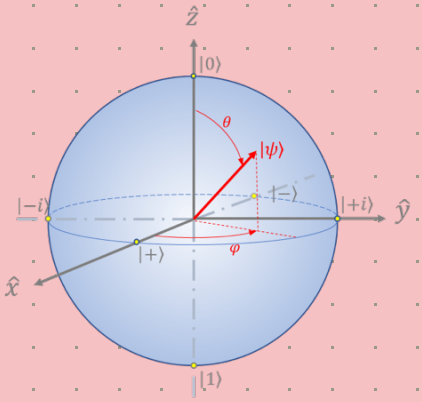


Beginner guide to quantum computing and machine learning

Welcome to quantum Realm!

Qubits:



- Similar to an electron in a cloud.
- No specific value (lies b/w 0 & 1)

Representation:

① **Dirac Notation:**

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

α, β → How much 0 & 1 are part of it.

α, β are amplitudes (complex no) of classical states $|0\rangle$ & $|1\rangle$

A ket

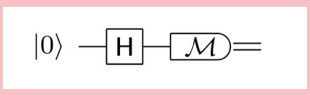
It represents a column vector (quantum state)

where, $|\alpha|^2 + |\beta|^2 = 1$ (similarly like probability adding upto 1)

when state $|\psi\rangle$ is measured $|0\rangle$ or $|1\rangle$ is observed.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

A Qubit state ($|\psi\rangle$) is a linear combination (or superposition) of these two.



1. A simple Quantum circuit acting as a perfect random bit generator.

! Dirac Notation

$\langle \ldots. |$ = Bra

$| \ldots. \rangle$ = ket

$\langle$ Bra | ket $\rangle$

Measuring a Qubit

- Probability of getting a 0 is $|\alpha|^2$
- Probability of getting a 1 is $|\beta|^2$

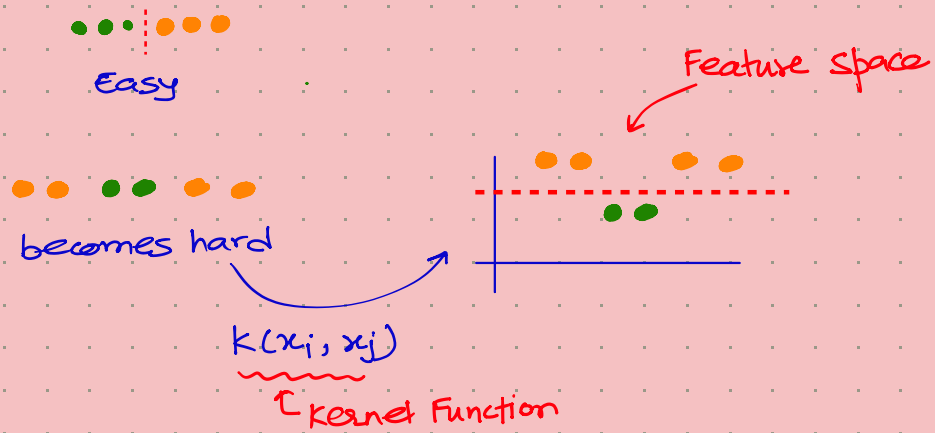
Quantum Register ($|\psi\rangle$)

- If there're 'n' qubits a quantum register lives in a 2^n Hilbert space

Grover's Algorithm

- An Algorithm developed by Lov Grover's in 1996
- Was used to provide a optimized solution to a why quadratic problem.

classification is easy when:



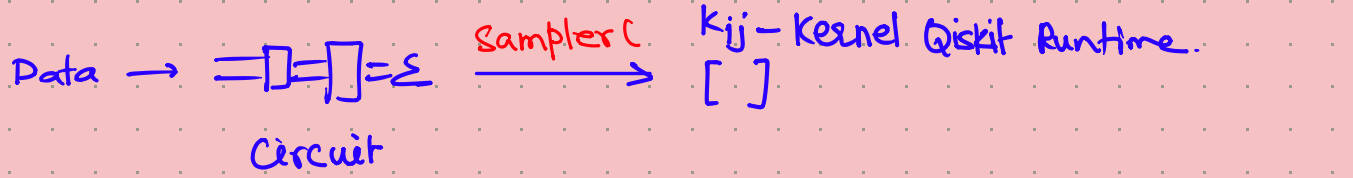
In 2021 IBM Research : Proved that 'Quantum Kernels' can provide exponential speed up over the classical counterparts for certain classes of classification problems.

Qiskit Runtime : to easily built QML Algo's with built tools such as 'Sampler Primitive' which primitives are unique to the IBM's Qiskit Runtime.

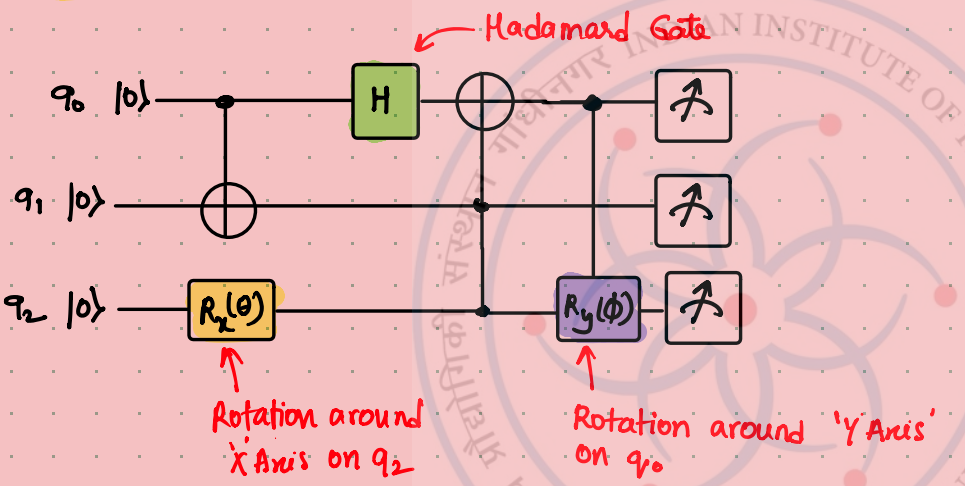
These are essentially predefined programs that help us to optimize workflows & execute them efficiently on quantum systems.

Linear Regression

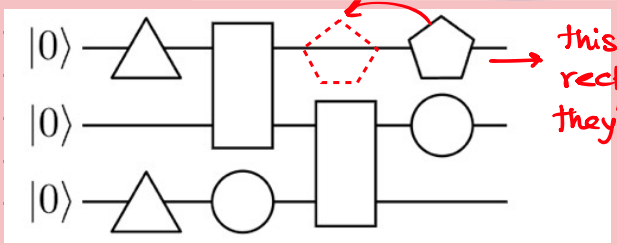
- we have data & we have encoded it in a quantum circuit



Quantum Circuits

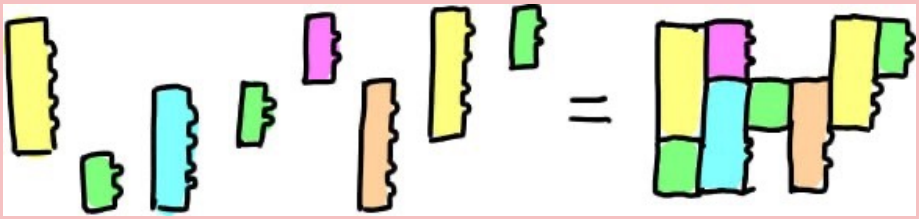


- Operations on qubits are often called 'Gates'



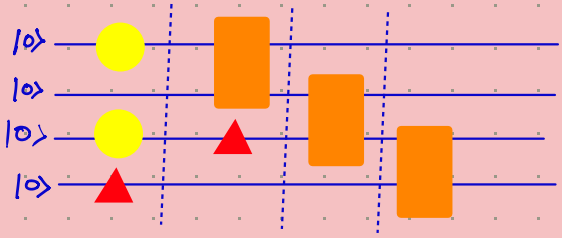
this pentagon can be shifted above the rectangle on qubit 0 & 1 to show that they're working parallelly.

- Imagine these shapes as operations on qubits, quantum operations which are performed on different qubits can be applied

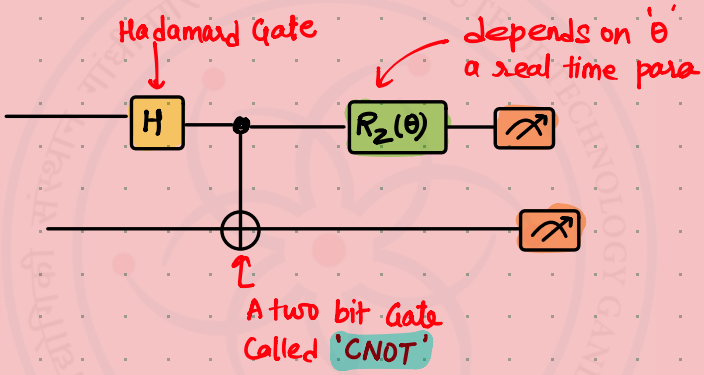


Exercise I.2.1. Draw the circuit diagram for a 4-qubit circuit from the following set of instructions:

Initialize all the qubits in
Apply a circle operation to qubit 0
Apply a circle operation to qubit 2
Apply a triangle operation to qubit 2
Apply a triangle operation to qubit 3
Apply a rectangle operation between qubits 0 and 1
Apply a rectangle operation between qubits 1 and 2
Apply a rectangle operation between qubits 2 and 3
Measure all the qubits



Depth is 4



In pennylane, unitary operations specified by a matrix can be implemented in a quantum circuit using the Qubit Unitary operation, Qubit operator is a paramatcrized Gate.

Unitary Matrices

$$UU^\dagger = U^\dagger U = I_n,$$

where I_n is the n-dimensional identity matrix. $U^\dagger$ is the notation for the conjugate transpose or Adjoint of U (i.e take the transpose of the matrix & replace each entry with its complex conjugate)

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

$|\alpha|^2$ is the probability of getting a $|0\rangle$ out of this Quantum state &

$|\beta|^2$ is the probability of getting a $|1\rangle$ out of this Quantum state.

Kronecker's product or Tensor Product $\otimes$

when there are many no's of Qubits in a System we use this product to get the combined result.

Polar Form

Express: $z = -\sqrt{3} - i$ in polar form.

If $z = x + iy$ is a complex No.

Polar form:

$$z = r(\cos\theta + i\sin\theta) \text{ where}$$

$$r = |z| = \sqrt{x^2+y^2} \text{ and } \theta = \text{Arg } z$$

$$-\pi \le \theta \le \pi$$

$$\tan\alpha = \left|\frac{y}{x}\right| \quad 0 \le \alpha \le \frac{\pi}{2}$$

III)

$x < 0 \quad y > 0$

$\theta = \pi - \alpha$

The Pauli Z Gate

Rz or 'Z rotation' is a quantum gate that modifies the relative path b/w $|0\rangle$ & $|1\rangle$ in a more general way.

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

$$R_z(\omega)|\psi\rangle = \alpha|0\rangle + \beta e^{i\omega}|1\rangle$$

The matrix representation of Rz is a bit-counter-intuitive at first.

$$RZ(\omega) = \begin{pmatrix} e^{-i\omega/2} & 0 \\ 0 & e^{i\omega/2} \end{pmatrix}$$

It doesn't changes the probability of a qubit to produce a $|0\rangle$ or $|1\rangle$ but changes the way it interacts with other qubits.

quantum support vector machine

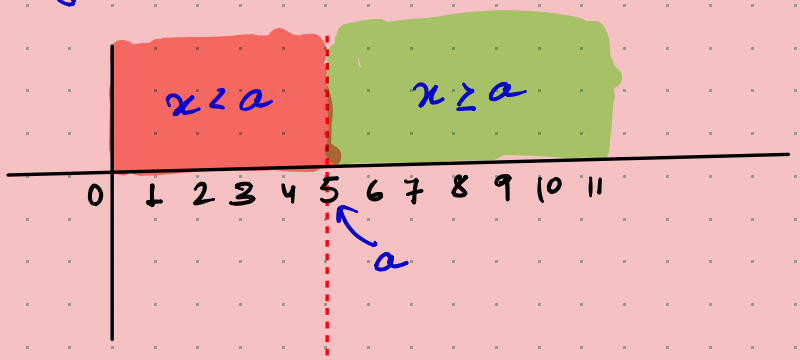
- most generally binary classification.

Classical Support Vector Machine

- SVM uses kernel tricks for classification.

Simplest classifier you could think:

- Imagine a classifier



lets try to do the same with real plane:

- Any line can be characterized in terms of a vector $\vec{w} \in R^2$ and a scalar $b \in R$ as set of points $\vec{x} = (x, y)$ such that $\vec{w} \cdot \vec{x} + b = 0$. Of course, we are using $\cdot$ to denote the scalar product

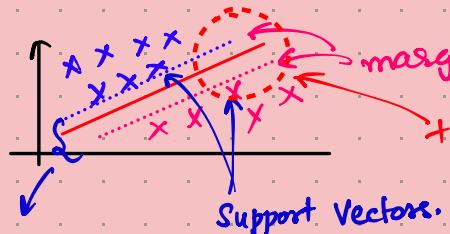
$$\vec{w} \cdot \vec{x} = w_1 x + w_2 y, \text{ provided that}$$

$$\vec{w} = (w_1, w_2)$$

- the vector $\vec{w}$ defines the normal or perpendicular direction to the line

① Classifier ② Regression.

Support Vector Classifier

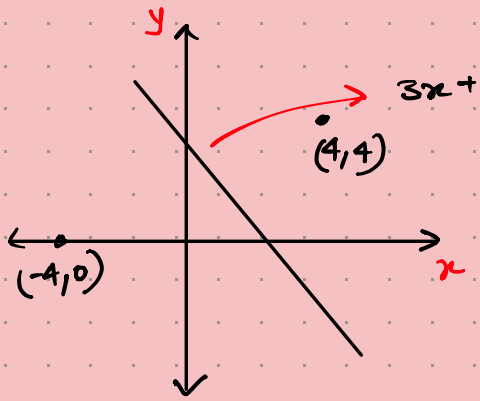


A line eqn.

$y = mx + c$

$y = \theta_0 + \theta_1 x$

$y = \beta_0 + \beta_1 x$



$3x + 2y + 4 = 0$

↳ Now for $(-4, 0)$ eqn gives:

$3(-4) + 2(0) + 4 = \text{line}$

$-12 + 0 + 4$

$-8 \rightarrow -ve$

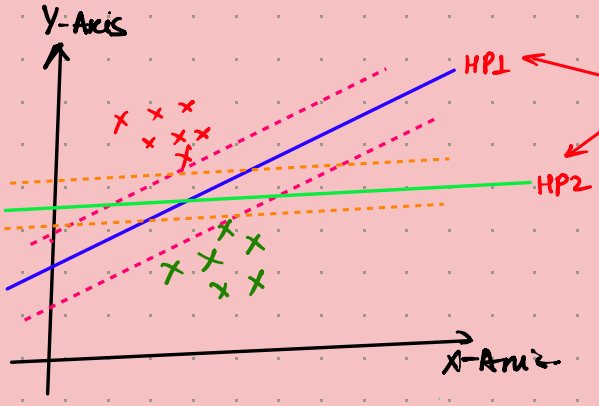
for $(4, 4)$:

$3(4) + 2(4) + 4 = 0$

$12 + 8 + 4$

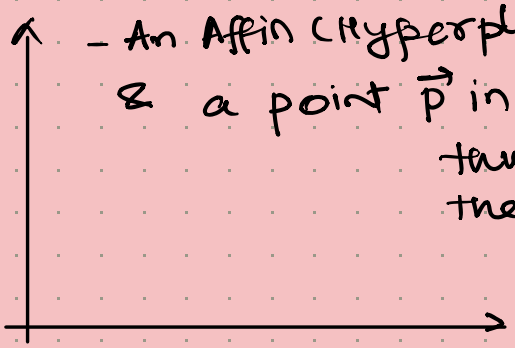
$24 \rightarrow +ve$

- here you can see anything below the line will give negative result & anything above it is producing +ve result.



both are valid! but the one with max distance b/w both margins is the best fit

These, hyperplanes in $\mathbb{R}^n$ is defined by some $\vec{w} \in \mathbb{R}^n$ and some constants $b \in \mathbb{R}$. In analogy what we saw in $\mathbb{R}^2$, their points are the $\vec{x} \in \mathbb{R}^n$ that satisfy the eqn of the form: $\vec{w} \cdot \vec{x} + b = 0$



- An Affin Hyperplane can be defined by vector $\vec{w}$ & a point $\vec{p}$ in the plane, thus a point $\vec{x}$ will belong to the hyperplane if & only if

$$\vec{x} = \vec{p} + \vec{v}$$

for some vector $\vec{v}$ which is orthogonal to $\vec{w}$.

$$\vec{w} \cdot (\vec{x} - \vec{p}) = 0$$

$$\vec{w} \cdot \vec{x} + (-\vec{w} \cdot \vec{p}) = \vec{x} \cdot \vec{w} + b = 0$$

where we implicitly define

$$b = -\vec{w} \cdot \vec{p}.$$

Mathematical Computation of SVM

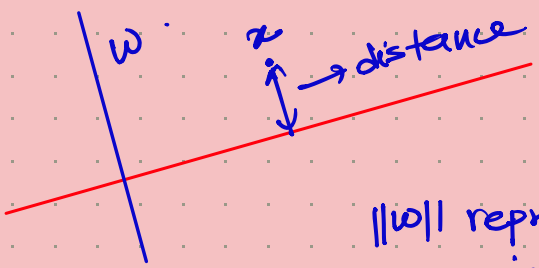
Dataset → Input feature vectors 'x' & their corresponding labels 'y'

eqn of Linear Hyper-plane can be written as:

$$w^T x + b = 0$$

where,

1) w is the normal vector to the Hyperplane
2) b is the offset or bias term representing the distance of Hyperplane from origin along the normal vector w.



$\|w\|$ represents the Euclidean norm of weight vector 'w'. Euclidean norm of the normal vector w.

Kernel

A Kernel is a function which Maps data into Higher dimensional space without explicitly computing the co-ordinates in that space.

Types

1) Linear kernel : For linear separability

2) Polynomial kernel: Maps data into polynomial space

3) Radial Basis Function. Transform data into space based on distance b/w data points

Linear SVM Classifier

distance b/w a Data point to the Hyperplane.

$$\hat{y} = \begin{cases} 1 & : \vec{w}\vec{x} + b \geq 0 \\ 0 & : \vec{w}\vec{x} + b \leq 0 \end{cases}$$

$\hat{y}$ is the predicted label of Data point.

Quantum Support Vector Machine



$P(|00\rangle) = 0.25$
$P(|01\rangle) = 0.25$
$P(|10\rangle) = 0.25$
$P(|11\rangle) = 0.25$

Kernel Trick: If data is not linearly separable in input data space we project it to a higher dimensional space to make it linearly separable in that higher dimensional space.

The Lagrangian Dual form

Maximize : $\sum_j a_j - \frac{1}{2} \sum_{j,k} y_j y_k \alpha_j \alpha_k (\bar{x}_j \cdot \bar{x}_k)$

↳ this is the part which we need to run on a Q.C

- we might use 'kernel Trick' so we need a feature map first that can project the data to a higher dimensional space

So, let
ϕ be a feature map.

$$\phi : R^n \rightarrow R^d \qquad \text{where } d \gg n$$

$$K(\vec{x}_j, \vec{x}_k) = (\phi(\vec{x}_j) \cdot \phi(\vec{x}_k))$$

K: Kernel Function; $\vec{x}_j, \vec{x}_k$: Input vectors; $(\phi(\vec{x}_j) \cdot \phi(\vec{x}_k))$: inner product (classically)

$$K(\vec{x}_j, \vec{x}_k) = (\phi(\vec{x}_j) \cdot \phi(\vec{x}_k))$$

$$K(\vec{x}_j, \vec{x}_k) = |\langle \phi(\vec{x}_j) | \phi(\vec{x}_k) \rangle|^2$$

Kernel Fidelity Alignment.

$|\langle \phi(\vec{x}_j) | \phi(\vec{x}_k) \rangle|^2$: inner product in a Quantum way.

Let $\phi(\vec{x})$ = Quantum circuit that creates $\phi(\vec{x})$ when acted on $|0\rangle$.

$$k(\vec{a}, \vec{b}) = |\langle \psi(a) | \psi(b) \rangle|^2 = |\langle 0 | \phi^\dagger(a) \phi(b) | 0 \rangle|^2$$

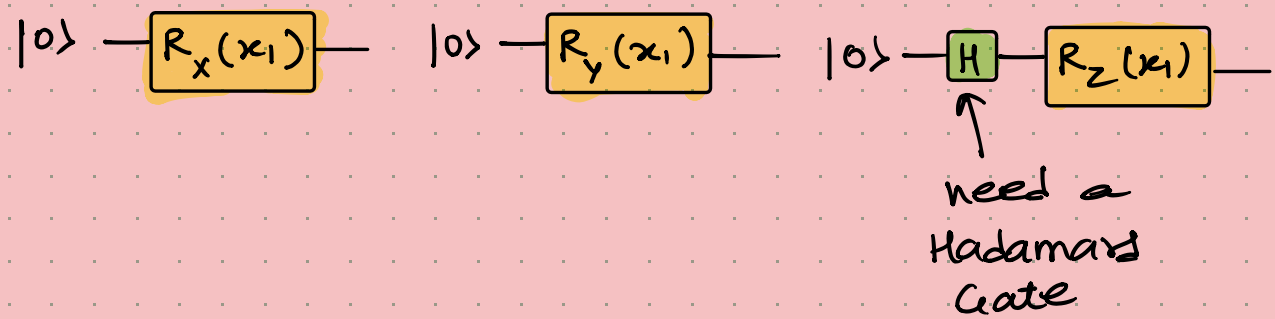
Feature Map

Angle Encoding

- simple yet powerful.
- If used on 'n' qubit circuit, it can take upto 'n' numerical inputs $x_1, \ldots x_n$.
- In angle encoding we are free to use any rotation gate of our choice

- Using R_z gate with initial state $|0\rangle$ will have no effect whatsoever. (definition of R_z Gate)

$$Z|0\rangle = +|0\rangle$$
$$Z|1\rangle = -|1\rangle$$



Points:

1) If a feature is being fed in a feature map it should be mapped within a certain interval:

for ex: if data is mapped b/w 0 & 4π.
then the data will be mapped in a wider region than if they were normalized b/w 0 and 1.

Amplitude Encoding

- this feature map can take 2^n inputs when implemented on an n-qubit circuit.

$$|\psi(\vec{x})\rangle = \frac{1}{\sqrt{\sum_k x_k^2}} \sum_{k=0}^{2^n-1} x_k |k\rangle$$

Input: $x_0, x_1, x_2 \ldots\ldots x_n$.

Results are normalized

Encoding (Classical Data → Classical Data)

Feature Map

Its a parameterized circuit $\phi(\vec{x})$ that depends on original data & thus can be used to prepare a state that depends on it.

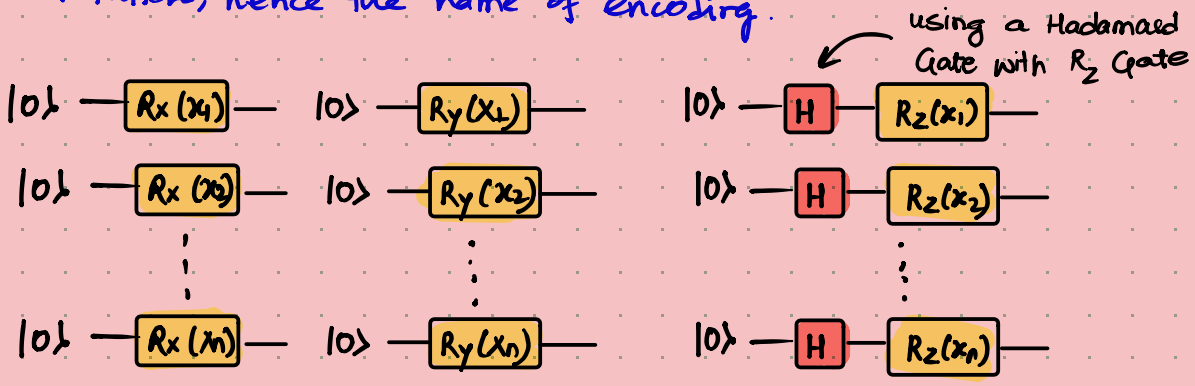
Feature map defines a mapping from: Classical Feature Space → Quantum Hilbert Space

Types of Feature Maps:

1. Angle Encoding

- When used on 'n' qubit circuit, this feature map can take upto 'n' numerical inputs $x_1, \ldots, x_n$.

Circuit: the action of its circuit consists in the application of a rotation gate on each qubit 'j' parameterized by value x_j, we are using x_j values angles in the rotation, hence the name of encoding.



Angle encoding for an input $(x_1, \ldots, x_n)$ using different rotation gates.

The variables that are fed to the Angle encoding feature Map should be normalized within an interval.

$0-4\pi$: If normalized b/w $0-4\pi$ then data will be mapped to a wider region of the feature space than if they will be normalized b/w 0 & 1.

What Actually Happen in Angle Encoding?

A Input (lets say x_j) goes into circuit.

→ $R_x(x_j) = e^{-i x_j X/2}$

A Rotation Gate.

↑ where x_j will act as an angle θ.

So this is how each classical input controls how much a qubit should rotate which keeps information in quantum state.

ex

Let feature 'x', Applying R_x, so it becomes

$$|0\rangle \rightarrow R_x(x)|0\rangle = \cos(x/2)|0\rangle + \sin(x/2)|1\rangle$$

So, the probabilities of getting 0 & 1 Encode the feature.

Suppose:

$$x = [0.1,\ 0.2,\ 0.3]$$

$$|\phi(x)\rangle = R_y(0.1) \otimes R_y(0.2) \otimes R_y(0.3)$$

BAM! It is now Angle Encoded!

Amplitude Encoding

Assume a Input Vector:

$$x = [x_0, x_1 \dots x_n]$$

$$n\text{-qubit} = [\log_2 N]$$

$$|x\rangle = \sum_{i=0}^{N-1} \tilde{x}_i |i\rangle, \quad \tilde{x}_i = \frac{x_i}{\|x\|_2}, \quad \|x\|_2 = \sqrt{\sum_{i=0}^{N-1} x_i^2}$$

What is Amplitude Encoding?

- A technique used to encode information into the probability amplitude of quantum states. In the quantum realm

Let,

Ex: $x = [2, 0, 1, 2]$

So, $|x| = \sqrt{2^2 + 0^2 + 1^2 + 2^2} = \sqrt{9} = 3$

Now, for 4 Numbers only 2 (log n) qubits will be required & it will be encoded as:

$$|\psi\rangle = \underbrace{\tfrac{2}{3}|00\rangle + \tfrac{0}{3}|01\rangle + \tfrac{1}{3}|10\rangle + \tfrac{2}{3}|11\rangle}_{x\text{-Amplitude encoded}}$$

ZZ - Feature Map : Amplitude Encoding

