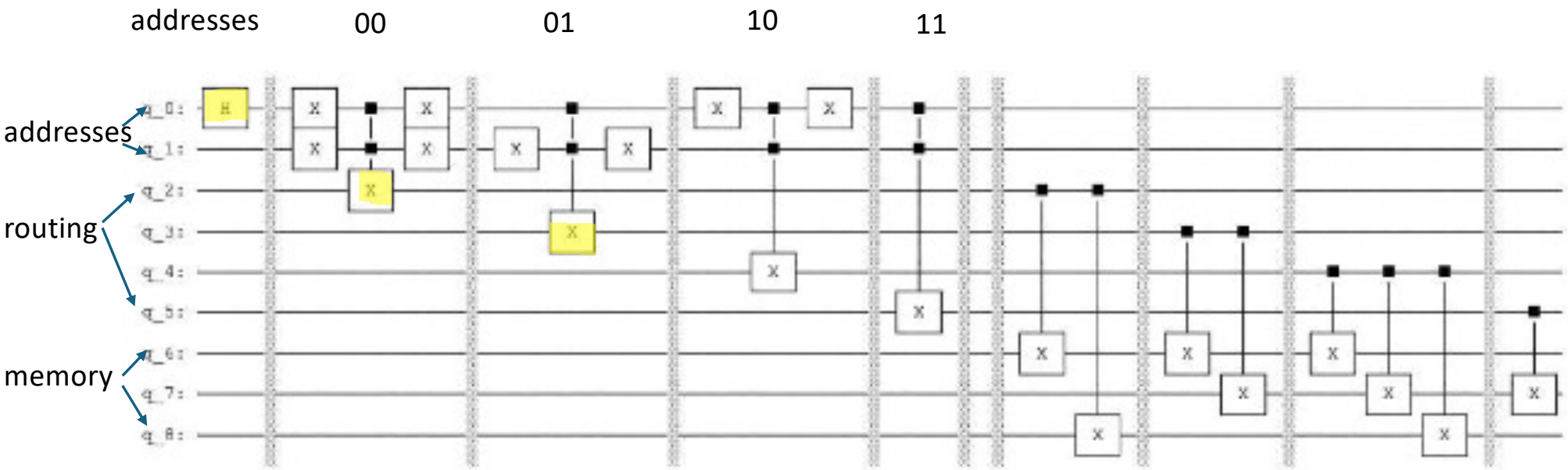
  with the time complexity ignoring the preparation cost of (due to the input problem) is *O(log(m))*

Bucket Brigade Architecture of qRAM

- The bucket brigade architecture of qRAM is inspired by the traditional RAM architecture and consists of 3 main parts:
  - Addressing qubits: Address for the memory cell we wish to read In order to read a memory cell the user has to encode the address of that memory cell on to the addressing qubits
  - The address can be in s**uperposition** so that **several patterns** can be read in one step
- Routing nodes: Determine the memory cell based upon the states of the addressing qubits
  - Once the addressing qubits have been set the states of the routing node qubits will be effected
  - Whatever qubit in the routing nodes is 1 will lead to the memory cell that we wish to read.
- Memory cells: Store and readout patterns.

qRAM for Binary Patterns

The addresses are ordered 00, 01, 10 and for the last pattern 11



The binary patterns can then be copied with CNOT gates, but they are **entangled** ☹

$|101\rangle_4, \quad |011\rangle_3, \quad |111\rangle_2, \quad |010\rangle_1$

entanglement between address and data

- To read classically *m* vectors the complexity is
  O(m), using qRAM **ignoring** the preparation cost it is $O(log_2(m))$
- *Why?*
  - Address register has $log_2(m)$ qubits

- Algorithmically, qRAM looks **very efficient……**
  - However, many quantum algorithms that assume qRAM are still debated in practicality
    - preparation coats are usually ignored, collapse during measurement is ignored

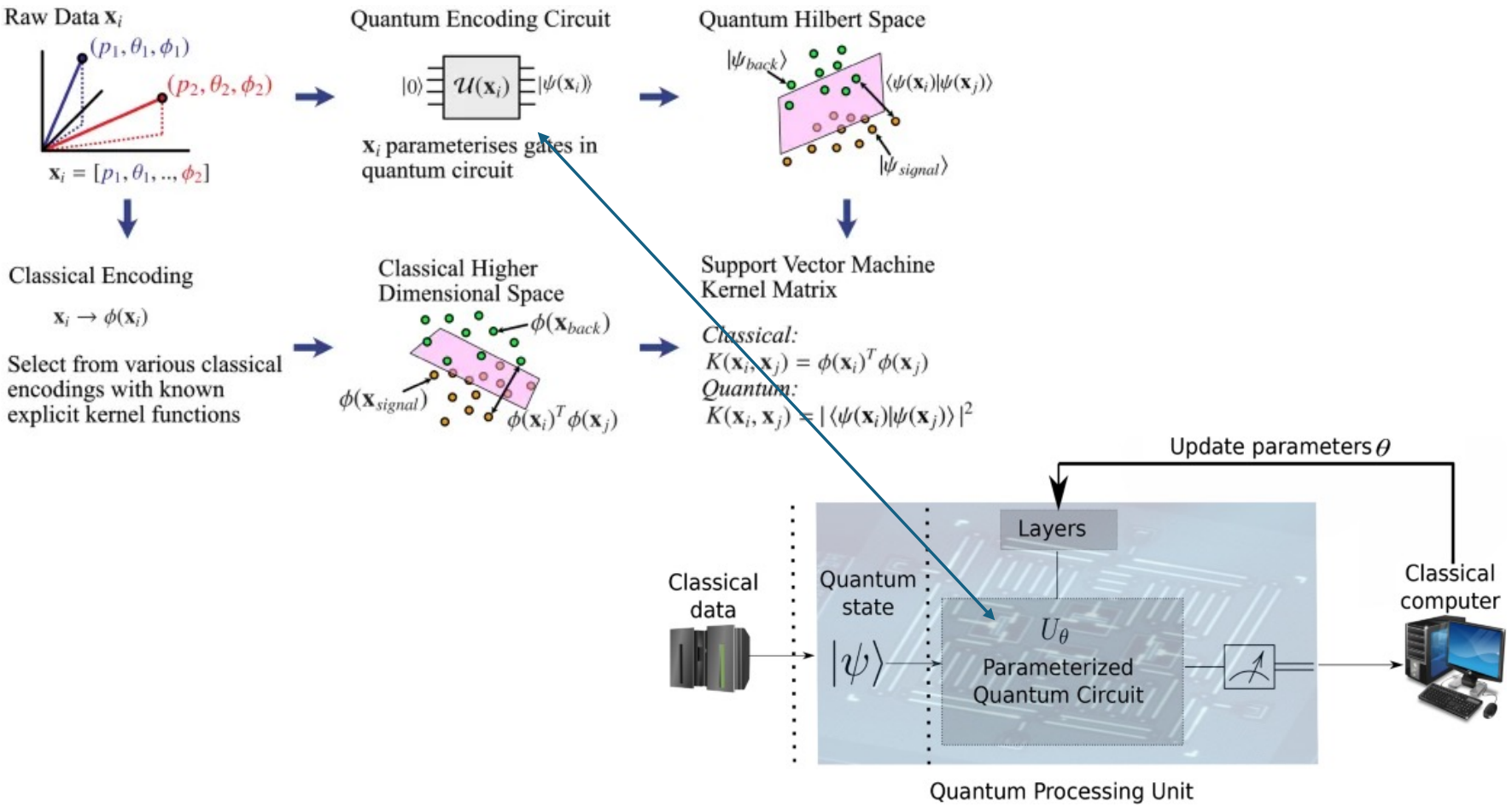
- No qRAM for amplitude coding:
  - An operation that would produce a copy of an arbitrary quantum state such as $|\psi\rangle$ is not possible; we cannot copy non-basis states because of the linearity of quantum mechanics

A Possible Solution?

- Most quantum machine learning algorithms suffer from the input destruction problem
- However, the input destruction problem has not yet been solved, and theoretical speed-ups are usually analyzed by ignoring the input problem, which is the **main bottleneck for data encoding**

- Overcome this problem by a quantum processor (quantum circuit) that calculates the expectation value over an ***extremely high dimensional space*** ⟶ ***Quantum Kernel***

Quantum Kernel and Variational Classifier



Quantum Kernels

- Quantum Kernels via Conventional Kernels
- $\Phi^T(x_i)\Phi(x)$ represents an inner product $\langle\Phi(x_i)|\Phi(x)\rangle$

$$k(\mathbf{x}, \mathbf{x}_i) = \langle \Phi(\mathbf{x}_i) | \Phi(\mathbf{x}) \rangle = \sum_{j=1}^{\infty} \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})$$

Conventional: represent the scalar operation by a kernel, for example the RBF kernel (*Φ in* infinite space)

$$k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2 \cdot \sigma}\right)$$

Quantum: map by $\Phi(x_i)|$ *and* $\Phi(x)$ *and* into **high dimensional** space (not infinite…), preform the scalar operation and estimate

Quantum Kernels

- A quantum computer can estimate a quantum kernel and the **estimate** can be used by a kernel method on a **classical computer**
- This is because the exponential quantum advantage in evaluating inner products allows to estimate the quantum kernel machines directly in the higher dimensional space

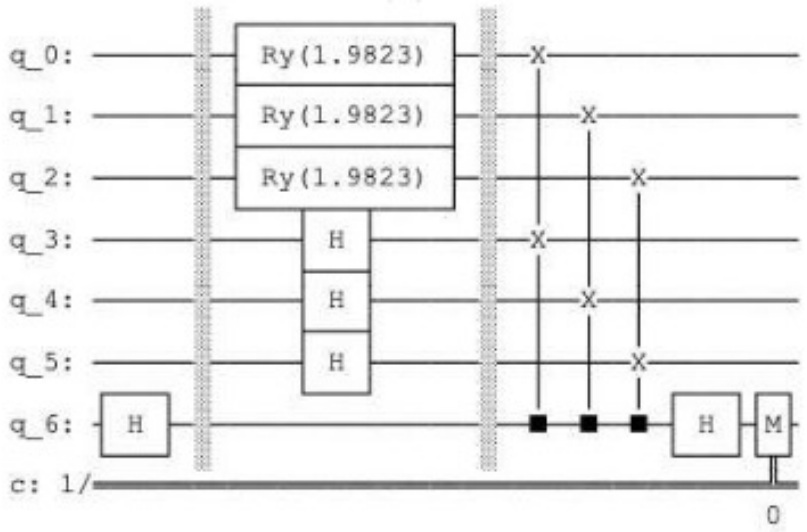
$$|\phi(\mathbf{x}\rangle) \to |\mathbf{x}\rangle \otimes \cdots \otimes |\mathbf{x}\rangle = |\mathbf{x}\rangle^{\otimes m}$$

- and define the homogenous polynomial kernel as

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle = \langle \mathbf{x} | \mathbf{y} \rangle \otimes \cdots \otimes \langle \mathbf{x} | \mathbf{y} \rangle = (\mathbf{x}^T \cdot \mathbf{y})^m.$$

$$\mathbf{x} = \begin{pmatrix} \sqrt{0.3} \\ \sqrt{0.7} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} \sqrt{0.5} \\ \sqrt{0.5} \end{pmatrix}.$$

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle = \langle \mathbf{x} | \mathbf{y} \rangle \otimes \langle \mathbf{x} | \mathbf{y} \rangle \otimes \langle \mathbf{x} | \mathbf{y} \rangle = (\mathbf{x}^T \cdot \mathbf{y})^3$$



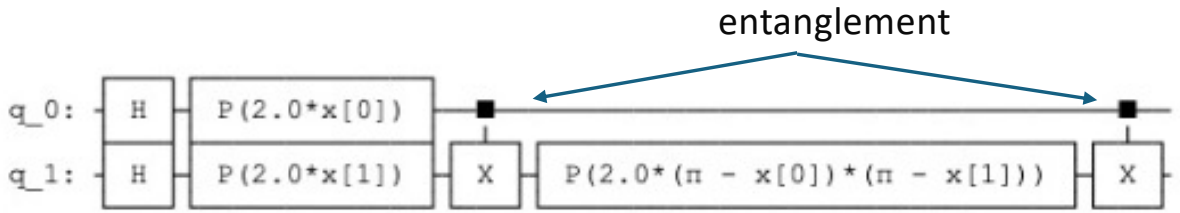
Using the swap test, after 10000 shots we measure $p(|0\rangle) = 0.9395$ with

$$k(\mathbf{x}, \mathbf{y}) = |(\mathbf{x}^T \cdot \mathbf{y})^3| \approx \sqrt{2 \cdot 0.9395 - 1} = 0.93755.$$

Quantum Kernels and the Inversion Test

- Quantum feature maps encodes classical data into quantum data via a *parametrized quantum circuit*
- The feature vector defines by *m* parameters of the parametrized quantum circuit $U_{\varphi(x)}$ with the dimension of $\varphi(x)$ being 2^m
- Parameterized quantum circuits based on **superposition** and **entanglement** are hard to simulate classically and could lead to an advantage over classical machine learning approaches

$$|\phi(\mathbf{x})\rangle = U_{\phi(\mathbf{x})}|0\rangle^{\otimes m}$$



- The *ZZFeatureMap* with *2* features, *x[0], x[1],* dimension of $\varphi(x)$ being 2^4 and depth 1

- The inversion test is based on the idea of estimating the fidelity (similarity) between two states
- For an input state $|0\rangle^{\otimes m}$, if we map it by parametrized quantum circuit $U_{\varphi(x)}$ with parameters that are defined by x
- then un-compute it by $U^{\dagger}{}_{\varphi(x)}$, the inverse of the parametrized quantum circuit $U_{\varphi(x)}$, then the probability of measuring the state $|0\rangle^{\otimes m}$ is **one**

- Quantum circuit U by **x** ($U_{\varphi(x)}$) , the inverse quantum circuit U† by y ($U^{\dagger}_{\varphi(y)}$)

$$U^{\dagger}_{\phi(\mathbf{y})} U_{\phi(\mathbf{x})} |0^{\otimes m}\rangle$$

- If ***x*** and ***y*** are similar, the probability of measuring $|0\rangle^{\otimes m}$ for the input $|0\rangle^{\otimes m}$ should be near 1
- If ***x*** and ***y*** differ a lot, this probability is smaller

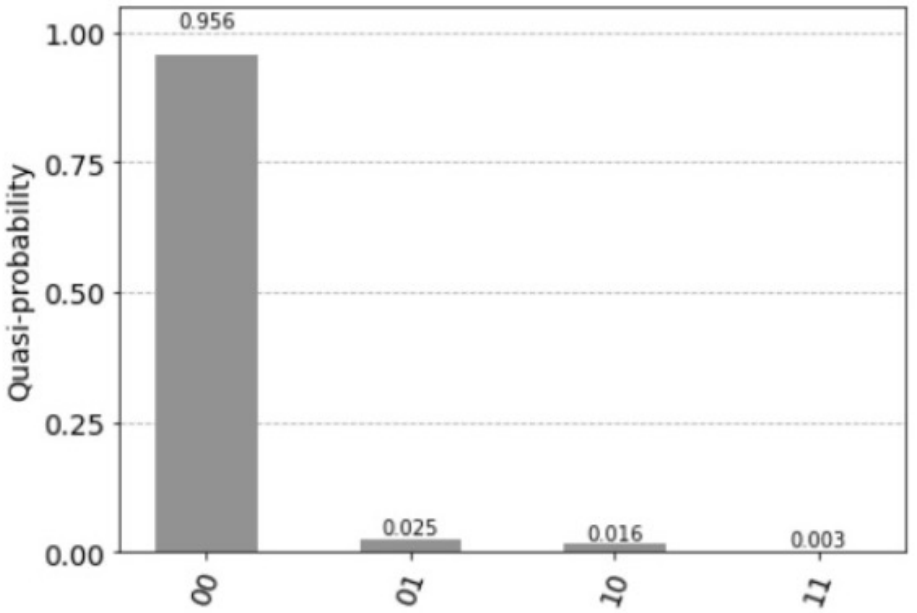
- The quantum kernel is represented as:

$$k(\mathbf{x}, \mathbf{y}) = |\langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle|^2 = |\langle 0^{\otimes m} | U^{\dagger}_{\phi(\mathbf{y})} | U_{\phi(\mathbf{x})} | 0^{\otimes m} \rangle|^2$$

- We measure the final state several times and record the number of $|0^{\otimes m}\rangle$ and estimate the value $k(\mathbf{x}, \mathbf{y})$

- After 10000 shots we measure p(|00⟩) = 0.956 with

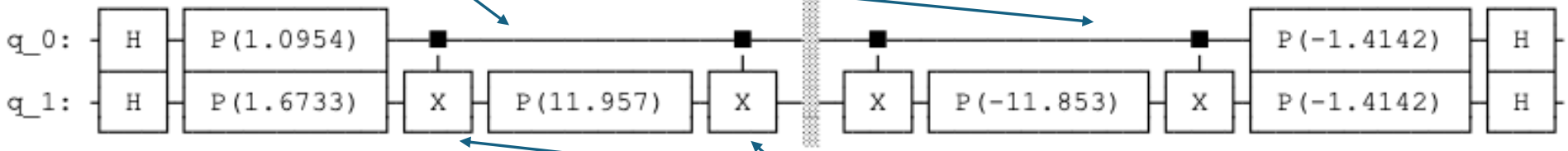
$$\mathbf{x} = \begin{pmatrix} \sqrt{0.3} \\ \sqrt{0.7} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} \sqrt{0.5} \\ \sqrt{0.5} \end{pmatrix}$$




```
data = [np.sqrt(0.3), np.sqrt(0.7)]
feature_map = ZZFeatureMap(2, reps=1)
feature_map = feature_map.assign_parameters(data) # <== here

data2 = [np.sqrt(0.5), np.sqrt(0.5)]
feature_map2 = ZZFeatureMap(2, reps=1).inverse()
feature_map2 = feature_map2.assign_parameters(data2) # <== here
```

$$k(\mathbf{x}, \mathbf{y}) = |\langle 0^{\otimes m} | U^{\dagger}_{\phi(\mathbf{y})} | U_{\phi(\mathbf{x})} | 0^{\otimes m} \rangle|^2 = 0.956.$$



entanglement

dimension of *φ(x)* being 2^4

LETTER

https://doi.org/10.1038/s41586-019-0980-2

Supervised learning with quantum-enhanced feature spaces

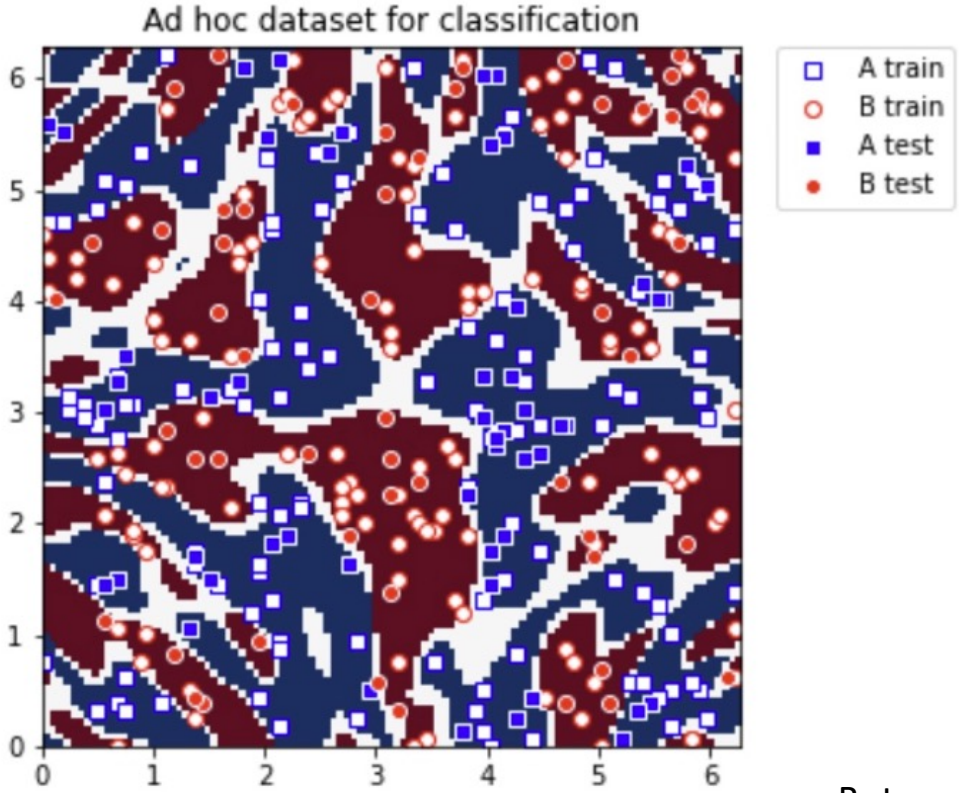
Vojtěch Havlíček[1,2], Antonio D. Córcoles[1]*, Kristan Temme[1]*, Aram W. Harrow[3], Abhinav Kandala[1], Jerry M. Chow[1] & Jay M. Gambetta[1]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. [2]Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, UK. [3]Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA, USA. *e-mail: adcorcol@us.ibm.com; kptemme@ibm.com

Quantum Support Vector Machine

- The quantum kernel is plugged into classical kernel methods like Support Vector Machine (Kernel Machine)
- Only the Gram matrix (kernel matrix) is determined by a quantum computer (or a simulation), the other parts of the computation are performed on a conventional computer
- Either one passes a function of the quantum kernel to a conventional algorithm, or one precomputes the training and testing kernel matrices
  - Often the actual internal representations of the quantum kernel are hidden from the outside.

- An 2 dimensional ad hoc toy data set of two dimensions with two classes is generated using *ZZFeatureMap.* Quantum kernel with SVM achieve **great results** on this toy data set

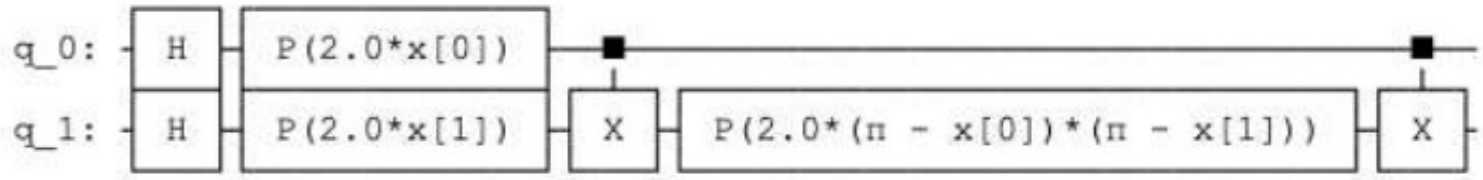


But…

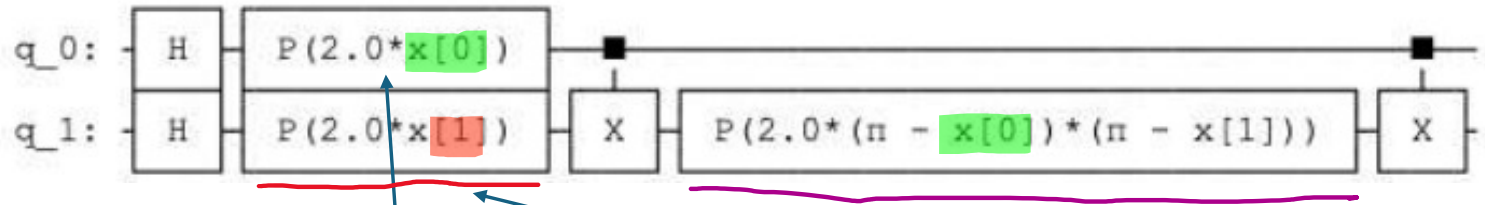
Overfitting

- Quantum kernels are intractable for higher dimension on a classical due to exponential growth of parameters
- They give an advantage for artificial generated problems *Havlicek et al. (2019)*!
- They can achieve lower **training** error on real data (-> **Overfitting**)
- However, this improvement leads to poor generalization on the test set and it seems that *classical models* **can outperform** quantum models
  - Why?

- We will analyze the quantum kernel based on ZZFeatureMaps in two dimensions
- First a Hadamard gate H_2 is applied to two qubits represented as



$$H_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$$



then the tensor product of two phase gates

$$P_1 = \begin{pmatrix} 1 & 0 \\ 0 & e^{2i \cdot x_1} \end{pmatrix}, \quad P_2 = \begin{pmatrix} 1 & 0 \\ 0 & e^{2i \cdot x_2} \end{pmatrix}.$$

on the two qubits is represented by the matrix

$$P_2 \otimes P_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{2i a1} & 0 & 0 \\ 0 & 0 & e^{2i a2} & 0 \\ 0 & 0 & 0 & e^{2i a1 + 2i a2} \end{pmatrix}.$$

The *ZZFeatureMap* F with 2 features and depth 1 corresponds to the product

$$F = CX \cdot (P_3 \otimes I) \cdot CX \cdot (P_2 \otimes P_1) \cdot H_2$$

and is represented by the matrix

$$F = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a1}} & -\frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a1}} & \frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a1}} & -\frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a1}} \\ \frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a2}} & \frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a2}} & -\frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a2}} & -\frac{1}{2}e^{2i(\pi-\mathrm{a1})(\pi-\mathrm{a2})+2i\mathrm{a2}} \\ \frac{1}{2}e^{2i\mathrm{a1}+2i\mathrm{a2}} & -\frac{1}{2}e^{2i\mathrm{a1}+2i\mathrm{a2}} & -\frac{1}{2}e^{2i\mathrm{a1}+2i\mathrm{a2}} & \frac{1}{2}e^{2i\mathrm{a1}+2i\mathrm{a2}} \end{pmatrix}$$

and the conjugate transpose matrix $B = F^*$ is given by

$$B = \begin{pmatrix} \frac{1}{2} & \frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b1}} & \frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b2}} & \frac{1}{2}e^{-2i\mathrm{b1}-2i\mathrm{b2}} \\ \frac{1}{2} & -\frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b1}} & \frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b2}} & -\frac{1}{2}e^{-2i\mathrm{b1}-2i\mathrm{b2}} \\ \frac{1}{2} & \frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b1}} & -\frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b2}} & -\frac{1}{2}e^{-2i\mathrm{b1}-2i\mathrm{b2}} \\ \frac{1}{2} & -\frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b1}} & -\frac{1}{2}e^{-2i(\pi-\mathrm{b1})(\pi-\mathrm{b2})-2i\mathrm{b2}} & \frac{1}{2}e^{-2i\mathrm{b1}-2i\mathrm{b2}} \end{pmatrix}.$$

and the mapping $U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle$ is described by

$$U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle = B \cdot F \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} =$$

$$\frac{1}{4}\begin{pmatrix} e^{2i(\pi x_1)(\pi x_2)+2ix_1-2i(\pi y_1)(\pi y_2)-2iy_1} + e^{2i(\pi x_1)(\pi x_2)+2ix_2-2i(\pi y_1)(\pi y_2)-2iy_2} + e^{2ix_1+2ix_2-2iy_1-2iy_2} + 1 \\ -e^{2i(\pi x_1)(\pi x_2)+2ix_1-2i(\pi y_1)(\pi y_2)-2iy_1} - e^{2i(\pi x_1)(\pi x_2)+2ix_2-2i(\pi y_1)(\pi y_2)-2iy_2} + e^{2ix_1+2ix_2-2iy_1-2iy_2} + 1 \\ -e^{2i(\pi x_1)(\pi x_2)+2ix_1-2i(\pi y_1)(\pi y_2)-2iy_1} + e^{2i(\pi x_1)(\pi x_2)+2ix_2-2i(\pi y_1)(\pi y_2)-2iy_2} - e^{2ix_1+2ix_2-2iy_1-2iy_2} + 1 \\ e^{2i(\pi x_1)(\pi x_2)+2ix_1-2i(\pi y_1)(\pi y_2)-2iy_1} - e^{2i(\pi x_1)(\pi x_2)+2ix_2-2i(\pi y_1)(\pi y_2)-2iy_2} - e^{2ix_1+2ix_2-2iy_1-2iy_2} + 1 \end{pmatrix}$$

with

$$\langle 00|U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle = \frac{1}{4}\Big(e^{2i(\pi x_1)(\pi x_2)+2ix_1-2i(\pi y_1)(\pi y_2)-2iy_1} + e^{2i(\pi x_1)(\pi x_2)+2ix_2-2i(\pi y_1)(\pi y_2)-2iy_2} + e^{2ix_1+2ix_2-2iy_1-2iy_2} + 1\Big).$$

$$\langle 00|U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle = \frac{1}{4}\cdot\Big(1+e^{2i(x_1+x_2-y_1-y_2)}+$$
$$2\cdot e^{i(2\pi(-x_1-x_2+y_1+y_2)+2x_1x_2+x_1+x_2-2y_1y_2-y_1-y_2)}\cdot\cos(x_1-x_2-y_1+y_2)\Big).$$

Notice that $\langle 00|U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle$ is not a kernel, since it is not symmetric. Only after the measurement using the Born rule a symmetric kernel function is present with

$$k(\mathbf{x},\mathbf{y}) = |\langle 00|U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle|^2 = \frac{1}{8}\Big(2+\cos\Big(2(x_1+x_2-y_1-y_2)\Big)+\cos\Big(2(x_1-x_2-y_1+y_2)\Big)+$$
$$2\Big((\cos\Big(2(x_1-y_1)\Big)+\cos\Big(2(x_2-y_2)\Big)\Big)\cdot$$
$$\cos\Big(2\pi(-x_1-x_2+y_1+y_2)+2x_1x_2-2y_1y_2\Big)\Big). \qquad)$$

Quantum kernels are intractable for higher dimension on a **classical computer** due to exponential growth

ZZFeatureMap with 3 features (mapping 3 dim in 2^3 entangled space):

$$p = \langle 000|U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|000\rangle = \frac{1}{8}e^{-2i(2\pi x_1+2\pi x_2+2\pi x_3+y_1(y_2+y_3+1)+y_2y_3+y_2+y_3)}\cdot$$

$$\Big(e^{2i(2\pi x_1+x_1+x_2+2\pi x_2+x_3+2\pi x_3+y_1y_2+y_1y_3+y_2y_3)}+$$

$$e^{2i(2\pi x_1+2\pi x_2+2\pi x_3+y_1(y_2+y_3+1)+y_2y_3+y_2+y_3)}+$$

$$e^{2i((x_1+\pi+1)x_2+(x_1+\pi+1)x_3+2\pi y_1+y_1+y_2y_3+\pi y_2+\pi y_3))}+$$

$$e^{2i(x_1(x_2+\pi+1)+(x_2+\pi+1)x_3+y_1y_3+\pi y_1+y_2+2\pi y_2+\pi y_3)}+$$

$$e^{2i(x_1(x_3+\pi+1)+x_2(x_3+\pi+1)+y_1y_2+\pi y_1+\pi y_2+y_3+2\pi y_3)}+$$

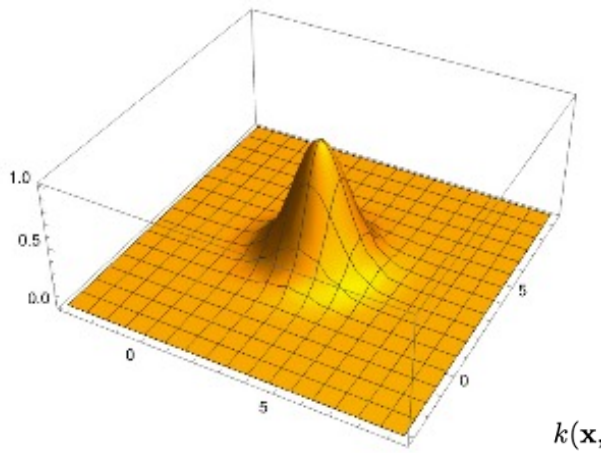
$$e^{2i(x_3(x_1+x_2+1)+\pi x_1+\pi x_2+y_1(y_2+\pi+1)+y_2+\pi y_2+2\pi y_3)}+$$

$$e^{2i(x_2(x_1+x_3+1)+\pi x_1+\pi x_3+y_1(y_3+\pi+1)+2\pi y_2+y_3+\pi y_3)}+$$

$$e^{2i(x_1(x_2+x_3+1)+\pi x_2+\pi x_3+2\pi y_1+y_2(y_3+\pi+1)+y_3+\pi y_3)}\Big)\,.$$

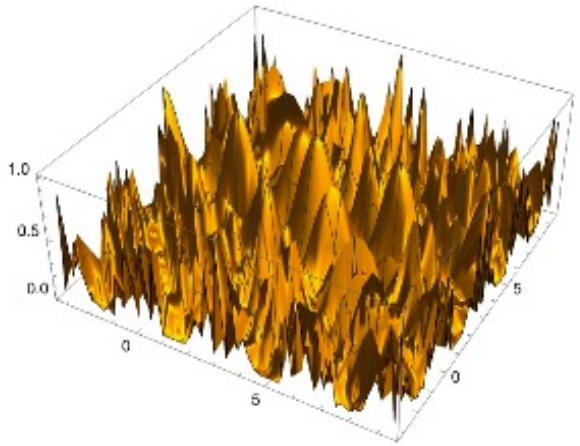
The quantum kernel

$$k(\mathbf{x},\mathbf{y}) = |\langle 000|U^{\dagger}_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|000\rangle|^2 = p\cdot p^*$$



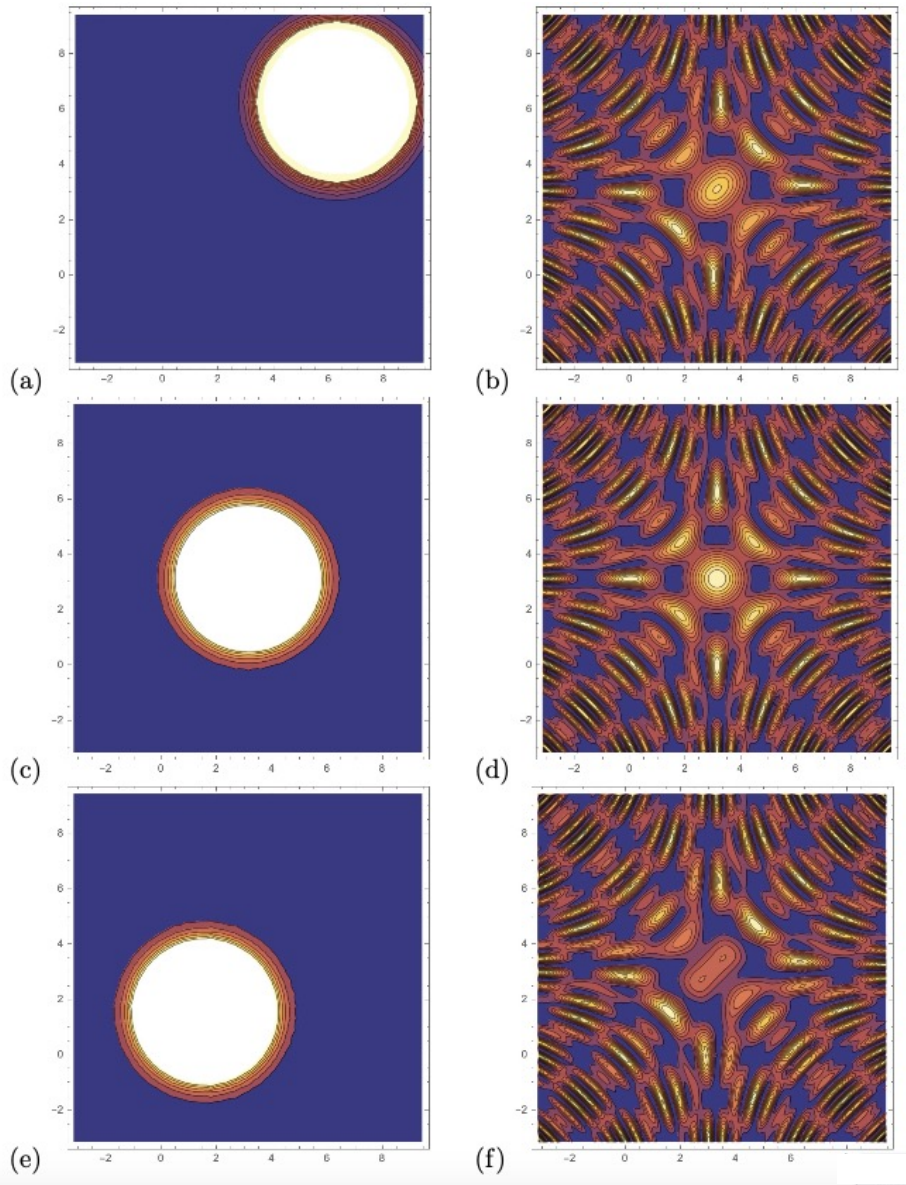
(a)

$$k(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2}\right)$$



(b)

$$k(\mathbf{x},\mathbf{y}) = |\langle 00|U^\dagger_{\phi(\mathbf{y})}|U_{\phi(\mathbf{x})}|00\rangle|^2 = \frac{1}{8}\left(2+\cos\left(2(x_1+x_2-y_1-y_2)\right)+\cos\left(2(x_1-x_2-y_1+y_2)\right)+\right.$$
$$2\left((\cos\left(2(x_1-y_1)\right)+\cos\left(2(x_2-y_2)\right)\right)\cdot$$
$$\left.\cos\left(2\pi(-x_1-x_2+y_1+y_2)+2x_1x_2-2y_1y_2\right)\right).$$



Problems

- The RBF kernel position in the space is defined by its center, this is not the case with the quantum kernel whose center remains fixed and instead its *wave distribution* changes
- This is why quantum kernels based on *ZZFeatureMaps* fail to generalize real world normal distributed data and are successful for artificial generated problems where the distribution of the data is *periodic*
- Using quantum kernels on real data requires new alternative machine learning techniques are required that do not minimize the error in the **Euclidean space** (like SVM or Perceptron) but in the **frequency space**

nature communications

Article

https://doi.org/10.1038/s41467-023-36159-y

Quantum machine learning beyond kernel methods

Received: 1 April 2022

Accepted: 18 January 2023

Published online: 31 January 2023

Check for updates

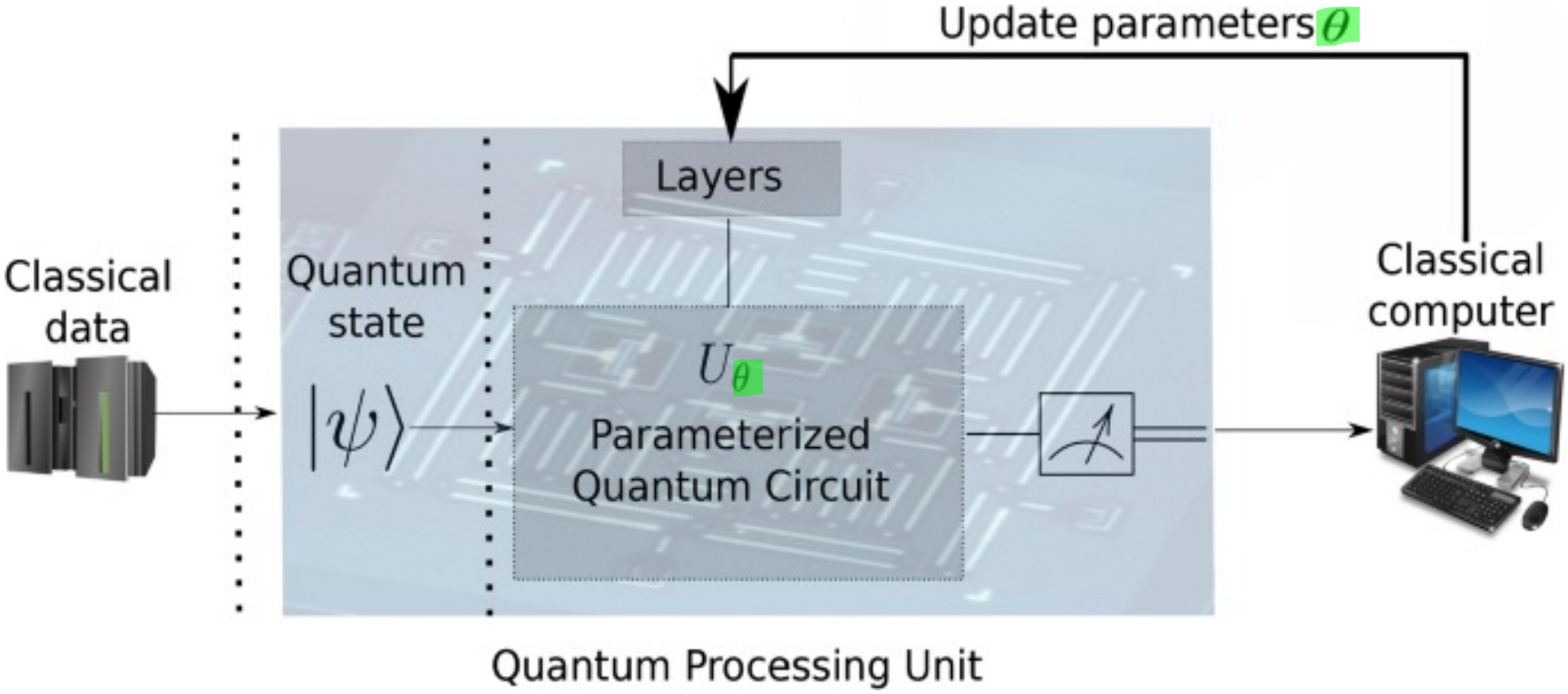
Sofiene Jerbi[1], Lukas J. Fiderer[1], Hendrik Poulsen Nautrup[1], Jonas M. Kübler[2], Hans J. Briegel[1] & Vedran Dunjko[3]

Machine learning algorithms based on parametrized quantum circuits are prime candidates for near-term applications on noisy quantum computers. In this direction, various types of quantum machine learning models have been introduced and studied extensively. Yet, our understanding of how these models compare, both mutually and to classical models, remains limited. In this work, we identify a constructive framework that captures all standard models based on parametrized quantum circuits: that of linear quantum models. In particular, we show using tools from quantum information theory how data re-uploading circuits, an apparent outlier of this framework, can be efficiently mapped into the simpler picture of linear models in quantum Hilbert spaces. Furthermore, we analyze the experimentally-relevant resource

Are Variational Quantum Classifiers the Solution?

- “Machine learning algorithms based on **parametrized quantum circuits** are prime candidates for near-term applications on noisy quantum computers. ”
- “Based on recent results from classical machine learning, we prove that linear quantum models must utilize exponentially more qubits than data re-uploading models in order to solve certain learning tasks, while **kernel methods** additionally **require exponentially more data points**.“

Variational Classifier



Hybrid Variational Algorithms

- Variational approaches are characterized using a **classical optimization algorithm** to iteratively update a parameterized quantum trial solution also called *ansatz* (from German *Ansatz* = approach)
- The **parameterized quantum** trial solution is defined by a parametrized quantum circuit like for example the *ZZFeatureMap*
- Simultaneous Perturbation Stochastic Approximation (SPSA)
  - The optimizer performs stochastic gradient approximation, which requires only **two** measurements of the loss function.

The variational quantum classifier for a binary classification problem, with input data vectors $\mathbf{x}_k$ of dimension m and binary output labels t_k with a training set

$$D = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots ,, (\mathbf{x}_N, t_N)\}, \quad t_k \in \{0, 1\}$$

For each input data vector $\mathbf{x}_k$ a quantum feature maps encodes classical data into quantum data via a parametrized quantum circuit [95]. circuit $U_{\phi(\mathbf{x}_k)}$ with m parameters

$$U_{\phi(\mathbf{x})_k}|0\rangle^{\otimes m}$$

Additionally we will use a variational quantum circuits that represents the free parameter $\mathbf{w}$ that will adapt during training

$$|\psi(\mathbf{x}_k, \mathbf{w})\rangle = U_{W(\mathbf{w})} \cdot U_{\phi(\mathbf{x})_k}|0\rangle^{\otimes m}.$$

Example

- We use the parameterized *qiskit* quantum circuit over two qubits $U_{\phi(x)}$ = *ZZFeatureMap* with repetition two where the parameter are defined by the data, two-dimensional vector ***x***
- The variational quantum circuits that represents the free parameter *w* that will adapt during training is the qiskit quantum circuit $U_{W(w)}$ = *TwoLocal*
- The *TwoLocal* circuit is a parameterized circuit consisting of alternating rotation layers and entanglement layers
  - The rotation layers are single qubit gates applied on all qubits
  - The entanglement layer uses two-qubit gates to entangle the qubits according to the definition
  - Two repetitions resulting in ***12*** free parameters

$U_{\varphi(\boldsymbol{x})}$ = *ZZFeatureMap,* two-dimensional vector ***x*** *representing the data*

x = [0.1, 0.1]

```
feature_map = ZZFeatureMap(feature_dimension=2, reps=2)
feature_map = feature_map.assign_parameters(x)
```

$U_{W(\boldsymbol{w})}$ = *TwoLocal,* ***12*** free parameters representing weights

```
weights =np.array([3.28559355, 5.48514978, 5.13099949,
                   0.88372228, 4.08885928, 2.45568528,
                   4.92364593, 5.59032015, 3.66837805,
                   4.84632313, 3.60713748, 2.43546])
two_local = TwoLocal(2, ['ry', 'rz'],'cz', reps=2)
two_local = two_local.assign_parameters(weights)
```

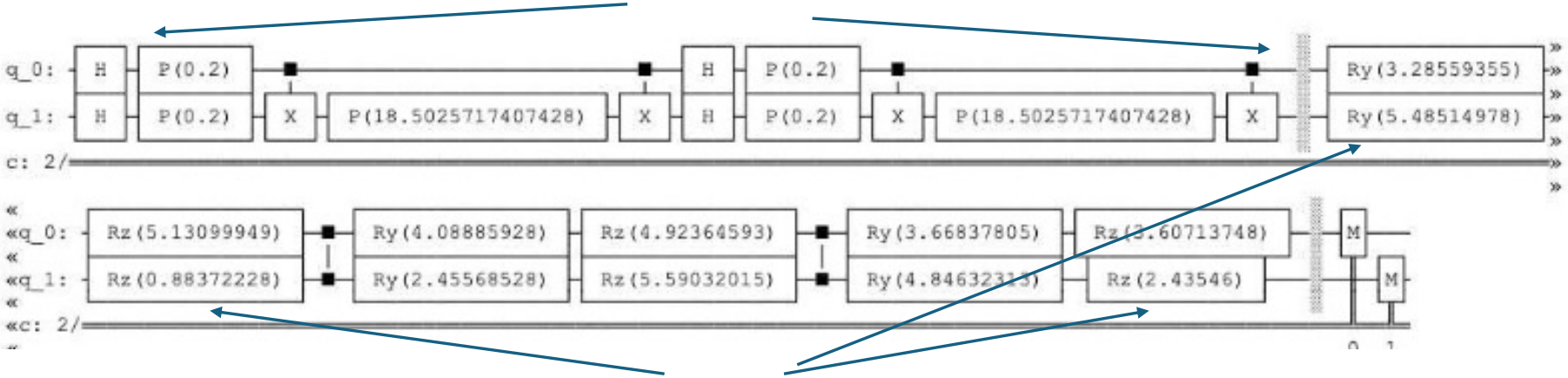
Circuit Definition

```
qc = QuantumCircuit(2,2)
qc.compose(feature_map, inplace=True)
qc.barrier()
qc.compose(two_local, inplace=True)
qc.barrier()
qc.measure(0,0)
qc.measure(1,1)
qc.decompose().draw(fold=130)
```

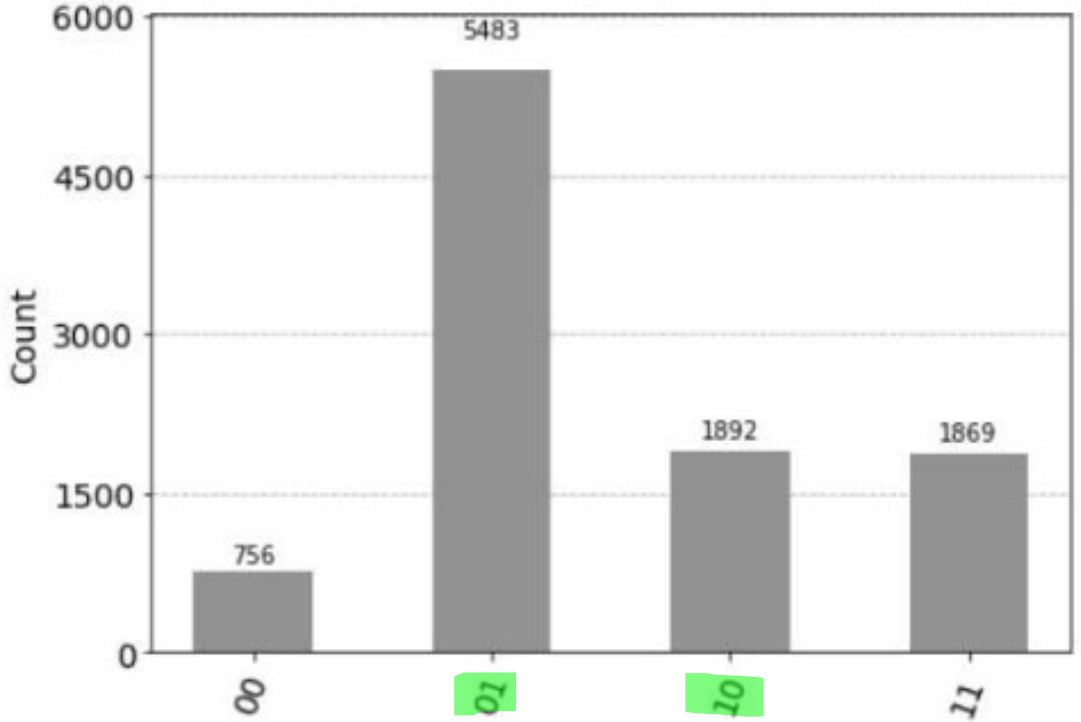
Quantum Circuit

- We define the binary output from the binary string by a parity function
  - A parity function is a Boolean function whose value is one if and only if the input vector has an odd number of ones

$U_{\varphi(x)}$ = *ZZFeatureMap,* two-dimensional vector ***x*** *representing the data*



$U_{W(w)}$ = *TwoLocal,* ***12*** free parameters representing weights



We perform *10000* shots. The string 01 appears *5483* times and the string 10 appears 1*892*. We define the binary output from the binary string by a parity function, $p(1) = o_k = (5483 + 1892)/1000 = 0.7375$

In Boolean algebra, a **parity function** is a Boolean function whose value is one if and only if the input vector has an odd number of ones.

Cross Entropy

$$o_k = p(C_1|\mathbf{x}_k) = \sigma\left(\mathbf{w}^T \cdot \mathbf{x}_k\right) = \sigma(net_k),$$

we can simply write

$$p(\mathbf{t}|\mathbf{x}_1, \cdots, \mathbf{x}_N, \mathbf{w}) = \prod_{k=1}^{N} o_k^{t_k} \cdot (1 - o_k)^{1-t_k}.$$

Having the likelihood, we can estimate the weights such that the likelihood is maximized. This is equivalent to minimizing the negative log likelihood, yielding the loss function

$$L(\mathbf{w}) = -\log(p(\mathbf{t}|\mathbf{w})) = -\sum_{k=1}^{N} \left(t_k \log o_k + (1 - t_k) \log(1 - o_k)\right).$$

The resulting loss function is exactly the cross entropy between targets and output

$$H(t, p) = -\sum_{k=1}^{N} \left(t_k \cdot \log(p(C_1|\mathbf{x}_k)) + \neg t_k \cdot \log(p(C_2|\mathbf{x}_k))\right).$$

Cross Entropy

Assuming that the training set consists of N observations

$$X = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k, \cdots, \mathbf{x}_N)^T$$

and respective target values represented as vectors of dimension K (since t is used as an index, we will use y_{kt} to indicate the specific target)

$$Y = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k, \cdots, \mathbf{y}_N)^T.$$

During training, each variational quantum circuit is trained individually with its target value y_{kt}

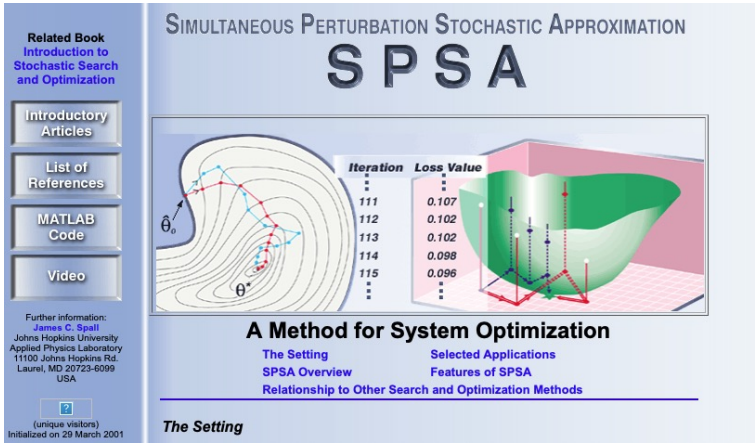
$$y_{kt} \in \{0, 1\}, \quad \sum_{t=1}^{K} y_{kt} = 1$$

The resulting cross entropy loss function is given by

$$L(\mathbf{w}) = -\sum_{k=1}^{N} \sum_{t=1}^{K} y_{kt} \cdot \log o_{kt},$$

SPSA Optimizer

- Simultaneous Perturbation Stochastic Approximation (SPSA) optimizer performs stochastic gradient approximation
  - https : //www.jhuapl.edu/SPSA/



- It requires **only two** measurements of the loss function, regardless of the dimension of the optimization problem

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\mathbf{w})$$

with the gradient operator is

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \cdots, \frac{\partial}{\partial w_D}\right]^T$$

$$g(\mathbf{w}) = \nabla L(\mathbf{w}) = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \cdots, \frac{\partial L}{\partial w_D}\right]^T.$$

SPSA use the iterative process with τ indicating the iteration starting at $\tau = 1$ with

$$\mathbf{w}_{\tau+1} = \mathbf{w}_\tau - \eta_\tau \cdot \hat{g}_\tau(\mathbf{w}_\tau)$$

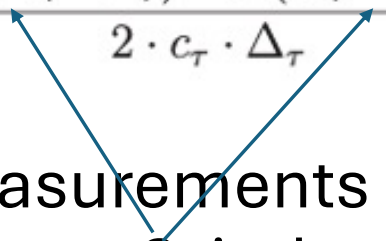
with η_τ being the learning rate that converges to zero and $\hat{g}_\tau(\mathbf{w}_\tau)$ estimation gradient $g\tau(\mathbf{w}_\tau)$ with

$$\hat{g}_\tau = \frac{L(\mathbf{w}_\tau + c_\tau \cdot \Delta_\tau) - L(\mathbf{w}_\tau - c_\tau \cdot \Delta_\tau)}{2 \cdot c_\tau \cdot \Delta_\tau}.$$

Δ_τ is a random perturbation vector and c_τ is a small positive number that decreases with τ.

SPSA Optimizer

$$\hat{g}_\tau = \frac{L(\mathbf{w}_\tau + c_\tau \cdot \Delta_\tau) - L(\mathbf{w}_\tau - c_\tau \cdot \Delta_\tau)}{2 \cdot c_\tau \cdot \Delta_\tau}$$

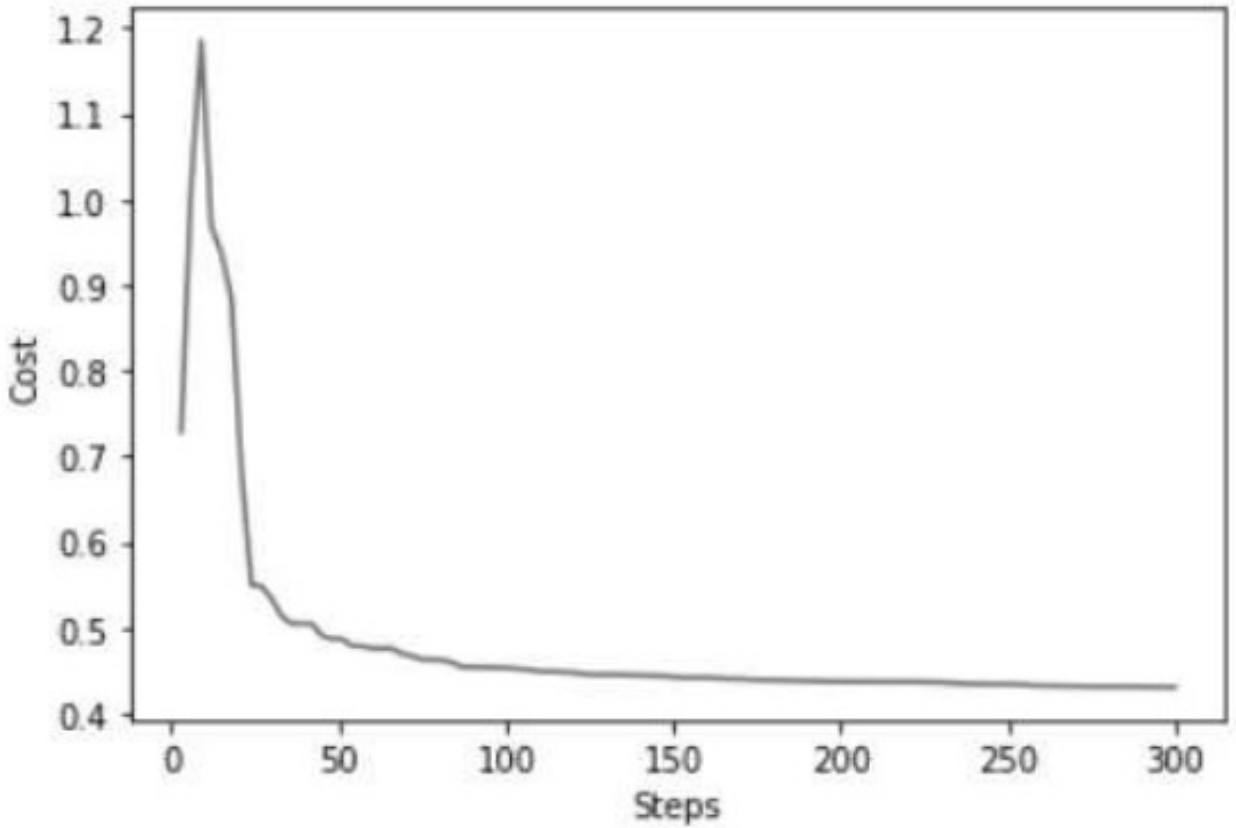


- The number of loss function measurements needed in the SPSA method for each is always ***2***, independent of the dimension
- SPSA with the random search direction does not follow the gradient path but *approximates* it
- When using a variational classifier on a quantum computer SPSA is therefore the most recommended choice since it can be used in the presence of noise

Qiskit Variational Quantum Classifier

only qiskit < v2

- Qiskit implements the variational quantum classifier (VQC) that can be embedded in classical machine learning tasks.
- In this simple qiskit example we create 10 two-dimensional training data points and 5 testing data points for **two classes** each
- We use the same classification circuit as before
- We one hot encode our labels, as required by the algorithm using cross entropy
- Then, we set up our classical optimizer and the VQC algorithm using the callback function



We plot the cost function with respect to optimization step, we can see it starts to converge to a minimum

 # score == accuracy

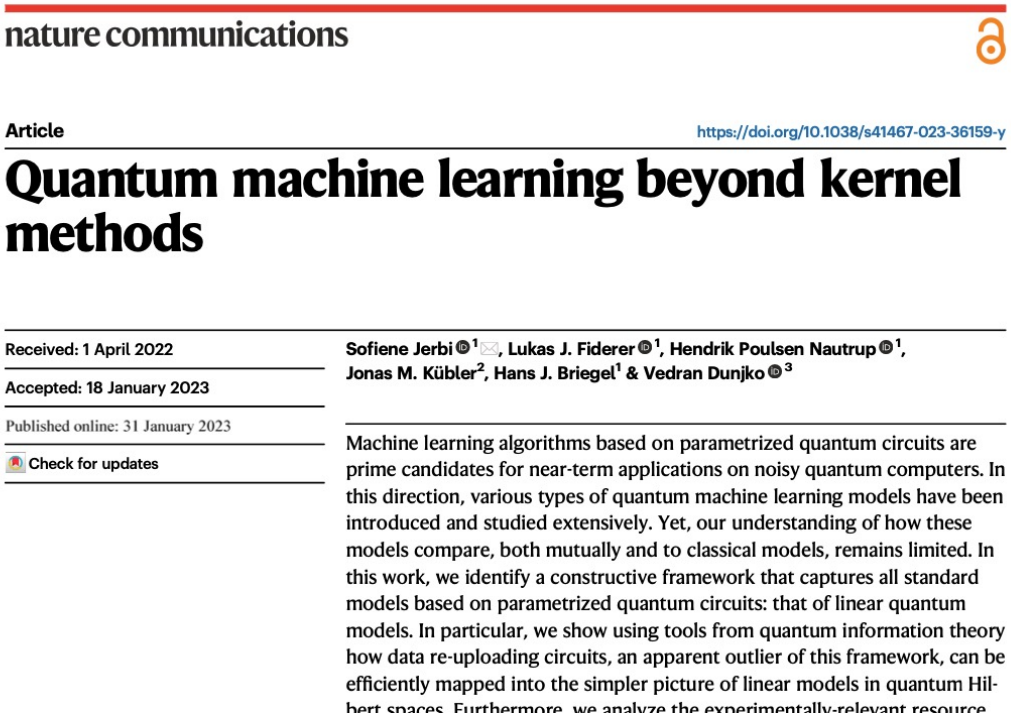
vqc.score(TEST_DATA, test_labels_oh)

- We test our trained VQC classifier Score indicates the mean accuracy, it determines for the test set true values, in our case the score is *0.9*

Conclusion

- Most quantum machine learning algorithms suffer from the input destruction problem
  - The efficient preparation of data is possible in part for sparse data (next lecture)
- The input destruction problem has not yet been solved, and theoretical speed-ups are usually analyzed by ignoring the input problem, which is the main bottleneck for data encoding.
- Other constraints:
  - Normalized representation of vectors (Cosine similarity)
  - Quantum kernels with a “periodic” receptive fields
- **Dilemma**: should we ignore or marginalize those constraints or not?
  - Optimist: In the feature they will be solved
  - Pessimist: Till now, no theoretical advantage over classical algorithms on real data

- Variational Quantum Classifier a solution to the **Dilemma?**



nature communications

Article

https://doi.org/10.1038/s41467-023-36159-y

Quantum machine learning beyond kernel methods

Received: 1 April 2022

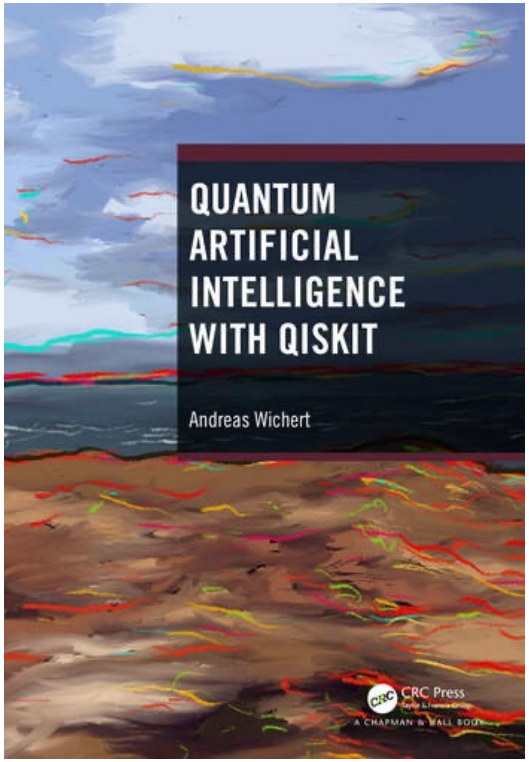
Accepted: 18 January 2023

Published online: 31 January 2023

Check for updates

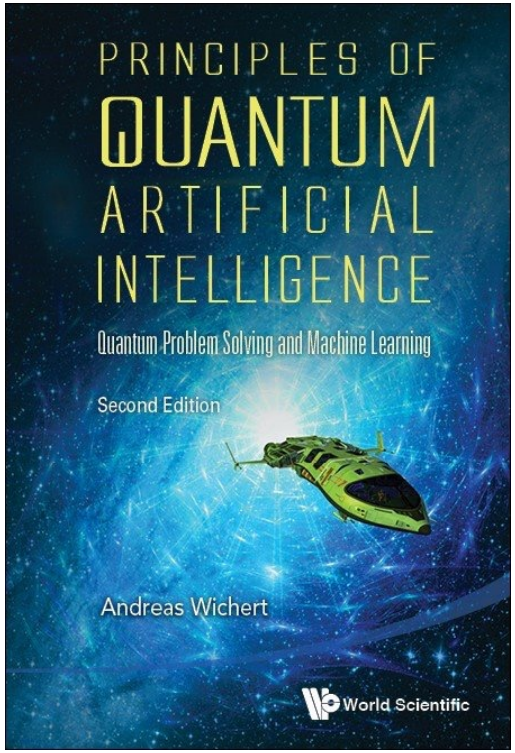
Sofiene Jerbi[1], Lukas J. Fiderer[1], Hendrik Poulsen Nautrup[1], Jonas M. Kübler[2], Hans J. Briegel[1] & Vedran Dunjko[3]

Machine learning algorithms based on parametrized quantum circuits are prime candidates for near-term applications on noisy quantum computers. In this direction, various types of quantum machine learning models have been introduced and studied extensively. Yet, our understanding of how these models compare, both mutually and to classical models, remains limited. In this work, we identify a constructive framework that captures all standard models based on parametrized quantum circuits: that of linear quantum models. In particular, we show using tools from quantum information theory how data re-uploading circuits, an apparent outlier of this framework, can be efficiently mapped into the simpler picture of linear models in quantum Hilbert spaces. Furthermore, we analyze the experimentally-relevant resource



- Chapter 16
- Chapter 17
- Chapter 18
- Chapter 23

Quantum Artificial Intelligence with Qiskit, A. Wichert, Chapman and Hall/CRC, 2024



- Chapter 14.3
- Chapter 14.4
- Chapter 14.5

Principles of Quantum Artificial Intelligence: Quantum Problem Solving and Machine Learning, 2nd Edition, A. Wichert, World Scientific, 2020