1- Single Q-Bit

A single q-bit is a vecton

$$|\psi\rangle = a|1\rangle + b|0\rangle$$

parametrized by 2 complex numbens that satisfy

$$|a|^2 + |b|^2 = 1$$

An operetor U must preserve this nonm and thus it must be unitany

$$U\,U^{\dagger} = I$$

2- Operations and gates

2.1- Pauli Matrices

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

2.2- Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{X + Z}{\sqrt{2}}$$

2.3 - Phase

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = T^2$$

2.4 - $\pi/8$ (T)

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}$$

2.5 - Rotations

$$R_{\hat{n}}(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \operatorname{sen}\left(\frac{\theta}{2}\right)(n_x X + n_y Y + n_z Z)$$

Where

$$\hat{n} = (n_x, n_y, n_z)$$

is an arbitrary unit vector representing the direction of rotation

2.5 - Axial decomposition

Given two non parallel unit vectors $\hat{m}$ and $\hat{n}$, any gate U can be decomposed as

$$U = e^{i\alpha} R_{\hat{n}}(\beta) R_{\hat{m}}(\gamma) R_{\hat{n}}(\delta)$$

for some α, β, γ and δ

2.6- Identities

$$HXH = Z$$

$$HYH = -Y$$

$$HZH = X$$

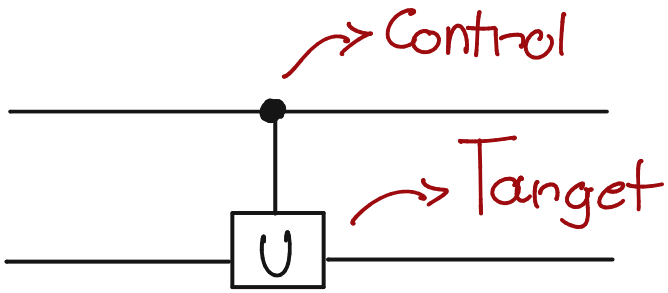
3- Controlled Operations

Consider two q-bits : control $|c\rangle$ and target $|t\rangle$.

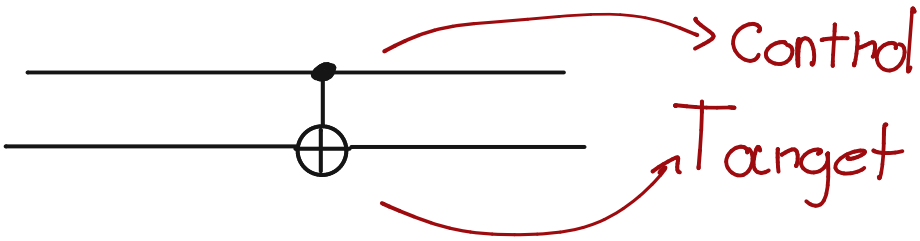
Given an arbitrary single q-bit operation U, its controled version is

$$|c\rangle|t\rangle \longrightarrow |c\rangle U^c |t\rangle$$

That is, if the control q-bit is set, then U is applied to the target, otherwise, the target is left alone



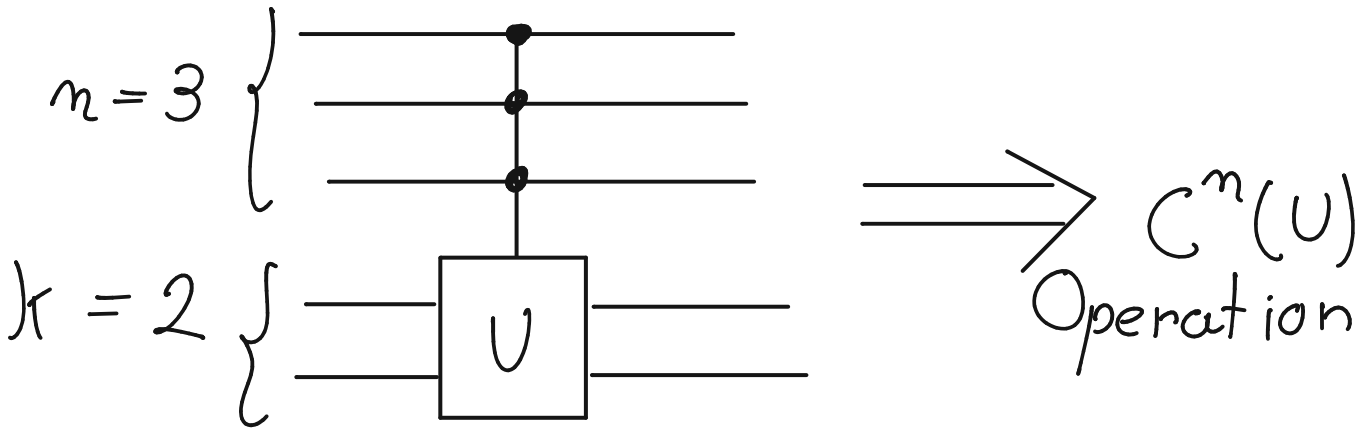
Example: CNOT, which flips the target if the control is set



Matrix representation for CNOT

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

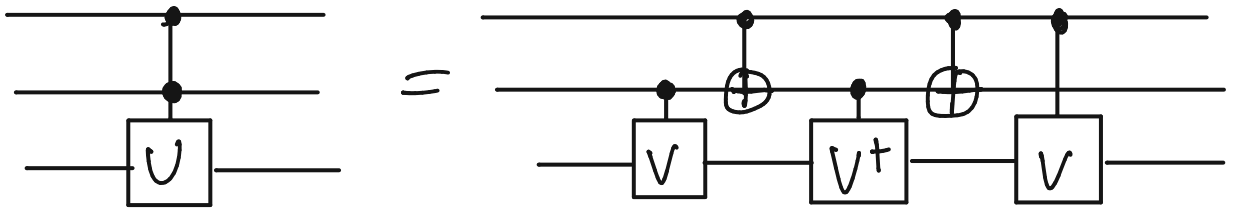
These operations generalize trivially to n controls and k targets



If U is a single unitary operation and V is such that

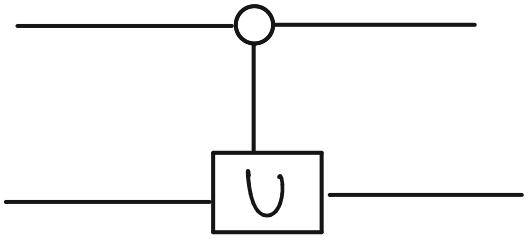
$$V^2 = U$$

The $C^2(U)$ operation becomes

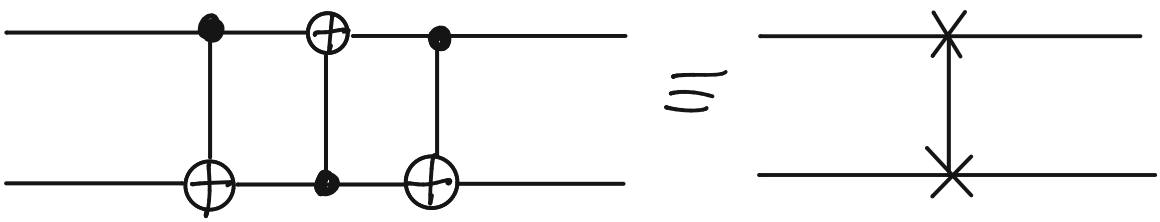


If $V = \frac{(i-1)(I+iX)}{2}$, this implements the Toffoli gate

In order to test if the control is $|0\rangle$ we can denote the circuit like



A useful gate is the swap gate. It swaps the states of two q-bits



4- Measurement

Measurements are denoted by 
There are 2 principles of measurement to remember:

(1) Any q-bit in an unterminated quantum wire may assumed to be measured

(2) Measurement can aways be moved from an intermediate state to the end of a cirquit
If the measurements are used in any stage of the circuit, classical control operations can be replaced by quantum ones

5 - Universal Gates

There are 2 sets that can be used to approximate any unitary operatoh with arbitrary accuracy. Note tha even tough we can approximate any gate, it does not mean that is easiy

H-S-CNOT-T or H-S-CNOT-Toffoli
↳ Stol. set

6- QFT

The DFT takes a set of N numbers $(x_0, \dots, x_{N-1})$ and maps them to the set $(y_0, \dots, y_{N-1})$ using the rule

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \exp\left(\frac{2\pi i j k}{N}\right)$$

The QFT is defined as

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i j k}{N}\right) |k\rangle$$

It can also be expressed as a unitary operation

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \exp\left(\frac{2\pi i j k}{N}\right) |k\rangle\langle j|$$

The QFT is simply a base change. We denote states in the computational basis by a tilde

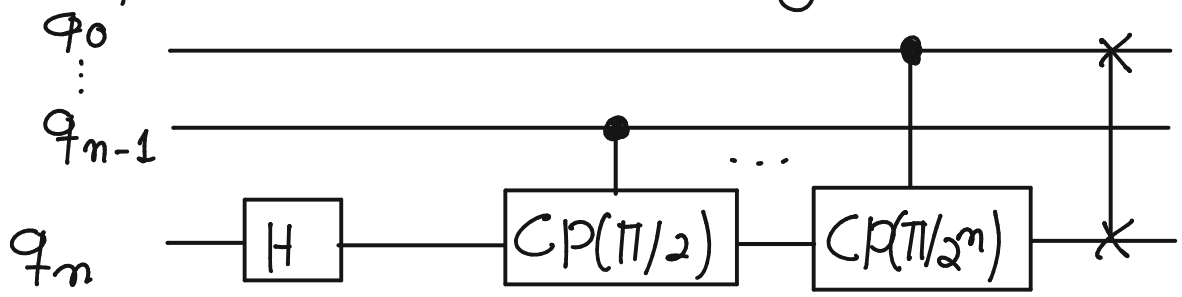
The implementation is based on the conditional phase gate $CP(\theta)$, given in 2 q-bit computational basis as

$$CP(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix}$$

Identity

Generalized phase

The QFT is the recursive application of the basic algorithm



Thus the algorithm starting at q-bit n with N total q-bits

1- Apply H to q_n

2- For i in $[n, 0[$ and step -1 apply $CP(\pi/2^{(n-it+1)}, i-1, n)$

3- Reset $n \to n-1$. Go to step 1 until $n = 0$

4- Swap $q_0, q_n \mid q_1, q_{n-1} \mid \dots$ in pairs

Quantum Machine Learning

ML can be split into 3 subfields

1) Supervised Learning

Given tuples of labeled data (x_i, y_i) the goal is to learn a function that maps $f: x \rightarrow y$ and generalizes to other inputs

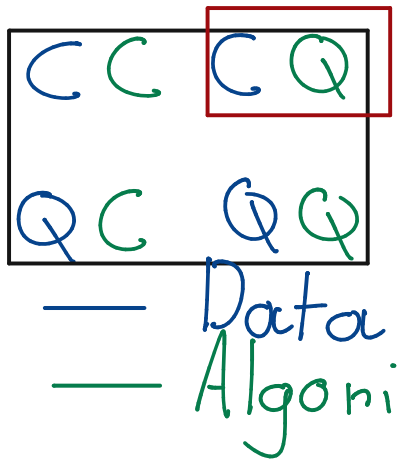
2) Unsupervised Learning

Given a set of unlabeled data we want to learn some structure of the data

3) Reinforcement Learning

Maximize a reward function with an env. that rewards us based on our actions

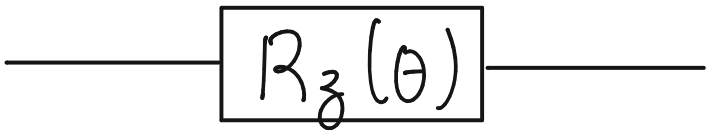
Approaches are classified based on data and algorithims:



→ Our research uses this approach. We will process classical data using quantum algorithims

Parametrized Quantum Circuit

This is a circuit where gates have a tunable parameter. Example: rotation



Mathematically, a parametrized gate is a unitary operation U acting on an initial state $|\phi_0\rangle$ and producing another state $|\phi_\theta\rangle$

$$|\phi_\theta\rangle = U_\theta |\phi_0\rangle$$

Expressibility of a PQC

Expressibility of a PQC refers

to the "size" of the subset of Hilbent space covered by a circuit

Imagine a circuit U produces all possible states in the Bloch sphere. This circuit is called a uniform distribution
Expressibility measures how much a circuit deviates from U

A possible distance measure is the Kullback - Leibler divergence

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

The KL divergence of probability distribution P from Q is the expected excess surprise of using Q as a model when the actual distribution is P
X represents the sample space

Entangling capacity

This measure is defined on the basis of the Meyen - Wallach measure Q:

Consider n qubits with basis labeled by $|b_1 \dots b_n\rangle$ with $b_j \in \{0,1\}$. For $b \in \{0,1\}$ define

$$i_j(b)|b_1, \dots b_n\rangle = \delta_{b b_j} |b_1 \dots \hat{b_j} \dots b_n\rangle$$

where $\hat{b_j}$ indicates the absence of b_j

We can see that $i_j(b)$ maps

$$i_j : \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes (n-1)}$$

For any $(u, v) \in (\mathbb{C}^2)^{\otimes (n-1)}$ we can expand

$$u = \sum u_x |x\rangle$$

$$v = \sum v_y |y\rangle$$

where $0 \leq (x, y) \leq 2^{n-1}$ are bit strings

Let

$$D(u, v) = \sum_{x<y} |u_x v_y - u_y v_x|^2.$$

For $\psi \in (\mathbb{C}^2)^{\otimes n}$, the MW measure is

$$\boxed{Q(\psi) \frac{4}{n} \sum_{j=1}^{n} D[i_j(0)\psi, i_j(1)\psi]}$$

The entanglement capability of a circuit is the average MW measure of the states generated by the circuit

$$Ent = \frac{1}{|S|} \sum_{\theta_i \in S} Q(|\psi_{\theta_i}\rangle)$$

Where $S = \{\theta_i\}$ is the set of sampled circuit parameter vectors

Hardware efficient constructions

Current quantum hardware is kinda bad. We need to build circuits that help the shitty hardware. These circuits have the following traits

- 1 2-qubit entangling gate
- 1-3 single qubit gate(s)
- The circuit is constructed from blocks of single qubit gates and entangling gates, applied to multiple or all circuits in parallel.

One sequence of single qubit and entangling block is called a layer

Data encoding

First, it is usefull to define the concept of features

A feature is a property of the things we are trying to learn about, encoded as numbers.

Lets consider a classical dataset of M samples, each with N features

$$\mathcal{X} = \{x^{(1)}, \dots, x^{(M)}\}$$

Each $x^{(i)}$ is thus an N-dimentional vector

Basis encoding

This schem represents a classical N bit string as a N bit state in the computational basis. For an N-bit string $x = (b_1, \dots, b_N)$ the corresponding N-qubit state is

$|x\rangle = |b_1 \dots b_N\rangle$ with $b_n \in \{0, 1\}$ for $n = 1, \dots N$

To encode $\mathcal{H}$ using this scheme, each data element needs to be a N bit string $x^{(m)} = (b_1, \dots, b_N)$ which can then be mapped to the state $|x^m\rangle = |b_1 \dots b_N\rangle$ with $b_n \in \{0, 1\}$ for $n = 1, \dots, N$ and $m = 1, \dots, M$.
The dataset is then a superposition of basis states

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

This scheme is simple but not very bit-efficient.

Amplitude encoding

In this scheme, data is encoded in the amplitude of the quantum state An N-dimentional data point x is nepre-sented as the amplitudes of a n-qubit state $|\psi_x\rangle$

$$|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$$

where $N = 2^n$ and $|i\rangle$ is the i-th computational basis state

To encode $\mathcal{H}$, we flatten it into a 1-D vector of lenght $N \times M$

$$\alpha = A\left(x_1^{(1)}, \dots, x_N^{(1)}, \dots, x_1^{(M)}, \dots, x_N^{(M)}\right)$$

where A is a normalization facton. The dataset then is

$$|\mathcal{H}\rangle = \sum_{i=1}^{N \times M} \alpha_i |i\rangle$$

where α_i are the elements of α and $|i\rangle$ are the computational basis states

To encode $N \times M$ amplitudes, we need

$$n \geq \log(NM)$$

qubits since a system of n qubits provides 2^n amplitudes.

This method uses few qubits but it uses the amplitude of the state which is not efficient

Angle encoding

This method encodes N features in the rotation angle of n qubits. The data point $x = (x_1, \dots, x_N)$ can be encoded as

$$|x\rangle = \bigotimes_{i=1}^{N} [\cos(x_i)|0\rangle + \sin(x_i)|1\rangle]$$

This encodes 1 point at a time but requires only N qubits or less. It is good for modern hardware

This encoding can be written as a unitary transformation

$$S_{x_j} = \bigotimes_{i=1}^{N} U(x_j^{(i)})$$

where

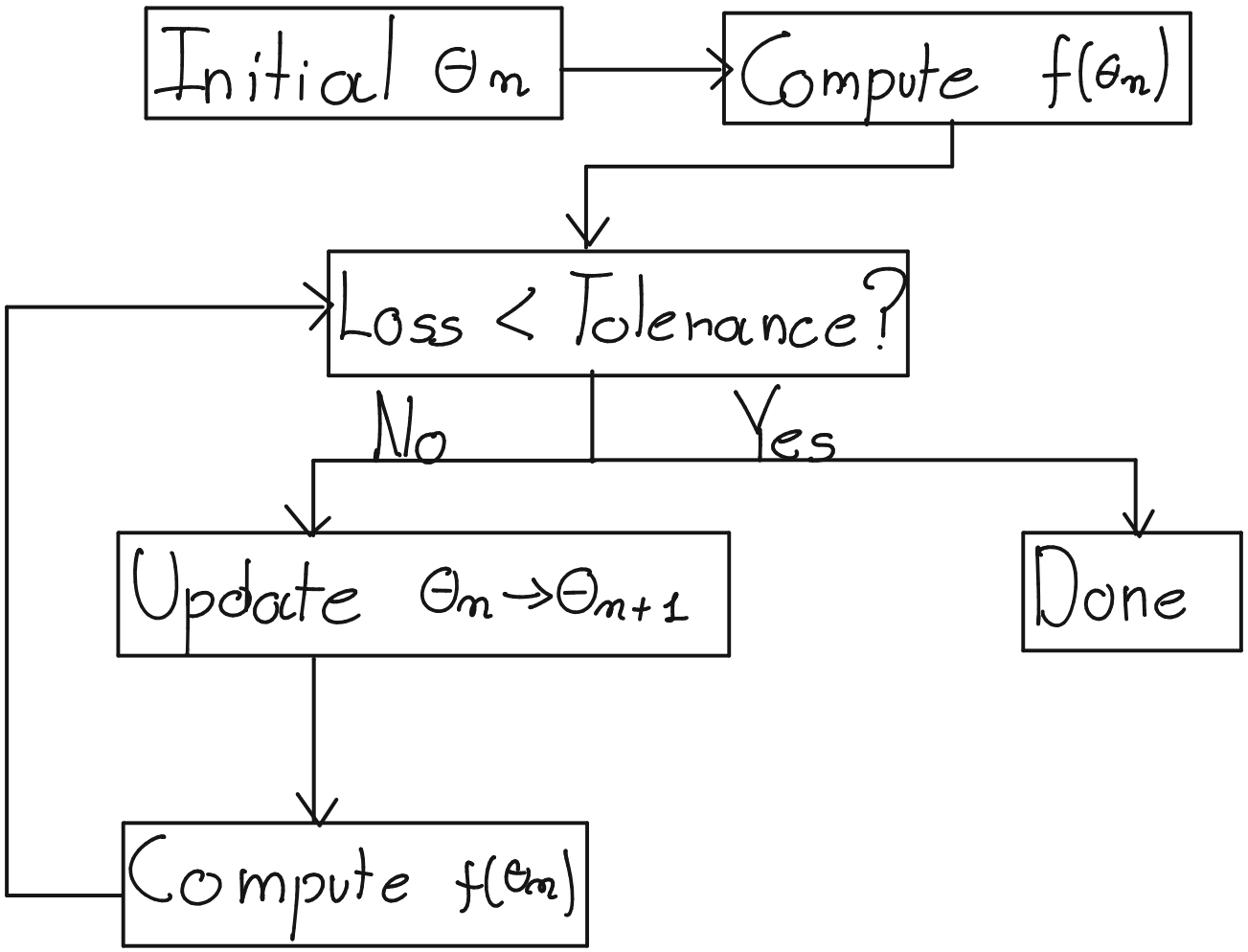
$$U(x_j^{(i)}) = \begin{bmatrix} \cos(x_j^{(i)}) & -\sin(x_j^{(i)}) \\ \sin(x_j^{(i)}) & \cos(x_j^{(i)}) \end{bmatrix}$$

Note that $U(x_j^{(i)}) = RY(2x_j^{(i)})$

This idea can be generalized to encode more features per qubit

Training PQCs

PQCs can be trained to learn an arbitrary function from the data. Mathematically this is equivalent to finding the minimization of a cost function $f(\vec{\theta})$, also known as a loss or objective function



The stepping of θ_n can be done via various algorithms. Gradient based

methods are the most popular

Gradient descent

Given $f(\vec{\theta})$ and $\vec{\nabla} f(\vec{\theta})$, the parameter evolution is given by

$$\boxed{\vec{\theta}_{n+1} = \vec{\theta}_n - \eta \vec{\nabla} f(\vec{\theta}_n)}$$

Here η is called learning rate. It is an hyper parameter: a parameter used to control the algorithim

Methods for computing gradients

1) Finite differences: Works regardless of function complexity, but may give wrong results if the function is noisy

2) Analytic Gradients: For simple circuits consisting only of Pauli Matrices, one can apply the shift parameter rule

$$\frac{\partial f}{\partial \theta_i} = \frac{f(\vec{\theta} + \pi/2 \vec{e}_i) - f(\vec{\theta} - \pi/2 \vec{e}_i)}{2}$$

3) Natura gradients: This method

Takes into account that the geometry of the parameten space is not uniform and uses a metric connection

$$\vec{\theta}_{n+1} = \vec{\theta}_n - \eta\, g^{-1}(\vec{\theta})\, \vec{\nabla} f(\vec{\theta}_n)$$

Where

$$g_{ij}(\vec{\theta}) = Re\left\{ \left\langle \frac{\partial \psi}{\partial \theta_i} \middle| \frac{\partial \psi}{\partial \theta_j} \right\rangle - \left\langle \frac{\partial \psi}{\partial \theta_i} \middle| \psi \right\rangle \left\langle \psi \middle| \frac{\partial \psi}{\partial \theta_j} \right\rangle \right\}$$

and $|\psi(\vec{\theta})\rangle$ is the parametrized state

$$|\psi(\vec{\theta})\rangle = U(\vec{\theta})|0 \dots 0\rangle$$

4) Simultaneous Perturbation Stochastic Approximation (SPSA)

Approximates the gradient by perturbing all dimentions at once by a random parameter. This works because we only care about convergence and not precise gradient values.

This method is $O(1)$ in the size of the parameter space and can be

extended to use the natural gradient given nise to the QNSPSA algorithim

In practice, today SPSA is used with an exponential learning rate to accelerate convergence

Supervised Learning

This is the task of learning a function from labeled training data from a set of training examples. Performance is calculated using a set of testing examples

Supervised learning can be further separated into Classification and Regression

1) Classification: Assigns data into categories

2) Regression: Understand the relationship between dependent and inde pendent variables

Feature Maps (QFMs)

A quantum feature map maps a feature vector $\vec{x}$ to a quantum Hilbert space The QFM des this using a PQC

Constructing QFMs that are hard to simulate classically is key in obtaning quantum advantage

There is a family of QFMs that are conjectured to be hard to simulate classically known as Pauli Feature Map. which consists in a sequence of Hadamard gates interleaved with entangling gates
A map of deapth d is given by

$$\mathcal{U}_\Phi(\vec{x}) = \prod_d U_\Phi(\vec{x})\, H^{\otimes n}$$

where

$$U_\Phi(\vec{x}) = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{k \in S} P_i \right)$$

The $U_\Phi(\vec{x})$ blocks are the ones encoding the classical data

In the entangling blocks, $U_\Phi(\vec{x})$,

$$P_i \in \{\hat{I}, \hat{X}, \hat{Y}, \hat{Z}\}$$

denots the Pauli matrices, the index S

$$S \in \left\{ \binom{n}{k} \text{ combinations}, \; k = 1, \ldots, n \right\}$$

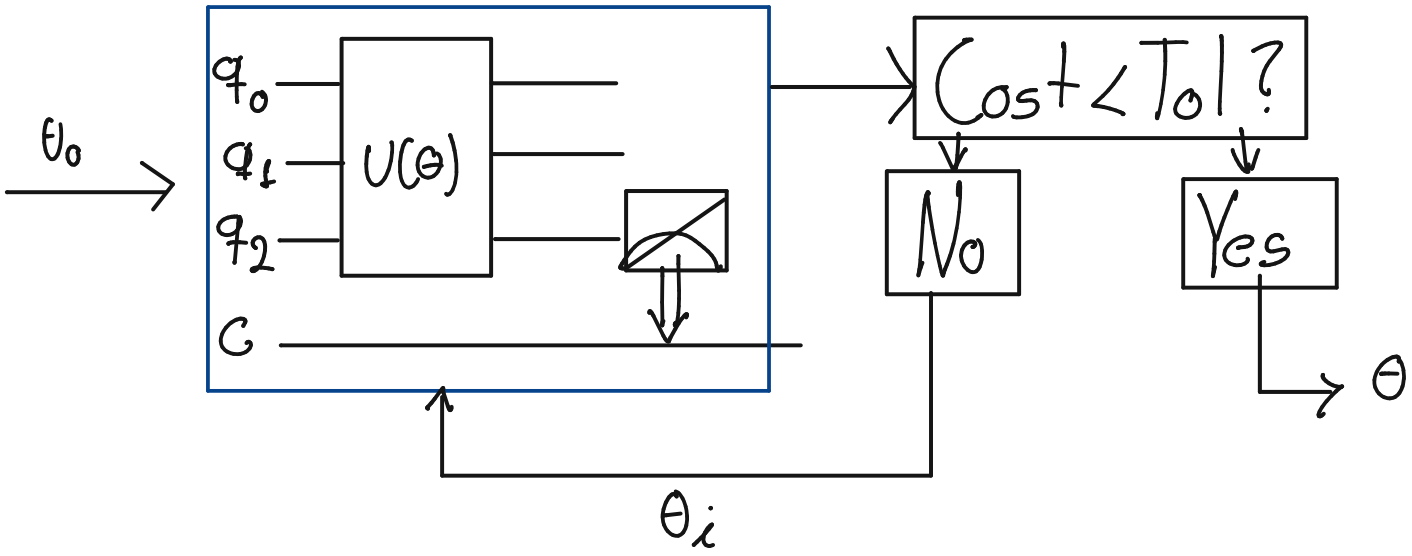
describes the conectivity between different qubits or data points, and by default the data mapping function $\phi_S(\vec{x})$ is

$$\phi_S : \vec{x} \mapsto \begin{cases} x_i & \text{if } S = \{i\} \\ (\pi - x_i)(\pi - x_j) & \text{if } S = \{i, j\} \end{cases}$$

when $k = 2$, $P_0 = \hat{Z}$, $P_1 = \hat{Z}\hat{Z}$, this is called the ZZ Feature Map

Variational Quantum Regression

In this approach, we use the QC to test an ansatz



Model problem:

Let us imagine that we want to fit

$$y = ax + b$$

We have the dataset

$$y = (y_0, \; y_1)$$

$$x = (x_0, \; x_1)$$

We construct the $|x\rangle$ state with amplitude encoding

$$|x\rangle = A\left(x_0|0\rangle + x_1|1\rangle\right) \; ; \; A = 1/\sqrt{x_0^2 + x_1^2}$$

We construct the ansatz state $|\psi\rangle$

$$|\psi\rangle = U(a, b)\,|x\rangle$$

We seek parameters p_1 and p_2 such that

$$|\psi(p_1, p_2)\rangle = U(p_1, p_2)|x\rangle = |y\rangle$$

Where $|y\rangle$ is our amplitude encoded tanget state

$$|y\rangle = B(y_0|0\rangle + y_1|1\rangle)\,;\; B = 1/\sqrt{y_0^2 + y_1^2}$$

Introduce the Projection Hamiltonian

$$H_p = \mathbb{I} - |y\rangle\langle y|$$

And the Cost Function

$$C = \langle\psi|H_p|\psi\rangle = \langle\psi|(\mathbb{I} - |y\rangle\langle y|)|\psi\rangle$$

$$= \langle\psi|\psi\rangle - \langle\psi|y\rangle\langle y|\psi\rangle$$

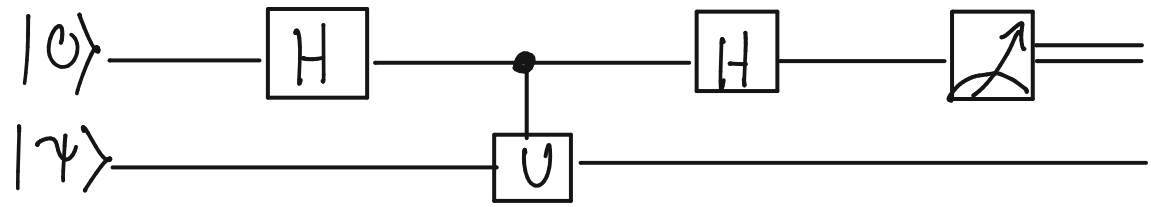
$$\Rightarrow C = \langle\psi|\psi\rangle - |\langle y|\psi\rangle|^2$$

Normalizing with $\langle\psi|\psi\rangle$, we get

$$\hat{C} = 1 - \frac{|\langle y|\psi\rangle|^2}{\langle\psi|\psi\rangle}$$

Hadamard Test

Given a unitary U and state $|\psi\rangle$ the circuit



Then

$P_{|0\rangle}(0) = 1/2[1 + Re(\langle\psi|U|\psi\rangle)$ and $P_{|0\rangle}(1) = 1/2[1 - Re(\langle\psi|U|\psi\rangle)]$

$$\langle\psi|\psi\rangle = \langle x|U^\dagger U|x\rangle = \langle x|x\rangle = 1$$

$$\langle y|\psi\rangle = \langle y|U|x\rangle$$

$$= AB(y_0\langle 0| + y_1\langle 1|)U(x_0|0\rangle + x_1|1\rangle)$$

$$= AB(y_0\langle 0| + y_1\langle 1|)(x_0 U|0\rangle + x_1 U|1\rangle)$$

$$= AB\left[y_0 x_0\langle 0|U|0\rangle + y_0 x_1\langle 0|U|1\rangle + y_1 x_0\langle 1|U|0\rangle + y_1 x_1\langle 1|U|1\rangle\right]$$

$$\Rightarrow \boxed{\langle y|\psi\rangle = AB\sum_{i=0}^{1}\sum_{j=0}^{1} y_i x_j \langle i|U|j\rangle}$$

If we initialize the circuit with $|x\rangle$

from $|0\rangle$, that is

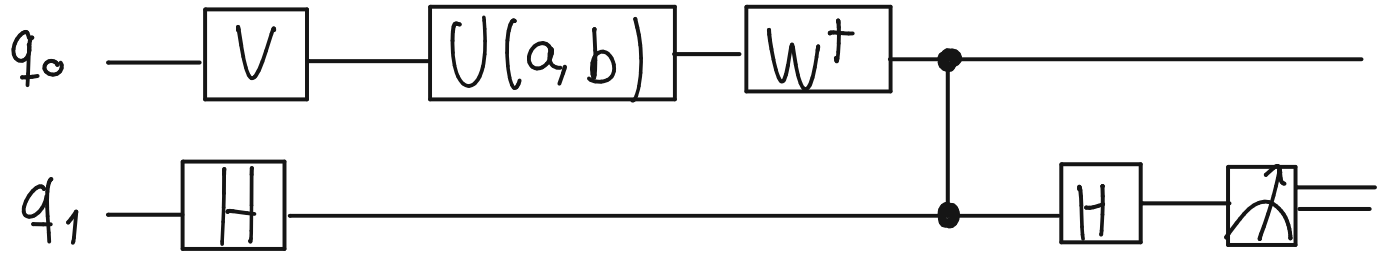
$$|\psi\rangle = U(a,b)\,V|0\rangle \quad ; \quad V|0\rangle = |x\rangle$$

and $|y\rangle = W|0\rangle$, we have

$$\langle y|\psi\rangle = \langle 0|W^\dagger U V|0\rangle,$$

which can be found via Hadamard test

The final $\hat{C}$ computing circuit is then



$$P_{q_1}(0) = \frac{1}{2}\left[1 + Re(\langle 0|W^\dagger U V|0\rangle)\right]$$

$$P_{q_1}(1) = \frac{1}{2}\left[1 - Re(\langle 0|W^\dagger U V|0\rangle)\right]$$

$$\Rightarrow Re(\langle 0|W^\dagger U V|0\rangle) = P(0) - P(1)$$

Generalization for higher dimentional inputs

Let $\vec{y} = (y_0, \dots, y_N)$

Where $\log_2(N+1) \in \mathbb{N}$ be the target state, encoded as

$$|\vec{y}\rangle = A \sum_{i=0}^{N} y_i |i_b\rangle$$

where $|i_b\rangle$ is the binary repr. of the integer i and

$$A = \left(\sum_{i=0}^{N} y_i^2 \right)^{-1/2}$$

Let $S(\vec{y})$ be a state prep. unitary such that

$$S(\vec{y})|\vec{0}\rangle = |\vec{y}\rangle$$

Let $\vec{x} = (x_0, \dots, x_N)$ be a input state, obtained by applying some function f, that is

$$\vec{y} = \vec{f}(\vec{x})$$

encoded as

$$|\vec{x}\rangle = B \sum_{i=0}^{N} x_i \, |i_b\rangle$$

where

$$B = \left[\sum_{i=0}^{N} x_i^2 \right]^{-1/2}$$

Let $U(\vec{x})$ be a state prep. unitary such that

$$U(\vec{x})|\vec{0}\rangle = |\vec{x}\rangle$$

We will model the target function $f(\vec{x})$ as

$$|\vec{x}(\vec{a})\rangle = V(\vec{a})|\vec{x}\rangle$$

Lets introduce the error tolerance

$$\varepsilon = \frac{1}{2} \, Tr\left[|\vec{y}\rangle\langle\vec{y}| - |\vec{x}(\vec{a})\rangle\langle\vec{x}(\vec{a})| \right]$$

for a given set of a values

Let

$$H_G = \mathbb{1} - |\vec{y}\rangle\langle y|$$

We then define the cost functions

$$C_G = \langle \vec{x}(\vec{\alpha}) | H_G | \vec{x}(\vec{\alpha}) \rangle$$

$$= \langle \vec{x}(\vec{\alpha}) | (\mathbb{1} - |\vec{y}\rangle\langle\vec{y}|) | \vec{x}(\vec{\alpha}) \rangle$$

$$= \langle \vec{x}(\vec{\alpha}) | \left[|\vec{x}(\vec{\alpha})\rangle - |\vec{y}\rangle\langle\vec{y}|\vec{x}(\alpha)\rangle \right]$$

$$= \langle \vec{x}(\vec{\alpha}) | \vec{x}(\vec{\alpha}) \rangle - \langle \vec{x}(\vec{\alpha}) | \vec{y}\rangle\langle\vec{y}|\vec{x}(\vec{\alpha})\rangle$$

$$= \langle \vec{x} | V^\dagger(\vec{\alpha}) V(\vec{\alpha}) | \vec{x} \rangle - \langle \vec{x}(\vec{\alpha}) | \vec{y}\rangle\langle\vec{y}|\vec{x}(\vec{\alpha})\rangle$$

$$= \underbrace{\langle \vec{x} | \vec{x} \rangle}_{=1} - \underbrace{\langle \vec{x} | V^\dagger(\vec{\alpha}) | \vec{y}\rangle\langle\vec{y}|V(\vec{\alpha})|\vec{x}\rangle}_{= |\langle \vec{y}|V(\vec{\alpha})|\vec{x}\rangle|^2}$$

$$\Rightarrow \boxed{C_G = 1 - |\langle \vec{0}|S^\dagger(\vec{y}) V(\vec{\alpha}) U(\vec{x})|\vec{0}\rangle|^2}$$

The quantity in brackets can be computed via Hadamard Test

Trace distance as cost

$$T(|\vec{y}\rangle, |\vec{x}(\vec{\alpha})\rangle) = \varepsilon = \sqrt{1 - |\langle \vec{y}|\vec{x}(\vec{\alpha})\rangle|^2}$$

$$\Rightarrow \boxed{\varepsilon = \left[1 - |\langle\vec{0}|S^\dagger(\vec{y}) V(\vec{\alpha}) U(\vec{x})|\vec{0}\rangle|\right]^{1/2}}$$

Angle encoding

We want to angle encode in the hopes that this fits general data

1 data point

Let $\vec{x} = (x_0)$ and $\vec{y} = (y_0)$ be the input and target states. Angle encoding $\vec{x}$ yields

$$|\vec{x}\rangle = R_y(2x_0)|0\rangle$$

$$= \left[\begin{array}{l} \cos(x_0)|0\rangle\langle 0| - \sin(x_0)|0\rangle\langle 1| + \sin(x_0)|1\rangle\langle 0| \\ + \cos(x_0)|1\rangle\langle 1| \end{array}\right] |0\rangle$$

$$\Rightarrow \boxed{|\vec{x}\rangle = \cos(x_0)|0\rangle + \sin(x_0)|1\rangle}$$

How to recover x_0?

$$\vec{z} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

$$\vec{z}\,|\vec{x}\rangle = \cos(x_0)|0\rangle - \sin(x_0)|1\rangle$$

$$\langle\vec{x}|\vec{z}\,|\vec{x}\rangle = \left[\langle 0|\cos(x_0) + \langle 1|\sin(x_0)\right]\left[\cos(x_0)|0\rangle - \sin(x_0)|1\rangle\right]$$

$$= \cos^2(x_0) - \operatorname{sen}^2(x_0) = \cos(2x_0)$$

$$\Rightarrow \quad \langle \vec{x} | \hat{Z} | \vec{x} \rangle = \cos(2x_0)$$

$$\hat{X} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

$$\hat{X}|\vec{x}\rangle = \operatorname{sen}(x_0)|0\rangle + \cos(x_0)|1\rangle$$

$$\langle \vec{x} | \hat{X} | \vec{x} \rangle = 2 \operatorname{sen}(x_0)\cos(x_0) = \operatorname{sen}(2x_0)$$

$$\hat{Y} = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$$

$$\hat{Y}|\vec{x}\rangle = -i \operatorname{sen}(x_0)|0\rangle + i \cos(x_0)|1\rangle$$

$$\langle \vec{x} | \hat{Y} | \vec{x} \rangle = 0$$

So, we can recover x_0 using

$$\langle \vec{x} | \hat{Z} | \vec{x} \rangle = \cos(2x_0) \Rightarrow \boxed{x_0 = \frac{1}{2}\cos^{-1}\left(\langle \vec{x} | \hat{Z} | \vec{x} \rangle\right)}$$

Or

$$\boxed{x_0 = \frac{1}{2}\operatorname{sen}^{-1}\left(\langle \vec{x} | \hat{Y} | \vec{x} \rangle\right)}$$

The cost function:

Let $|\vec{y}\rangle = R_y(y_0)|0\rangle$

$$\Rightarrow |\vec{y}\rangle = \cos(y_0)|0\rangle + \operatorname{sen}(y_0)|1\rangle$$

Let $|\vec{x}(\theta)\rangle = U(\theta)|\vec{x}\rangle$

$$\langle \vec{y} | \vec{x}(\theta)\rangle = \langle \vec{y} | U(\theta) | \vec{x}\rangle$$

$$= \langle \vec{y} | \left(\cos(x_0) U(\theta)|0\rangle + \operatorname{sen}(x_0) U(\theta)|1\rangle \right)$$

$$= \cos(x_0)\langle \vec{y}|U(\theta)|0\rangle + \operatorname{sen}(x_0)\langle \vec{y}|U(\theta)|1\rangle$$

$$\langle \vec{y}|U(\theta)|0\rangle = \cos(y_0)\langle 0|U(\theta)|0\rangle + \operatorname{sen}(y_0)\langle 1|U(\theta)|0\rangle$$

$$\langle \vec{y}|U(\theta)|1\rangle = \cos(y_0)\langle 0|U(\theta)|1\rangle + \operatorname{sen}(y_0)\langle 1|U(\theta)|1\rangle$$

Thus

$$\begin{aligned}\langle \vec{y}|\vec{x}(\theta)\rangle = {} & \cos(x_0)\cos(y_0)\langle 0|U(\theta)|0\rangle \\ & + \cos(x_0)\operatorname{sen}(y_0)\langle 1|U(\theta)|0\rangle \\ & + \operatorname{sen}(x_0)\cos(y_0)\langle 0|U(\theta)|1\rangle \\ & + \operatorname{sen}(x_0)\operatorname{sen}(y_0)\langle 1|U(\theta)|1\rangle\end{aligned}$$