Quantum Neural Networks

Antoine (Jack) Jacquier

(Imperial College London)

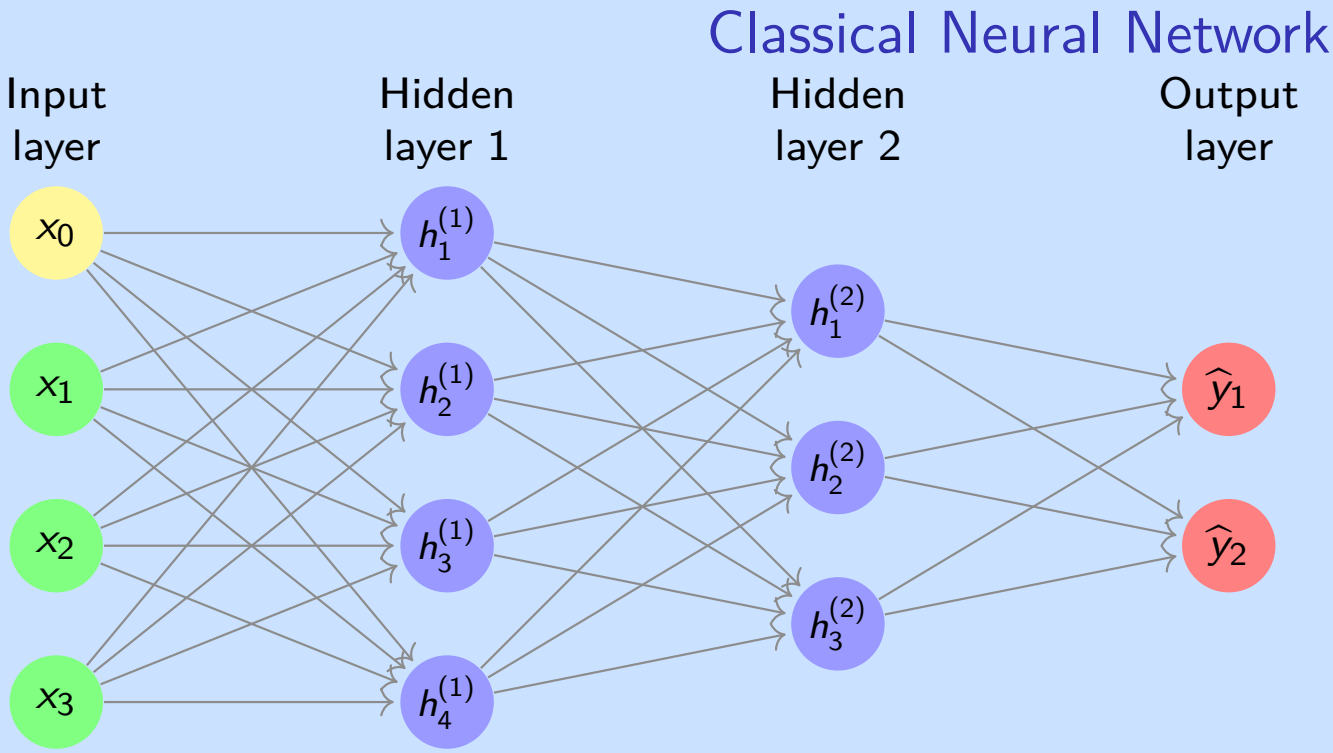
January 16, 2025

Overview

Quantum Neural Networks

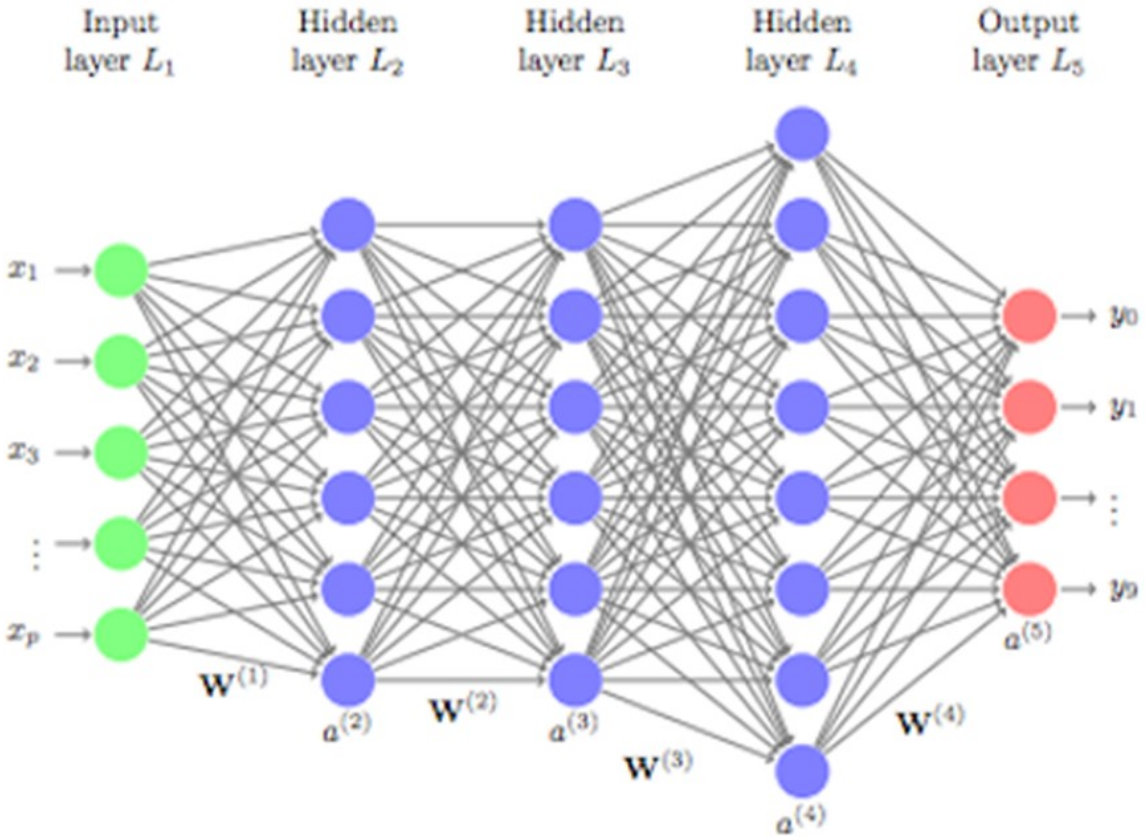
Classical Neural Networks

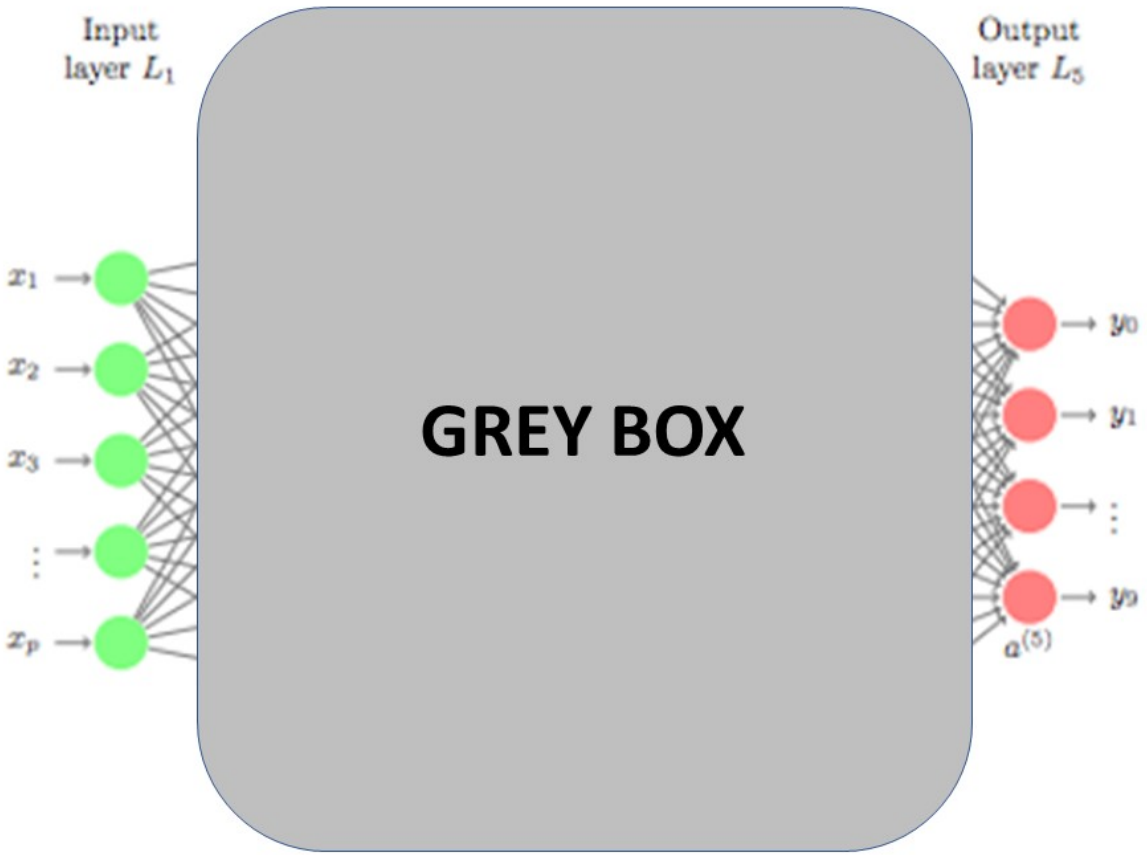
- First logical neuron (**perceptron**) was developed by McCulloch (neuroscientist) and Pitts (logician) in 1943.
- Concept of a NN/**machine learning** appears to have first been proposed by Turing (*Intelligent Machinery*, 1948).
- 1958: Rosenblatt implemented the perceptron (with 3 layers).
- Ivakhnenko and Lapa suggested the first **deep learning** algorithm (with arbitrarily many layers) arose in 1965.
- 1982: Hopfield networks.
- 1980s-1990s: Hinton with backpropagation, deep learning
- 2014: Generative Adversarial Networks
- Since 2017: Large Language models
- They are now ubiquitous, in particular for:
  - Function approximation / time series prediction;
  - Classification problems (pattern recognition, decision making);
  - Data processing (filtering).

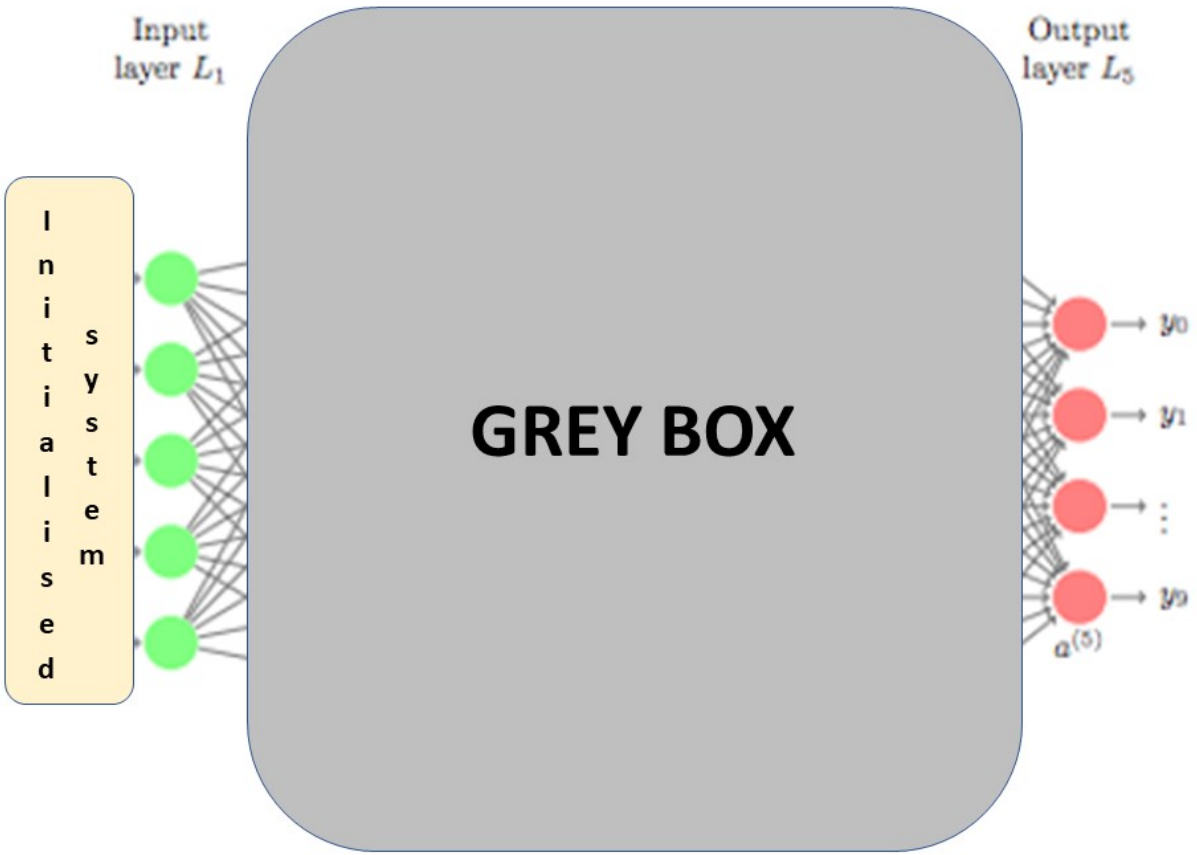


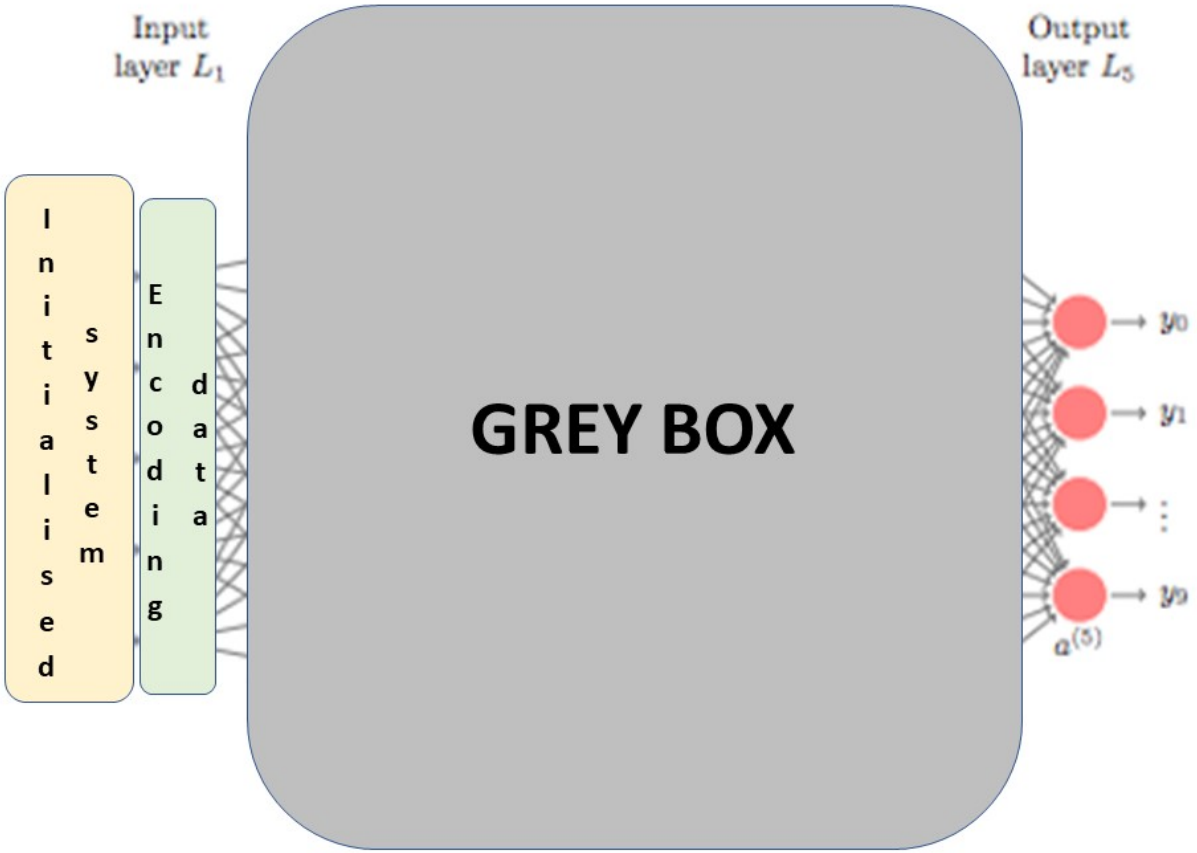
ANN with one input layer, 2 hidden layers and one output layer.

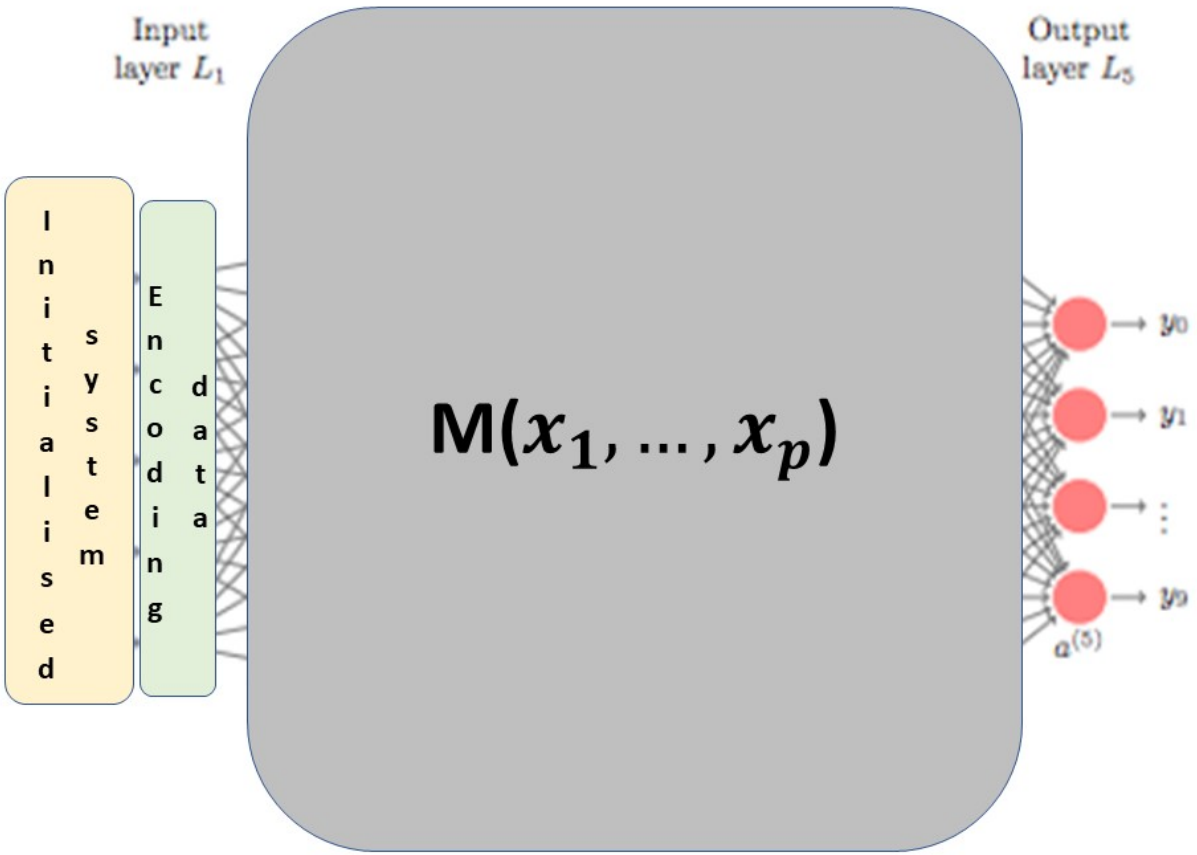
- $\widehat{\mathrm{y}} = L_3 \circ \sigma \circ L_2 \circ \sigma \circ L_1(\mathrm{x})$, where each linear function L_k is of the form $L_k(\mathrm{z}) = \mathrm{A}_k \mathrm{z} + \mathrm{b}_k$, for matrices A_k and vectors b_k with appropriate dimensions, and σ is a (non-linear) activation function (example $\sigma(x) = \max(0, x)$ for ReLU).
- Universal approximation theorem.
- Generalisations (Recurrent NN, Convolutional NN, LSTM, ...)

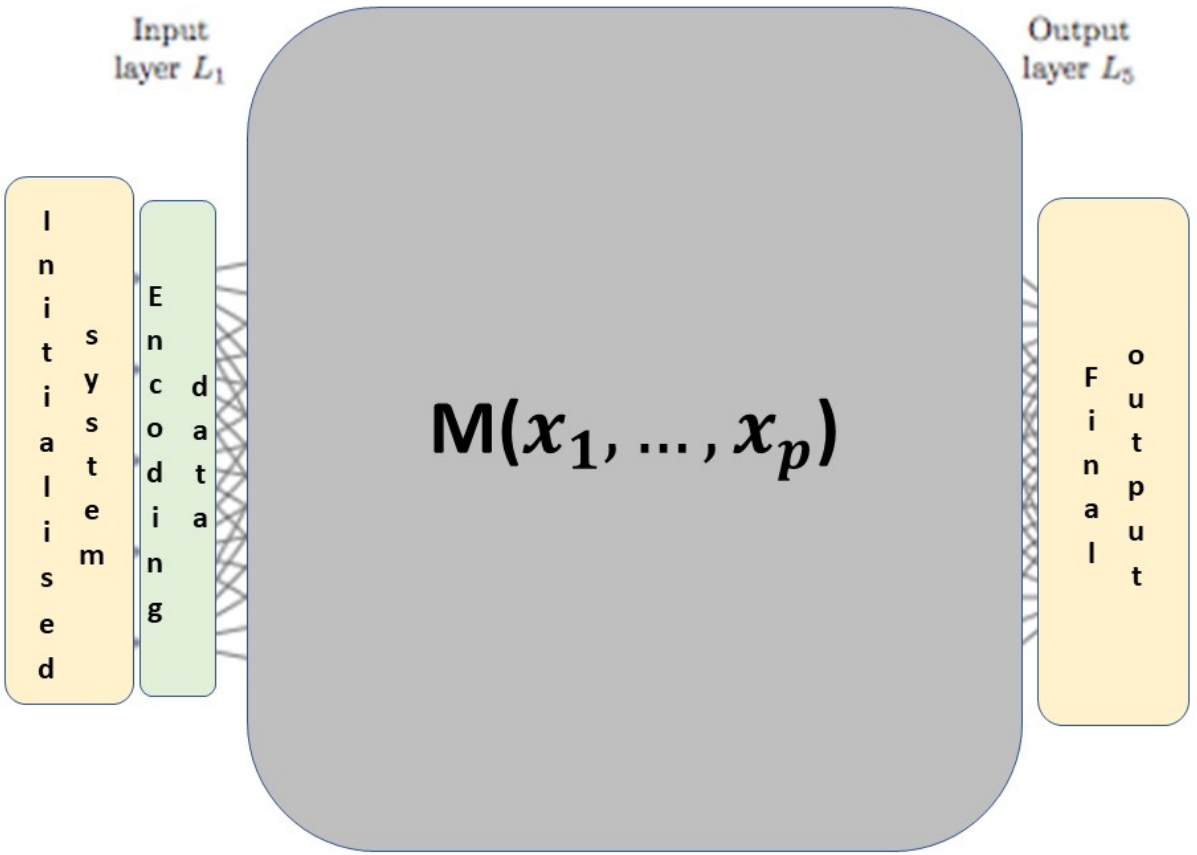

Input layer L_1
Hidden layer L_2
Hidden layer L_3
Hidden layer L_4
Output layer L_5
x_1
x_2
x_3
x_p
y_0
y_1
y_9
$\mathbf{W}^{(1)}$
$\mathbf{W}^{(2)}$
$\mathbf{W}^{(3)}$
$\mathbf{W}^{(4)}$
$a^{(2)}$
$a^{(3)}$
$a^{(4)}$
$a^{(5)}$

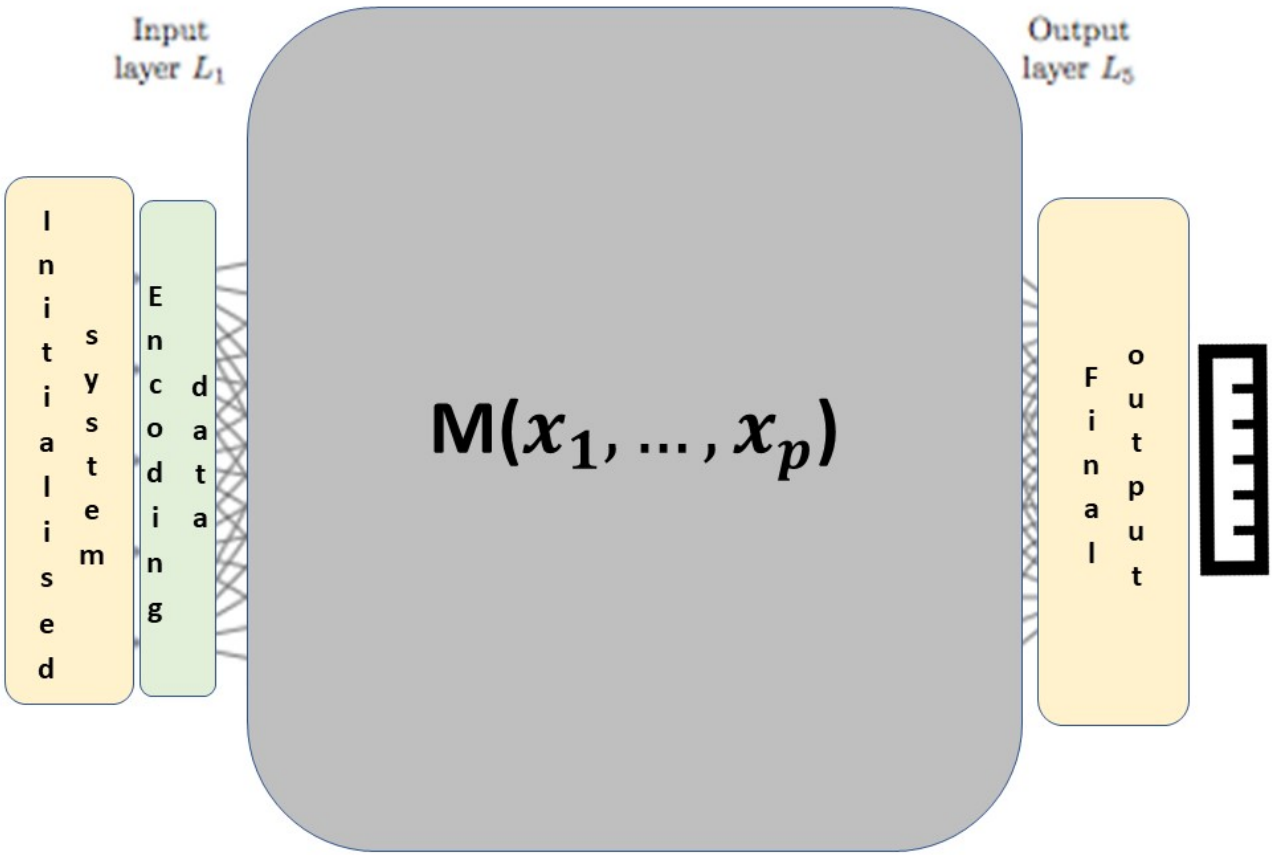

Input
layer L_1
Output
layer L_5
x_1
x_2
x_3
x_p
GREY BOX
y_0
y_1
y_9
$a^{(5)}$

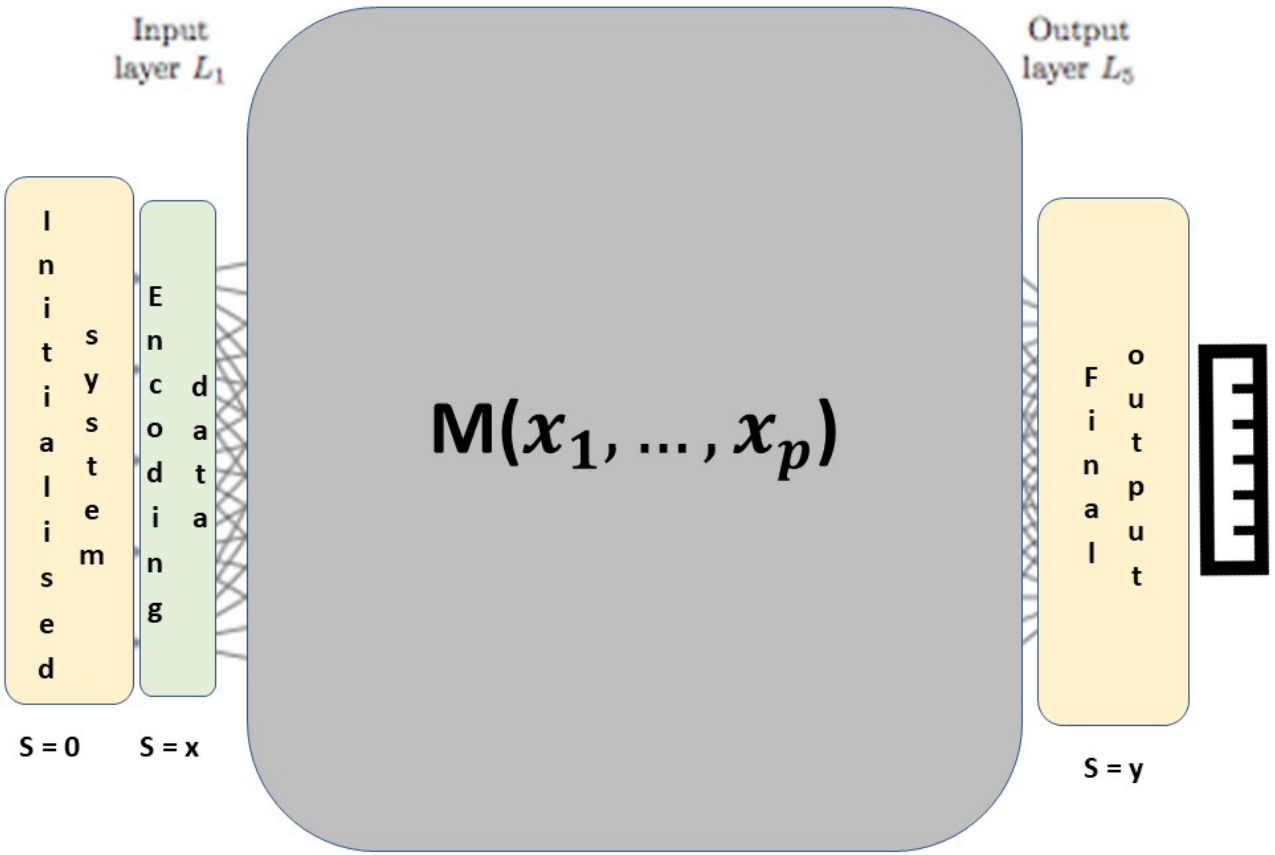

Input layer L_1
Output layer L_5
Initialised system
GREY BOX
y_0
y_1
y_9
$a^{(5)}$

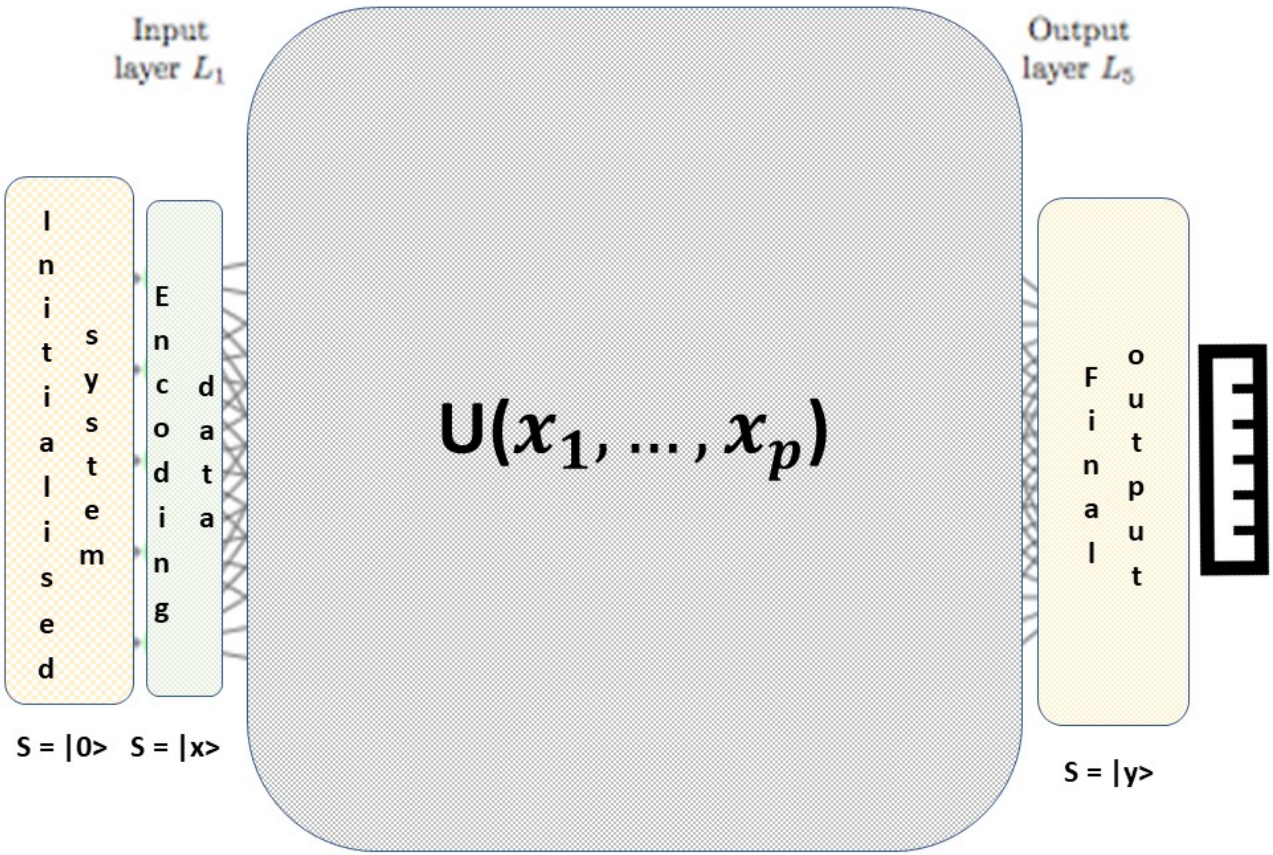

Input layer L_1
Output layer L_5
Initialised system
Encoding data
GREY BOX
y_0
y_1
y_9
$a^{(5)}$

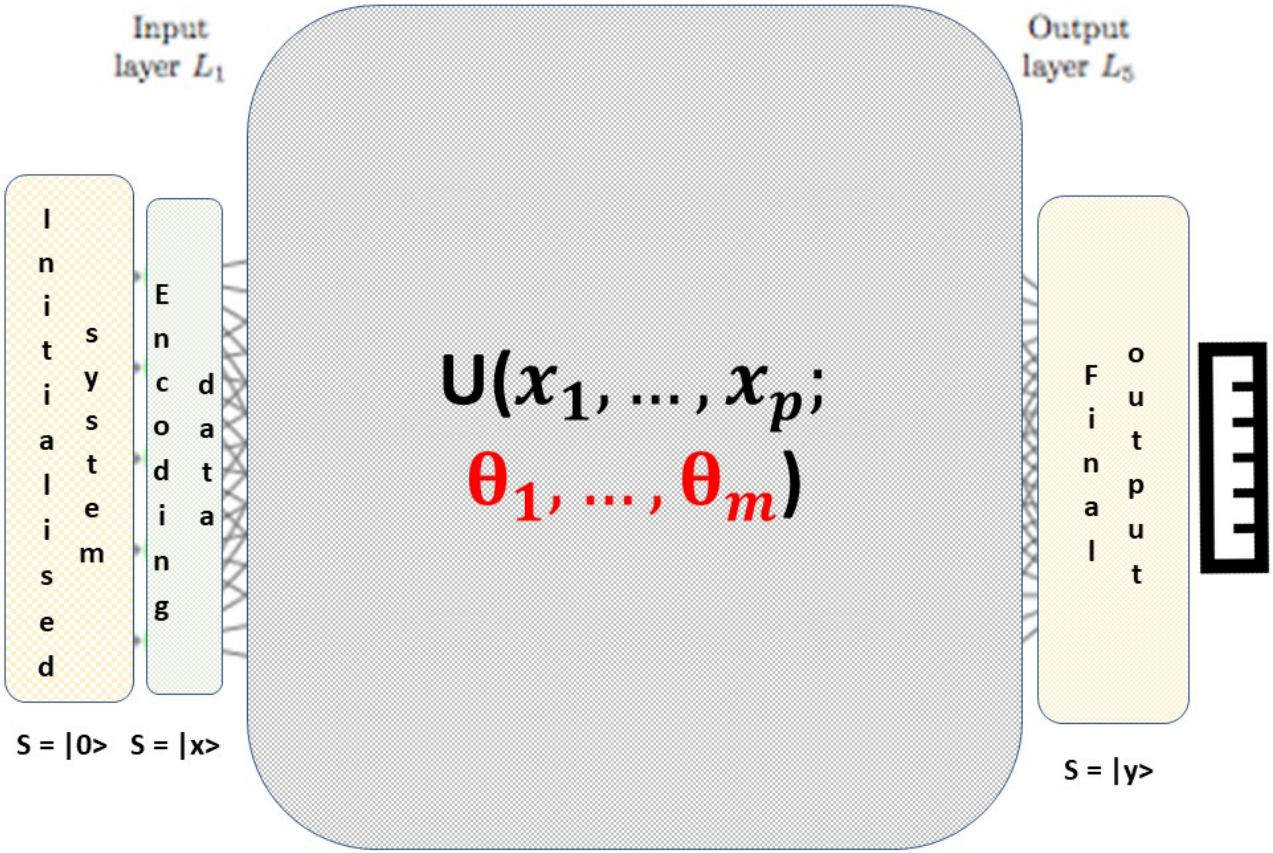

Input layer L_1
Output layer L_5
Initialised system
Encoding data
$\mathbf{M}(x_1, \ldots, x_p)$
y_0
y_1
y_9
$a^{(5)}$


Input layer L_1
Output layer L_5
Initialised system
Encoding data
$\mathbf{M}(x_1, \ldots, x_p)$
Final output


Input layer L_1
Output layer L_5
Initialised system
Encoding data
$\mathbf{M}(x_1, \ldots, x_p)$
Final output


Input layer L_1
Output layer L_5
Initialised system
Encoding data
$\mathbf{M}(x_1, \ldots, x_p)$
Final output
S = 0
S = x
S = y


Input layer L_1
Output layer L_5
Initialised system
Encoding data
$U(x_1, \ldots, x_p)$
Final output
S = |0>
S = |x>
S = |y>


Input layer L_1
Output layer L_5
Initialised system
Encoding data
$U(x_1, \ldots, x_p; \theta_1, \ldots, \theta_m)$
Final output
S = |0>
S = |x>
S = |y>

Quantum Neural Network

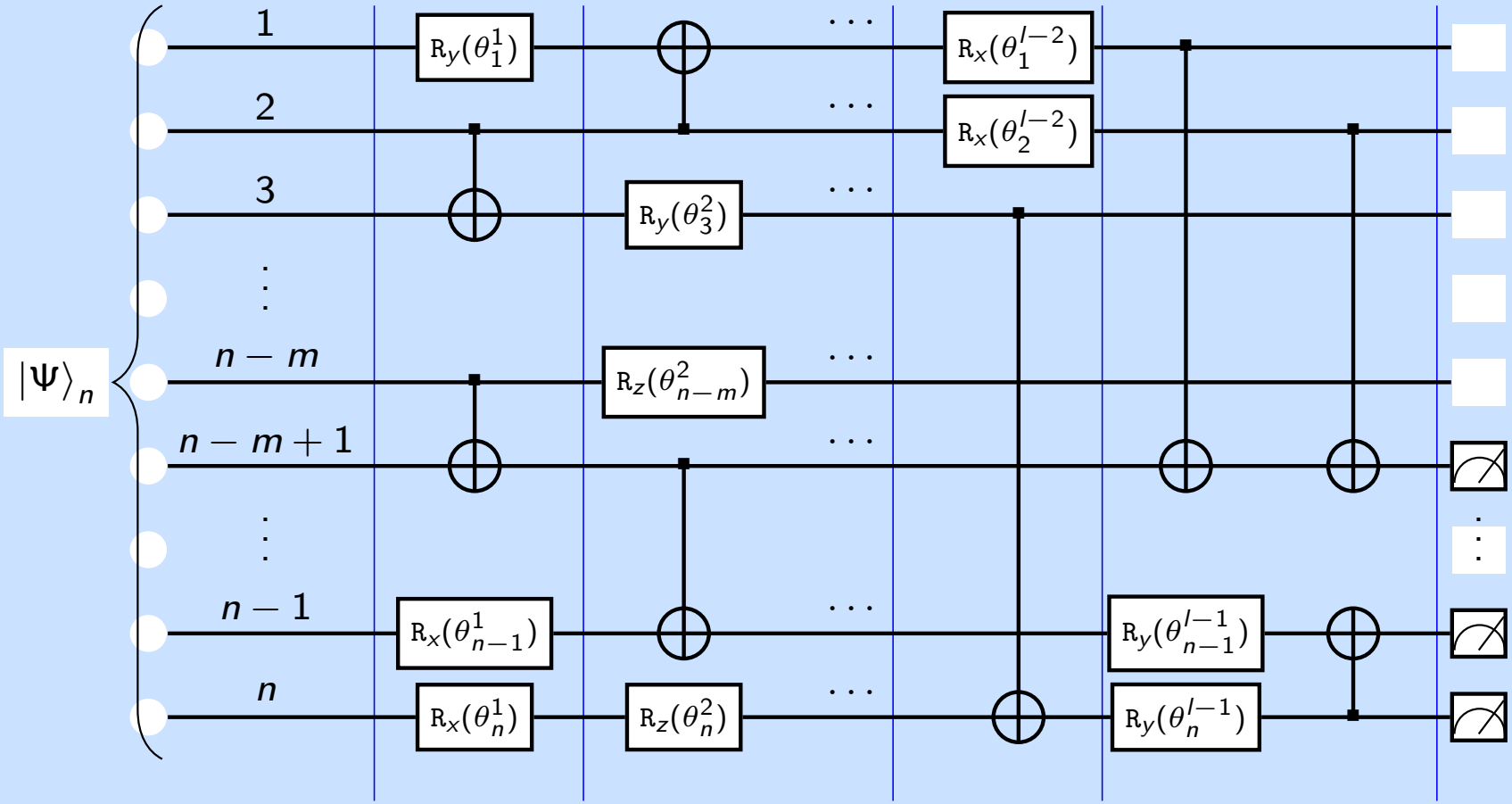


Diagramme of a quantum neural network – parameterised quantum circuit.

QNN output

Since measurement only produces a single sample, the quantum circuit needs to be run many times to obtain good enough statistics.

Example: with 2 qubits, a QNN-classifier predicts one of the four possible class labels ("0", "1", "2" and "3"). After 1,000 runs, we observe the following:

Measured bitstring	Class label	Number of observations
00	0	100
01	1	550
10	2	200
11	3	150

The most likely class label is class "1" (with probability 55%). The other probabilities, for the other possible class values, may be useful for other purposes.

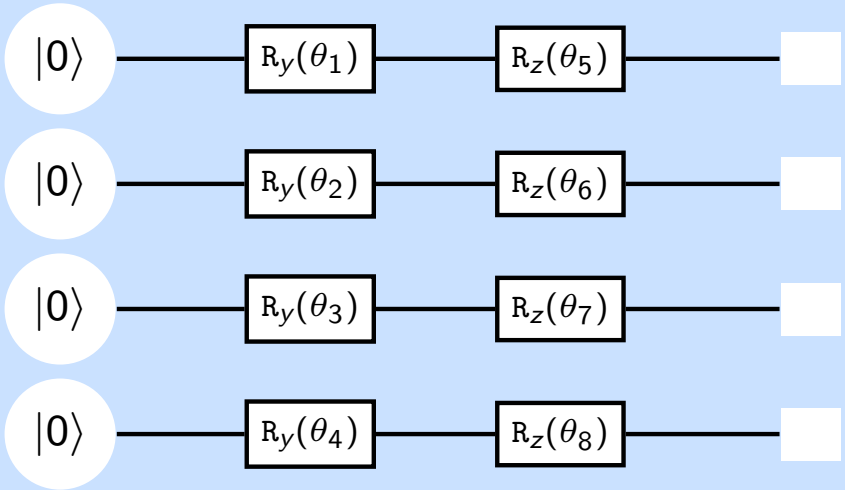
Data encoding

QNN input

The input (sample from the training dataset or a new unseen sample) must be encoded in the initial quantum state.

Feature encoding:

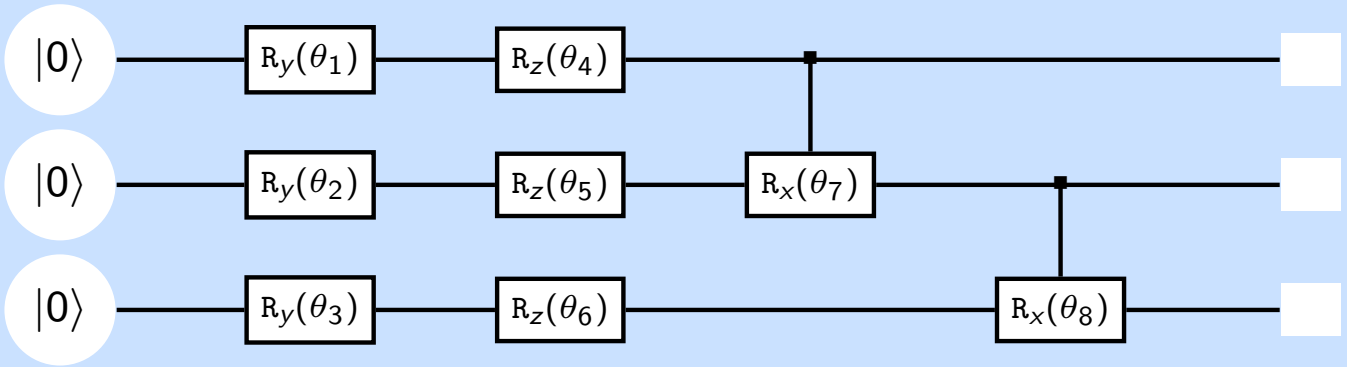
- Initialise a qubit state as $|0\rangle$
- Apply an adjustable gate (rotation) with an adjustable parameter (rotation angle) supporting one-to-one mapping of the feature value.



4-qubit quantum circuit for 8-feature sample encoding.

Superdense feature encoding

Most of the information in large quantum systems can be stored in *correlations*. In our setup above, we can reduce the number of necessary qubits to just 3 if we use entanglement:



3-qubit quantum circuit for 8-feature sample encoding.

In principle, since the n-qubit state can be uniquely described by specifying 2^n probability amplitudes, we only need n qubits to encode 2^n features.

However, this superdense encoding is not always practical nor desirable.

Binary inputs into basis states

Consider a real number $x \in \mathbb{R}$ approximated with the binary representation

$$x \approx \widehat{x} = (x_{\mathfrak{i}}, x_{\mathfrak{i}-1}, \cdots, x_{-\mathfrak{d}}) := (-1)^{x_{\mathfrak{i}}} \left(\sum_{j=0}^{\mathfrak{i}-1} x_j 2^j + \sum_{j=1}^{\mathfrak{d}} x_{-j} 2^{-j} \right) \longmapsto |x_{\mathfrak{i}} x_{\mathfrak{i}-1} \cdots x_{-\mathfrak{d}}\rangle =: |x\rangle ,$$

for $\mathfrak{i}, \mathfrak{d} \in \mathbb{N}$.

Consider now $\mathrm{x} = (x^1, \ldots, x^N) \in \mathbb{R}^N$, and concatenate all the binary approximations $\widehat{x}^1, \ldots, \widehat{x}^N$ into

$$\left(x^1_{\mathfrak{i}}, x^1_{\mathfrak{i}-1}, \cdots, x^1_{-\mathfrak{d}}, \cdots, x^N_{\mathfrak{i}}, x^N_{\mathfrak{i}-1}, \cdots, x^N_{-\mathfrak{d}} \right) \in \{0,1\}^{(1+\mathfrak{i}+\mathfrak{d})N},$$

to obtain a quantum state representation with $(1+\mathfrak{i}+\mathfrak{d})N$ qubits of the form

$$|x^1_{\mathfrak{i}} x^1_{\mathfrak{i}-1} \cdots x^1_{-\mathfrak{d}} \cdots x^N_{\mathfrak{i}}, x^N_{\mathfrak{i}-1} \cdots x^N_{-\mathfrak{d}}\rangle .$$

The encoding circuit simply only requires the $\mathtt{X}$ gate as

$$|0\rangle^{\otimes(1+\mathfrak{i}+\mathfrak{d})N} \longmapsto \bigotimes_{l=1}^{N} \bigotimes_{k=-\mathfrak{d}}^{\mathfrak{i}} \mathtt{X}^{x^l_k} |0\rangle^{\otimes(1+\mathfrak{i}+\mathfrak{d})N}$$

It however requires a large number of qubits and is in general not efficient. Indeed, for a given dimension N, there are 2^N possible basis states. If a dataset contains only M points with M much smaller than N, the quantum representation will therefore be sparse.

Superposition encoding

Consider again $(\mathrm{x}^1, \ldots, \mathrm{x}^M)$, with $\mathrm{x}^k = (x_1^k, \ldots, x_n^k) \in \{0,1\}^n$. We use a quantum system of the form

$$|\psi_0\rangle := |0\rangle^{\otimes n} |00\rangle |0\rangle^{\otimes n}.$$

The encoding works recursively. Note that again $\left(\bigotimes_{i=1}^n \mathtt{X}^{x_i^1}\right) |0\rangle^{\otimes n} = |x_1^1 \cdots x_n^1\rangle = |\mathrm{x}^1\rangle$.

$$|\psi_1\rangle := \frac{|0\rangle^{\otimes n} |00\rangle |0\rangle^{\otimes n}}{\sqrt{2}} + \frac{|0\rangle^{\otimes n} |01\rangle |\mathrm{x}^1\rangle}{\sqrt{2}} \longrightarrow \frac{|0\rangle^{\otimes n} |00\rangle |\mathrm{x}^1\rangle}{\sqrt{2}} + \frac{|0\rangle^{\otimes n} |01\rangle |0\rangle^{\otimes n}}{\sqrt{2}}.$$

After m steps, we arrive at

$$|\psi_m\rangle := \frac{1}{\sqrt{M}} \sum_{k=1}^{m} |0\rangle^{\otimes n} |00\rangle |\mathrm{x}^k\rangle + \sqrt{\frac{M-m}{M}} |0\rangle^{\otimes n} |01\rangle |0\rangle^{\otimes n}.$$

Lemma There exists a unitary operator $\mathtt{U}$ such that

$$\mathtt{U} |\psi_m\rangle = \frac{1}{\sqrt{M}} \sum_{k=1}^{m+1} |0\rangle^{\otimes n} |00\rangle |\mathrm{x}^k\rangle + \sqrt{\frac{M-(m+1)}{M}} |0\rangle^{\otimes n} |01\rangle |0\rangle^{\otimes n} =: |\psi_{m+1}\rangle$$

Angle encoding

With $\mathrm{x} = (x_1, \ldots, x_N) \in \mathbb{C}^N$ normalised so that $x_i \in [-\pi, \pi)$ for each $i = 1, \ldots, N$, angle encoding works by constructing the map

$$\mathrm{x} \longmapsto \bigotimes_{i=1}^{n} \Big(\cos(x_i) \, |0\rangle + \sin(x_i) \, |1\rangle \Big).$$

It only requires one rotation gate for each qubit, hence encodes as many features as the number of qubits.

The small variant

$$\mathrm{x} \longmapsto \bigotimes_{i=1}^{n} \Big(\cos(x_{2i-1}) \, |0\rangle + \mathrm{e}^{\mathrm{i}x_{2i}} \sin(x_{2i-1}) \, |1\rangle \Big),$$

requiring one extra phase gate, allows to encode $2n$ features with the same number n of qubits.

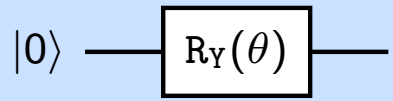
Encoding a probability distribution

GOAL: Given a probability distribution $(p_0, \ldots, p_{2^n-1})$ over 2^n points, such that $p_i \in (0,1)$ for each $i \in \{0, \ldots, 2^n - 1\}$ and $\sum_{i=0}^{2^n-1} p_i = 1$, we would like to construct a quantum circuit, represented by a unitary gate U, such that

$$\mathrm{U} \left|0\right\rangle^{\otimes n} = \sum_{i=0}^{2^n-1} \sqrt{p_i} \left|i\right\rangle .$$

$n = 1$ qubit

Consider a probability distribution over two points, with associated probability masses $p_0, p_1 \in (0,1)$ such that $p_0 + p_1 = 1$. Define $\theta_0 := 2\,\mathrm{acos}(\sqrt{p_0})$. The circuit



generates the state $\psi = \sqrt{p_0}\,|0\rangle + \sqrt{1-p_0}\,|1\rangle$, as desired.

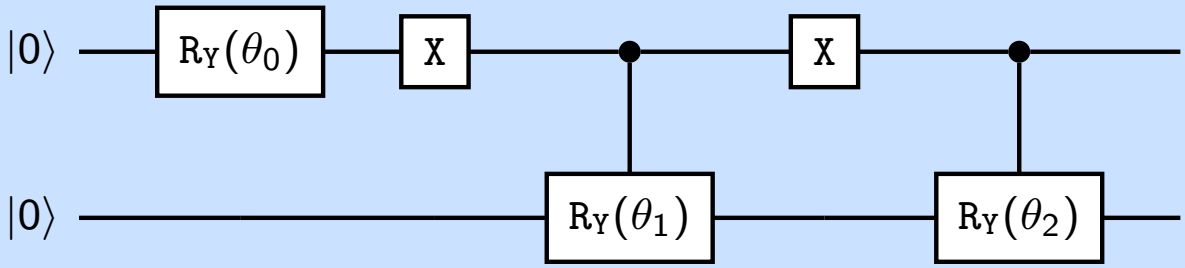
Indeed, with $\mathtt{Y} = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}$, then $\mathtt{Y}^2 = \mathtt{I}$ and, for $\theta \in \mathbb{R}$,

$$\mathrm{R_Y}(\theta) = \exp\left\{-\frac{\mathrm{i}\theta}{2}\mathtt{Y}\right\} = \cos\left(\frac{\theta}{2}\right)\mathtt{I} - \mathrm{i}\sin\left(\frac{\theta}{2}\right)\mathtt{Y},$$

so that $\mathtt{Y}\,|0\rangle = \mathrm{i}\,|1\rangle$ and

$$\begin{aligned}\mathrm{R_Y}(\theta_0)\,|0\rangle &= \left\{\cos\left(\frac{\theta_0}{2}\right)\mathtt{I} - \mathrm{i}\sin\left(\frac{\theta_0}{2}\right)\mathtt{Y}\right\}|0\rangle \\ &= \cos\left(\frac{\theta_0}{2}\right)|0\rangle + \sin\left(\frac{\theta_0}{2}\right)|1\rangle \\ &= \sqrt{p_0}\,|0\rangle + \sqrt{1-p_0}\,|1\rangle\,.\end{aligned}$$

$n = 2$ qubits and probabilities $p_{00}, p_{01}, p_{10}, p_{11}$



and we let $\theta_1 := 2\,\mathrm{acos}(\sqrt{q_1})$ and $\theta_2 := 2\,\mathrm{acos}(\sqrt{q_2})$, for some $q_1, q_2 \in (0, 1)$.

$$
\begin{aligned}
\mathtt{U}\,|00\rangle &= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)(\mathtt{X}\otimes\mathtt{I})\,{}_c\mathtt{R}_\mathtt{Y}(\theta_1)(\mathtt{X}\otimes\mathtt{I})(\mathtt{R}_\mathtt{Y}(\theta_0)\otimes\mathtt{I})\,|00\rangle \\
&= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)(\mathtt{X}\otimes\mathtt{I})\,{}_c\mathtt{R}_\mathtt{Y}(\theta_1)(\mathtt{X}\otimes\mathtt{I})\left(\sqrt{p_0}\,|0\rangle + \sqrt{1-p_0}\,|1\rangle\right)|0\rangle \\
&= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)(\mathtt{X}\otimes\mathtt{I})\,{}_c\mathtt{R}_\mathtt{Y}(\theta_1)\left(\sqrt{p_0}\,|1\rangle + \sqrt{1-p_0}\,|0\rangle\right)|0\rangle \\
&= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)(\mathtt{X}\otimes\mathtt{I})\left(\sqrt{p_0}\,|1\rangle\,\mathtt{R}_\mathtt{Y}(\theta_1)\,|0\rangle + \sqrt{1-p_0}\,|0\rangle\,|0\rangle\right) \\
&= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)(\mathtt{X}\otimes\mathtt{I})\left(\sqrt{p_0}\,|1\rangle\left[\sqrt{q_1}\,|0\rangle + \sqrt{1-q_1}\,|1\rangle\right] + \sqrt{1-p_0}\,|0\rangle\,|0\rangle\right) \\
&= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)(\mathtt{X}\otimes\mathtt{I})\left(\sqrt{p_0 q_1}\,|10\rangle + \sqrt{p_0(1-q_1)}\,|11\rangle + \sqrt{1-p_0}\,|00\rangle\right) \\
&= {}_c\mathtt{R}_\mathtt{Y}(\theta_2)\left(\sqrt{p_0 q_1}\,|00\rangle + \sqrt{p_0(1-q_1)}\,|01\rangle + \sqrt{1-p_0}\,|10\rangle\right) \\
&= \sqrt{p_0 q_1}\,|00\rangle + \sqrt{p_0(1-q_1)}\,|01\rangle + \sqrt{1-p_0}\,|1\rangle\,\mathtt{R}_\mathtt{Y}(\theta_2)\,|0\rangle \\
&= \sqrt{p_0 q_1}\,|00\rangle + \sqrt{p_0(1-q_1)}\,|01\rangle + \sqrt{1-p_0}\,|1\rangle\left[\sqrt{q_2}\,|0\rangle + \sqrt{1-q_2}\,|1\rangle\right] \\
&= \sqrt{p_0 q_1}\,|00\rangle + \sqrt{p_0(1-q_1)}\,|01\rangle + \sqrt{(1-p_0)q_2}\,|10\rangle + \sqrt{(1-p_0)(1-q_2)}\,|11\rangle\,.
\end{aligned}
$$

$$\mathtt{U}\,|00\rangle = \sqrt{p_0 q_1}\,|00\rangle + \sqrt{p_0(1-q_1)}\,|01\rangle + \sqrt{(1-p_0)q_2}\,|10\rangle + \sqrt{(1-p_0)(1-q_2)}\,|11\rangle\,.$$

Identifying this output with the desired distribution $\sqrt{p_{00}}\,|00\rangle + \sqrt{p_{01}}\,|01\rangle + \sqrt{p_{10}}\,|10\rangle + \sqrt{p_{11}}\,|11\rangle$, we deduce

$$\begin{cases} p_{00} &= p_0 q_1, \\ p_{01} &= p_0(1-q_1), \\ p_{10} &= (1-p_0)q_2, \\ p_{11} &= (1-p_0)(1-q_2) \end{cases}$$

Note further that $p_0 = p_{00} + p_{01}$ and $p_1 = p_{10} + p_{11}$ by the total law of probability, and $p_0 + p_1 = 1 =$, so that

$$q_1 = \frac{p_{00}}{p_{00}+p_{01}} \qquad \text{and} \qquad q_2 = \frac{p_{10}}{p_{10}+p_{11}}.$$

and therefore the rotation angles read

$$\theta_1 = 2\,\mathrm{acos}\left(\frac{p_{00}}{p_{00}+p_{01}}\right) \qquad \text{and} \qquad \theta_2 = 2\,\mathrm{acos}\left(\frac{p_{10}}{p_{10}+p_{11}}\right).$$

This methodology can be generalised to n qubits iteratively.

Training Quantum Neural Networks

Training QNN

Training QNN consists of specifying and executing a procedure to determine an optimal configuration of the adjustable parameters $\boldsymbol{\theta}$.

Assume a QNN with n quantum registers with l layers of adjustable quantum gates where each adjustable gate is controlled by a single parameter θ_i^j, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, l\}$.

In this case $\boldsymbol{\theta} \in \mathcal{M}_{nl}$ is an $n \times l$ matrix of adjustable network parameters:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1^1 & \ldots & \theta_1^l \\ \vdots & \ddots & \vdots \\ \theta_n^1 & \ldots & \theta_n^l \end{bmatrix}.$$

Goal: Find an optimal $\boldsymbol{\theta}$ that minimises a given cost function.

This can be done in many different ways that broadly falls into two category: *differentiable* and *non-differentiable* learning.

The differentiable learning of QNN

Step 1: Choosing the cost function.

Let $\mathbf{y} = (y_1, \ldots, y_K)$ be a vector of binary labels and let $\mathbf{f} = (f_1(\boldsymbol{\theta}), \ldots, f_K(\boldsymbol{\theta}))$ a vector of binary classifier predictions for the training dataset consisting of K samples.

Example of cost function:

$$L(\boldsymbol{\theta}) = \frac{1}{2} \sum_{k=1}^{K} (y_k - f_k(\boldsymbol{\theta}))^2 .$$

Step 2: Iterative update of the adjustable parameters by gradient descent:

$$\theta_i^j \leftarrow \theta_i^j - \eta \frac{\partial L(\hat{\theta})}{\partial \theta_i^j}, \qquad \text{for each } \; i = 1, \ldots, n, \;\; j = 1, \ldots, l.$$

with η being a given *learning rate*.

The finite difference scheme for gradient calculation

The gradient can be calculated numerically by finite difference:

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i^j} = \frac{L(\theta_1^1, \ldots, \theta_i^j + \Delta\theta, \ldots, \theta_n^l) - L(\theta_1^1, \ldots, \theta_i^j - \Delta\theta, \ldots, \theta_n^l)}{2\Delta\theta},$$

with error $\mathcal{O}((\Delta\theta)^2)$.

- Physical characteristics of NISQ devices put restrictions on the size of the increment ($\Delta\theta > 0.1$ radians).
- The rest of the training routine follows standard classical algorithm of training NN through backpropagation.

Parameter shift rule

The cost function gradient with respect to the parameter θ_i^j reads

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i^j} = -\sum_{k=1}^{K} (y_k - f_k(\boldsymbol{\theta})) \frac{\partial f_k(\boldsymbol{\theta})}{\partial \theta_i^j},$$

so that calculating it reduces to calculating $\partial_{\theta_i^j} f_k(\boldsymbol{\theta})$. Let $|\psi_k\rangle$ be the quantum state that encodes the k-th sample from the training dataset and $\mathtt{U}(\boldsymbol{\theta})$ the unitary operator that represents the sequence of QNN gates transforming the initial state $|\psi_k\rangle$. Then the expected value of the measurement operator $\mathtt{M}$ is given by

$$f_k(\boldsymbol{\theta}) = \langle\psi_k|\, \mathtt{U}^\dagger(\boldsymbol{\theta})\, \mathtt{M}\, \mathtt{U}(\boldsymbol{\theta})\, |\psi_k\rangle\,.$$

If θ_i^j only affects a single gate $\mathtt{G}(\theta_i^j)$, then $\mathtt{U}(\boldsymbol{\theta}) = \mathtt{V}\mathtt{G}(\theta_i^j)\mathtt{W}$, where $\mathtt{W}$ and $\mathtt{V}$ are gate sequences preceding and following $\mathtt{G}(\theta_i^j)$. Absorb $\mathtt{V}$ into the Hermitian observable $\mathtt{Q} = \mathtt{V}^\dagger\mathtt{M}\mathtt{V}$ and write $|\phi_k\rangle = \mathtt{W}\,|\psi_k\rangle$, so that

$$f_k(\boldsymbol{\theta}) = \langle\phi_k|\, \mathtt{G}^\dagger(\theta_i^j)\mathtt{Q}\mathtt{G}(\theta_i^j)\, |\phi_k\rangle\,.$$

Then (writing $\partial_i^j := \partial_{\theta_i^j}$)

$$\partial_i^j f_k(\boldsymbol{\theta}) = \partial_i^j \langle\phi_k|\, \mathtt{G}^\dagger(\theta_i^j)\mathtt{Q}\mathtt{G}(\theta_i^j)\, |\phi_k\rangle = \langle\phi_k|\left(\partial_i^j \mathtt{G}(\theta_i^j)\right)^\dagger \mathtt{Q}\mathtt{G}(\theta_i^j)\, |\phi_k\rangle + \langle\phi_k|\, \mathtt{G}^\dagger(\theta_i^j)\mathtt{Q}\left(\partial_i^j \mathtt{G}(\theta_i^j)\right) |\phi_k\rangle\,.$$

Write $\mathtt{B} := \mathtt{G}(\theta_i^j)$ and $\mathtt{C} := \dfrac{\partial \mathtt{G}(\theta_i^j)}{\partial \theta_i^j}$ and notice that

$$\begin{aligned}
&\langle \phi_k | \, \mathtt{C}^\dagger \mathtt{Q} \mathtt{B} \, | \phi_k \rangle + \langle \phi_k | \, \mathtt{B}^\dagger \mathtt{Q} \mathtt{C} \, | \phi_k \rangle \\
&= \frac{1}{2} \left(\langle \phi_k | \, (\mathtt{B} + \mathtt{C})^\dagger \mathtt{Q} (\mathtt{B} + \mathtt{C}) \, | \phi_k \rangle - \langle \phi_k | \, (\mathtt{B} - \mathtt{C})^\dagger \mathtt{Q} (\mathtt{B} - \mathtt{C}) \, | \phi_k \rangle \right).
\end{aligned}$$

Therefore, if we can find the way to implement the operator $\mathtt{B} \pm \mathtt{C}$ as part of an overall unitary evolution then we can evaluate the gradient directly.

The parameter-shift rule [Schuld, 2018]

Since $\mathtt{G}(\theta_i^j)$ is unitary, then $\mathtt{G}(\theta_i^j) = \exp\left(-\mathrm{i}\theta_i^j \Gamma\right)$ for some Hermitian Γ, hence

$$\partial_i^j \mathtt{G}(\theta_i^j) = -\mathrm{i}\Gamma \exp\left(-\mathrm{i}\theta_i^j \Gamma\right) = -\mathrm{i}\Gamma \mathtt{G}(\theta_i^j).$$

Substituting into above yields

$$\partial_i^j f_k(\boldsymbol{\theta}) = \langle \varphi_k | \, \mathrm{i}\Gamma \mathtt{Q} \, | \varphi_k \rangle + \langle \varphi_k | \, \mathtt{Q}(-\mathrm{i}\Gamma) \, | \varphi_k \rangle \,,$$

where $|\varphi_k\rangle = \mathtt{G}(\theta_i^j) \, |\phi_k\rangle$. If Γ has just two distinct eigenvalues we can shift the eigenvalues to $\pm r$, since the global phase is unobservable:

$$\begin{aligned}
\partial_i^j f_k(\boldsymbol{\theta}) &= r \left(\langle \varphi_k | \, \frac{\mathrm{i}\Gamma}{r} \mathtt{Q}\mathtt{I} \, | \varphi_k \rangle - \langle \varphi_k | \, \mathtt{I}\mathtt{Q} \frac{\mathrm{i}\Gamma}{r} \, | \varphi_k \rangle \right) \\
&= \frac{r}{2} \left[\langle \varphi_k | \left(\mathtt{I} - \frac{\mathrm{i}}{r}\Gamma \right)^\dagger \mathtt{Q} \left(\mathtt{I} - \frac{\mathrm{i}}{r}\Gamma \right) | \varphi_k \rangle - \langle \varphi_k | \left(\mathtt{I} + \frac{\mathrm{i}}{r}\Gamma \right)^\dagger \mathtt{Q} \left(\mathtt{I} + \frac{\mathrm{i}}{r}\Gamma \right) | \varphi_k \rangle \right],
\end{aligned}$$

where $\mathtt{B} := \mathtt{I}$ and $\mathtt{C} := -\frac{\mathrm{i}}{r}\Gamma$,

Noting that

$$\mathtt{G}\left(\mp\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}\left(\mathtt{I} \pm \frac{\mathrm{i}}{r}\Gamma\right),$$

the 'parameter shift rule', with the shift $s = \pi/(4r)$, then reads

$$\partial_i^j f_k(\boldsymbol{\theta}) = r\left(\langle\phi_k|\,\mathtt{G}^\dagger\left(\theta_i^j + \frac{\pi}{4r}\right)\mathtt{Q}\mathtt{G}(\theta_i^j + s)\,|\phi_k\rangle - \langle\phi_k|\,\mathtt{G}^\dagger\left(\theta_i^j - \frac{\pi}{4r}\right)\mathtt{Q}\mathtt{G}(\theta_i^j - s)\,|\phi_k\rangle\right).$$

Example: If $\Gamma \in \{\mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$, then $r = 1/2$, $s = \pi/2$, and

$$\partial_i^j f_k(\boldsymbol{\theta}) = \frac{1}{2}\left[\langle\phi_k|\,\mathtt{G}^\dagger\left(\theta_i^j + \frac{\pi}{2}\right)\mathtt{Q}\mathtt{G}\left(\theta_i^j + \frac{\pi}{2}\right)|\phi_k\rangle - \langle\phi_k|\,\mathtt{G}^\dagger\left(\theta_i^j - \frac{\pi}{2}\right)\mathtt{Q}\mathtt{G}\left(\theta_i^j - \frac{\pi}{2}\right)|\phi_k\rangle\right].$$

Not necessarily faster than FD, but may produce a more accurate estimate of the cost function gradient, because of current NISQ hardware limited precision.

Non-differentiable learning of QNN

Differentiable learning, although quite powerful in many cases, may not work for non-convex cost function with many local minima separated by tall barriers.

Non-differentiable learning: *Particle Swarm Optimisation* (PSO) algorithm.

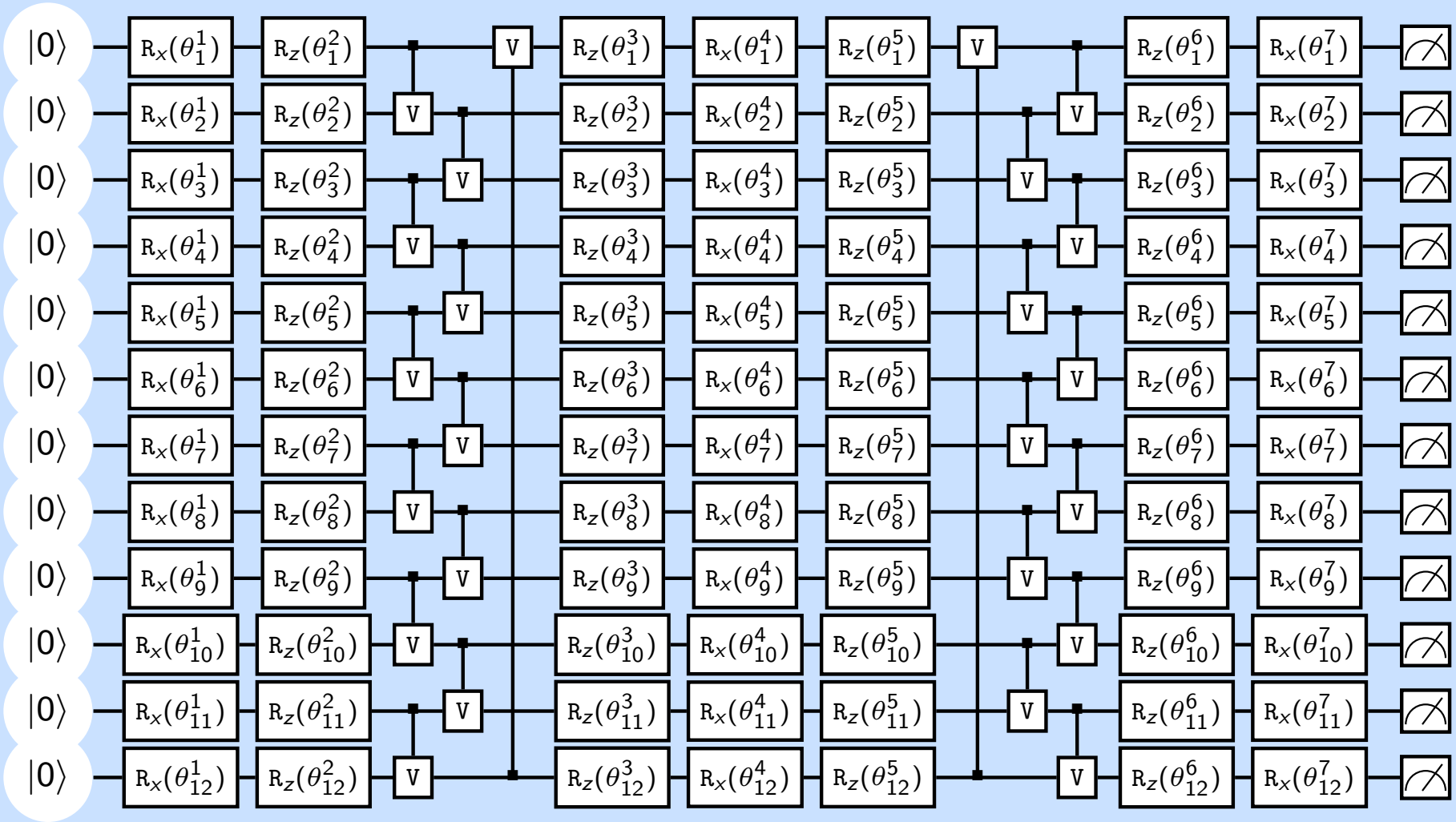
PSO belongs to a wide class of evolutionary search heuristics where at each iteration (*generation*) the population of solutions (*chromosomes*) is evaluated in terms of their fitness with respect to the environment.

Quantum Circuit Born Machine

Quantum Circuit Born Machine

- The Quantum Circuit Born Machine (QCBM) is a parameterised quantum circuit where a layer of adjustable one-qubit gates is followed by a layer of fixed two-qubit gates, a pattern that can be repeated any number of times building a progressively deeper circuit.
- Input: a quantum state $|0\rangle^{\otimes n}$.
- Final layer: measurement operators producing a bitstring sample from the learned distribution.
- To specify the QCBM architecture: number of layers, type of adjustable gates, type of fixed gates for each layer.

QCBM architecture



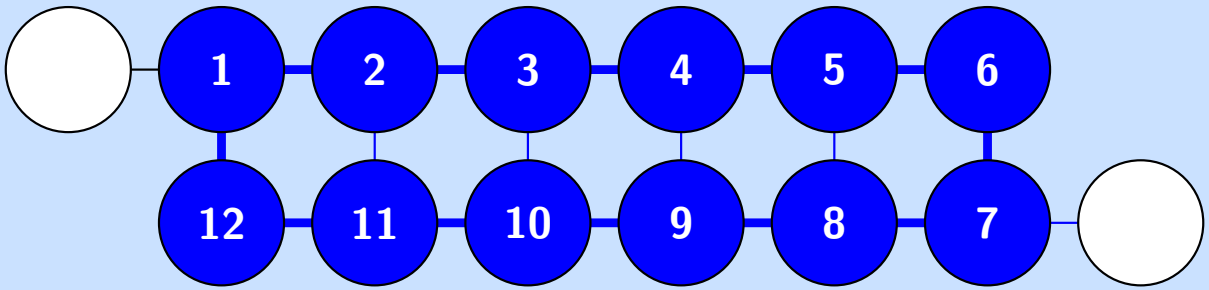
QCBM(12, 7): 12 quantum registers, 7 layers of adjustable gates.

QPU embedding

The 12×7 QCBM architecture is compatible with the limited connectivity observed in the current generation of quantum processors.

This architecture can be supported by, e.g., IBM's Melbourne system.

12 shaded qubits correspond to 12 quantum registers. The thick lines represent connections used in QCBM ansatz; the thin lines represent all other available qubit connections.



IBM's Melbourne system

Training QCBM

Training of QCBM follows the same principles as that of QNN: minimisation of a cost function. The main difference is the form of the cost function:

- QNN-based classifier: the cost function represents the classification error
- QCBM: the cost function represents the distance between two probability distributions (distribution of training samples and distribution of generated samples).

Let $\boldsymbol{\theta}$ denote the set of adjustable QCBM parameters, $p_{\boldsymbol{\theta}}(\cdot)$ the QCBM distribution and $\pi(\cdot)$ the data distribution. Then we can define the *maximum mean discrepancy* cost function $L(\boldsymbol{\theta})$ as

$$L(\boldsymbol{\theta}) := \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\boldsymbol{\theta}},\mathbf{y}\sim p_{\boldsymbol{\theta}}}[K(\mathbf{x},\mathbf{y})] - 2\mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\boldsymbol{\theta}},\mathbf{y}\sim \pi}[K(\mathbf{x},\mathbf{y})] + \mathop{\mathbb{E}}_{\mathbf{x}\sim \pi,\mathbf{y}\sim \pi}[K(\mathbf{x},\mathbf{y})],$$

where $K(\cdot,\cdot)$ is a *kernel function* – a measure of similarity between points in the sample space.

Classical and quantum kernels

Popular choice of kernel function is the Gaussian mixture:

$$K(\mathbf{x}, \mathbf{y}) = \frac{1}{c} \sum_{i=1}^{c} \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma_i}\right),$$

for some $c \in \mathbb{N}$ and where σ_i, $i = 1, \ldots, c$, are the bandwidth parameters of each Gaussian kernel.

We can also explore the possibility of using *quantum kernels*. Quantum kernels can provide an advantage over classical methods for kernels that are difficult to compute on a classical device.

For example, we can consider a quantum kernel method, which uses a quantum circuit $\mathtt{U}(x)$ to map real data into a quantum state $|\phi\rangle$ via a *feature map* $\phi : \mathbf{x} \to |\phi(\mathbf{x})\rangle$:

$$|\phi(\mathbf{x})\rangle = \mathtt{U}(\mathbf{x}) |0\rangle^{\otimes n}.$$

The kernel function is then defined as the squared inner product

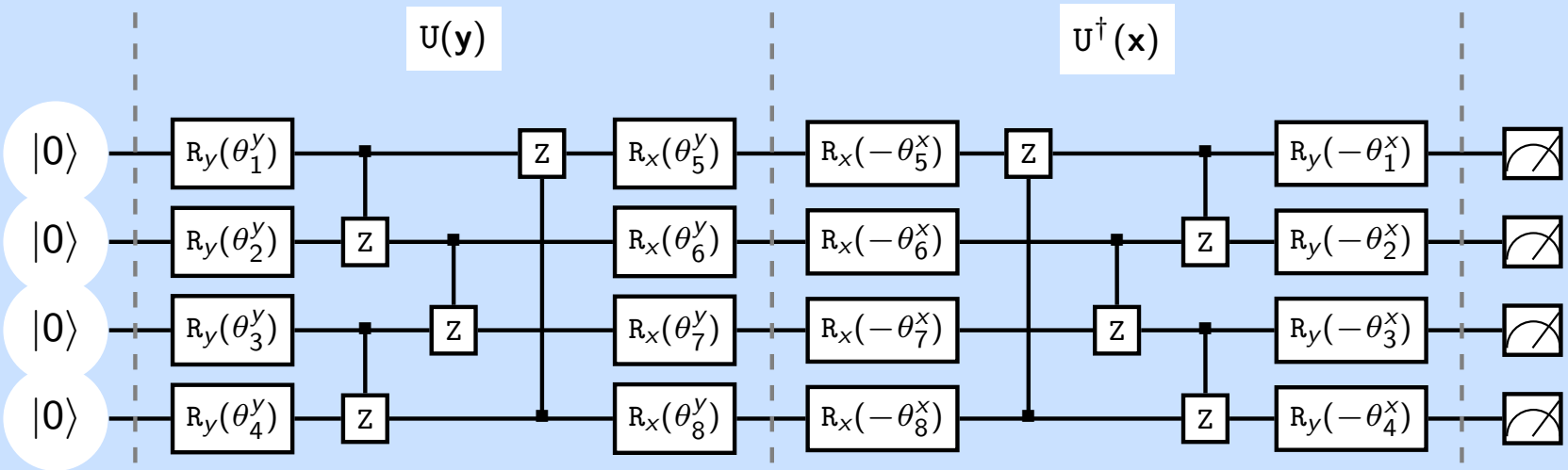
$$K(\mathbf{x}, \mathbf{y}) = |\langle \phi(\mathbf{x}) | \phi(\mathbf{y}) \rangle|^2.$$

Calculation of quantum kernel on a quantum computer

The quantum kernel takes the form (with $|\mathbf{0}\rangle \equiv |0\rangle^{\otimes n}$):

$$K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{0} | \, \mathtt{U}^\dagger(\mathbf{x}) \mathtt{U}(\mathbf{y}) \, | \mathbf{0} \rangle \, ,$$

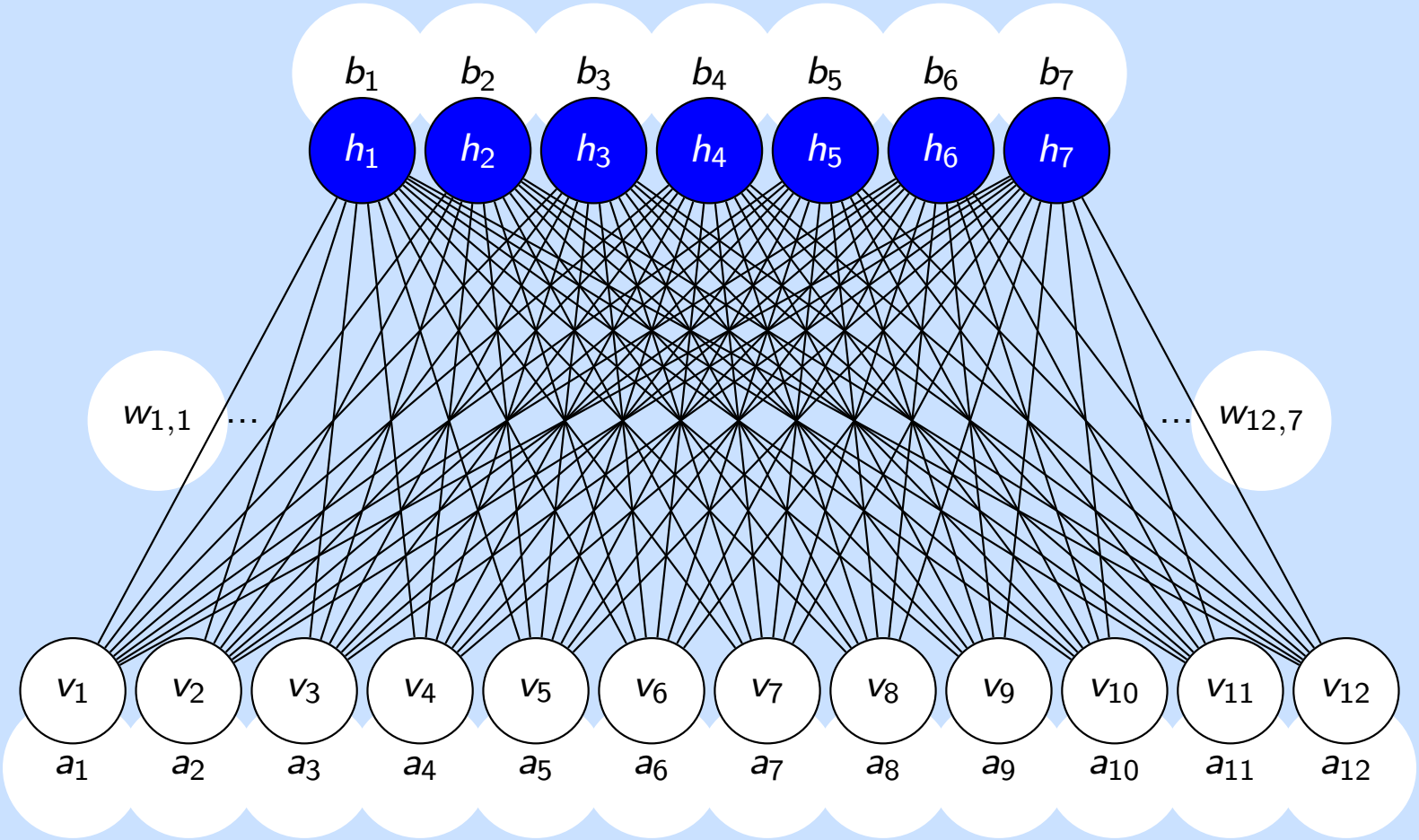
and is the probability of measuring the all-zero outcome.



Schematic quantum kernel circuit. Note that $\mathtt{R}_y(\theta)^\dagger = \mathtt{R}_y(-\theta)$ and ${}_c\mathtt{Z}^\dagger = {}_c\,\mathtt{Z}$.

Classical counterpart – Restricted Boltzmann Machine

QCBM performance can be compared to that of RBM. Both operate on the binary representation of the dataset, with similar number of adjustable parameters.



RBM(12, 7)

Expressive power???

- For deep neural networks, the power of a model can be quantified by the *Vapnik-Chervonenkis dimension*.
- Another popular approach: *Fisher information*.
- *Entanglement entropy* of a bipartite system ρ_{AB} is defined as

$$\mathcal{S}(\rho_A) = -\mathrm{Tr}(\rho_A \log(\rho_A)) = -\mathrm{Tr}(\rho_B \log(\rho_B)) = \mathcal{S}(\rho_B),$$

where $\rho_A = \mathrm{Tr}_B(\rho_{AB})$ is the reduced density matrix of system A and $\rho_B = \mathrm{Tr}_A(\rho_{AB})$ is the reduced density matrix of system B.

Quantum Universal Approximation (joint with L. Gonon)

For $n \in \mathbb{N}$ and $\boldsymbol{\theta} \in \mathbb{R}^{3n}$, define

$\mathtt{C}_{\mathfrak{n}}(\boldsymbol{\theta}) := \mathtt{U}(\boldsymbol{\theta}, \mathrm{x})\mathtt{V}$ acting on $\mathfrak{n} = \lceil \log_2(4n) \rceil$ qubits:

Theorem For any $R > 0$, $f \in \mathcal{F}_R = \cdots$ and $n \in \mathbb{N}$, there exists $\boldsymbol{\theta}$ such that

$$\left(\int_{\mathbb{R}^d} \left| f(\mathrm{x}) - f^R_{n,\boldsymbol{\theta}}(\mathrm{x}) \right|^2 \mu(\mathrm{dx}) \right)^{1/2} \leq \frac{\mathrm{Const}}{\sqrt{n}},$$

where the function $f^R_{n,\boldsymbol{\theta}}$ is constructed via $\mathtt{C}_{\mathfrak{n}}(\boldsymbol{\theta})$.

Example: Given a (measurement) operator $\mathcal{M}$, the expectation of $\mathcal{M}$ is given by $\langle \psi | \mathcal{M} | \psi \rangle$. So one can define $f^R_{n,\boldsymbol{\theta}}(\mathrm{x}) := \langle \psi(\boldsymbol{\theta}, \mathrm{x}) | \mathcal{M} | \psi(\boldsymbol{\theta}, \mathrm{x}) \rangle$.

Note: Similar result with *reservoir* quantum neural networks.