QML-Mod4-From Quantum Computing to QML

Riccardo Marega

March 2025

Indice

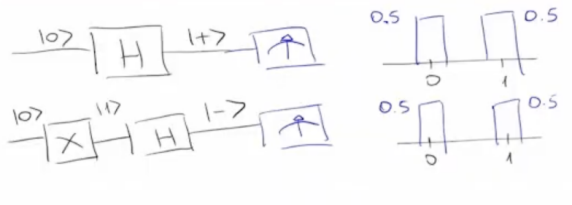
1 From Quantum Computing to QML-23/05/2025

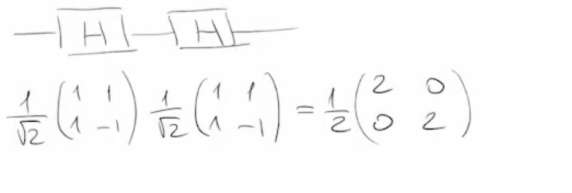
1.1 Hadamard Gate (H)

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

$$\begin{cases} H \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ H \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{cases}$$

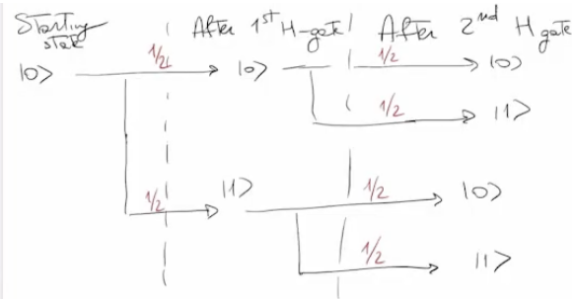
Two Hadamard gates placed one after the other compose the identity gate.



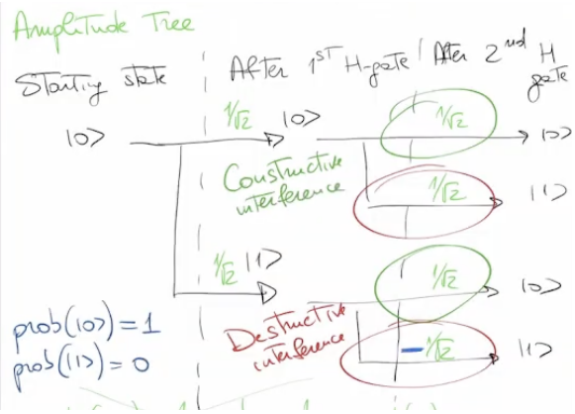


$= \textit{Identity matrix}$

Using amplitude tree we get that, starting from $|0\rangle$, after applying a Hadamard gate we have a probability of 0.5 of getting either $|1\rangle$ or $|0\rangle$. Suppose now applying an Hadamard gate what we get is an overall probability of finding half of the time $|0\rangle$ and the other half $|1\rangle$ but that is not what we expected given the fact that we told that applying twice in a row an Hadamard gate corresponds to obtain the identity gate.



The problem is that we have to be careful when applying the Hadamard gate to $|1\rangle$ because the probability of finding $|1\rangle$ is not $\frac{1}{2}$ but it is $|\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}|^2$ (first all the amplitudes are summed than the modulus square is computed). We talk in this case of destructive interference while for $|0\rangle$ we have constructive interference.



Probability is fully characterized by its magnitude while the Amplitude has also a Phase factor.

1.2 Composite systems

$$H_{1+\ldots+n} = H_1 \otimes \ldots \otimes H_n$$

1.2.1 Tensor Product

$$\psi_1 \otimes \psi_2 = \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \otimes \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \alpha_1\,\alpha_2 \\ \alpha_1\,\beta_2 \\ \beta_1\,\alpha_2 \\ \beta_1\,\beta_2 \end{pmatrix}.$$

Definition If $|\psi\rangle = (2^n) = \begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix} \otimes ... \otimes \begin{pmatrix}\alpha_n\\ \beta_n\end{pmatrix}$ than it is called a **product state**.

$$\begin{cases}\textbf{product states} & 2n\\ \textbf{entangled states} & 2^n\end{cases}$$

We start by computing

$$|0\rangle_1 \otimes |0\rangle_2 = |00\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

in the same way

$$|0\rangle \otimes |1\rangle = |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

and so on for the other combinations.
In general a two qubit state has the form

$$\begin{pmatrix}a_{00}\\a_{01}\\a_{10}\\a_{11}\end{pmatrix}$$

where $a_{ij} \in C$. We recall that is possible to expand it on a given basis

$$|\psi\rangle = a_{00}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + a_{01}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} + a_{10}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} + a_{11}\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Examples: Can we decomposed the following states in a tensor product?

$$|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |0+\rangle$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\1\\1\end{pmatrix} = |1+\rangle$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = ?$$

In order to solve that we have to solve for

$$\begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix}\otimes\begin{pmatrix}\alpha_2\\ \beta_2\end{pmatrix}$$

applying the conditions given from the problem. What we find is that it is impossible to find $\alpha_1, \alpha_2, \beta_1$ and β_2 able to solve for the previous problem: the state is said to be entangled.
Another example is

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 0\\ 0\\ -1\end{pmatrix}=\frac{1}{\sqrt{2}}(|00\rangle-|11\rangle)$$

and that is also an entangled state, moreover, being the relative phase maximum ($e^{i\phi}=-1$) that is called **maximum entangled state**.
We can compute the remaining combinations and we'll find that both of them are also entangled states.
Maximum entangled states (Bell states) compose an orthonormal basis

$$\begin{cases}|\beta_{00}\rangle\\ |\beta_{10}\rangle\\ |\beta_{01}\rangle\\ |\beta_{11}\rangle\end{cases}$$

1.3 How gates are composed?

$$1\otimes 1=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}=\begin{pmatrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} & 0\\ 0 & \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\end{pmatrix}$$

In the same way

$$X\otimes 1=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\otimes\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} & 0\end{pmatrix}$$

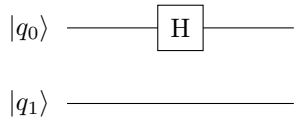
and so on for all other possible gate combinations.
Indeed, we continue by considering the case:

$$H=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\quad;\quad I=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

We get:

$$H\otimes I=\frac{1}{\sqrt{2}}\begin{bmatrix}1\cdot I & 1\cdot I\\ 1\cdot I & -1\cdot I\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}I & I\\ I & -I\end{bmatrix}$$

$$H \otimes I = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$



For n qubits on which an Hadamard transformation is applied we have:

$$\frac{1}{\sqrt{2}} \sum_{j=0}^{2^n-1} |j\rangle$$

1.3.1 Two-qubit gate

We want a gate able to interact with two qubits, transforming them at the same time. The most general n-qubit gate is described by $2^n \times 2^n$ unitary matrix.

C-U (Controlled U) gate in this case we have a control qubit and a target qubit. Mathematically this gate is represented by

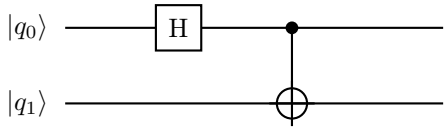
$$|xy\rangle \rightarrow |x\rangle U^x |y\rangle$$

$$\begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 0 \\ 0 & U \end{pmatrix}$$

The C-NOT gate is the controlled NOT gate and it is composed assigning U = X. We can now compute as excercise:

$$U_{CNOT} \times (X \otimes X) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

1.3.2 Hadamard and CNOT gate



$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$H \otimes I_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$U_{\text{total}} = \text{CNOT} \left(H \otimes I_2 \right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

2 From Quantum Computing to QML part 2 -24/05/2025

Suppose being in the state $\beta_{00} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and performing a measurement: the probability of measuring $|00\rangle$ is the same of $|11\rangle$, which is $\frac{1}{2}$.

2.1 Bell Measurement

To perform a Bell measurement we apply to the state β first the CNOT gate followed by the Hadamard gate. What we get is:

$$\beta_{00} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \xrightarrow{CNOT+H} |00\rangle$$

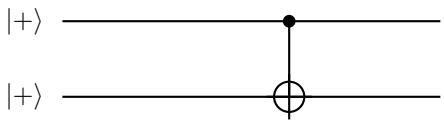
$$\beta_{01} = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \xrightarrow{CNOT+H} |01\rangle$$

$$\beta_{10} = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \xrightarrow{CNOT+H} |10\rangle$$

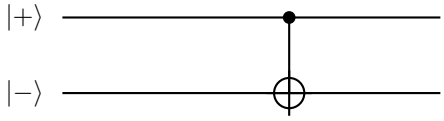
$$\beta_{11} = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \xrightarrow{CNOT+H} |11\rangle$$

2.2 CNOT and $|\pm\rangle$ states

Both qubits are in a superposition.



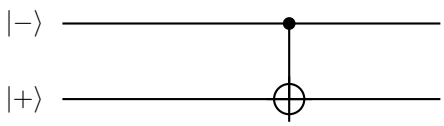
$$|++\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \xrightarrow{CNOT} \frac{1}{2}(|00\rangle + |01\rangle + |11\rangle + |10\rangle) = |++\rangle$$



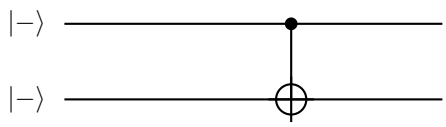
$$|+-\rangle \xrightarrow{CNOT} \frac{1}{2}(|00\rangle - |01\rangle + |11\rangle - |10\rangle) = \frac{1}{2}|[0\rangle \otimes (|0\rangle - |1\rangle) + |1\rangle \otimes (|0\rangle - |1\rangle)]$$

$$= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |--\rangle.$$

This is called phase kick-back: the target remains the same but is the controller who is changed (also called π shift).



$$|-+\rangle \xrightarrow{CNOT} \frac{1}{2}(|00\rangle + |01\rangle - |11\rangle - |10\rangle) = |-+\rangle.$$

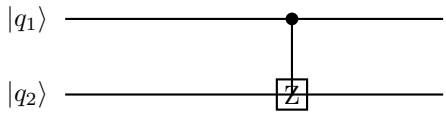


$$|--\rangle \xrightarrow{CNOT} \frac{1}{2}(|00\rangle - |01\rangle - |11\rangle + |10\rangle) = \frac{1}{\sqrt{2}}|0\rangle \otimes (|0\rangle - |1\rangle) + \frac{1}{\sqrt{2}}|1\rangle(|0\rangle - |1\rangle) = |+-\rangle.$$

Once again we are affecting the control.

2.3 CZ

The CZ gate is a gate that, as the CNOT, uses a control gate on a qubit and applies a Z gate on the other qubit.



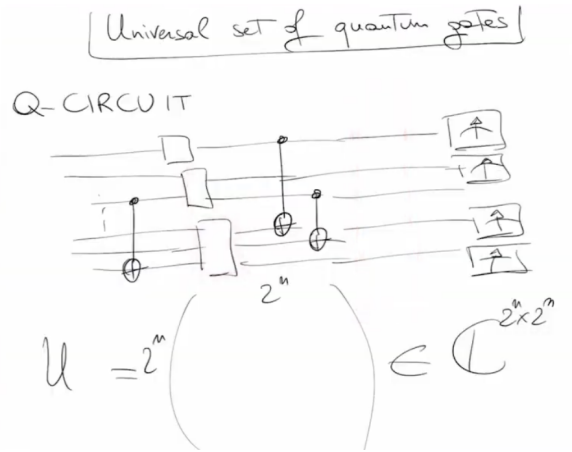
$$\begin{cases} |00\rangle \to |00\rangle \\ |01\rangle \to |01\rangle \\ |10\rangle \to |10\rangle \\ |11\rangle \to -|11\rangle \end{cases}$$

Symmetric role of control and target qubits.
Note that applying $\xrightarrow{H-Z-H} \equiv \xrightarrow{X}$ as also $\xrightarrow{H-X-H} \equiv \xrightarrow{Z}$. It's possible proving that by simply computing all the matrix multiplications.
The most general type of gate is a unitary matrix $\in C^{2^n \times 2^n}$. It's possible to demonstrate that this unitary matrix can be decomposed thanks to the tensor product decomposition.

Theorem:

$$\{CX, X, R_Z(\sigma), H\}$$

is called the universal set of quantum gates and their combination (tensor product combination) permits to construct an infinite type of gates.
The operation of expanding a gate on the universal set is called transpiling.



Theorem Given a gate characterized by an error ϵ and a quantum circuit composed on N gates applied to n qubits, than

$$\epsilon_{tot} \leq N\epsilon.$$

2.4 Quantum algorithms

2.4.1 Superdense coding

Suppose having A and B sharing the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

A takes the first qubit while B takes the second one. Suppose now A wants to send to B two bits (00,01,10 or 11). If A wants to send 00 she won't have to do any transformation to her qubit, if she wants to send 01 she will have to apply a X gate, if she wants to send 10 she will have to apply a Z gate and to send 11 she will have to apply firs a X gate followed by a Z gate. For each of these cases she will also have to apply a CNOT followed by an Hadamard gate on the first qubit. If now B performs a measurement on the two qubits, he will find the bits A intended to send:

Superdense Coding

Alice $\Rightarrow$ Bob

$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(|00\rangle + |11\rangle\Big) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}\Big(|00\rangle + |10\rangle\Big) \xrightarrow{H} H|+\rangle|0\rangle = |00\rangle$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(|10\rangle + |01\rangle\Big) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}\Big(|11\rangle + |01\rangle\Big) \xrightarrow{H} H|+\rangle|1\rangle = |01\rangle$$

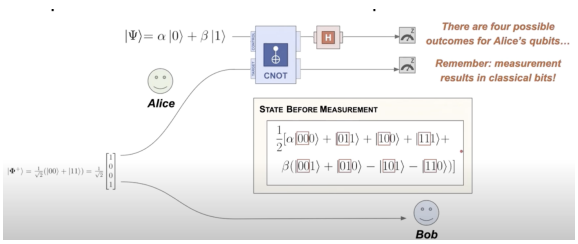
$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(|00\rangle - |11\rangle\Big) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}\Big(|00\rangle - |10\rangle\Big) \xrightarrow{H} H|-\rangle|0\rangle = |10\rangle$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(|10\rangle - |01\rangle\Big) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}\Big(|11\rangle - |01\rangle\Big) \xrightarrow{H} H|-\rangle|1\rangle = |11\rangle$$

What happened is that we sent two bits of classical information just sending a single qubit.

2.4.2 Quantum Teleportation

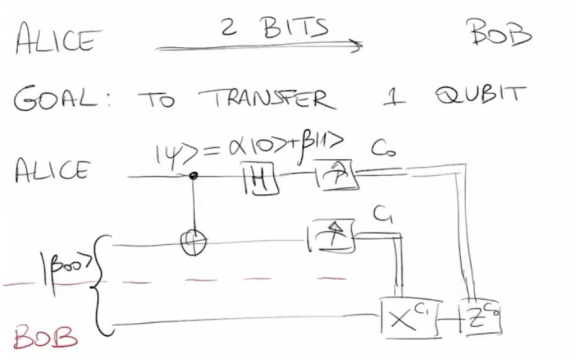
In order to perform quantum teleportation we'll need two entangled qubits (each party has one half of the entangled pair), a 'message' qubit that will be sent from one point to another and a classical communication line between both parties for the transmission of two classical bits. Suppose that after entangling a two qubits A (which we suppose having another qubit on her own) and B both take a qubit. What happens now is that A applies a CNOT qubit using her previous qubit (message qubit) as the control qubit and the second qubit (taken from the entagled pair) as the target. After that, A applies an Hadamard gate on her message qubit. After that she performs a measurement and the possible results she can measure are the one the follows:



what we now see is that:

$$\begin{cases} \text{A measurement: } 00 \rightarrow \text{B state: } \alpha|0\rangle + \beta|1\rangle \\ \text{A measurement: } 01 \rightarrow \text{B state: } \alpha|1\rangle + \beta|0\rangle \\ \text{A measurement: } 10 \rightarrow \text{B state: } \alpha|0\rangle - \beta|1\rangle \\ \text{A measurement: } 11 \rightarrow \text{B state: } \alpha|1\rangle - \beta|1\rangle \end{cases}$$

A now sends her two classical bits. B, in order to recover $|\psi\rangle$, has to apply a Z gate if the first bit is 1 and has to apply an X gate if the second bit is 1.



What we are witnessing now is the no-cloning theorem.

3 Quantum algorithms -30/05/2025

3.1 Quantum parallelism

We start by considering two Hadamard gates: $|00\rangle \xrightarrow{HH} |++\rangle$, $|01\rangle \xrightarrow{HH} |+-\rangle$ and $|11\rangle \xrightarrow{HH} |--\rangle$. In general for a single Hadamard gate we can write

$$H|x\rangle = \frac{1}{\sqrt{2}}[|0\rangle + (-1)^x|1\rangle]$$

where x = 0,1. We can apply now n Hadamard gates on n qubits

$$|x_1x_2...x_n\rangle \xrightarrow{H_1...H_n} \frac{1}{2^{\frac{n}{2}}}\sum_y (-1)^{xy}|y\rangle$$

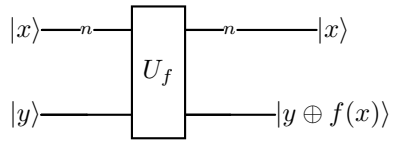
with y containing all n-bit strings. xy is called bitwise product and is defined as

$$xy = x_1y_1 + ... + x_ny_n.$$

We note how this works like a XOR gate.
Suppose now having a function $f(x) : \{0,1\}^n \to \{0,1\}$: $|x\rangle \to |f(x)\rangle$ where x represents a generic n-bit string. We prepare a quantum state $|x\rangle$ representing the input and aim to compute $f(x)$ into a second qubit:

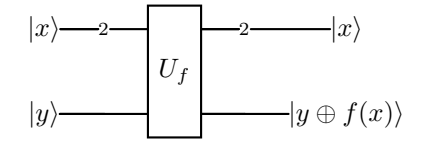
$$|x\rangle \to |f(x)\rangle.$$

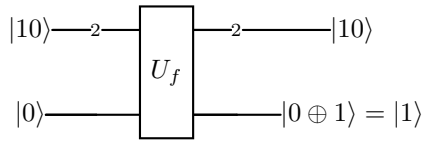
Lets define $U_f : |x\rangle|y\rangle \to |x\rangle|y + f(x)\rangle$.

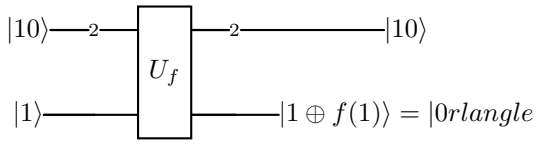


where $\oplus$ represents the XOR gate (that to ensure that the transformation defined by U_f is unitary i.e. reversible).

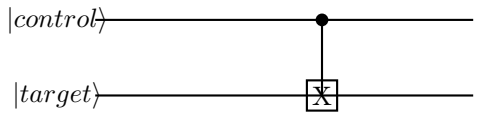
What follows is an example where n = 2.



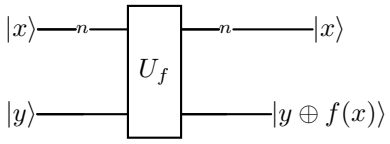




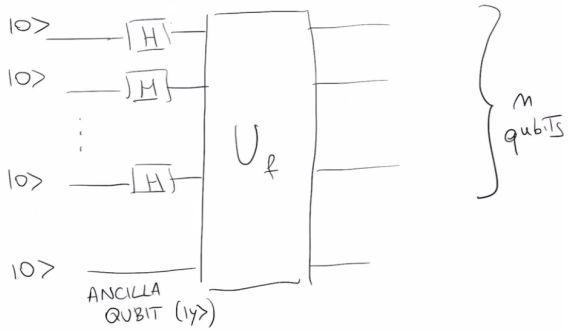
When f(x) = 1, U_f is a not-gate on $|y\rangle$, while when f(x) = 0 $U_f = 1$.
U_f is like a C-X gate on y but with the control on f(x) instead of x (as in C-X):



that is switched on if control c = 1, while



the X-gate acts on y if f(x) = 1.
Lets now consider the following circuit



$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$$

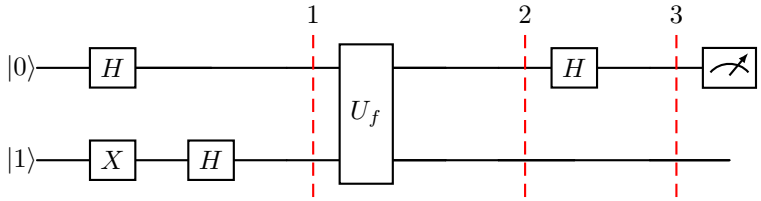
where x is all possible bit-string. Overall the state can be written as

$$\Rightarrow \frac{1}{\sqrt{2^n}} \sum_x |x\rangle f(x)$$

This is called quantum parallelism.
If we make now n measurements we loose these specific combinations.

3.2 Deutsch-Josza algorithm

Given $f : \{0,1\}^n \to \{0,1\}$ where f(x) can be either constant or balanced ($f(x) = 0$ for half values of x and 1 otherwise). The goal of this algorithm is to say whether it is constant or balanced. The question is: how many function evaluations do we need classically? $2^{n-1} + 1 \to O(2^n)$. The best classical algorithm has a computational complexity of $O(2^n)$. What we are going to see is that using quantum computing we'll get a computational power of $O(1)$ (just one evaluation), we'll witness what is called quantum exponential speedup.



At checkpoint 1 we have $|\psi_1\rangle = |+-\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2}\sum_{x=0}^{1} |0\rangle(|0\rangle - |1\rangle)$.
At checkpoint 2 we have $|\psi_2\rangle = \frac{1}{2}\sum_{x=0}^{1} |x\rangle \otimes (|0 XOR f(0)\rangle - |1 XOR f(1)\rangle) = \frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$.
At checkpoint 3 we have $|\psi_3\rangle = \frac{1}{2}\sum_x (-1)^{f(x)} \frac{1}{2^{\frac{n}{2}}} \sum_y (-1)^{f(x)}|y\rangle(|0\rangle - |1\rangle) =$

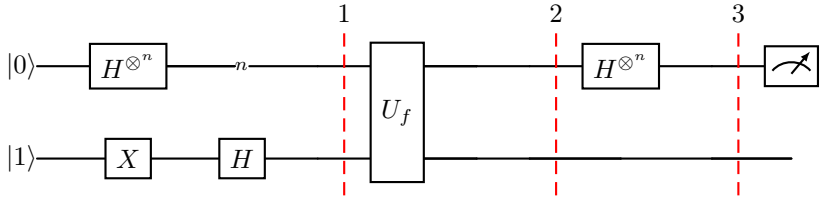
$$= \frac{1}{\sqrt{2^3}} \sum_x (-1)^{f(x)} \sum_y (-1)^{f(x)}|y\rangle|-\rangle$$

What's the probability of finding 0 measuring the first qubit?

$$prob(|0\rangle_1) = |\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}|^2$$

$$\begin{cases} 0 & \text{if f(x) is balanced} \\ 1 & \text{if f(x) is constant} \end{cases}$$

n-bit case $f(x) : \{0,1\}^n \to \{0,1\}$.



$$|\psi_1\rangle = \frac{1}{2^{\frac{n+1}{2}}} \sum_{x=0}^{2^n-1} |x\rangle(|0\rangle - |1\rangle) = \frac{1}{2^{\frac{n+1}{2}}} \sum_x |x\rangle|-\rangle$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_x (-1)^{f(x)}|x\rangle|-\rangle$$

$$\underbrace{|\psi_3\rangle = \frac{1}{2^n}\sum_x (-1)^{f(x)}\sum_y (-1)^{xy*}|y\rangle|-\rangle}_{\text{n-qubit state}}$$

Recalling the Hadamard transformation on a 1-qubit basis stete:

$$H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{xy}|y\rangle,$$

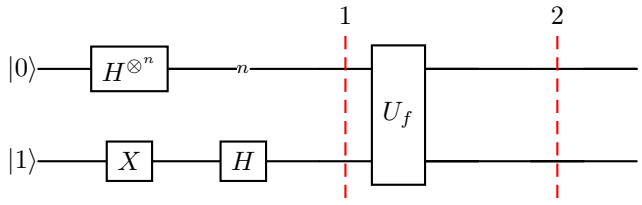
we substitute:

$$|\psi_3\rangle = \frac{1}{2\sqrt{2}}\sum_{x=0}^{1}\sum_{y=0}^{1}(-1)^{f(x)}(-1)^{xy}|y\rangle\otimes(|0\rangle - |1\rangle).$$

Grouping the summations we have:

$$|\psi_3\rangle = \frac{1}{\sqrt{8}}\sum_{Y=0}^{1}(\sum_{x=0}^{1}(-1)^{f(x)+xy})|y\rangle\otimes(|0\rangle - |1\rangle).$$

This form allows us to observe constructive or destructive interference depending on whether f is constant or balanced. The probability of $|0\rangle^{\otimes^n} = |\frac{1}{2^n}\sum_x^{2^n-1}(-1)^{f(x)}|^2$.
The solution is that 1 run of the quantum circuit (f-evaluation) followed by a final measurement gives that if all bits are 0 than the function is constant otherwise if at least one bit is 1 than the function is balanced.
We note that ψ_2 is a state composed by a linear combination where, if f(x)=1, has a negative sign.



$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle|-\rangle$$

We are moving from this state

$$|+\rangle^{\otimes^n} = \frac{1}{2^{\frac{n}{2}}}\sum_x |x\rangle.$$

to a state where the only difference is the negative sign. What happens is that we are marking $|x\rangle$ with a minus sign if f(x)=1. What happens is that if f(x) is constant, all $(-1)^{f(x)}$ are the same and therefore the amplitudes add up constructively, especially at $|0\rangle$, giving probability 1. However, if

f(x) is balanced, half the terms are +1 and half -1 ⇒ they cancel out, causing destructive interference at $|0\rangle$ and probability 0. In this way, the final measurement reveals the answer:

$$\begin{cases} |0\rangle \rightarrow \text{function is constant} \\ |1\rangle \rightarrow \text{function is balanced} \end{cases}$$

3.2.1 Oracle

In quantum computing, an oracle is a "black box" unitary operation that encodes a function f(x) in such a way that it can be queried quantumly. The oracle doesn't reveal how f is implemented, but allows us to evaluate it coherently on superpositions.
While using oracles, the computational complexity is given in terms of the number of oracle queries. The Deutsch–Jozsa algorithm relies on this oracle to extract global properties of a function $f(x) : 0,1^m \rightarrow 0,1$, which is promised to be either constant or balanced. By applying the oracle to a superposition of all inputs and exploiting interference, the algorithm can determine the nature of f(x) using only one quantum query.

3.2.2 Oracle types and problem classes

Oracles are powerful tools in quantum algorithms, and they can be designed to solve two broad classes of computational problems:

- **Decision problems:** The oracle helps decide a yes/no question about the input. For example, in the D-J algorithm, the oracle answers to the question: "Is the function balanced or constant?"

- **Function problems:** The oracle encodes a function, and the goal is to find something about its structure or output. For example, in the Simon's algorithm, the goal is to find a hidden period of a function.

In both cases, the oracle provides a mechanism to manipulate quantum states in a way that encodes the solution into interference patterns. However, the power of the oracle depends on the overall algorithmic structure and the ability to interpret the result after the measurement.
It is important to note that while oracles are theoretical constructs, they abstract away the implementation of f(x) and focus on how quantum mechanics can transform problem-solving via superposition and interference.

3.3 Constructing Oracles for Deutsch–Jozsa

To use the Deutsch–Jozsa algorithm in practice, we must implement a unitary operator U_f for a given function $f(x)$. This is a key challenge: for *each* choice of function f, we must define a

corresponding circuit that implements U_f such that

$$U_f|x\rangle|y\rangle = |x\rangle|\, y \oplus f(x)\rangle.$$

This means the oracle applies a *bit-flip* on the second qubit if and only if $f(x) = 1$. The implementation of U_f must be *reversible* and *unitary* to preserve quantum coherence.

Case 1: Constant Function

Consider the case where

$$f(x) = 0 \quad \forall x \quad \text{or} \quad f(x) = 1 \quad \forall x.$$

This represents a constant function.

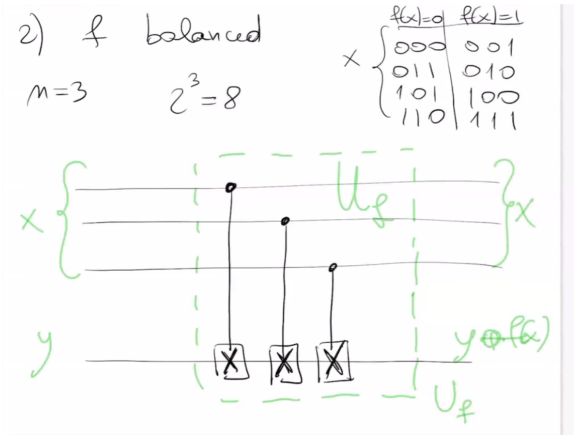
- If $f(x) = 0\ \forall x$, the oracle does *nothing* to the second qubit:

$$U_f|x\rangle|y\rangle = |x\rangle|\, y \oplus 0\rangle = |x\rangle|y\rangle.$$

- If $f(x) = 1\ \forall x$, the oracle applies an X gate to the second qubit, flipping it for all inputs:

$$U_f|x\rangle|y\rangle = |x\rangle|\, y \oplus 1\rangle = |x\rangle|\overline{y}\rangle.$$

Case 2: Balanced Function



Now consider the case where $f(x)$ is *balanced*: it returns 0 for half the inputs and 1 for the other half.

For example, if $m = 1$, we define

$$f(0) = 0, \quad f(1) = 1.$$

Then the oracle U_f must implement

$$U_f|0\rangle|y\rangle = |0\rangle|\, y \oplus 0\rangle = |0\rangle|y\rangle,$$

$$U_f|1\rangle|y\rangle = |1\rangle|\, y \oplus 1\rangle = |1\rangle|\overline{y}\rangle.$$

This corresponds to applying an X gate to the second qubit *controlled* by the first qubit being in $|1\rangle$: in circuit notation, a CNOT gate where

- the control is the input qubit $|x\rangle$,
- the target is the ancilla qubit $|y\rangle$.

Such a gate-based construction is typical for encoding balanced functions. More complex balanced functions (e.g. $m = 2$) require larger circuits composed of Toffoli or multiple-controlled gates.

Example: $m = 2$, Balanced Function

Let

$$f(00) = 0, \quad f(01) = 1, \quad f(10) = 1, \quad f(11) = 0.$$

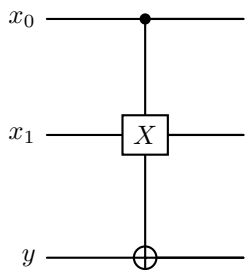
We must design U_f such that:

- If $x = 01$ or $x = 10$, apply an X gate to the ancilla qubit y.
- Otherwise, do nothing.

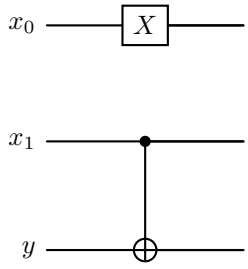
This requires:

1. Use X gates on control lines to convert conditions like $x = 01$ or $x = 10$ into AND logic.
2. Apply a Toffoli (multi-controlled X) gate on the ancilla.
3. Uncompute the auxiliary X gates to restore the original input qubits.

To conditionally flip y when $x = 01$:



and similarly for x=10:



You can stack these blocks to form a full oracle for any balanced function. This approach generalizes to larger m, where you may need Toffoli gates or their decompositions into basic gates. The key is always to encode the positions where $f(x) = 1$ as control logic for flipping the ancilla.

3.4 Search problem

Search problems are a major class of computational tasks where we are given a dataset of size $N = 2^m$ and we want to identify one or more elements that satisfy a specific condition.
The size of the dataset grows exponentially with the number of input bits (m). The goal is to search for some specific items within this exponentially large set, ideally using fewer steps than classically possible.
Classically, search problems require O(N) steps in the worst case. Quantum algorithms, such as Grover's algorithm, offer a quadratic speedup, solving them in $O(\sqrt{N})$ steps.

3.4.1 SAT-problem

In these type of problems we have string of bits $x-1, x_2, x_3$ (in this case, we are considering a 3-SAT problem) satisfying a set of Boolean conditions.
We want to keep in mind that for all these kind of problems we are just checking whether a solution is present, which is exponentially simpler than finding the solution to the problem.

3.5 Grover's Algorithm

Grover's algorithm is a quantum algorithm designed to solve unstructured search problems. Given a function.

$$f : \{0,1\}^n \to \{0,1\}$$

such that there exists a unique input x_0 for which $f(x_0) = 1$, and f(x)=0 for all other inputs, Grover's algorithm finds x_0 in $O(\sqrt{N})$ queries, where $N = 2^n$. We are given black-box access to a function $f(x)$. Our goal is to find the unique value x_0 such that:

$$f(x_0) = 1$$

This is known as an oracle problem, where the function f is implemented as a quantum oracle U_f, acting as:

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle$$

This version of the oracle applies a phase flip to the solution state $|x_0\rangle$, and leaves all others unchanged.

3.5.1 Key Steps of Grover's Algorithm

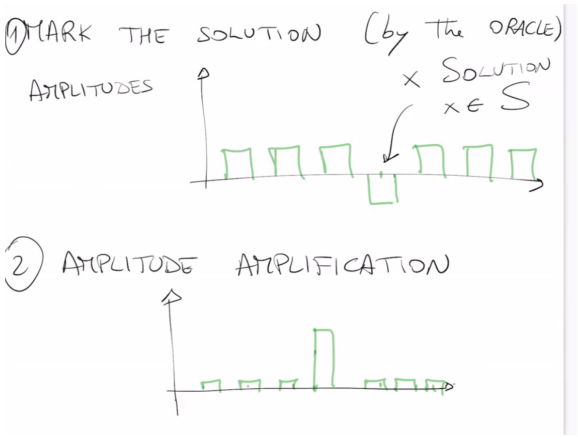
- Initialize the system in a uniform superposition over all N basis states:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

- Grover Iteration: Repeatedly apply the following two operations:
  - Oracle Mark the solution: Applies a phase flip to the target state (the marked solution), effectively multiplying its amplitude by -1.
  - Amplitude Amplification: Also called the "inversion about the mean," this operation reflects all amplitudes around their average value, increasing the marked state's amplitude and reducing the others.

  The process works as follows:
  - Let α be the average amplitude across all basis states.
  - For each state $|x\rangle$, compute the new amplitude as: $2\alpha - a_x$
  - This causes the marked state's (negative) amplitude to increase significantly, while the others decrease slightly.

  The effect is visualized in the diagram below:



As iterations continue, the amplitude of the marked state grows while others diminish. After about $\sqrt{N}$ iterations, measuring the quantum state will return the marked solution with high probability.

- Repeat the Grover iteration $\sim \sqrt{N}$ times.
- Measure the system. With high probability, the result will be x_0.

3.5.2 Quantum Circuit

The Grover circuit consists of:

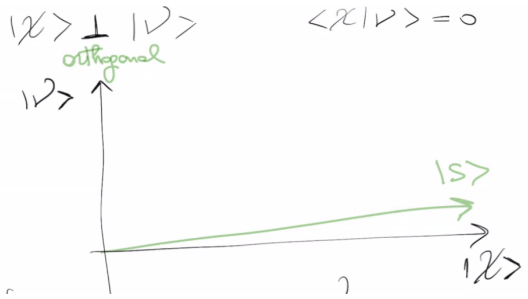
- Hadamard gates to create the initial uniform superposition.
- The oracle U_f to mark the correct state.
- The diffusion operator to amplify its amplitude.

4 Quantum Algorithms part 2 -31/05/2025

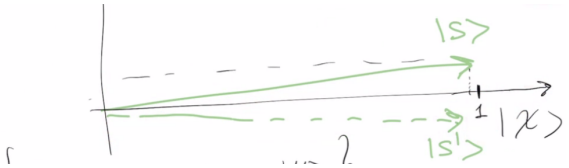
To better understand how Grover's Algorithm works, it's helpful to adopt a geometric perspective of quantum states as vectors in a Hilbert space.

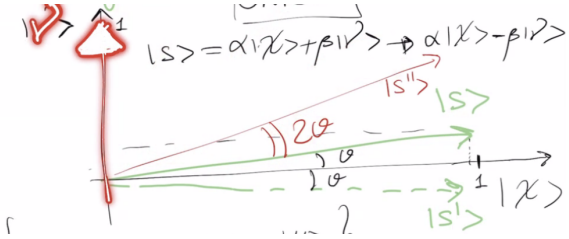
4.0.1 Example

Given 3 bits, we have 8 instances (000,001,...). Suppose that the solution is 100 for example, we should run classically the algorithm O(N) times (via random guessing). We define now a state $|\nu\rangle = |100\rangle$ and a state $|\chi\rangle = \frac{1}{\sqrt{7}}(|000\rangle + ...)$ where in general $|s\rangle = \frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle$.



The oracle is marking the solution. Since $|s\rangle = \alpha|\chi\rangle + \beta|\nu\rangle$, after the oracle we get $\alpha|\chi\rangle - \beta|\nu\rangle$.



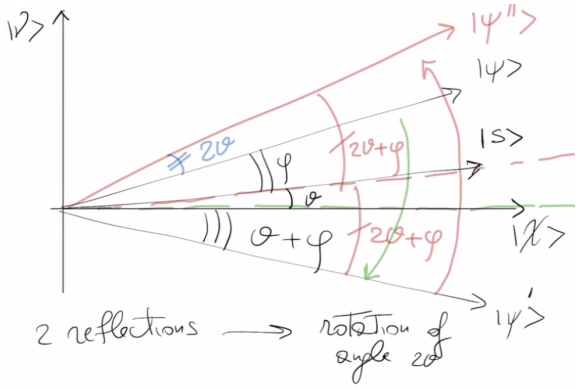


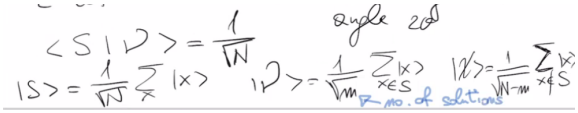
And that is given by using an amplifier (or diffuser).
What we want now is to make it more precise.
The angle θ between the state vector $|s\rangle$ and the non-solution axis $|\chi\rangle$ determines the probability amplitude of the correct answer. At the beginning, $|\theta\rangle$ is small because $|s\rangle$ is mostly aligned with

$|\chi\rangle$, due to the rarity of the solution.
Each Grover iteration consists of two reflections:

- the oracle reflects the hyperplane orthogonal to $|\nu\rangle$, flipping the sign of the solution component;
- the diffusion operator reflects the state about the initial state $|s\rangle$.

Together, these two reflections rotate the state vector by an angle of 2θ toward $|\nu\rangle$. After approximately $\frac{\pi}{4}\sqrt{N}$ iterations, the state vector aligns closely to $|\nu\rangle$. At this point, a measurement will return the correct solution with high probability.





We proceed now to calculate the quadratic speed up via θ. We start by considering the case where m = 1 (where m represents the number of solutions).
From the geometry of Grover's algorithm, we know that the amplitude of the solution state is:

$$\langle\chi|s\rangle = sin\theta = \sqrt{\frac{k}{N}}$$

where k is the number of solutions in dataset of size N. For real-world application, $N \to \infty$ and $k << N$, the angle becomes small and we can approximate:

$$\theta \sim \sqrt{\frac{k}{N}}.$$

Since each Grover iteration rotates the state vector by an angle of 2θ, the number of iterations required to reach close alignment with the solution $|\nu\rangle$ is approximately:

$$R \sim \frac{\pi}{4\theta} \sim \frac{\pi}{4}\sqrt{\frac{N}{k}}.$$

This shows that Grover's algorithm achieves a quadratic speedup, reducing the number of queries from O(N) in the classical case to $O(\sqrt{\frac{N}{k}})$ in the quantum case.
We now move to the case where m > 1.
When the oracle marks more than one solution, that is when there are m > 1 values of x for which f(x)=1, Grover's algorithm can still be applied, but the analysis requires slight modifications.

4.1 Adjusted State Decomposition

In the case of multiple solutions, we define:

- $|\nu\rangle$: the normalized superposition of all m solution states.
- $|\chi\rangle$: the normalized superposition of all $N - m$ non-solution states.

The initial state $|s\rangle$ is still the uniform superposition over all N states:

To compute the coefficient $\alpha = \langle\chi|s\rangle$, we use the definition of inner product between superpositions:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$$

$$|\nu\rangle = \frac{1}{\sqrt{m}} \sum_{x \in \text{solutions}} |x\rangle$$

$$|\chi\rangle = \frac{1}{\sqrt{N-m}} \sum_{x \notin \text{solutions}} |x\rangle$$

Then:

$$\langle\nu|s\rangle = \left\langle \frac{1}{\sqrt{m}} \sum_{y \in \text{solutions}} \langle y| \,\middle|\, \frac{1}{\sqrt{N}} \sum_x |x\rangle \right\rangle = \frac{1}{\sqrt{Nm}} \sum_x \sum_{y \in \text{solutions}} \langle y|x\rangle$$

The inner product $\langle y|x\rangle$ is 1 only when $x = y$, so the double sum counts exactly m matches:

$$\langle\nu|s\rangle = \frac{1}{\sqrt{Nm}} \cdot m = \sqrt{\frac{m}{N}}$$

4.1.1 Rotation Angle and Number of Iterations

As in the single-solution case, Grover's iteration rotates the state vector within the plane spanned by $|\nu\rangle$ and $|\chi\rangle$ by an angle of 2θ at each step, where:

$$\sin\theta = \alpha = \sqrt{\frac{m}{N}} \implies \theta \approx \sqrt{\frac{m}{N}} \text{ for small } \theta$$

The optimal number of iterations becomes:

$$R \approx \frac{\pi}{4}\sqrt{\frac{N}{m}}$$

After R Grover iterations, the state vector aligns closely with $|\nu\rangle$, and measuring yields one of the m solutions with high probability.

Summary

Grover's algorithm with multiple solutions maintains a quadratic speedup over classical search. The main differences are:

- The solution subspace has dimension m
- The number of iterations must be adapted to m
- The final measurement yields one of the m valid solutions

This generalization is crucial for applying Grover's algorithm to broader search problems where multiple correct results may exist.

4.2 Grover Oracle

In this section, we introduce how to build the Grover oracle U_f used to mark solutions in the search space.

We consider an example with 2 qubits:

- Input space size: $N = 2^2 = 4$
- Inputs: $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$
- Suppose the solution is $|11\rangle$
- Thus $m = 1$ (number of solutions)

We define:

- The uniform superposition state:

$$|s\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

- The non-solution component:

$$|\chi\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)$$

- The solution state:

$$|w\rangle = |11\rangle$$

The Grover oracle U is defined such that:

$$U|x\rangle = \begin{cases} |x\rangle & \text{if } x \notin S \\ -|x\rangle & \text{if } x \in S \end{cases}$$

Where S is the solution set.

In this case, only $|11\rangle$ is flipped:

$$U|00\rangle = |00\rangle, \quad U|01\rangle = |01\rangle, \quad U|10\rangle = |10\rangle, \quad U|11\rangle = -|11\rangle$$

This phase flip applied by the oracle marks the solution and allows the amplitude amplification to selectively boost its probability.

The number of required iterations is approximately:

$$\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor = \left\lfloor \frac{\pi}{2} \right\rfloor \approx 1 \text{ or } 2$$

4.2.1 Controlled-$f(x)$ Gates in the Oracle

To implement the Grover oracle U_f, we must conditionally apply a phase flip to the solution states. This is typically achieved using a controlled gate that checks whether x belongs to the solution set S.

In practice, this is implemented using a multi-controlled Z gate, often referred to as a $C^n(Z)$ or controlled-Z gate, acting only when $x = x_0$ for the marked solution.

This requires a circuit construction that evaluates $f(x)$ in quantum logic and flips the phase of the state when $f(x) = 1$.

If $f(x)$ is efficiently classically computable, one can:

- Build a quantum circuit that outputs $f(x)$ into an ancilla qubit.
- Use that ancilla to control a Z gate.
- Uncompute $f(x)$ to restore the ancilla to its original state (cleaning up entanglement).

4.3 Amplitude Amplifier (Diffusion Operator)

The amplitude amplifier (or diffusion operator) is the second key component in each Grover iteration, responsible for reflecting the state about the average amplitude. It amplifies the amplitude of marked states and suppresses the rest, steering the quantum state toward the solution.

This can be geometrically interpreted as a reflection with respect to the initial state $|s\rangle$.
To implement the diffusion operator, we use the following sequence:

1. Apply Hadamard gates $H^{\otimes n}$ to transform $|s\rangle$ into $|0\rangle^{\otimes n}$
2. Apply a multi-controlled-Z gate (or $C^n(Z)$) to flip the sign of $|0\rangle^{\otimes n}$, leaving other basis states unchanged
3. Apply Hadamard gates again to return to the original basis

This sequence results in the following transformation:

$$D = 2|s\rangle\langle s| - I$$

4.3.1 Circuit description

To reflect about $|s\rangle$, we use the identity:

$$\text{Reflection w.r.t. } |s\rangle = H^{\otimes n} \cdot (2|0\rangle\langle 0| - I) \cdot H^{\otimes n}$$

This is implemented as:

- Initial state $|\psi\rangle \rightarrow H^{\otimes n}|\psi\rangle$
- Apply $C^n(Z)$: flips the sign of $|0\rangle$
- Apply $H^{\otimes n}$ again: transforms back to original basis

This implements the diffusion operator and completes the Grover iteration: oracle + diffuser.

4.4 Example: Full Grover Circuit with Oracle and Diffuser

The diagram illustrates a full implementation of Grover's algorithm for 2 qubits, searching for the solution $|11\rangle$.

4.4.1 Initial State Preparation

Apply Hadamard gates to all qubits to create the uniform superposition:

$$|s\rangle = H^{\otimes 2}|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

4.4.2 Grover Iteration $G = D \cdot U_f$

- **Oracle** U_f (green): A controlled-Z gate flips the phase of the solution state $|11\rangle$ by applying a phase shift of -1.
- **Diffuser** D (red): Implements reflection about the mean amplitude via:

$$D = H^{\otimes n} \cdot (2|0\rangle\langle 0| - I) \cdot H^{\otimes n}$$

This involves:

1. Applying Hadamard to all qubits
2. Applying X gates to invert around $|0\rangle$
3. Applying controlled-Z
4. Undoing X and Hadamard gates

4.4.3 Measurement

After 1 or 2 Grover iterations, the probability of measuring the correct state $|11\rangle$ becomes very high. The final measurement yields the solution with near certainty, as depicted in the histogram.

4.5 Grover Applied to Binary Sudoku

4.5.1 Binary Sudoku 2×2

We consider a simple *binary Sudoku* puzzle defined on a 2×2 grid. Each cell can take a value of either 0 or 1. Let the variables representing the four cells be:

$$Q_0, Q_1, Q_2, Q_3$$

which are laid out in the grid as:

$$\begin{matrix} Q_0 & Q_1 \\ Q_2 & Q_3 \end{matrix}$$

Constraints For a valid binary Sudoku configuration: - Each *row* must contain one 0 and one 1 - Each *column* must contain one 0 and one 1

This creates a total of 4 binary constraints, which the oracle will evaluate.

4.5.2 Oracle Construction

The oracle U_f will evaluate all 4 constraints and return 1 *only if all are satisfied*. That is:

$$f(Q_0, Q_1, Q_2, Q_3) = \begin{cases} 1 & \text{if all constraints are satisfied} \\ 0 & \text{otherwise} \end{cases}$$

Step 1: Prepare Ancillae Create 4 ancilla qubits A_1, A_2, A_3, A_4 to store the XOR results of the following constraints:

- $A_1 = Q_0 \oplus Q_1$ - $A_2 = Q_2 \oplus Q_3$ - $A_3 = Q_0 \oplus Q_2$ - $A_4 = Q_1 \oplus Q_3$

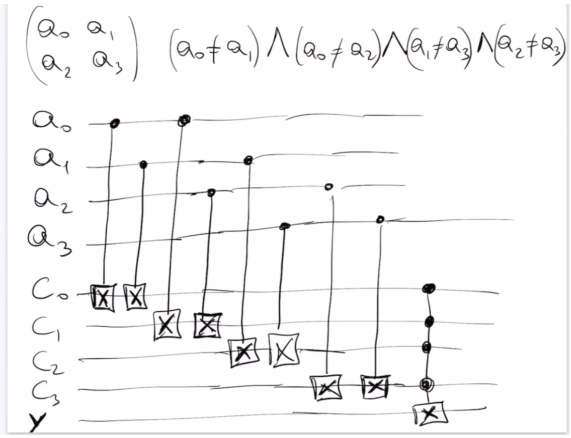
These ancillae must all be in state $|1\rangle$ for the constraints to be satisfied.

Step 2: Compute the XORs Each XOR is implemented using two CNOT gates:

```
CNOT(Q0, A1); CNOT(Q1, A1)
CNOT(Q2, A2); CNOT(Q3, A2)
CNOT(Q0, A3); CNOT(Q2, A3)
CNOT(Q1, A4); CNOT(Q3, A4)
```

Step 3: Apply Multi-Controlled CNOT Use a multi-controlled CNOT Gate (or a controlled phase flip) that is activated when all ancillae are in state $|1\rangle$, flipping the global phase of the state if all constraints are satisfied. This type of multi-controlled operation is often implemented using a *Toffoli gate*, also known as a *CCNOT gate*, when only two control qubits are involved.

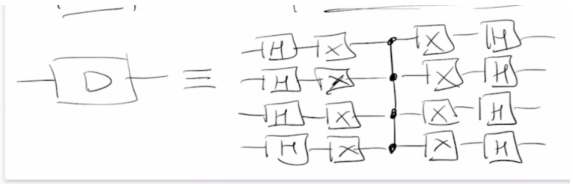
If I have the ancilla qbit (y), not it means that all conditions are satisfied



4.6 The Diffuser (Amplitude Amplification)

Once the oracle marks the correct solutions by flipping their sign, we need to amplify their probability amplitude. This is done via the *diffusion operator*, sometimes referred to as *Grover's diffuser*.

The diffusion operator performs this transformation:



Operationally, this is implemented as:

1. Apply Hadamard gates to all data qubits
2. Apply X gates to all data qubits
3. Apply a multi-controlled Z gate (phase flip) targeting the $|0\rangle^{\otimes n}$ state
4. Apply X gates again to all data qubits
5. Apply Hadamard gates again

This sequence reflects all state amplitudes about the average, amplifying those marked by the oracle.

The number of times this diffuser is applied depends on the number of valid solutions. For $N = 2^n$ total states and m solutions:

$$R \approx \frac{\pi}{4}\sqrt{\frac{N}{m}}$$

iterations are typically required to maximize the measurement probability of a correct solution.

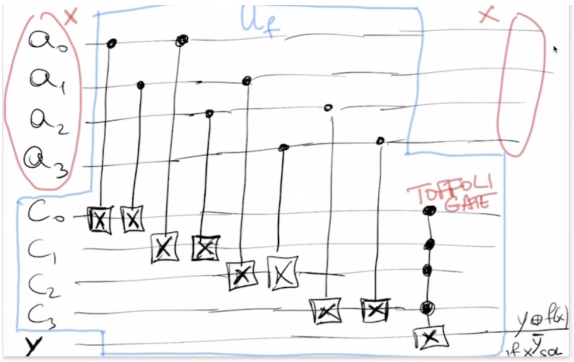
Final Construction of the Oracle for Binary Sudoku

To complete the oracle for this specific Binary Sudoku configuration, we follow these steps:

- Create ancilla qubits A_1, A_2, A_3, A_4 to check the XOR constraints:
  - $A_1 = Q_0 \oplus Q_1$,
  - $A_2 = Q_2 \oplus Q_3$,
  - $A_3 = Q_0 \oplus Q_2$,
  - $A_4 = Q_1 \oplus Q_3$.

- Add a fifth ancilla qubit A_5 initialized in state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
- Use Toffoli gates to check that all constraints are satisfied simultaneously by:
  - Combining A_1, A_2 into one Toffoli
  - Combining A_3, A_4 into another Toffoli
  - Then combining the outputs of those into a final Toffoli that flips the phase of A_5
- Uncompute all ancillae to return them to $|0\rangle$, preserving reversibility.

This completes the construction of the oracle, allowing Grover's algorithm to amplify only the configurations that satisfy all Binary Sudoku constraints.



5 Introduction to quantum machine learning -6/06/2025

Quantum machine learning is an interdisciplinary field that combines concepts from machine learning and quantum computing. The goal is to leverage quantum algorithms and quantum data representations to enhance or accelerate machine learning tasks.
An useful way to categorize approaches in QML is by considering whether the data and the algorithms are classical or quantum.

5.0.1 Building blocks

Data encoding (embedding) In quantum machine learning, data encoding (or embedding) refers to the process of mapping classical information (such as text, images or sound) into quantum states so it can be processed by a quantum computer.
Classical digital information ca be represented as vector of features. In the quantum setting, this information is encoded into the amplitudes of a quantum state.
For an m-qubit system, a general quantum state can be written as:

$$|\psi\rangle = \sum_{i=0}^{2^m-1} x_i |i\rangle$$

where $|i\rangle$ represents the computational basis states (bit-strings) and x_i are the amplitudes which encode the information.
This process allows classical data (such as text, images or sound) to be embedded into a quantum system for further quantum processing.

5.1 Basis encoding

Basis encoding is one of the simplest ways to encode classical data into quantum states: a classical bit string (e.g. 001011) is directly mapped into a quantum computational basis state (following the previous example, $|\psi\rangle = |001011\rangle$). In this way for m classical bits, m qubits will be needed.
Mathematically, a set of N bit strings ca be encoded as:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} |x^i\rangle$$

where $N = 2^m$ and $|x^i\rangle$ are computationally basis stats.
The advantage of using this type of encoding is that it is simple and has a direct mapping, while the disadvantage is that it requires as many qubits as the bits in the data, and it can be challenging to prepare and measure such states for large m.

5.2 Amplitude encoding

Amplitude encoding is a method where classical data is encoded into the amplitudes of a quantum state.
Given a classical data vector of N values, the data is mapped as:

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

where $|i\rangle$ are the computational basis states and x_i are the (normalized) values.

5.3 Amplitude Encoding

For m qubits, you can represent 2^m amplitudes, allowing you to encode an exponentially large amount of data with a linear number of qubits.

Key points:

- Very efficient in terms of qubit usage: m qubits $\rightarrow 2^m$ amplitudes.
- **Advantage:** Compact representation of large datasets.
- **Disadvantage:** Preparing arbitrary amplitude-encoded states can be challenging in practice. Also Measurement is complicated (more sensitive of error)

This method is widely used in quantum machine learning for its efficiency in representing high-dimensional data.

5.3.1 Angle Encoding

Angle encoding (or parametric encoding) is a method where classical data is encoded into the angles of quantum gates, typically using rotation gates.

Given a classical data vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, each feature x_i is encoded as the angle of a rotation applied to a qubit:

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

This means the quantum state is constructed as:

$$|x\rangle = \begin{pmatrix}\cos(x_1)\\ \sin(x_1)\end{pmatrix} \otimes \begin{pmatrix}\cos(x_2)\\ \sin(x_2)\end{pmatrix} \otimes \cdots \otimes \begin{pmatrix}\cos(x_N)\\ \sin(x_N)\end{pmatrix}$$

- Each classical value x_i is mapped to a rotation angle for a single qubit.
- For N features, N qubits are typically used.

Key points:

- **Advantage:** Simple to implement on quantum hardware; widely used in variational quantum circuits.
- **Disadvantage:** The number of qubits grows linearly with the number of features.

A common way to implement angle encoding is by using rotation gates, such as the $R_y(\theta)$ gate, where the angle θ is set according to the data value x_i.

For example, applying a rotation to the initial state $|0\rangle$:

$$U(x_i)|0\rangle = \begin{pmatrix} \cos(x_i) \\ \sin(x_i) \end{pmatrix}$$

More generally, a rotation around the y-axis is given by:

$$R_y(\theta) = e^{-i\frac{\theta}{2}\sigma_y} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

where typically $\theta = 2x_i$ for encoding the data value x_i.

This approach allows each classical parameter to be mapped to a quantum rotation, efficiently embedding classical data into quantum circuits.

Angle encoding is commonly used in quantum machine learning models to efficiently map classical data into quantum circuits using parameterized gates.

5.3.2 Why do we get cos and sin when applying the rotation, and where does the R_y rotation come from?

When we apply a quantum rotation such as $R_y(\theta)$ to a qubit state, we are rotating the state vector on the Bloch sphere around the y-axis. The $R_y(\theta)$ gate is defined as:

$$R_y(\theta) = e^{-i\frac{\theta}{2}\sigma_y} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

where σ_y is the Pauli-Y matrix.

If we apply $R_y(\theta)$ to the basis state $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, we get:

$$R_y(\theta)|0\rangle = \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{pmatrix}$$

If we set $\theta = 2x_i$, then:

$$R_y(2x_i)|0\rangle = \begin{pmatrix} \cos x_i \\ \sin x_i \end{pmatrix}$$

This shows that by rotating the qubit by an angle $2x_i$ around the y-axis, we obtain a quantum state whose amplitudes are $\cos(x_i)$ and $\sin(x_i)$. This is why, in angle encoding, classical data x_i is mapped directly to rotation angles, and the resulting quantum state has components $\cos(x_i)$ and $\sin(x_i)$.

Summary:

- The R_y rotation comes from the physics of qubits and their representation on the Bloch sphere.
- Applying $R_y(2x_i)$ to $|0\rangle$ produces a state with amplitudes $\cos(x_i)$ and $\sin(x_i)$, encoding the classical value x_i into the quantum state.

5.3.3 Arbitrary (General) Encoding

Arbitrary encoding refers to the most flexible approach for embedding classical data into quantum circuits. In this method, the parameters of quantum gates—such as rotation angles for R_x, R_y, R_z gates—are set according to the values of the classical data.

- Each data feature can control the angle of a rotation gate on a specific qubit.
- More complex circuits can use combinations of single-qubit rotations and multi-qubit gates (like CNOT, CZ) to entangle qubits and encode correlations between features.

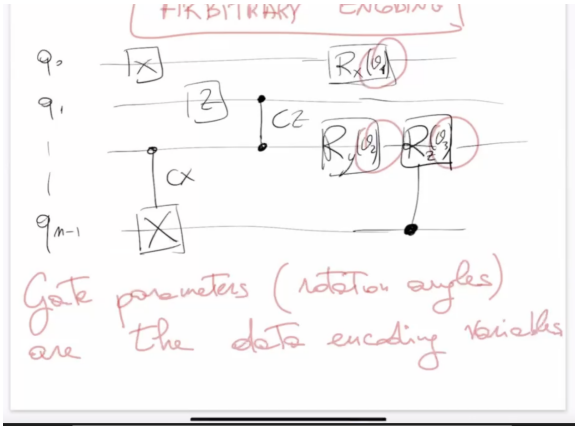


Figura 1: Arbitrary encoding circuit example

For example, in the circuit above:

- The classical data is mapped to the angles θ_1, θ_2, θ_3, ... of the rotation gates (R_x, R_y, R_z).
- The transformations R_x and R_y are **rotation operators** that act on a single qubit, rotating its state around the x and y axes of the Bloch sphere.

Rotation around the x axis (R_x):

$$R_x(\theta) = \exp\left(-i\frac{\theta}{2}X\right) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

Rotation around the y axis (R_y):

$$R_y(\theta) = \exp\left(-i\frac{\theta}{2}Y\right) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

These transformations allow you to encode classical data into the rotation parameters θ applied to the qubits.

- Entangling gates (like CX, CZ) can be used to introduce dependencies between qubits, allowing for more expressive encodings.

Key points:

- **Flexibility:** Any quantum gate parameter (rotation angle, phase, etc.) can be used as a data encoding variable.
- **Expressiveness:** By combining different gates and entanglement, arbitrary encoding can represent complex data structures and correlations.
- **Usage:** This approach is common in variational quantum circuits and quantum neural networks, where the circuit structure and parameters are tailored to the learning task.

In summary, arbitrary encoding allows you to design custom quantum circuits where the gate parameters are directly determined by the classical data you want to encode.

5.4 FRQI (Flexible Representation of Quantum Images)

The Flexible Representation of Quantum Images (FRQI) is a method for encoding classical images into quantum states. In FRQI, both the pixel position and the color (or grayscale) information are encoded using quantum resources:

$$|I(\theta)\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (\cos\theta_i|0\rangle + \sin\theta_i|1\rangle) \otimes |i\rangle$$

- **Pixel position** is encoded in the computational basis state $|i\rangle$ (basis encoding).
- **Color information** (e.g., grayscale value) is encoded in the angle θ_i using angle encoding, where each pixel's color determines the rotation angle.

This approach allows quantum computers to represent and process images efficiently, leveraging both basis and angle encoding within a single quantum state.

5.4.1 Example: FRQI for a 2x2 Image

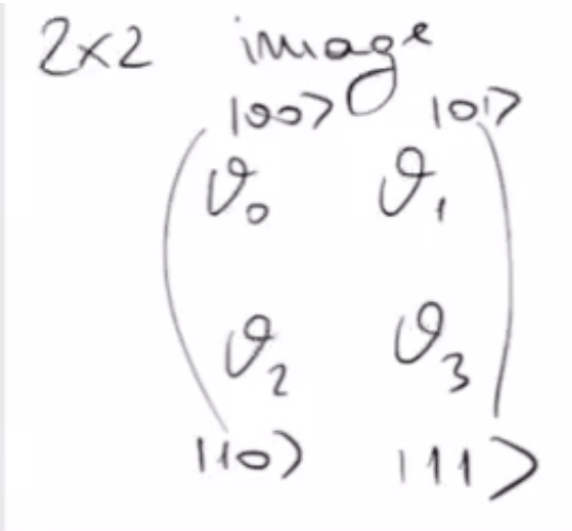


Figura 2: FRQI example for a 2x2 image

Let's see how the FRQI encoding works for a simple 2×2 image. Suppose the image has four pixels with grayscale values $\theta_0, \theta_1, \theta_2, \theta_3$.

The quantum state is:

$$|I\rangle = \frac{1}{2}\Big[(\cos\theta_0|0\rangle+\sin\theta_0|1\rangle)\otimes|00\rangle+(\cos\theta_1|0\rangle+\sin\theta_1|1\rangle)\otimes|01\rangle+(\cos\theta_2|0\rangle+\sin\theta_2|1\rangle)\otimes|10\rangle+(\cos\theta_3|0\rangle+\sin\theta_3|1\rangle)\otimes|11\rangle\Big]$$

- Each basis state $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ encodes the pixel position.
- The corresponding angle θ_i encodes the grayscale (or color) value for each pixel.

5.4.2 How to Realize FRQI Image Encoding

To encode an image using the FRQI method, you need to represent both the pixel positions and the color (intensity) information in a quantum state.

- For an image with $2^n \times 2^n$ pixels:
  - You need $2n$ qubits to encode the pixel positions (basis encoding).
  - You need 1 additional qubit to encode the color intensity (angle encoding).

Total qubits required: $2n + 1$

Step-by-step procedure: 1. Initialize the qubits:

- Start with all qubits in the $|0\rangle$ state.

2. Apply Hadamard gates to the $2n$ position qubits:

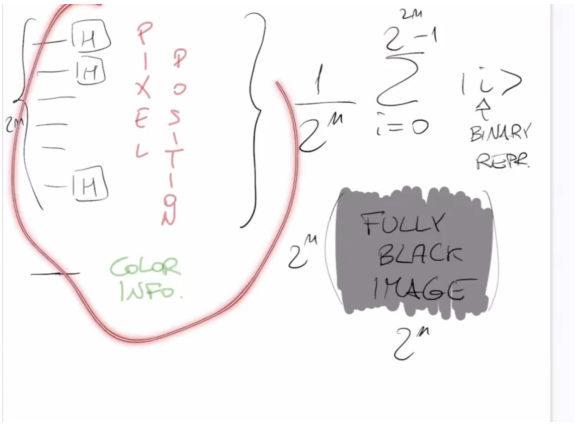


Figura 3: Hadamard gates applied to position qubits

- This creates a uniform superposition over all possible pixel positions:

$$H^{\otimes 2n}|0\rangle^{\otimes 2n} = \frac{1}{\sqrt{2^{2n}}} \sum_{i=0}^{2^{2n}-1} |i\rangle$$

Note: The state

$$|0\rangle \otimes \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

is the output obtained by applying Hadamard gates to the two position qubits. This operation creates a linear combination (superposition) of all four possible pixel positions.

the image is now black

3. Now, to encode the color information for each pixel, you apply a controlled rotation $R_y(2\theta_i)$ to the color qubit, conditioned on the pixel position $|i\rangle$.

For example, for pixel position $|i\rangle$, you apply:

$$R_y(2\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix}$$

This rotation changes the color

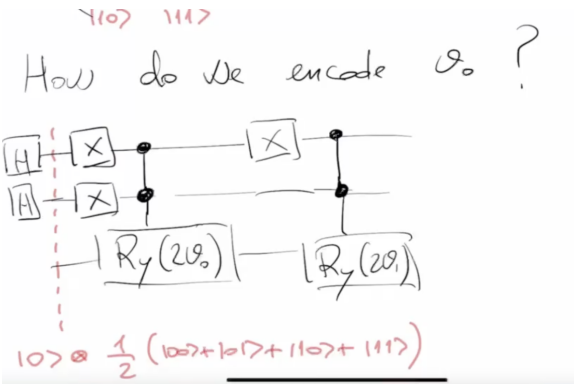


Figura 4: Controlled rotation for color encoding

To encode the value θ_0 for the pixel at position $|00\rangle$, you need to activate the control only when both position qubits are in the state $|1\rangle$.

This is achieved by applying X gates to flip the qubits from $|0\rangle$ to $|1\rangle$ (so that the control is "on" when both are $|1\rangle$), and then using a multi-controlled $R_y(2\theta_0)$ gate on the color qubit.

After the first controlled rotation (box), the state is:

$$\frac{1}{2}\left[(\cos(\theta_0)|0\rangle + \sin(\theta_0)|1\rangle) \otimes |00\rangle + |01\rangle + |10\rangle + |11\rangle\right]$$

This means only the color qubit associated with the pixel at position $|00\rangle$ has been rotated according to θ_0, while the other positions remain unchanged.

Now, to encode the next pixel value (for example, θ_1 at position $|01\rangle$), you repeat a similar procedure:

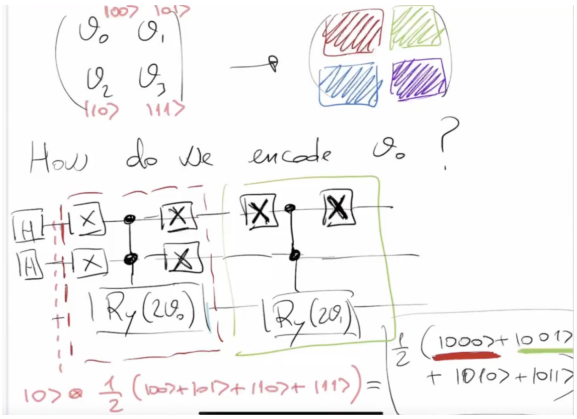


Figura 5: Encoding next pixel value

- Apply X gates as needed to activate the control only when the position qubits are in the $|01\rangle$ state.
- Apply a multi-controlled $R_y(2\theta_1)$ gate to the color qubit.

After this second controlled rotation, the state becomes:

$$\frac{1}{2}\left[(\cos(\theta_0)|0\rangle + \sin(\theta_0)|1\rangle) \otimes |00\rangle + (\cos(\theta_1)|0\rangle + \sin(\theta_1)|1\rangle) \otimes |01\rangle + |10\rangle + |11\rangle\right]$$

and finally

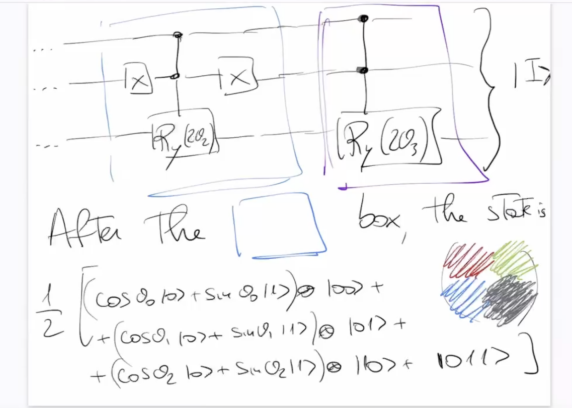


Figura 6: Final encoding circuit

This process is repeated for all pixel positions, so that each pixel's color information is encoded in the corresponding basis state by the appropriate controlled rotation.

At the end, after applying all the controlled rotations, the final state will be:

$$|I\rangle = \frac{1}{2}\Big[(\cos\theta_0|0\rangle + \sin\theta_0|1\rangle)\otimes|00\rangle + (\cos\theta_1|0\rangle + \sin\theta_1|1\rangle)\otimes|01\rangle + (\cos\theta_2|0\rangle + \sin\theta_2|1\rangle)\otimes|10\rangle + (\cos\theta_3|0\rangle + \sin\theta_3|1\rangle)\otimes|11\rangle\Big]$$

This is the FRQI-encoded quantum state, where each pixel position is associated with its corresponding color (or grayscale) value.

5.5 Measurement

Once the FRQI state has been prepared, you can perform a measurement to extract information about the image.

- Measuring the **position qubits** gives you the pixel position (e.g., $|00\rangle$, $|01\rangle$, etc.).

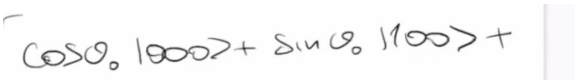


Figura 7: Measurement histogram

Note: For each pixel position (e.g., $|00\rangle$), you may find multiple quantum states (like $|000\rangle$ and $|100\rangle$) corresponding to the same pixel position. The amplitudes for these states are different: one is associated with $\cos\theta_i$ and the other with $\sin\theta_i$

N.B. The color information can be read in the y axis of the amplitude histogram: the probability of measuring the color qubit in $|0\rangle$ or $|1\rangle$ for each pixel position reflects the encoded grayscale (or color) value.

This allows you to reconstruct the image by sampling the quantum state and analyzing the measurement statistics.

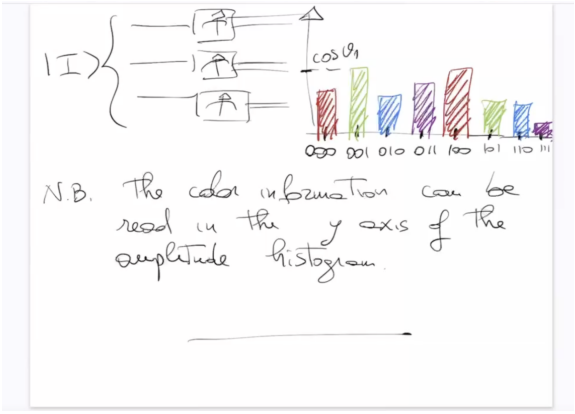


Figura 8: Image reconstruction from quantum measurements

5.6 NEQR (Novel Enhanced Quantum Representation for Digital Images)

NEQR is an alternative to FRQI in which pixel color values are encoded directly using **basis encoding**, instead of angle encoding.

In NEQR:

- Each pixel's **position** is encoded in the computational basis states (basis encoding).
- Each pixel's **color value** (e.g., grayscale value) is represented using an L-bit binary string encoded into qubits via basis encoding.

The quantum state is:

$$|I\rangle = \frac{1}{\sqrt{2^{2n}}} \sum_{i=0}^{2^{2n}-1} |x_i\rangle_{\text{color}} \otimes |i\rangle_{\text{position}}$$

- $|i\rangle$ encodes the pixel position.
- $|x_i\rangle$ is the L-bit binary representation of the pixel color value.

Example: 8-bit Grayscale Image

- Grayscale values range from 0 (black) to 255 (white).
- Example encoding:

Grayscale Value	Binary Encoding	Description
0	00000000	Black
100	01100100	Dark shade
200	11001000	Light shade
255	11111111	White

Example State Structure

$$|I\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle \otimes |10000000\rangle + |11\rangle \otimes |11001000\rangle + \dots\right)$$

- $|00\rangle, |11\rangle \rightarrow$ pixel position.
- $|10000000\rangle, |11001000\rangle \rightarrow$ pixel color in basis encoding.

6 Implementing NEQR (Normalized Encoding of Quantum Representations)

The **Normalized Encoding of Quantum Representations (NEQR)** is a method for encoding classical data, such as images, into quantum states. This encoding allows quantum algorithms to process classical data efficiently, leveraging the principles of quantum superposition and entanglement.

6.1 Quantum Circuit for NEQR

The image illustrates the quantum circuit for NEQR encoding. The circuit uses Hadamard gates (H) to create a superposition state, representing pixel positions in an image. The encoded quantum state is given by:

$$\frac{1}{2} \left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

Where:

- Each basis state (e.g., $|00\rangle$, $|01\rangle$, etc.) corresponds to a pixel position in the image.
- The amplitude of each basis state can be adjusted to encode pixel intensity values.

6.2 Steps to Implement NEQR

1. Initialize Qubits:

- Allocate qubits to represent pixel positions and intensities.
- For an $n \times n$ image, use $\log_2(n)$ qubits for pixel positions and additional qubits for intensity values.

2. Apply Hadamard Gates:

- Use Hadamard gates (H) to create a superposition of all possible pixel positions. This ensures that all positions are represented simultaneously in the quantum state.

3. Encode Pixel Intensities:

- For an L-bit grayscale image, allocate L qubits to represent the intensity (e.g., 8 qubits for 8-bit grayscale).
- For each pixel position, use multi-controlled X (NOT) gates to set the color qubits according to the binary representation of the pixel's intensity value.
- This ensures that, for every basis state representing a pixel position, the associated color qubits encode the correct grayscale value in binary.

4. Measurement:

- Measure the quantum state to retrieve both pixel positions and intensity values. Each measurement yields a bitstring where:
  - The first $\log_2(n)$ bits represent the pixel position.
  - The following L bits represent the pixel intensity (e.g., 8 bits for 8-bit grayscale).
- By repeatedly measuring the quantum state, you can reconstruct the classical image by collecting the frequency of each observed bitstring.

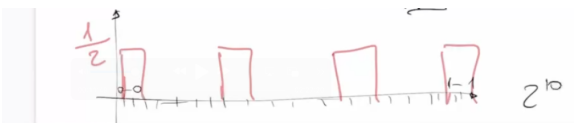


Figura 9: NEQR measurement histogram

Note: In the measurement histogram, all bars have the same height because the NEQR encoding creates a uniform superposition over all pixel positions. The histogram's x-axis encodes both the pixel position (first bits) and the color value (last bits), while the y-axis shows the probability (which is uniform if all pixels are equally likely).

6.3 Example Circuit

For a 2×2 image:

- Use 2 qubits for pixel positions ($|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$).
- Apply Hadamard gates to both qubits:

$$H \otimes H \cdot |00\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

then we apply the coding for color corresponding to each pixel position

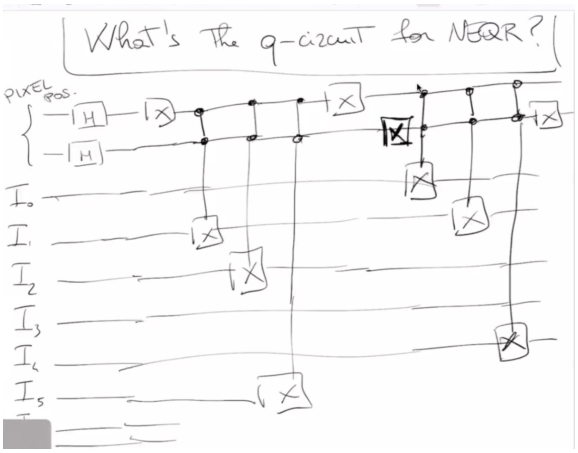


Figura 10: NEQR circuit example for 2x2 image

6.4 Advantages of NEQR

- **Efficient Encoding**: NEQR allows classical data to be encoded into quantum states with minimal resources.
- **Parallel Processing**: Quantum algorithms can process all pixel positions simultaneously due to superposition.
- **Scalability**: NEQR can be extended to larger images by increasing the number of qubits.

6.5 Challenges

- **Hardware Limitations**: Implementing NEQR requires quantum hardware capable of handling a large number of qubits.
- **Noise Sensitivity**: Quantum states are sensitive to noise, which can affect the accuracy of the encoding.

NEQR is a foundational technique for quantum image processing, enabling quantum algorithms to work with classical

7 Quantum Facial Expression Recognition

Facial expression recognition is a critical task in computer vision, and quantum computing offers unique advantages for processing and classifying facial features. This chapter explores how quantum algorithms can be applied to facial expression recognition using quantum states and graph-based representations.

7.1 Process Overview

1. Image Preprocessing:

- The facial image is preprocessed to extract key features, such as landmarks.
- Typically, 68 facial landmarks are identified, representing critical points on the face (e.g., eyes, nose, mouth).

2. Feature Extraction:

- The landmarks are used to construct a graph representation of the face.
- Each node in the graph corresponds to a facial landmark, and edges represent relationships between landmarks (e.g., distances or connections).

3. Classification:

- Quantum states are used to encode the graph structure.
- The classification task involves distinguishing between different facial expressions (e.g., happy vs. sad) based on the quantum representation.

7.2 Graph Representation

The facial landmarks are encoded as a graph G, where:

- $g(i, j) = 1$ if (i, j) is a link (i.e., there is a connection between landmarks i and j).
- $g(i, j) = 0$ otherwise.

The graph can be represented as a quantum state:

$$|G\rangle = \sum_{i,j} g(i, j)|i\rangle|j\rangle$$

Where:

- $|i\rangle$ and $|j\rangle$ represent the quantum states corresponding to the nodes (landmarks).

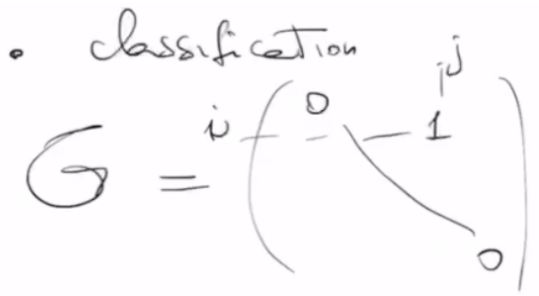


Figura 11: Quantum facial expression recognition graph representation

7.3 Quantum Classification

The quantum state $|G\rangle$ is used to classify facial expressions:

- For a "happy" expression, the quantum state might have specific amplitude distributions.
- For a "sad" expression, the amplitude distributions differ.

The classification is performed by computing the inner product (overlap) between the quantum state $|G\rangle$ representing the input facial graph and reference quantum states $|G_{\text{sad}}\rangle$ or $|G_{\text{happy}}\rangle$ corresponding to known "sad" or "happy" expressions. The squared magnitude of these inner products indicates how closely the input matches each reference:

$$\langle G_{\text{sad}}|G\rangle^2 \quad \text{vs.} \quad \langle G_{\text{happy}}|G\rangle^2$$

The input is classified as the expression with the highest overlap.

7.4 Advantages of Quantum Facial Expression Recognition

- **Efficient Feature Encoding**: Quantum states can encode complex graph structures efficiently.
- **Parallel Processing**: Quantum algorithms can process multiple features simultaneously due to superposition.
- **Scalability**: Quantum methods can handle large datasets and complex relationships between features.

7.5 Challenges

- **Noise Sensitivity**: Quantum states are sensitive to noise, which can affect classification accuracy.
- **Hardware Limitations**: Current quantum hardware may struggle with large-scale graph encoding.

8 Swap Test for Face Comparison

The **Swap Test** is a quantum algorithm for measuring the similarity (overlap) between two quantum states. In facial expression recognition, it compares quantum-encoded faces (e.g., $|\phi\rangle$ and $|\psi\rangle$) to quantify how alike they are.

8.1 Circuit Description

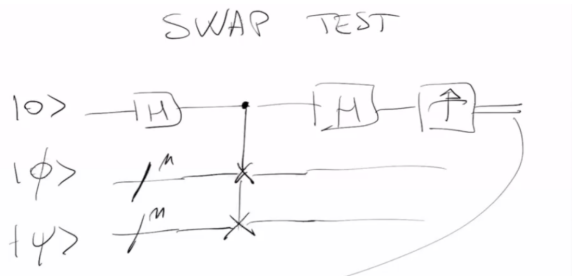


Figura 12: Swap Test circuit for face comparison

The Swap Test circuit consists of:

1. Ancilla Qubit:

- An additional qubit initialized in the state $|0\rangle$.

- This qubit is used to control the swap operation between the two input states.

2. Input States:

- Two quantum states $|\phi\rangle$ and $|\psi\rangle$ representing the faces to be compared.
- These states encode features extracted from facial landmarks.

3. Hadamard Gates:

- A Hadamard gate (H) is applied to the ancilla qubit at the beginning and end of the circuit to create and measure superposition.

4. Controlled-SWAP Gate:

- The ancilla qubit controls the swap operation between $|\phi\rangle$ and $|\psi\rangle$.
- If the ancilla qubit is in state $|1\rangle$, the two states are swapped.

5. Measurement:

- The ancilla qubit is measured at the end of the circuit to determine the similarity between $|\phi\rangle$ and $|\psi\rangle$.

8.2 Quantum State Evolution

1. Initial State: The system starts in the state:

$$|0\rangle \otimes |\phi\rangle \otimes |\psi\rangle$$

2. After First Hadamard Gate: The ancilla qubit is placed in superposition:

$$\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes |\phi\rangle \otimes |\psi\rangle$$

3. Controlled-SWAP Operation: If the ancilla qubit is $|1\rangle$, the states $|\phi\rangle$ and $|\psi\rangle$ are swapped.

4. Final Hadamard Gate: The ancilla qubit is transformed back to the computational basis.

5. Measurement: The probability of measuring $|0\rangle$ on the ancilla qubit is:

$$P(|0\rangle) = \frac{1}{2} \left(1 + |\langle\phi|\psi\rangle|^2\right)$$

The overlap $|\langle\phi|\psi\rangle|^2$ quantifies the similarity between the two states.

8.3 Interpretation of Results

- $|\langle\phi|\psi\rangle|^2 = 1$: The two states are identical (perfect match).
- $|\langle\phi|\psi\rangle|^2 = 0$: The two states are orthogonal (no similarity).

8.4 Application to Face Comparison

1. Quantum Encoding: Faces are encoded as quantum states based on features extracted from facial landmarks.

2. Similarity Measurement: The Swap Test is used to compare the quantum states of two faces. The result indicates whether the faces represent the same expression or different expressions.

8.5 Advantages of the Swap Test

- **Efficient Similarity Measurement**: The Swap Test directly computes the overlap between two quantum states without requiring full state tomography.
- **Scalability**: It can be applied to high-dimensional quantum states, making it suitable for complex facial features.

8.6 Challenges

- **Transpilation overhead**: Transpiling circuits with multi-controlled gates generates many basic gates, leading to deeper, more error-prone circuits on current quantum hardware.

9 Quantum Edge Detection

Quantum edge detection leverages quantum parallelism to efficiently identify image boundaries. By encoding images into quantum states using probability-based methods, quantum algorithms can process all pixels simultaneously, offering significant speedups over classical approaches. This section introduces the principles and advantages of quantum edge detection using quantum probability-based image encoding.

9.1 Classical vs Quantum Edge Detection

For an image with $2^m \times 2^m$ pixels:

- **Classical Complexity**: Classical edge detection algorithms typically require $O(2^{2m})$ operations, as they process each pixel individually.
- **Quantum Complexity**: Quantum edge detection can achieve $O(1)$ complexity by encoding the entire image into a quantum state and processing it in parallel.

9.2 Quantum Probability-Based Image Encoding (QPIE)

The image is encoded into a quantum state using probabilities derived from pixel intensities. The encoding process ensures that the quantum state represents the image efficiently.

9.2.1 Encoding Formula

Each pixel intensity $I_{x,y}$ is normalized to compute the probability amplitude C_i:

$$C_i = \frac{I_{x,y}}{\sqrt{\sum_{x,y} I_{x,y}^2}}$$

Where:

- $I_{x,y}$ is the intensity of the pixel at position (x, y).
- $\sum_{x,y} I_{x,y}^2$ ensures normalization, making $C_i \in [0, 1]$.

9.2.2 Quantum State Representation

The image is encoded as a quantum state:

$$|I\rangle = \sum_{x,y} c_{x,y}|x\rangle|y\rangle$$

Where:

- $|x\rangle|y\rangle$ represents the position of the pixel.
- $c_{x,y}$ represents the normalized intensity of the pixel.

9.3 Quantum Hadamard Edge Detection (QHED)

The **Quantum Hadamard Edge Detection (QHED)** algorithm leverages quantum principles, such as superposition and parallelism, to efficiently detect edges in images.

9.3.1 Key Components of QHED

1. Hadamard Gates:

- The Hadamard gate (H) creates a superposition of states, enabling parallel processing of pixel information.
- For a qubit in state $|0\rangle$, applying H results in:

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

- For a qubit in state $|1\rangle$, applying H results in:

$$H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

2. Binary Representation:

- Each pixel is represented as a binary string of 2^m bits, where m is the number of qubits used to encode the image.
- For example, a pixel intensity $I_{x,y}$ is encoded as:

$$|b_{2^m-1}b_{2^m-2}\dots b_1\rangle$$

- The **Least Significant Bit (LSB)** plays a critical role in detecting intensity differences.

3. Edge Detection Formula:

$$\Delta C = |C_{b_{2^m-1}\dots b_0} - C_{b_{2^m-1}\dots b_1 1}|$$

Where:

- $C_{b_{2^m-1}\dots b_0}$ represents the normalized intensity of the current pixel.
- $C_{b_{2^m-1}\dots b_1 1}$ represents the normalized intensity of the adjacent pixel.

9.3.2 Thresholding

To classify a pixel as part of an edge:

- If $\Delta C > \Delta C_{\text{th}}$, the pixel is classified as an **edge**.
- Otherwise, the pixel is classified as **no edge**.

Here:

- ΔC_{th} is the threshold value, which can be adjusted based on the desired sensitivity of edge detection.

9.3.3 Quantum Representation of Thresholding

The thresholding operation can be implemented using quantum gates:

- The Hadamard gate (H) is applied to create superposition states for comparing pixel intensities.
- The comparison is performed in parallel across all pixels using quantum operations.

9.3.4 Final Quantum State

The quantum state after thresholding represents the edges in the image:

$$\frac{1}{2^{m-1}} \otimes H = \frac{1}{\sqrt{2}}$$

This state encodes the positions of edge pixels, which can be measured to retrieve the edge map.

9.4 Matrix Representation in Quantum Edge Detection

The image illustrates how matrix operations are used in quantum edge detection to compute intensity differences and identify edges. This step involves transforming the quantum state into a column vector and applying specific matrix operations to extract edge information.

9.4.1 Quantum State Transformation

The quantum state encoding pixel intensities is represented as a 2^m-dimensional column vector:

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ \vdots \\ C_{2^m-1} \end{bmatrix}$$

9.4.2 Hadamard Matrix Transformation

To compute intensity differences, a Hadamard matrix is applied to the column vector. The Hadamard matrix is defined as:

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

For a larger system, the Hadamard matrix is extended to $2^m \times 2^m$ dimensions, with Hadamard transformations along its diagonal. For example, a 4×4 Hadamard matrix looks like:

$$H_4 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

9.4.3 Application of Hadamard Matrix

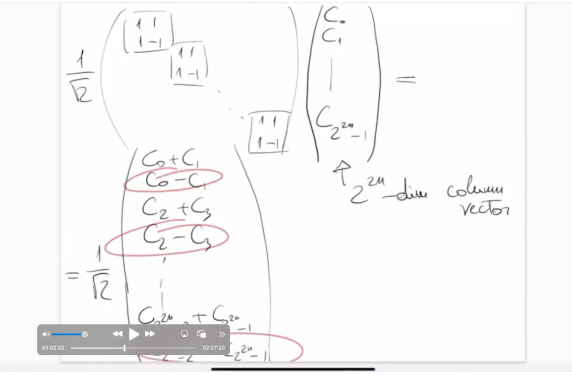


Figura 13: Hadamard matrix application for edge detection

The Hadamard matrix is applied to the quantum state vector to compute sums and differences of adjacent pixel intensities:

$$H \cdot \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} C_0 + C_1 \\ C_0 - C_1 \\ C_2 + C_3 \\ C_2 - C_3 \end{bmatrix}$$

9.4.4 Resulting Vector

The resulting vector contains the sums and differences of adjacent pixel intensities (only even indices), which are essential for edge detection:

$$\begin{bmatrix} C_0 + C_1 \\ C_0 - C_1 \\ C_2 + C_3 \\ C_2 - C_3 \end{bmatrix}$$

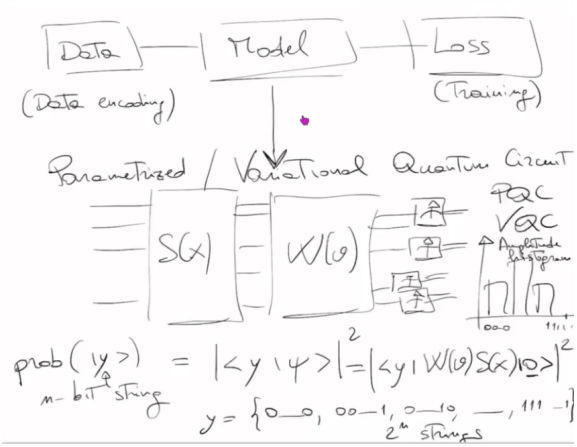
For odd indices, a permutation operation is applied to obtain:

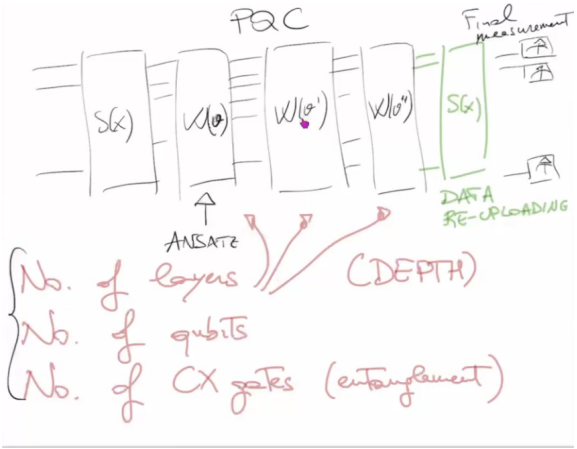
$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_{2^m-1} \\ C_0 \end{bmatrix}$$

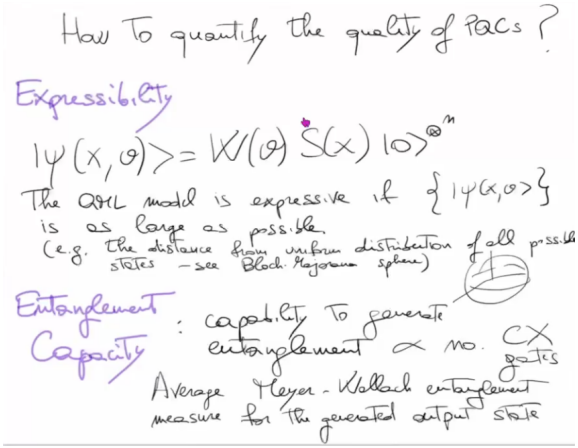
In this case the resulting vector includes odd indices:

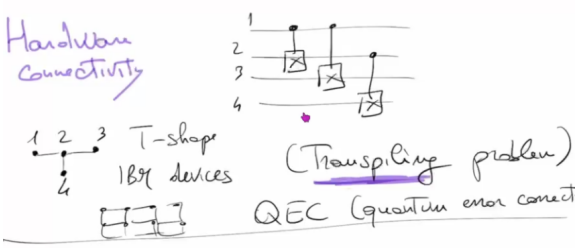
$$\begin{bmatrix} C_1 + C_2 \\ C_1 - C_2 \\ C_3 + C_4 \\ C_3 - C_4 \end{bmatrix}$$

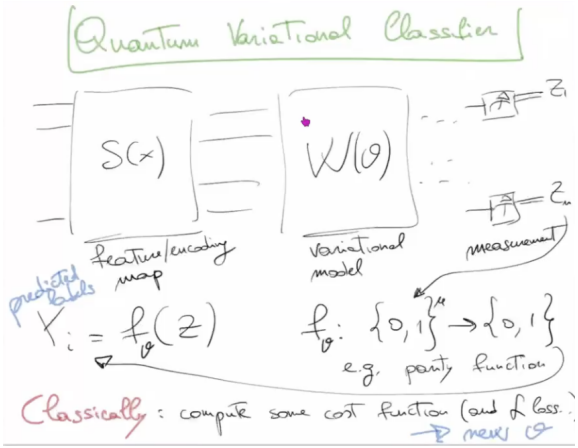
10 Introduction to quantum machine learning pt 2 -7/06/2025






How To quantify the quality of PQCs ?
Expressibility
$|\psi(x,\theta)\rangle = W(\theta)\, S(x)\, |0\rangle^{\otimes n}$
The QML model is expressive if $\{|\psi(x,\theta)\rangle\}$ is as large as possible.
(e.g. the distance from uniform distribution of all possible states – see Bloch Hyperbolic sphere)
Entanglement Capacity : capability to generate entanglement ∝ no. CX gates
Average Meyer-Wallach entanglement measure for the generated output state


Hardware connectivity
1
2
3
4
1 2 3 T-shape
4
IBM devices
(Transpiling problem)
QEC (quantum error correct


Quantum Variational Classifier
S(x)
W(θ)
feature/encoding map
variational model
measurement
predicted labels
$y_i = f_\theta(z)$
$f_\theta : \{0,1\}^n \to \{0,1\}$
e.g. parity function
Classically : compute some cost function (and & loss.)
new θ

QUANTUM KERNEL THEORY

What do we mean for classical kernel methods?

GOAL: To solve ML Tasks by means of a similarity measure between data points

Def. KERNEL

$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

↑ positive semi-definite bivariate function

Dataset $\mathcal{D} = \{x_1, \dots, x_M\} \subset \mathcal{X}$

Def. GRAM matrix

$K_{mm'} = K(x_m, x_{m'}) \quad x_m, x_{m'} \in \mathcal{D}$

K is a positive semi-definite matrix

$K \geq 0 \qquad K(x_m, x_{m'}) \geq 0$

$K(x_m, x_{m'}) = K(x_{m'}, x_m)$ *

KERNEL EXAMPLES

1) LINEAR $x^T x'$ $\quad x, x' \in \mathbb{R}^N$

2) POLYNOMIAL $(x^T x' + c)^p$ $\quad p \in \mathbb{N}$

3) GAUSSIAN $e^{-\gamma \|x - x'\|^2}$ $\quad \gamma \in \mathbb{R}^+$

4) EXPONENTIAL $e^{-\gamma \|x - x'\|}$ $\quad c \in \mathbb{R}$

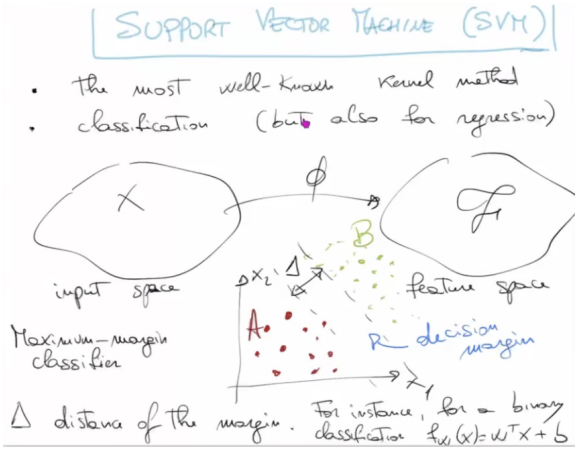
5) SIGMOID $\tanh(x^T x' + c)$

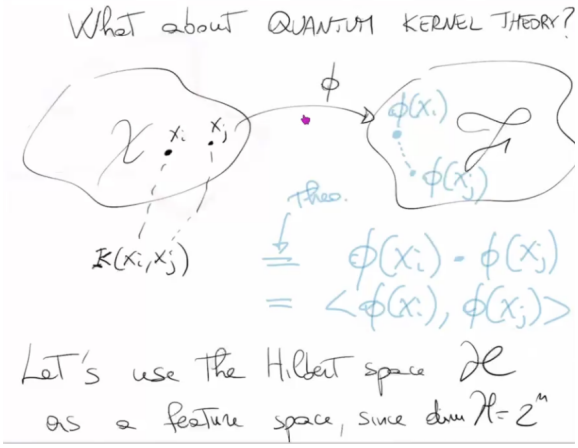
Theo. Any kernel function can be always written as an inner product of data that were mapped into a feature space $\mathcal{F}$

$\phi : \mathcal{X} \to \mathcal{F} \qquad K(x, x') = \phi(x) \cdot \phi(x')$

(KERNEL TRICK $K \to K_{NEW}$) $= \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$


DATA SPACE
FEATURE SPACE
KERNEL METHOD
K-nearest neighbour
It's the simplest classification algorithm
A
B
One selects the K closest Training inputs to the new input x


SUPPORT VECTOR MACHINE (SVM)
the most well-known kernel method
classification (but also for regression)
input space
feature space
Maximum-margin classifier
decision margin
A
B
Δ distance of the margin. For instance, for a binary classification


What about QUANTUM KERNEL THEORY?
Theo.
Let's use the Hilbert space
as a feature space, since

$K(x_i, x_j) = \langle f(x_i) | f(x_j) \rangle$

Quantum feature map

$\Phi : x \longrightarrow |f(x)\rangle$

↑ by a PQC

It becomes very promising if Φ is very hard to be simulated by classical computing (HPC)

$K(x_i, x_j) = \langle \phi(x_j)^\dagger | \phi(x_i) \rangle$ as a measure of similarity (K is large when x_i and x_j are close)

Which PQC ?

$\text{Prob}(|0\ldots0\rangle) = \langle 0\ldots0 | \psi(x_i, x_j) \rangle =$

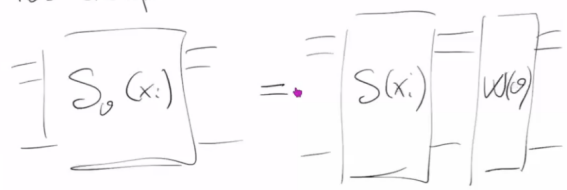
$= |\langle 0\ldots0 | S^\dagger(x_i) S(x_j) | 0\ldots0 \rangle|^2 =$

$= |\langle \phi(x_i) | \phi(x_j) \rangle|^2 = K(x_i, x_j)$

QUANTUM FEATURE MAP — Q-KERNEL

where $|\phi(x_i)\rangle \equiv S(x_i)|0\ldots0\rangle$

For example



SVM → Q-SVM

K-nearest neighbour → Q K-nearest neighbour

QUANTUM KERNEL ALIGNMENT

Unsupervised QML

Principal component analysis (PCA)

Clustering

Variational auto-encoders (VAE)

↓

QUANTUM VERSIONS

How to train PQCs

Variational Training

Methods
- gradient-based
  - very slow convergence speed
  - no guarantee to get the optimal solution
- gradient-free
  - better to achieve global minima but they require higher computational capacity
  - (when: - the loss function is expensive or noisy
    - the derivative information is unavailable / impractical

What happens for PQCs?

$$\frac{\partial f}{\partial \vartheta_i} \overset{\text{ANALYTICAL}}{=} \frac{f\left(\vec{\vartheta} + \frac{\pi}{2}\vec{e}_i\right) - f\left(\vartheta - \frac{\pi}{2}\vec{e}_i\right)}{2}$$

NO APPROXIMATION AS IN FD

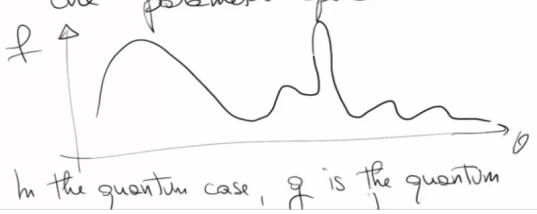
$$\vec{e}_i = (0 \; \cdots \; 0 \; \underset{\uparrow i}{1} \; 0 \; \cdots \; 0)$$

PARAMETER SHIFT RULE

Natural gradient

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

where g is the metric in the parameter space



In the quantum case, g is the quantum

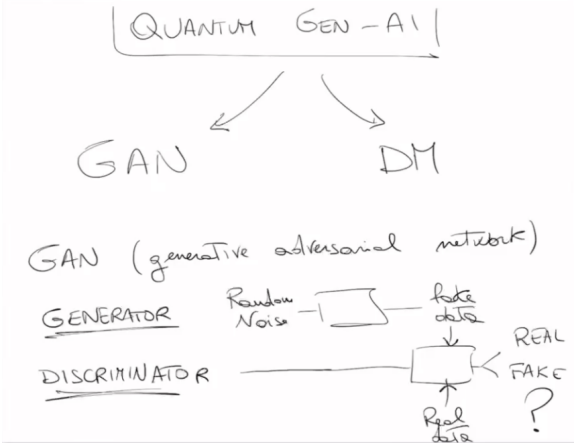
SPSA (Simultaneous Perturbation Stochastic Approximation)

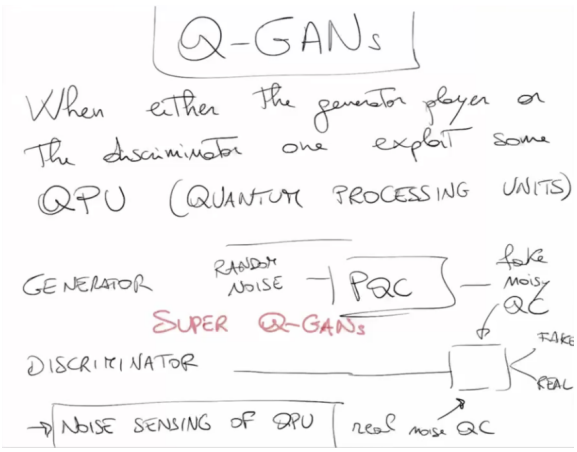
$$\vec{\nabla} f(\vartheta) = \left(\frac{\partial f}{\partial \vartheta_1}; \; \cdots \; , \frac{\partial f}{\partial \vartheta_n}\right)$$

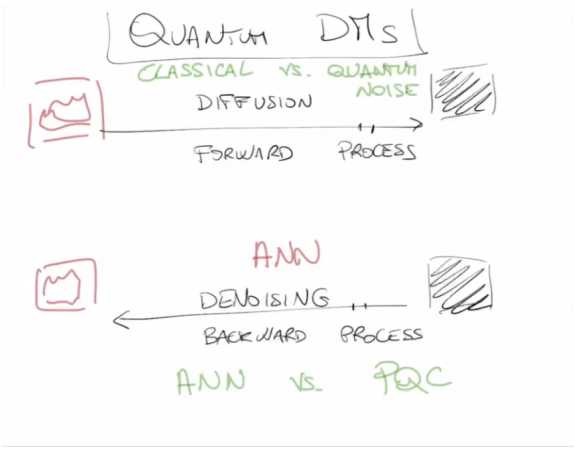
2n PQC

$$\simeq \frac{1}{2\varepsilon} \vec{\Delta}^{-1} \left[f\left(\vec{\vartheta} + \varepsilon\vec{\Delta}\right) - f\left(\vec{\vartheta} - \varepsilon\vec{\Delta}\right) \right]$$

where $\vec{\Delta} \in \{+1, -1\}^n$ random perturbation


QUANTUM GEN-AI
GAN
DM
GAN (generative adversarial network)
GENERATOR
Random Noise
fake data
DISCRIMINATOR
REAL
FAKE
Real data
?


Q-GANs
When either the generator player or the discriminator one exploit some QPU (QUANTUM PROCESSING UNITS)
GENERATOR
RANDOM NOISE
PQC
fake noisy QC
SUPER Q-GANs
DISCRIMINATOR
FAKE
REAL
NOISE SENSING OF QPU
real noise QC


QUANTUM DMs
CLASSICAL VS. QUANTUM NOISE
DIFFUSION
FORWARD PROCESS
ANN
DENOISING
BACKWARD PROCESS
ANN VS. PQC