Introduction to
Quantum Artificial Neural Networks

Summer School: 23.06 – 5.09.2025 (26.06)
Faculty of Automation and Computer Science, POLITEHNICA Bucharest
I.V. Grossu

This work focuses on providing **intuitive** representations and analogies to facilitate the assimilation of specific **Quantum Computing and Artificial Neural Networks (ANN)** concepts by students and other interested individuals. First, the classical **artificial neuron** is discussed along with its limitations to linearly separable problems. Various multi-dimensional intuitive representations are considered for a better understanding of the **Multi-Layer Perceptron (MLP)** and the corresponding **Backpropagation** training algorithm. The main Quantum Computing concepts are discussed in relation to the **"Semaphores Analogy"** and the **Bloch Sphere**. A **single-qubit neural circuit for solving Exclusive-Or (XOR)** is also presented (Grossu - 2020).

Introduction to Artificial Neural Networks

- Introduction
  - Multi-Dimensional Spaces
  - Fuzzy Logic Basics
  - ANN Overview
- Artificial Neuron
  - The Mc-Culloch & Pitts Model
  - Graphical Interpretation and Examples
  - The Sigmoid Activation Function
- Artificial Neural Network
  - Neuron Limitations
  - The Connectionist Paradigm
  - The Multi-Layer Perceptron (MLP)
- Training MLP by Backpropagation
  - Gradient Descent Methods Basic Concepts
  - Algorithm Description

References [1-3]

Introduction
Multi-Dimensional Spaces

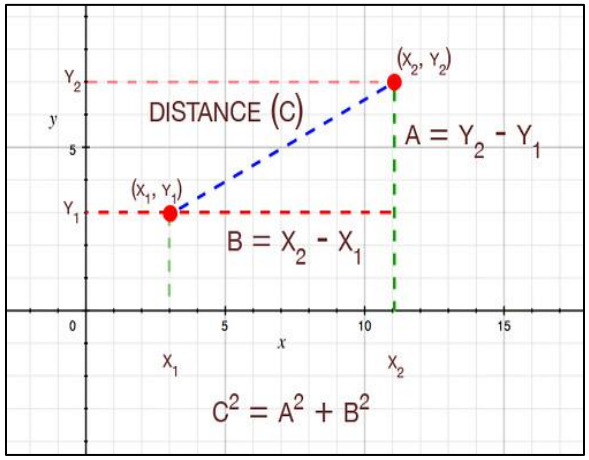
- **Cartesian Product:** considering two sets, A and B:

$$A \times B = \{(X, Y) | X \in A, \quad Y \in B\}$$

- **Plane**: ***R^2 = R X R =>*** *(x, y)*
  - Distance: Pythagoras

$$d(A, B) = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

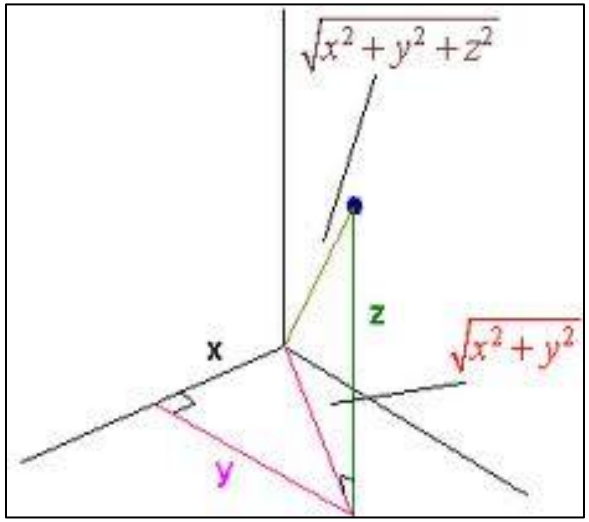
  - Circle: all equidistant points with respect to a fixed point
  - Line: **ax + by + c = 0**



- **3D Space:** ***R^3 = R X R X R =>*** *(x, y, z)*
  - Distance: Pythagoras applied 2 times

$$d(A, B) = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2}$$

  - Sphere**:** all equidistant points with respect to a fixed point
  - Plane: **ax + by + cz + d = 0**



Introduction
Multi-Dimensional Spaces

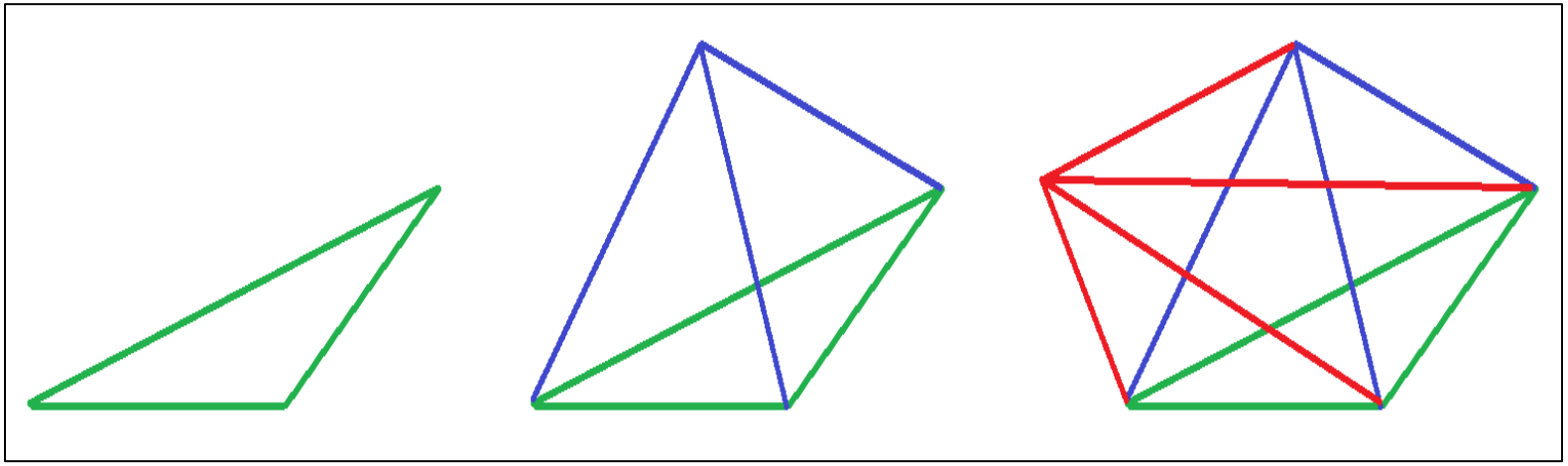
- ***R^n = R X R X … X R =>*** *(x_1, x_2, …, x_n)*
  - Distance: Generalized Pythagoras

$$d(A,B) = \sqrt{\sum_{i=1}^{n} (X_{2i} - X_{1i})^2}$$

  - Hyper-Sphere**:** All equidistant points with respect to a fixed point
  - Hyper-Plane:

$$a_0 + \sum_{i=1}^{n} a_i X_i = 0$$

- ***C^n = C X C X … X C =>*** *(z_1, z_2, …, z_n)*



Hyper-Tetrahedron

Introduction
Fuzzy Logic Basics

- **Characteristic function of a classical set:**

$$f_A(x) = \begin{cases} 1, & X \in A \\ 0, & X \notin A \end{cases}$$

- **Classical Point -** the set containing the point *y*:

$$f_Y(X) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases}, \qquad y \in X$$

- **Distance:**

$$d(y, z)$$

- **Fuzzy Set *F* =>** a pair *(X, f)*
  - *X is a classical set (the universe of F)*
  - ***f:X->[0,1] -*** the **characteristic function** of *F*

- **Fuzzy Point -** the set containing the point *y* in proportion *a*:

$$f_Y^a(X) = \begin{cases} a, & x = y \\ 0, & x \neq y \end{cases}, \quad a \in [0,1], \quad y \in X$$

- **Distance for fuzzy points:**

$$d(f_y^a, f_z^b) = \min(a, b) \times d(y, z)$$

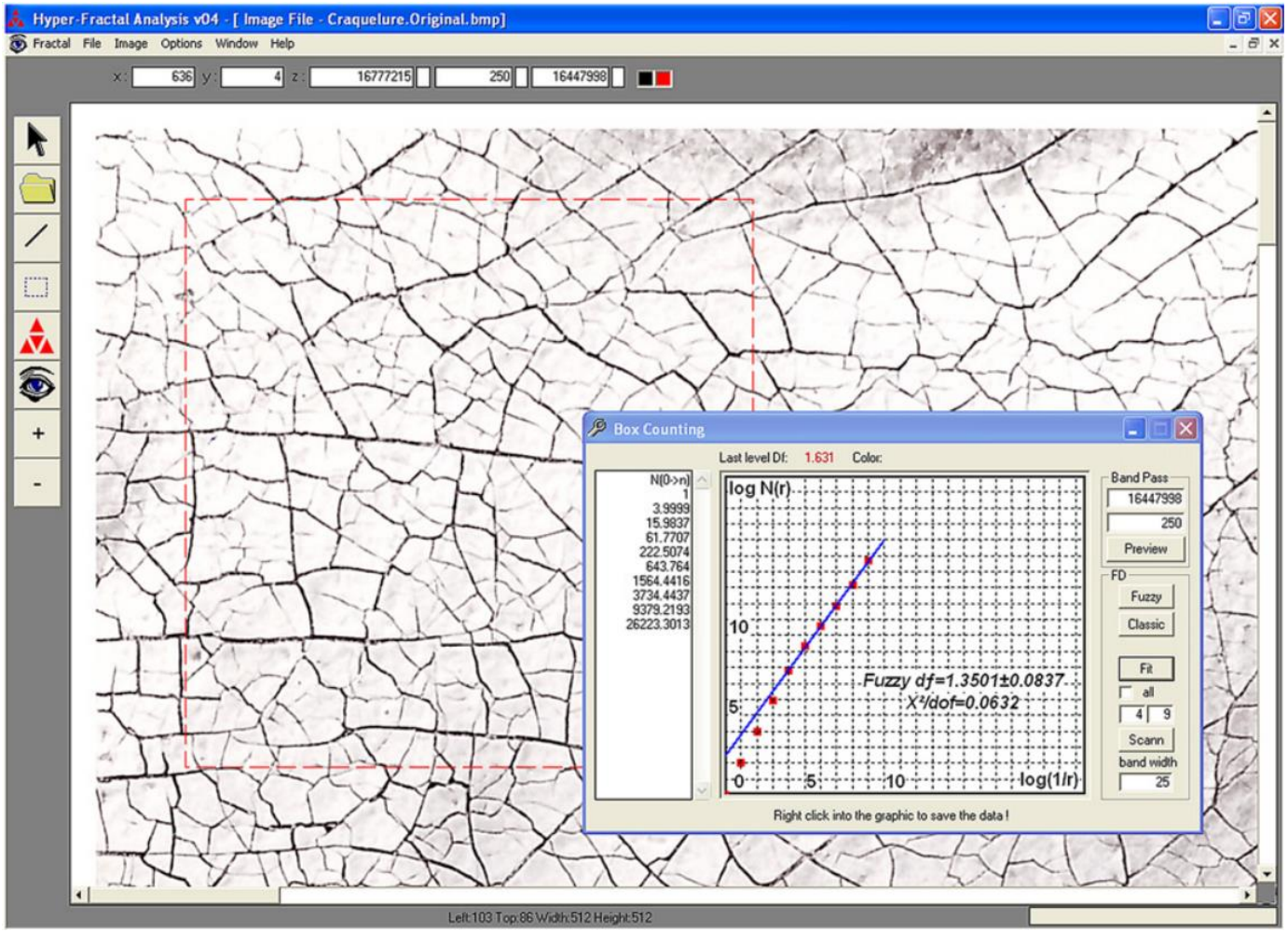
$$f_{A \cap B}(X) = \min(f_A(X), f_B(X))$$

$$f_{A \cup B}(X) = \max(f_A(X), f_B(X))$$

Introduction

Fuzzy Logic - Example

For 256 shades of gray images, one could consider black points (0) are 100% part of the fuzzy set, while white points (255) are not contributing to the information of interest [2]. The **membership degree** of all other pixels will be obtained by linear interpolation.



Introduction
Artificial Neural Networks (ANN) Overview

- Inspired by Natural Neural Networks
- As opposed to a classical approach, **does not implement a model but detect it from data**
- Are able to “learn” **f : R^n --> R^m** functions
- Are able to provide the expected output **even for noisy/incomplete information**
- Learning paradigms
  - Associator: e.g. Bidirectional Associative Memories (BAM)
  - Classificator: e.g. Multi-Layer Perceptron (MLP), Kohonen
  - Other
- Training paradigms
  - **Supervised**: the expected output is known for a set of input vectors (training set)
  - **Unsupervised**: the expected output and the number of classes is not known
  - Semi supervised learning (a mix between the previous two)

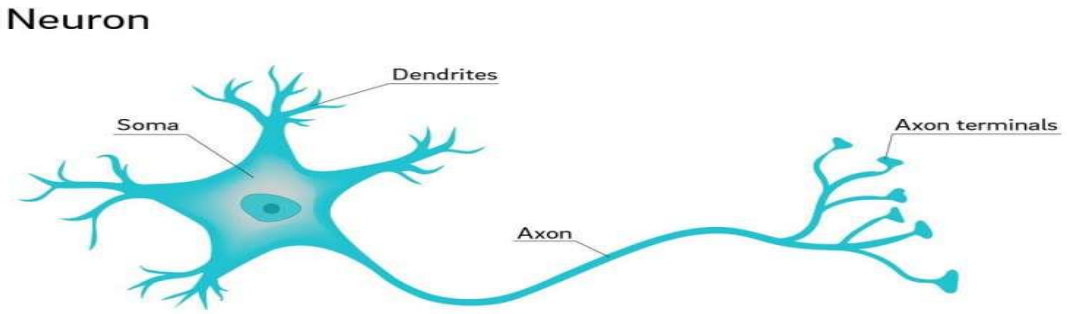
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Artificial Neuron

The Mc-Culloch & Pitts Model

Neuron model (McCulloch - logician & Pitts – neurobiologist, 1943)



WEIGHTS

THRESHOLD

X_1 W_1 t

X_2 W_2

INPUT VECTOR

W_n Y

OUTPUT

if ***(S > t)***

Y = 1

else

Y = 0

X_n

ACTIVATION

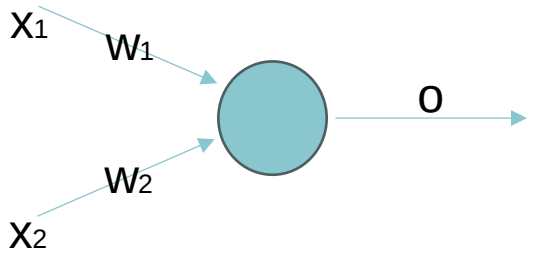
$$S = \sum_{i=1}^{n} W_i X_i$$

Artificial Neuron
Graphical Interpretation and Examples

2D: the neuron is “splitting” the space in two half-planes (classes)

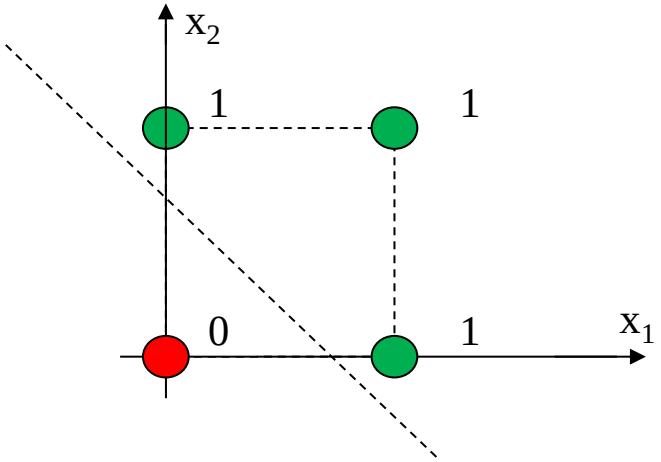
$$x_1 w_1 + x_2 w_2 + t = 0 \Rightarrow$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{t}{w_2} = a x_1 + b$$



Example: One could use a 2D neuron for “learning” the OR function:

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1



Interesting: We get the expected output even for points around the training set e.g. (0.1, 0.05)

Generalization: in an nD space the separation set is a hyper-plane

Artificial Neuron
The Sigmoid Activation Function

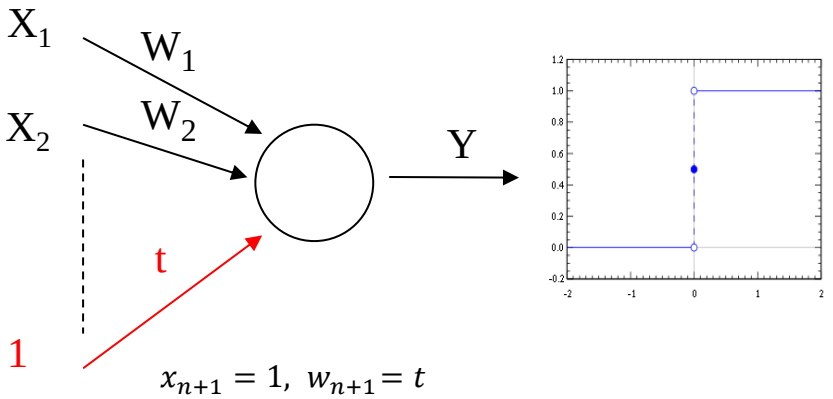
One could replace the threshold *t* with an additional virtual input, which value is always one:

- Thus, during the training process **the threshold could be adjusted together with the weights**

$$Y = f(S') = f(\sum_{i=1}^{n} w_i\, x_i + t) = f(\sum_{i=1}^{n+1} w_i\, x_i)$$

Step Activation Function

$$f(x) = \begin{cases} 1, & X \geq 0 \\ 0, & X < 0 \end{cases}$$

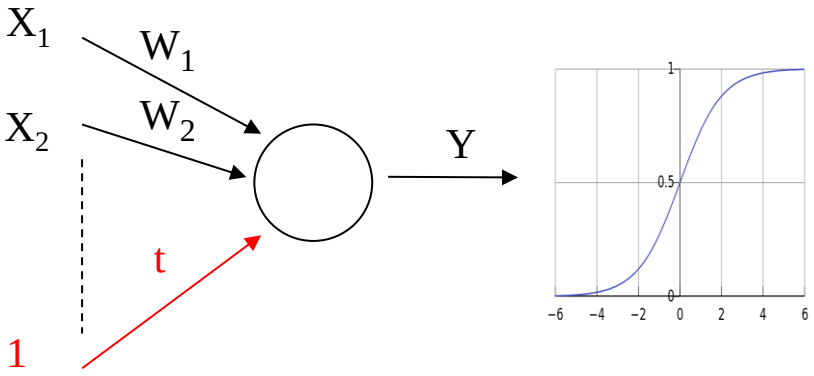


One could consider a sigmoid activation function (either unipolar => values in [0,1] or bipolar [-1,1]):

- **Differentiable** – important, for example, in implementing **gradient descent training algorithms**
- **Continuous** – important in implementing **fuzzy logic**

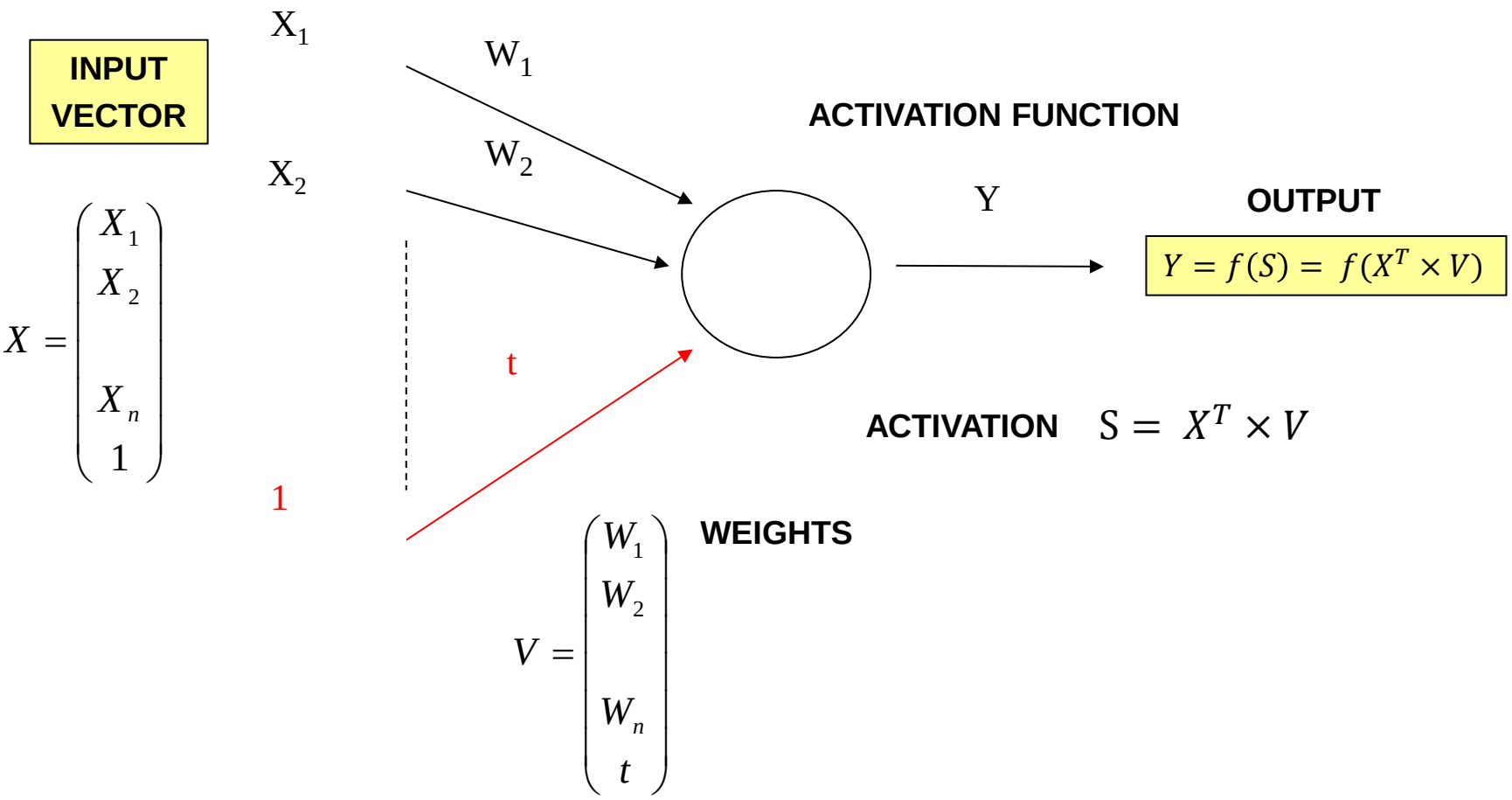
Sigmoid Activation Function

$$f(X) = \frac{1}{1 + e^{-KX}}, \quad K > 0$$



Artificial Neuron

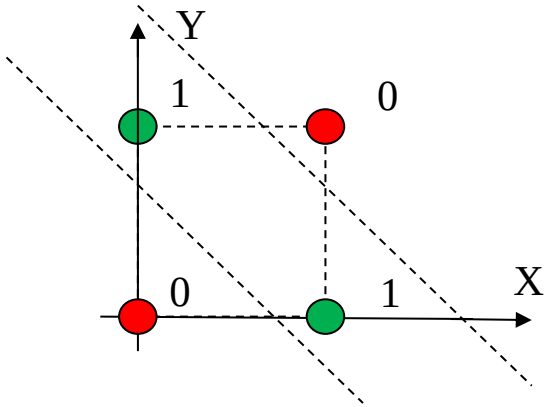
Matrix Representation

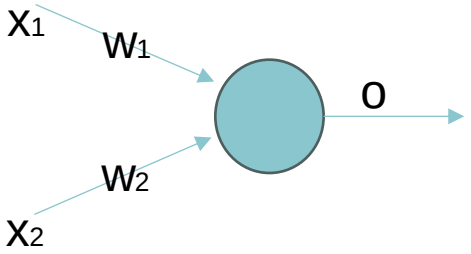


Artificial Neural Networks
Neuron Limitations

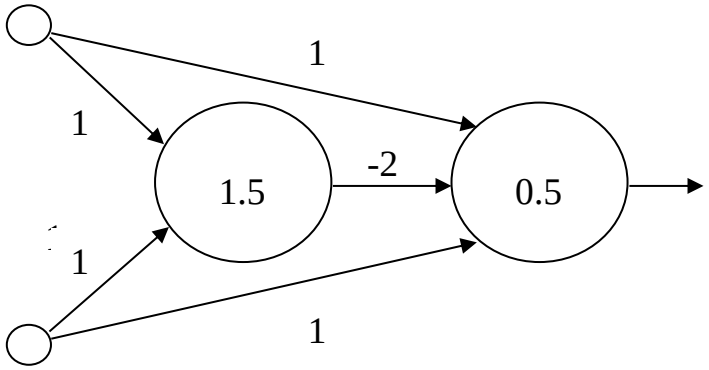
XOR: is not a linearly separable problem => **Cannot be solved with a single neuron**

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

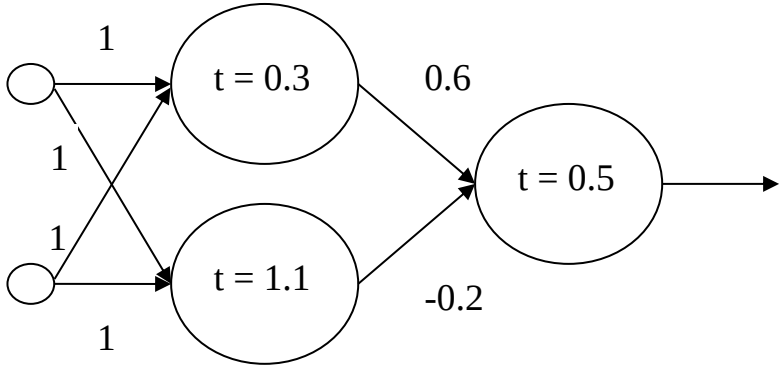




The idea of connecting more neurons => **The connectionist paradigm**



Example 1: ANN for XOR

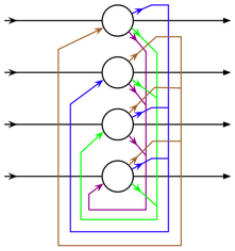


Example 2: MLP for XOR

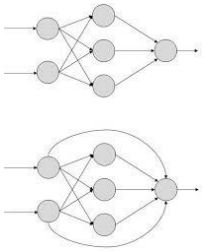
Artificial Neural Networks
The Connectionist Paradigm

ANN Topology

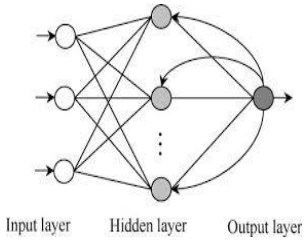
- **Classification by ANN connectivity**
  - Completely interconnected (e.g. Hopfield)
  - Locally connected (e.g. MLP)
- **Classification by ANN data propagation**
  - Feed-Forward
  - Feed-Back (recurrent)



Hopfield



Feed-Forward



Feed-Back

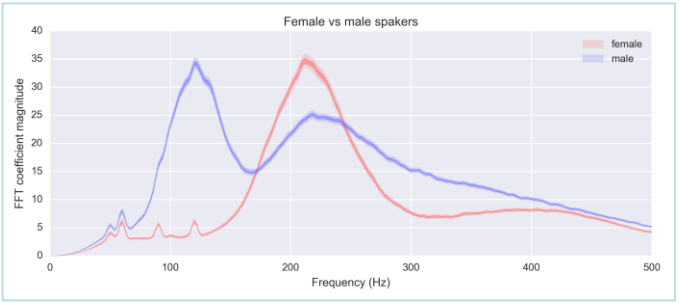
Training paradigms

- **Supervised learning:** the desired output is known for a set of input vectors => training set
  - Example: gradient descent methods (e.g. Back Propagation): based on minimizing the error seen as a function of all ANN weights
- **Unsupervised learning** (e.g. Kohonen): the output and the number of classes are not known
  - **Hebb's learning principle**: the synaptic weight between *i & j* is increased whenever the neurons are simultaneously stimulated: $\Delta w_{ij} = c y_i y_j$ *or* $\Delta w_{ij} = c S(y_i) S(y_j)$, where *c* is the learning rate, *y* the activation, and *S* the activation function
  - **Competitive learning:** more neurons are participating to the privilege of modifying their weights. e.g. **Winner-take-all** => for each input, select the winner (the neuron with maximum activation) and adjust only its weights
- **Semi-Supervised**:
  - Mix: unsupervised + provide some labeled data

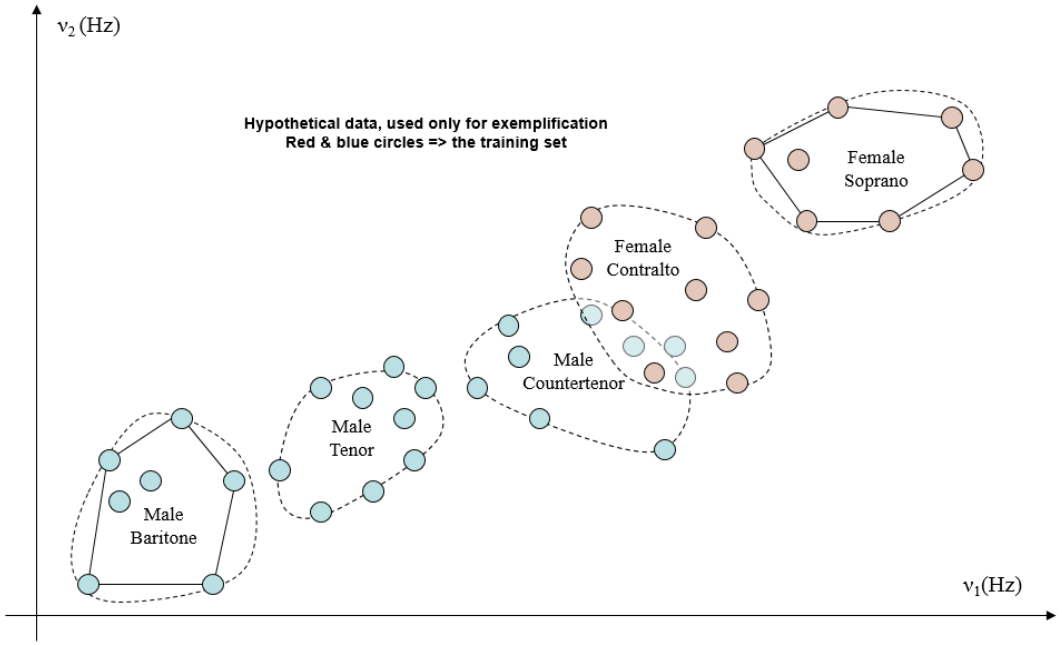
The Multi-Layer Perceptron
Example

- **Example:** considering the vector ($v_1, \ldots, v_n$) with the most significant frequencies (peaks) of a voice spectrum, our intend is to distinguish between male and female voices:

$$f(\vec{v}) = \begin{cases} 0, & \text{male} \\ 1, & \text{female} \end{cases}$$



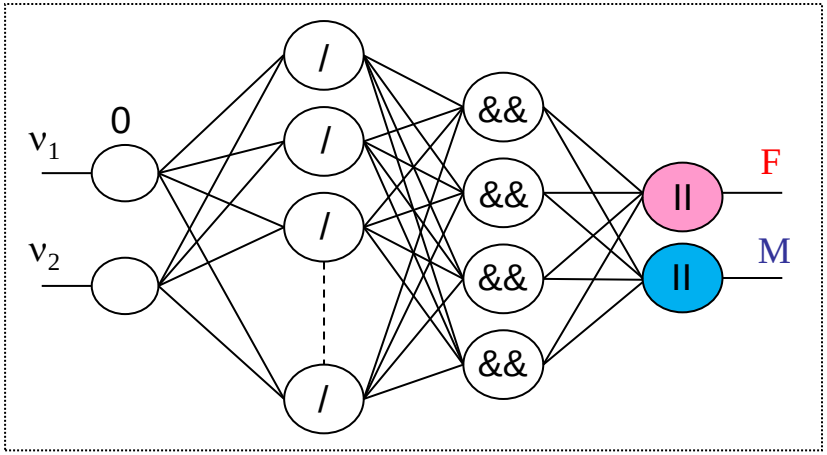
- The 2 classes correspond to a set of hyper-clusters in the multi-dimensional space of frequencies.
- For simplicity, we'll consider only two frequencies ($n=2$):

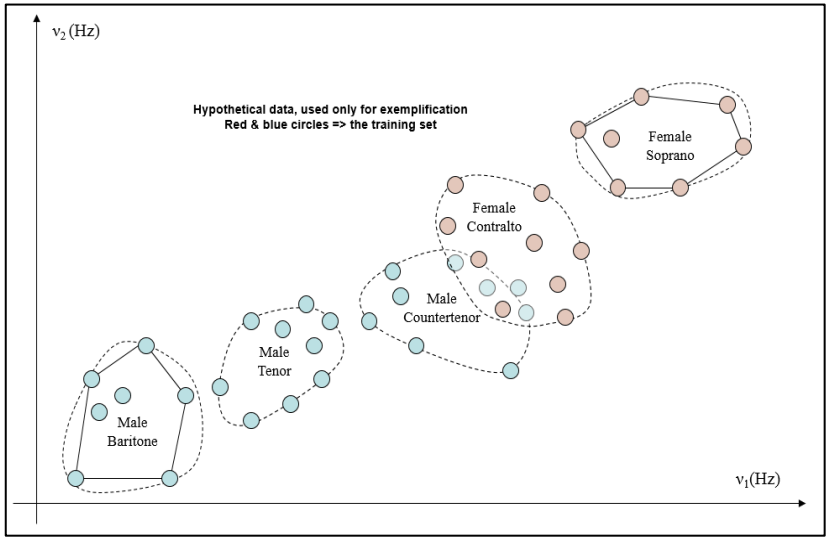


The Multi-Layer Perceptron
Graphical Interpretation

Intuitive representation:

- **The "zero" layer**: only a signal transmitter (number of neurons = number of input values)
- **The first hidden layer**: could be related to the **polygons edges** which will approximate each cluster
  - **Convolutional Neural Networks**: When the dimension of the input space is high (e.g. in image analysis 100x100 pixels => 10,000 dendrites per neuron), in order to avoid the difficulties involved by training an ANN with a huge number of weights, one could consider partially interconnecting the first layer (biology inspired model). For example, splitting the image in regions of 5x5 pixels, **with shared edges**, will result in only 25 dendrites per neuron.
- **The second hidden layer**: might act as a set of **AND operators (INTERSECTION)** in order to define each polygon (cluster approximation)
- **The output layer**: may act as **OR operators (UNION)** in order to define each class as a set of all corresponding clusters

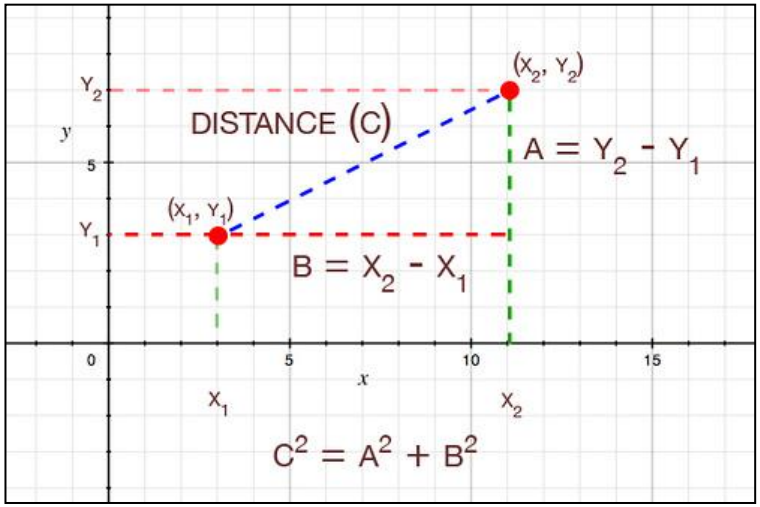




Backpropagation
Basic Concepts - Distance

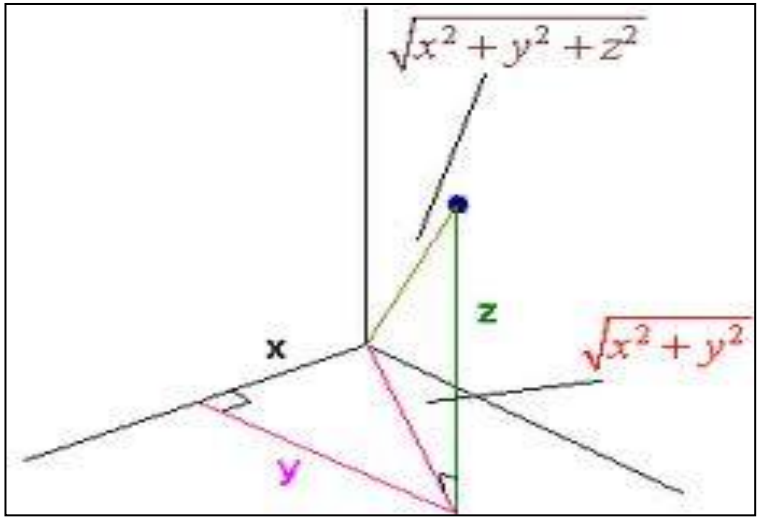
- **2D Pythagoras:**

$$d(A,B) = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$



- **3D Pythagoras 2 times:**

$$d(A,B) = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2}$$



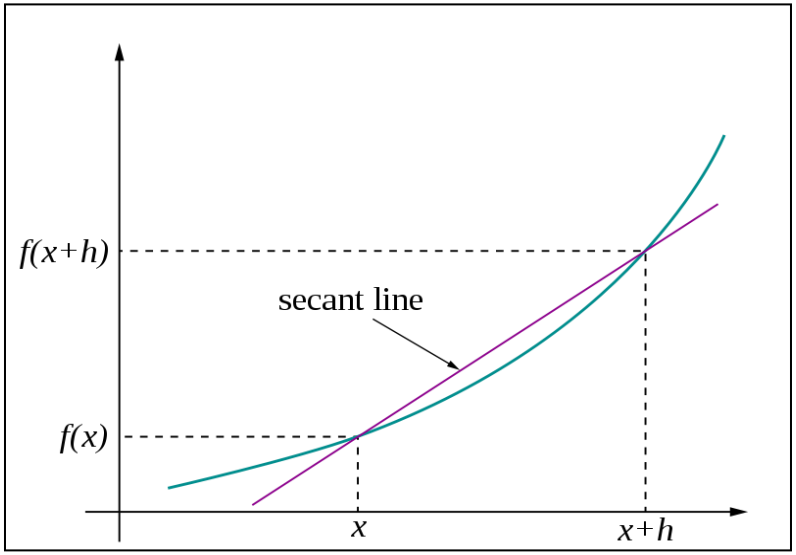
- **Generalization:**

$$d(A,B) = \sqrt{\sum_{i=1}^{n} (X_{2i} - X_{1i})^2}$$

Backpropagation
Basic Concepts - Gradient

- **Derivative**: the “velocity” with respect to a variable (time, space, temperature etc.)

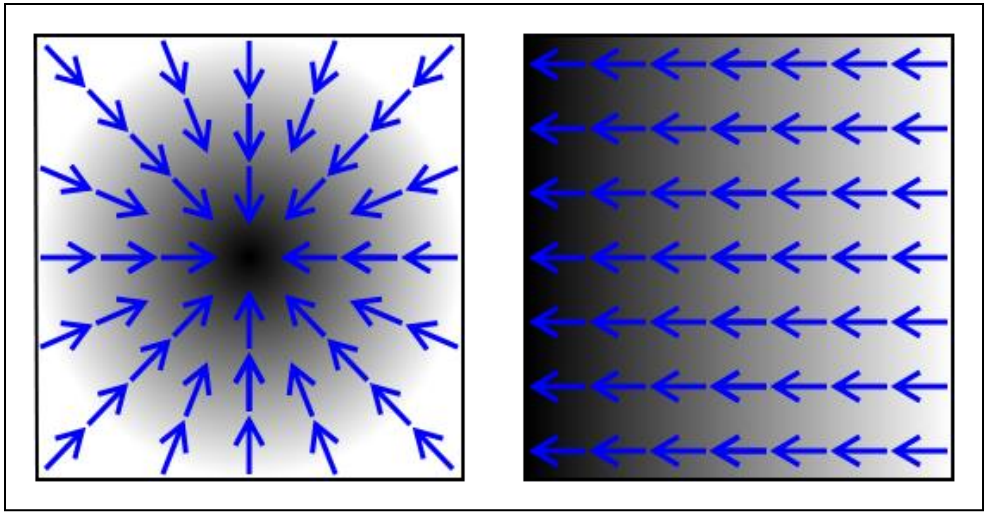
$$f'(x) = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x} \Rightarrow \Delta f(x) \approx \Delta x \, f'(x)$$



- **Gradient of a scalar *f*** : a **vector** indicating the direction on which *f* raises most quickly

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \quad \frac{\partial f}{\partial x_2}, \ldots, \quad \frac{\partial f}{\partial x_n} \right)$$

$$\nabla f = \left(\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y} \right)$$



Backpropagation
Basic Concepts

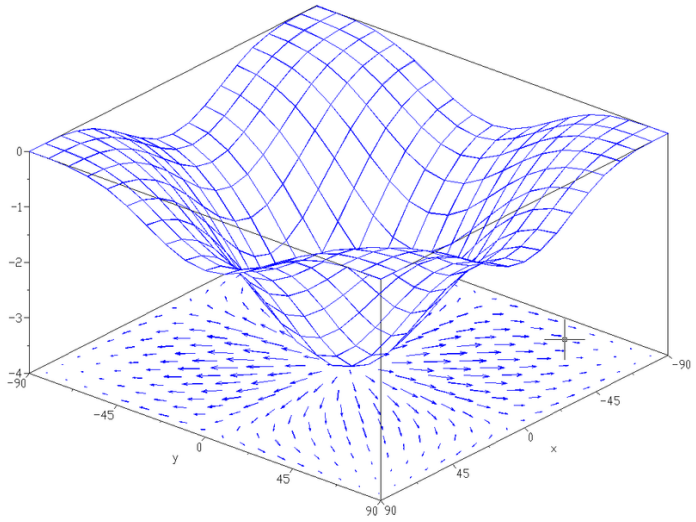
- **Supervised Learning:**
  - Training Set = {(x_1,d_1) $(x_2,d_2)\ldots$ (x_n,d_n)}
    - x_i is the input ("labeled information")
    - d_i is the **desired**, known output

- **Gradient Descent Methods:**
  - The main goal is minimizing a criteria function *J* (the classification error). One could consider the square distance between the output and the expected value:

$$J(k) = \sum_{i=1}^{n} e_{iK}^2 = \sum_{i=1}^{n} d^2(o_i, d_i)_K = \sum_{i=1}^{n} (o_i - d_i)_K^2$$

  - The ANN weights vector (v) is adjusted proportionally with the anti-gradient of *J*
    - ***J* is a function of all neurons weights**
    - *c* is the "learning rate" (which could be variable)

$$\boldsymbol{v^{K+1} = v^K - c\nabla J(v^K) = v^K - c\left.\frac{\partial J(v)}{\partial v}\right|_{v=v^K}}$$



Backpropagation
Algorithm Description

Backpropagation – all computation details are available in [1] (p.192-227).

- Choose the **learning rate *c in (0,1)***, a **target error**, and a **maximum number of iterations**
- Initialize the weight matrix (random small values)
- **While** the **perceptron is not trained** and the ***number of iterations < Max***
  - **For each** vector ***(x,d)*** in the training set (*d* => desired value)
    - Compute the hidden layer activation

$$h_q = \sum_{i=1}^{n} x_i v_{iq}$$

    - Compute the hidden layer output

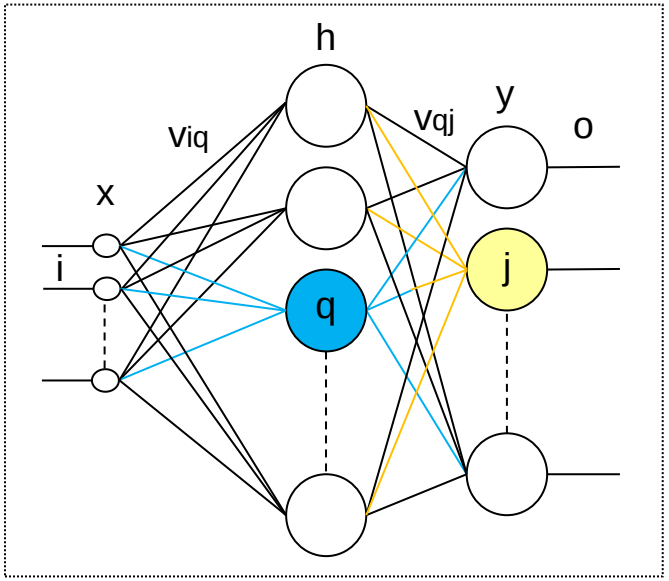
$$i_q = S_q^h(h_q)$$

    - Compute the output layer activation

$$y_j = \sum_{q} i_q v_{qj}^o$$

    - Compute the ANN output

$$o_j = S_j^o(y_j)$$



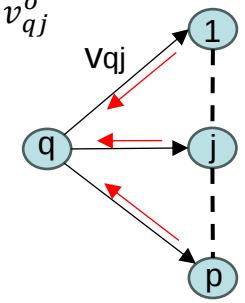
Backpropagation
Algorithm Description

- Compute the output “error signal”
- Compute the hidden “error signal”
- Adjust the output layer weights
- Adjust the hidden layer weights

$$\delta_j = (d_j - o_j)\frac{\partial S_j^o(y_j)}{\partial y_j}$$

back propagation

$$\delta_q^h = \frac{\partial S_q^h(h_q)}{\partial h_q}\sum_j \delta_j v_{qj}^o$$



$$v_{qj} \mathrel{+}= c i_q \delta_j$$

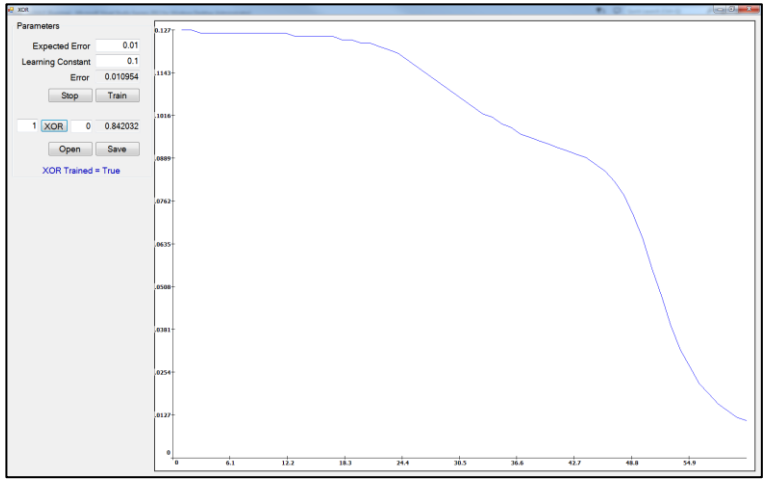
$$v_{iq} \mathrel{+}= c x_i \delta_q^h$$

- If *E* <= *target error*, the neuron is trained
- *E = 0*

$$E = \frac{1}{2}\sum_{j=1}^{p}(d_j - o_j)^2$$

Remarks:

- The training set is presented to ANN multiple times
- The algorithm **requires differentiable activation functions**
- For minimizing the risk of getting stuck into a **local minimum** one could use various methods (e.g. variable learning rate, simulated annealing, “weights shaking” etc.).



Introduction to Quantum Computing

- Qubit
- Quantum Register
- Quantum Entanglement
- Unitary Operators
- Measurement
- The Bloch Circle
- Examples of single qubit Gates
- Superposition or Statistical Mix?
- The Bloch Sphere
- Examples of two-qubits Gates
- Quantum Entanglement – Tests

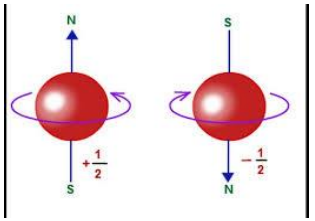
References [4-6]

Qubit

As opposed to classical bit, **one qubit has both 0 and 1 values at the same time**, which is quite counter intuitive (e.g. a sphere spinning in both directions at the same time).

A qubit could be represented as **a unit vector in C^2**:

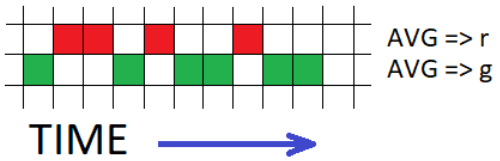
$$\Psi = \alpha|0> + \beta|1>, \quad |\alpha|^2 + |\beta|^2 = 1$$



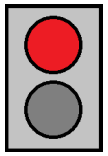
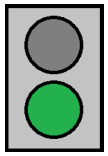
In Introduction to Quantum Computing [6], it was presented the **"Semaphores ANALOGY"**

- Qubit => semaphore which **randomly** switches from red to green and back with probabilities:

  r (for red), and *g* (for green) : ***r + g = 1***



- Supposing the semaphore switches fast enough from one color to another, the eye will "see" the average. Thus, one will perceive it is both red and green at the same time, but with different intensities:

$$|0> = \begin{bmatrix}1\\0\end{bmatrix}$$  $$|1> = \begin{bmatrix}0\\1\end{bmatrix}$$ 

- **Remark on Schrodinger's cat.** A cat could switch its status from alive to dead, but the opposite way is usually difficult to imagine.

Quantum Register

- In the frame of the same "semaphores analogy", one could consider two independent semaphores, **randomly** switching, enough fast, from red to green and back, with the corresponding probabilities:

$$(r_1, g_1), \text{ and } (r_2, g_2)$$

- The probabilities of compound states could be obtained by **multiplication of individual probabilities**:

$$r_1 * r_2,\ r_1 * g_2,\ g_1 * r_2,\ g_1 * g_2$$

- One could use the **Kronecker product**:

$$\begin{bmatrix} r_1 \\ g_1 \end{bmatrix} \otimes \begin{bmatrix} r_2 \\ g_2 \end{bmatrix} = \begin{bmatrix} r_1 \begin{bmatrix} r_2 \\ g_2 \end{bmatrix} \\ g_1 \begin{bmatrix} r_2 \\ g_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} r_1 r_2 \\ r_1 g_2 \\ g_1 r_2 \\ g_1 g_2 \end{bmatrix}$$

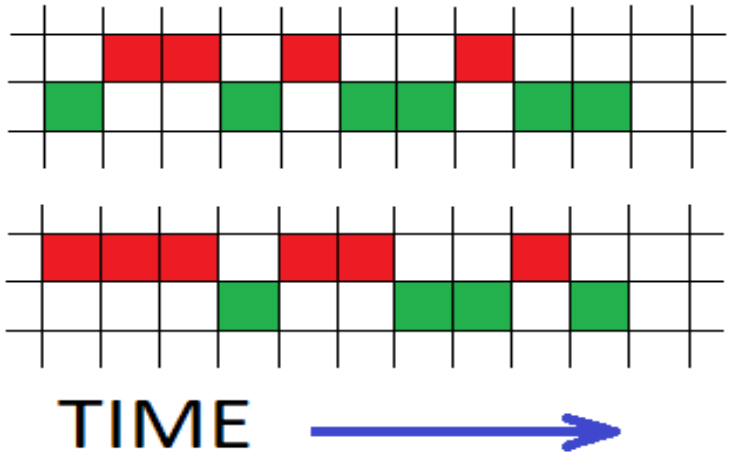
$$r_1 r_2 + r_1 g_2 + g_1 r_2 + g_1 g_2 = (r_1 + g_1)(r_2 + g_2) = 1$$

$$p_{rr} = r_1 r_2, \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$p_{rg} = r_1 g_2, \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$p_{gr} = g_1 r_2, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$p_{gg} = g_1 g_2, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



Quantum Entanglement

Considering **two qubits** in the following state:

$$|\boldsymbol{\Psi}> = \frac{|\mathbf{0},\mathbf{0}> + |\mathbf{1},\mathbf{1}>}{\sqrt{\mathbf{2}}}$$

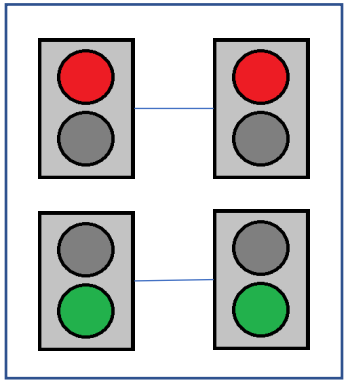
Which is the state of each qubit in this case?

$$|\Psi> = (a|0> + b|1>) \otimes (c|0> + d|1>) = \begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix}$$ (Kronecker product)

$$\Rightarrow \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix}$$ **no solution!**

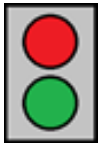
In "semaphores analogy", one could consider two "**synchronized semaphores**" with: *$r_1 = g_1 = 0.5$, and $r_2 = g_2 = 0.5$*

As the two semaphores are **not independent**, it is not possible any more to obtain the compound states probabilities by multiplying individual probabilities.

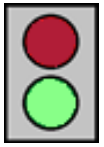


Unitary Operators & Measurement

- **Unitary Operator:** in ”semaphores analogy”, is a “device” (which has also an **associated matrix**) capable of changing the “red/green” probabilities of one quantum register:



$r + g \Rightarrow r' + g' = 1$

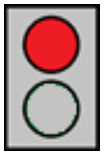


- **A quantum logic gate** (or simply quantum gate) is a unitary operator, operating on a small number of qubits.
  - Unlike many classical logic gates, **quantum gates are reversible.**

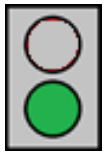
- **Quantum algorithm:** a prescription of a sequence of unitary operators applied to an initial state:

$$|\Psi_n> = U_n \dots U_1 |\Psi_1>$$

- **Measurement:** in ”semaphores analogy”, is a “switch” used for stopping some or all semaphores from oscillating. The semaphores will “freeze” in a single state. Thus, considering a singe qubit, one will find either red (with probability *r*) or green (with probability g).



OR



The Bloch Circle

Qubit: $\Psi = \alpha|0> + \beta|1>, \quad |\alpha|^2 + |\beta|^2 = 1$

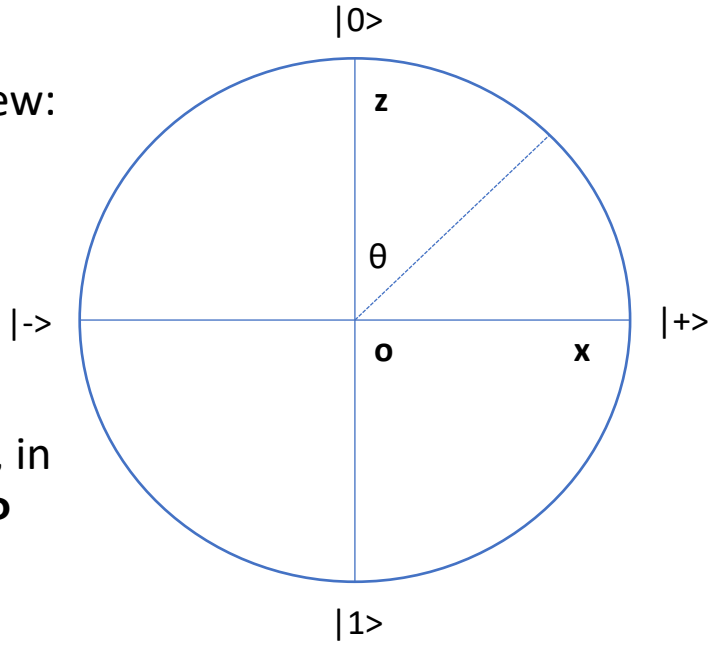
- Without limiting generality, one can consider:

$$\Psi = \cos\frac{\theta}{2}|0> + \sin\frac{\theta}{2}|1>, \text{ as } \cos^2\frac{\theta}{2} + sin^2\frac{\theta}{2} = 1$$

- Why θ/2 ?
  - It is more convenient from a graphical point of view:

$$|+> = \frac{|\mathbf{0}> + |\mathbf{1}>}{\sqrt{\mathbf{2}}} = \cos\frac{\pi}{2}|0> + \sin\frac{\pi}{2}|1>$$

- Why using the OZ axis?
  - As we'll see later, the qubit could be represented, in fact, on a sphere with polar coordinates: **r=1, θ, ϕ**



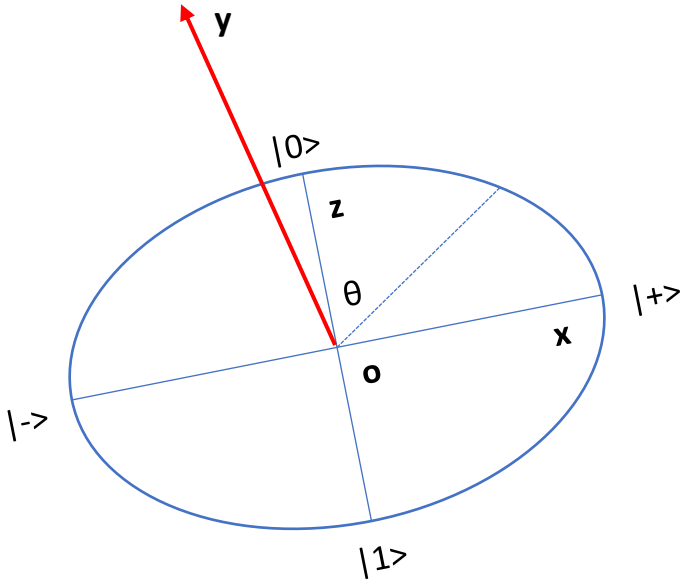
Examples of single qubit Gates

Reset: **|0>** **not a gate** but an irreversible operation

Pauli $\boldsymbol{X = NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ => $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$

$X|0> = |1>,\ X|1> = |0>$

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



$H|0> = \frac{|\mathbf{0}> + |\mathbf{1}>}{\sqrt{\mathbf{2}}} = |+>$ $H|1> = \frac{|\mathbf{0}> - |\mathbf{1}>}{\sqrt{\mathbf{2}}} = |->$

Rotations around the 3 axes: Rx, Ry, Rz

Regarding other logic gates, consult **IBM Quantum Learning [5].**

Examples of single qubit Gates

How to obtain any desired state?

For example, by rotations:

Ry(Pi/6) => θ/2 = Pi/12 => 0.965|0> + 0.258|1> =>

p(0) = 0.965² = 93.3%, p(1) = 0.258² = 6.7%

NOT

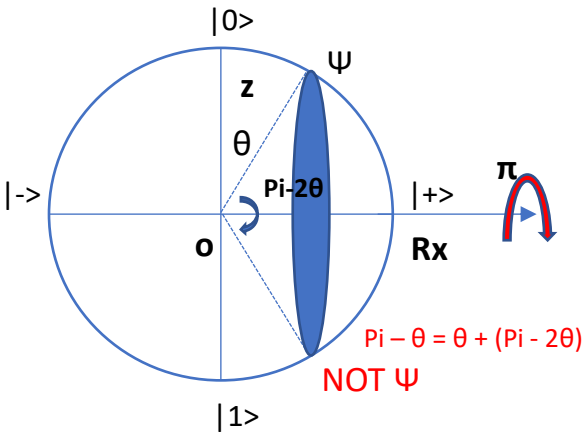
X(α|0> + β|1>) = **Ry(Pi-2θ)**Ψ = **Rx(Pi)**Ψ = β|0> + α|1>

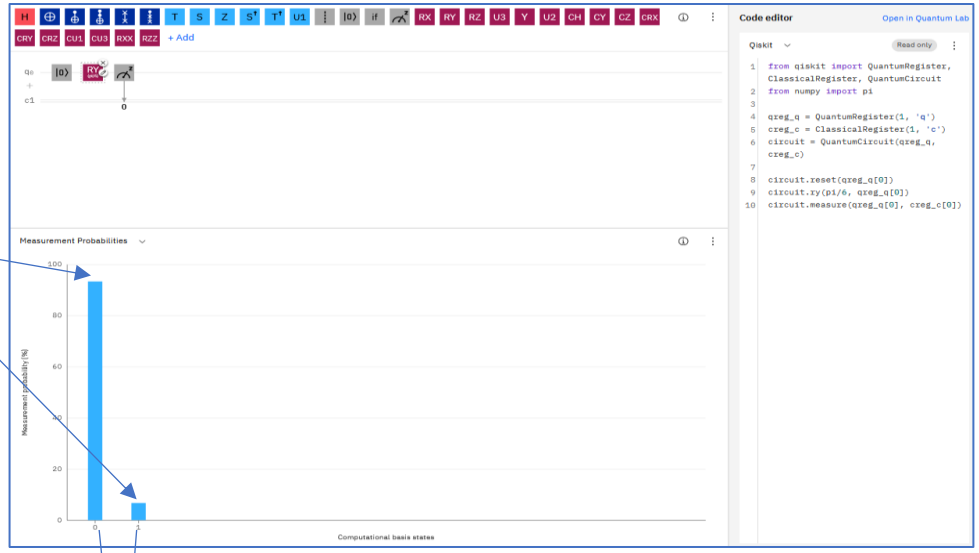
cos(x) = sin(Pi/2-x)

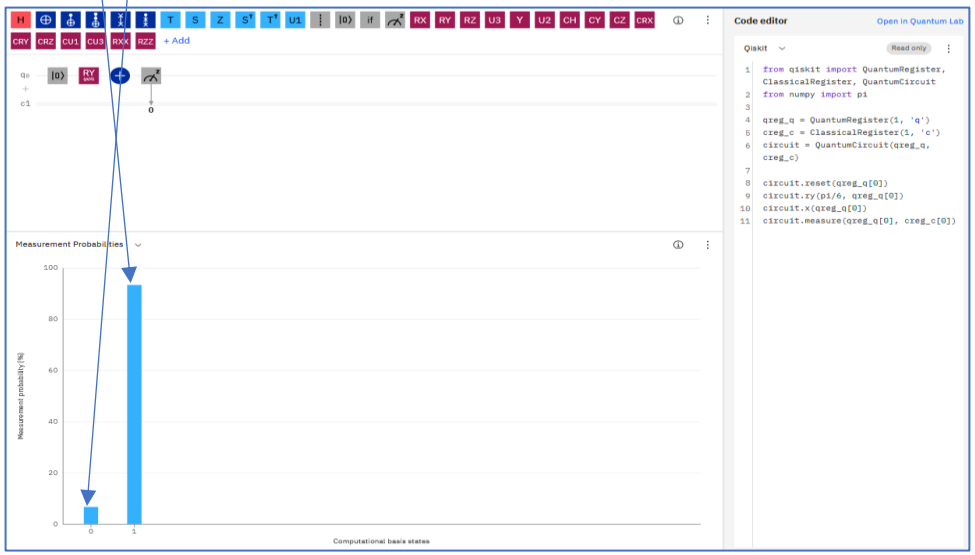
θ'= Pi – θ =>

cos(θ'/2) = cos(Pi/2- θ/2) = sin(θ/2) = β

sin(θ'/2) = sin(Pi/2- θ/2) = cos(θ/2) = α





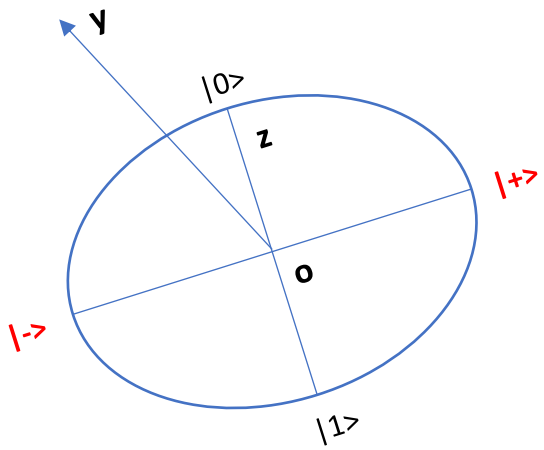


Superposition or Statistical Mix?

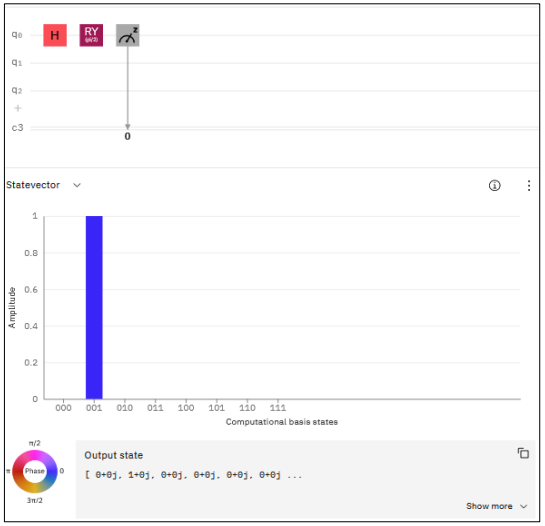
Both 0 and 1 or randomly chained series of 0 and 1?

Let's take the **|+>** = H|0> state. Considering the measurement limitations, if each qubit would have been either 0 or 1, with 50% probability, the measurement results would be the same…

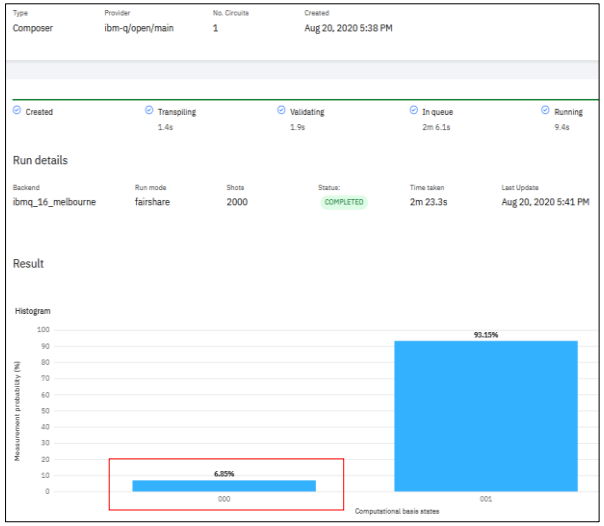
[0] [1] [0] [0] [1] [0] [1] [1] [1] [0] [1] [1] [0] [0]	[01] [01] [01] [01] [01] [01] [01] [01] [01] [01]
If one qubit would have been either 0 or 1, with equal probabilities: ⇒ **Ry gate would be applied, successively, to either |0> or |1>**	**Each qubit is both 0 and 1** ⇒ **Ry gate is applied to the |+> vector**
Ry(Pi/2)|0> = |+> Ry(Pi/2)|1> = |->	Ry(Pi/2)|+> = |1>
Measurement: p(0) = p(1) = 0.5	**Measurement: p(0) = 0, p(1) = 1**



THEORY =>



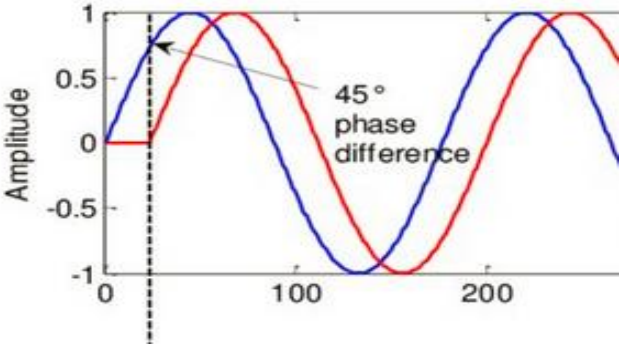
QUANTUM COMPUTER =>



The Bloch Sphere

Qubit: $\Psi = \alpha|0> + \beta|1>, \quad |\alpha|^2 + |\beta|^2 = 1$

- α and β could be separated also by a **phase difference** (wave functions)
- One could consider a circle for each θ => **The Bloch Sphere**
- Using complex numbers is convenient in this case:



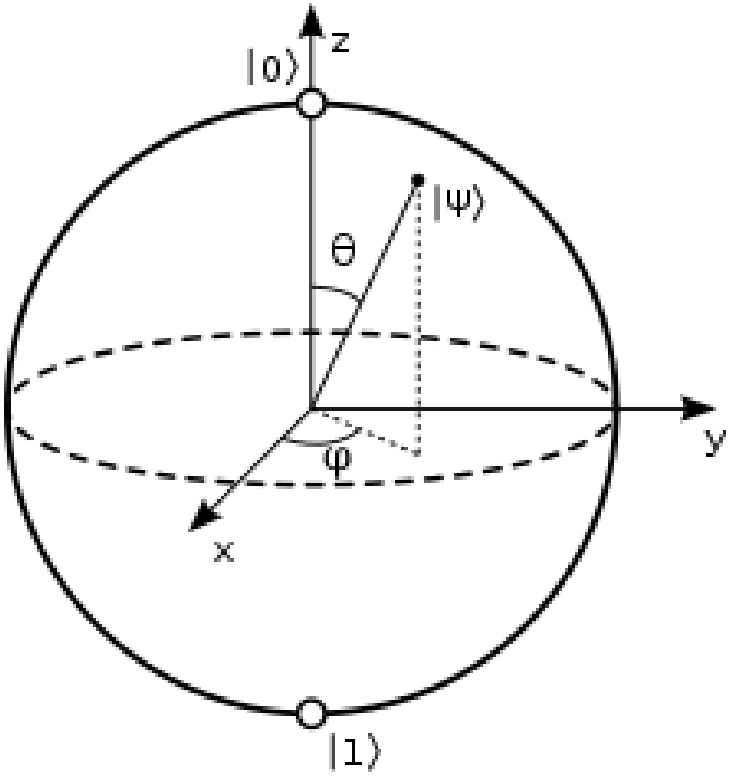
$$\Psi = \cos\frac{\theta}{2}|0> + e^{i\varphi}\sin\frac{\theta}{2}|1>$$

$$e^{i\varphi} = \cos\varphi + i\,\sin\varphi$$

$$|\Psi|^2 = \left|\cos\frac{\theta}{2}\right|^2 + \left|e^{i\varphi}\sin\frac{\theta}{2}\right|^2 = \cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2}\left|e^{i\varphi}\right|^2$$

$$= \cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2}(\cos^2\varphi + \sin^2\varphi)$$

$$= \cos^2\frac{\theta}{2} + \sin^2\frac{\theta}{2} = 1$$



Examples of two-qubits Gates

Let's test the Kronecker product

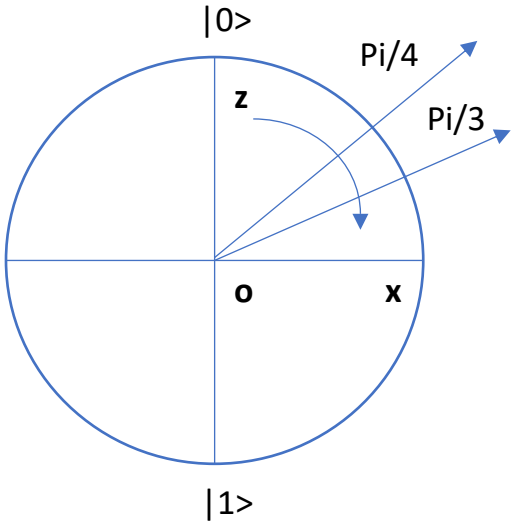
Pi/3 => cos(Pi/6)|0> + sin(Pi/6)|1> = 0.86|0> + 0.5|1>

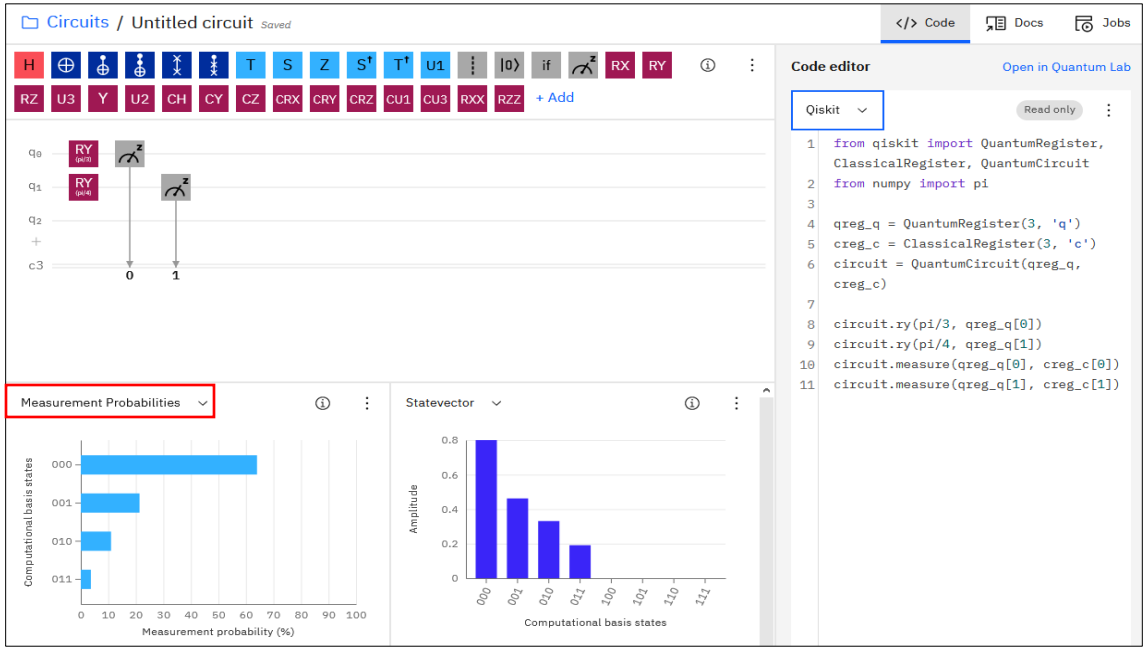
=> p(0) = 0.86^2 = **0.75**, p(1) = 0.5^2 = **0.25**

Pi/4 => cos(Pi/8)|0> + sin(Pi/8)|1> = 0.92|0> + 0.38|1>

=> p(0) = 0.92^2 = **0.85**, p(1) = 0.38^2 = **0.14**

$$\begin{bmatrix}0.8535\\0.1464\end{bmatrix} \otimes \begin{bmatrix}0.75\\0.25\end{bmatrix} = \begin{bmatrix} r_1 \begin{bmatrix} r_2 \\ g_2 \end{bmatrix} \\ g_1 \begin{bmatrix} r_2 \\ g_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix}0.6401\\0.2133\\0.1098\\0.0366\end{bmatrix}$$

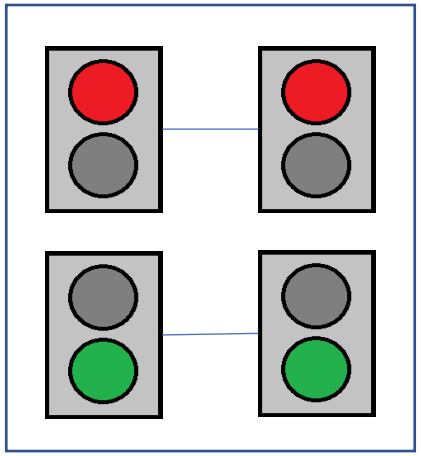




Quantum Entanglement – Tests

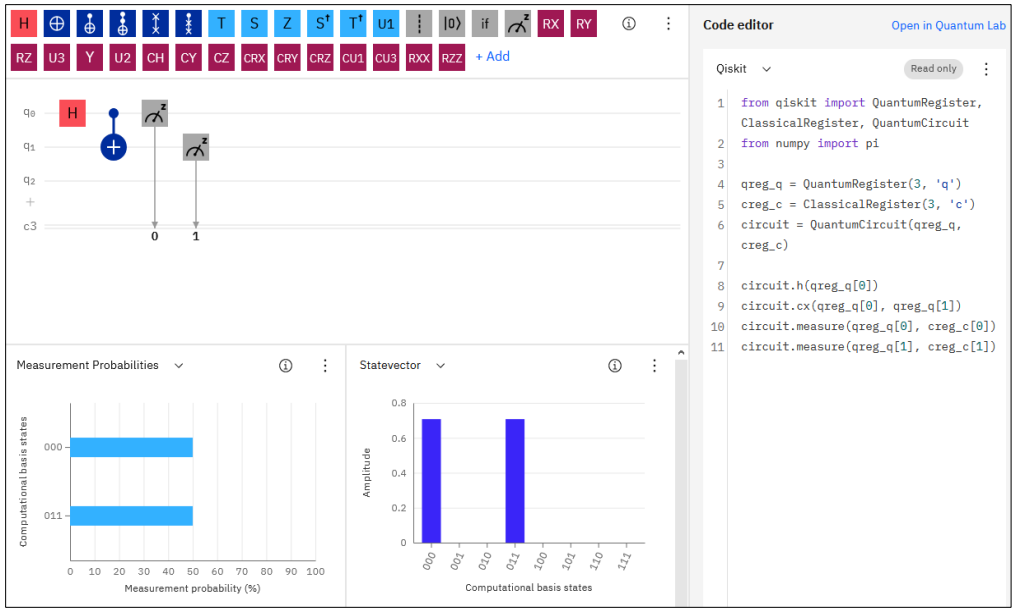
1. **H (Hadamard)** => $|\Psi> = \frac{|0> + |1>}{\sqrt{2}}$ (diagonal state)

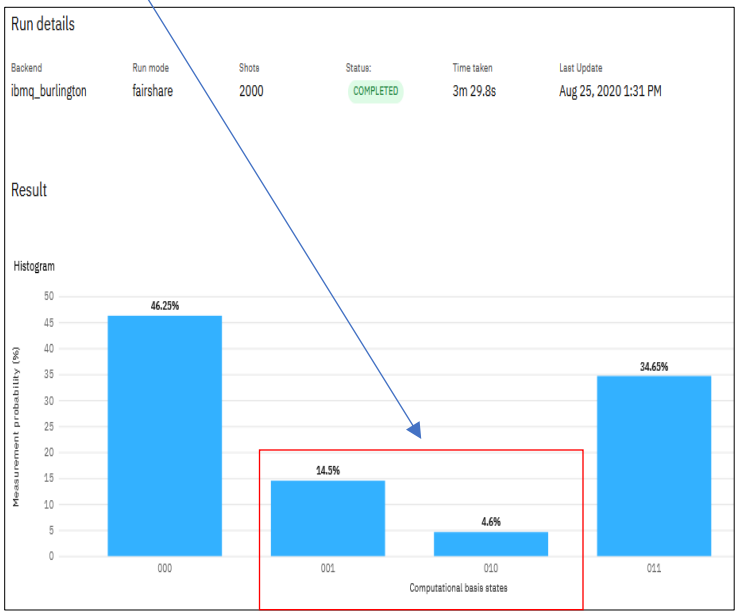
2. **CNOT (Conditional Not)** => $|\Psi> = \frac{|0,0> + |1,1>}{\sqrt{2}}$ Bell State



The deviation from the expected result could be considered a quantum computer precision test

Similarly, one could obtain: $|\Psi> = \frac{|0,1> + |1,0>}{\sqrt{2}}$





Introduction to Quantum Neural Networks

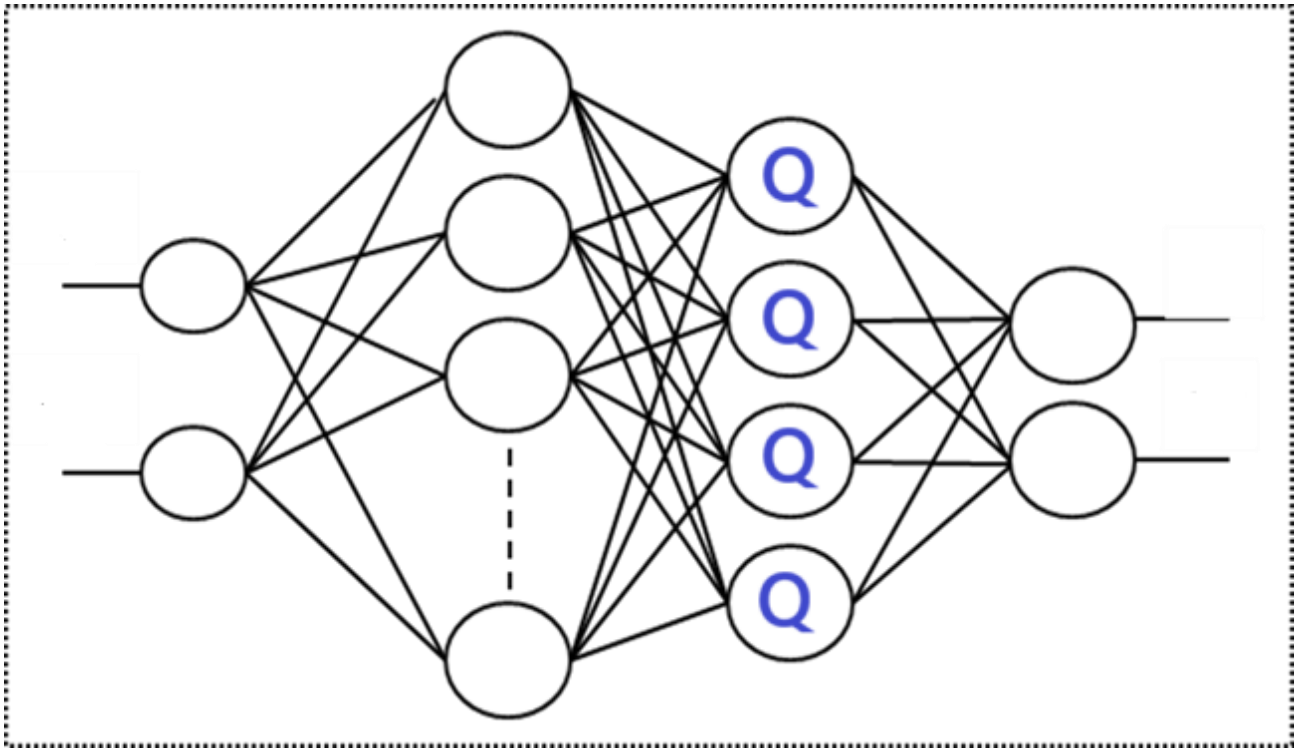
- Basic Concepts
- The classical artificial neuron limitations
- Single qubit neural circuit for solving Exclusive-OR

References [5-8]

Basic Concepts

• Hybrid quantum-classical neural networks => **implementing hidden layers with parameterized quantum circuits** (circuits which rotation angles, for each gate, are specified by the components of a classical input vector).

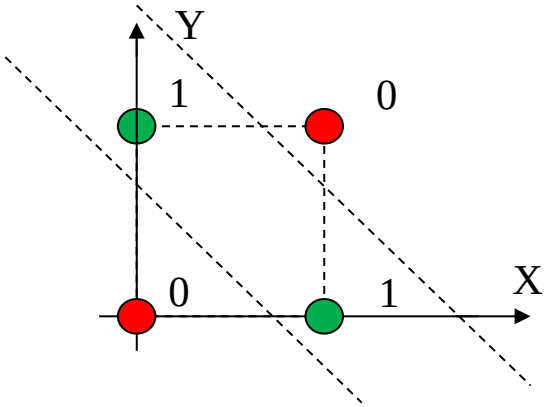
• One could calculate the gradient as the difference between the circuit evaluated at $\vartheta + s$ and $\vartheta - s$, where ϑ represents the circuit parameters, and s is a macroscopic shift. Thus, **the circuit could be trained using gradient descent methods,** such as backpropagation.

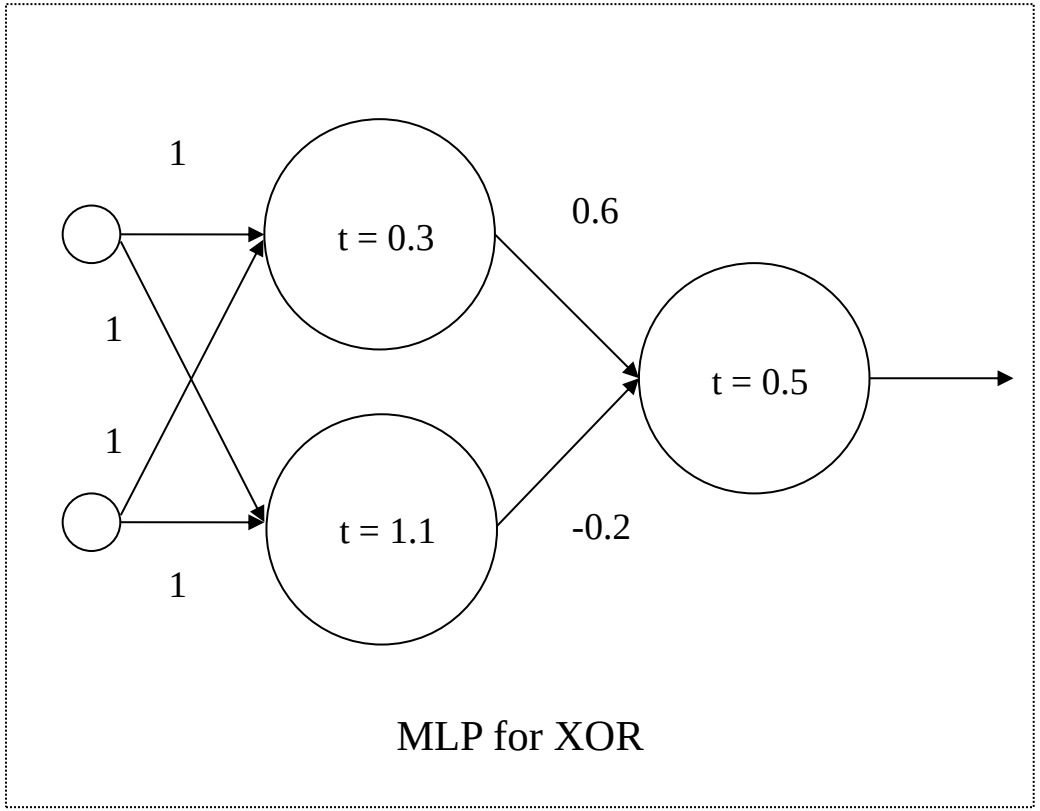


The classical neuron limitations

- **Classical neurons are limited to linearly separable problems**
- For linearly non-separable problems (e.g. XOR) one could use neural networks (e.g. MLP)
- Recent studies are discussing the **capability of individual human neocortical pyramidal neurons of classifying linearly non-separable inputs** [7]

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0



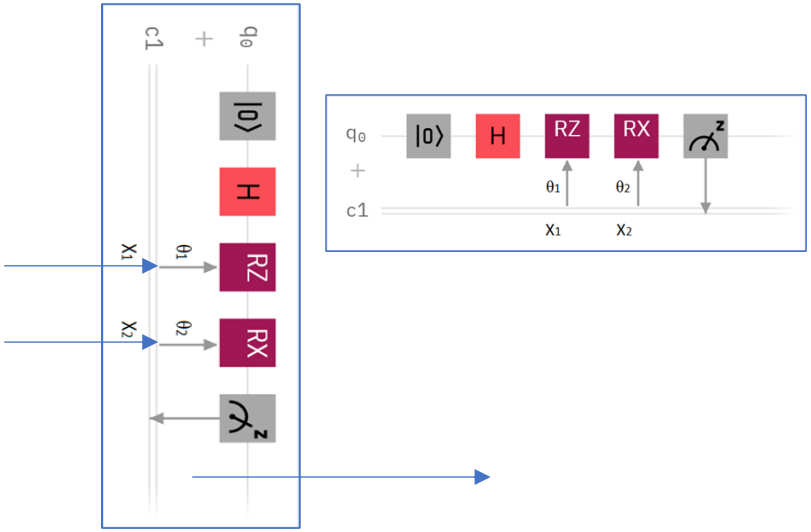


Single qubit neural circuit for solving Exclusive-OR

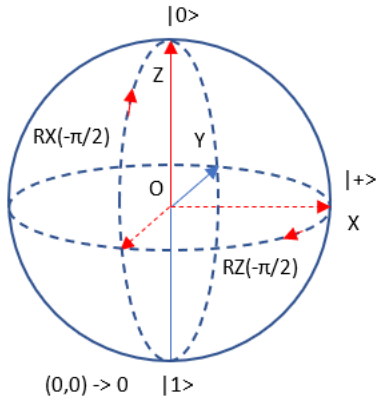
- A particular parametrized quantum circuit with two rotation gates was considered [8]:

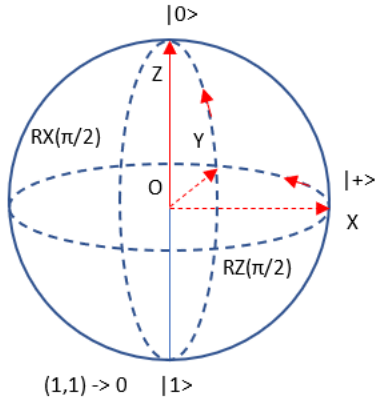
RZ(θ1 * x1 + α), and RX(θ2 * x2 + α), where (x1, x2) is the input vector, (θ1, θ2) are the "dendrites" trainable weights, and α is also a trainable parameter.

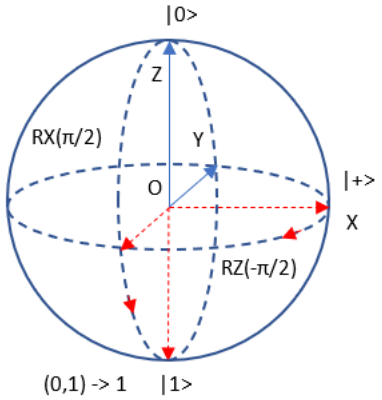
- For XOR, one solution could be immediately observed on the Bloch Sphere: $\vartheta_1 = \vartheta_2 = \pi$, for $\alpha = -\pi/2$

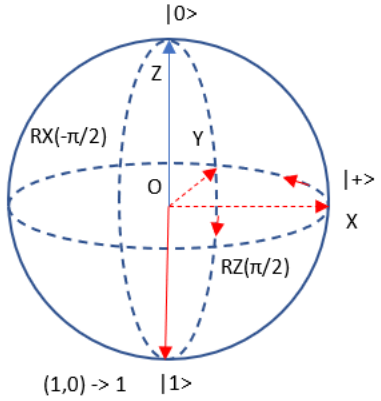


x_1	x_2	32q ibmq_qasm_simulator	5q ibmqx2
0.0	0.0	0.0000	0.0585
0.0	1.0	1.0000	0.8805
1.0	0.0	1.0000	0.8875
1.0	1.0	0.0000	0.0585









References

1. D. Dumitrescu, H. Costin, Retele Neuronale Teorie si Aplicatii, Teora, Romania, 1996
2. I.V. Grossu, S.A. El Shamali, Hyper-Fractal Analysis v04: Implementation of a fuzzy box-counting algorithm for image analysis of artistic works, Computer Physics Communications, Volume 184, Issue 7, July 2013, Pages 1812–1813
3. D. Felea et al., A first version of a Python MultiLayer Perceptron Deep Learning Neural Network to accurately and rapidly classify the Gravitational Waves, as a part of the 'Low Latency Pipeline' of the LISA Experiment, preprint, October 2020, DOI: 10.13140/RG.2.2.30357.35046
4. R. Portugal, Quantum Walks and Search Algorithms, Quantum Science and Technology, Springer, New York 2013, DOI 10.1007/978-1-4614-6336-8
5. https://learning.quantum.ibm.com/
6. I.V. Grossu, Introduction to Quantum Computing, Presentation, DOI: 10.13140/RG.2.2.29728.43524
7. A. Gidon, T.A. Zolnik, P. Fidzinski et al., Dendritic action potentials and computation in human layer 2/3 cortical neurons, Science, Jan 2020, Vol. 367, Issue 6473, pp. 83-87, DOI: 10.1126/science.aax6239
8. I.V. Grossu, Single qubit neural quantum circuit for solving Exclusive-OR, MethodsX, Elsevier, https://doi.org/10.1016/j.mex.2021.101573