Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Simulations of a quantum perceptron on IBM-Qiskit

Supervisor:
Prof. Elisa Ercolessi

Submitted by:
Valentina Monzali

Academic year 2022/2023

Index

Introduction

Artificial intelligence has been one of the principal aims of scientific research in the last years. Its target is to emulate in some way human intelligence, broadening the horizons to tasks that until '50s nobody thought could be performed by a machine.
One of the principal fields of this research is given by neural networks, which are systems that have the aim of independently elaborate the correct response over a stimulus never encountered before. This thesis will take into consideration this type of systems, in particular a neural network called perceptron, which has the task of recognizing and cataloguing external stimuli. The perceptron is built of single building blocks connected to each other, and each of these units corresponds to an artificial neuron. The first theorization of it was given by F. Rosenblatt (1958, [17] and [18]), with some assumptions made on each neuron unit. Each artificial neuron takes some inputs and gives as output a binary response. In addition, weights are associated to every input channel since it is supposed that in biological systems there are information channels of different importance which have to contribute more or less in the calculus of the response.
In classical perceptrons input channels are physical wires, and a weight is associated to each of them. This feature is different for what concerns quantum artificial neuron models. In fact, while in classical circuits information is brought through physical wires which transport classical bits (characterized by a binary value), quantum wires transport qubits, which, since they can transport also superposition of states (as discussed in this thesis), through the same number of channels can bring more information. In quantum artificial neurons a weight is associated not to each quantum channel, but to every basis state of the Hilbert space generated by the qubits involved in the circuit as input channels. Therefore if the qubits are n, the total number of equivalent input wires is given by 2^n. It is clear there is an exponential advantage in the quantity of information processed.
In fact, trying to implement a plausible quantum artificial neuron has been object of research in the last years, even if for what concerns quantum computers there are still many technological limits. This research is being deeply investigated because, as explained before, there are some tasks for which the quantity of information that can be simultaneously processed in a quantum computer is exponentially greater, or the processing time is exponentially smaller. This does not mean it is always convenient to

replace classical computer with quantum ones, but there are some suggestions that using quantum computer for artificial intelligence could get us closer to the functioning of biological networks.
This has inspired the principal aim of this thesis, which is to try to simulate the functioning of a quantum perceptron. It has been tested through the realization of a learning procedure. This consists in teaching to the network to correctly recognize a pattern, in order to finally obtain a machine that is able to recognize an image. This procedure is fundamental in neural networks since it permits to the system, once it has been performed, to correctly recognize also patterns not encountered during learning. To this aim, this thesis will at first go through the explanation of some perceptron models in the first two chapters, and gives also in the Appendix some fundamental notions of quantum mechanics and quantum computing necessary for this treatment. In fact, the first chapter tries to explain how classical artificial neural networks have been thought and developed until now, in order to arrive to the theory of implementation of a perceptron and give some examples. Then, in the second chapter the classical perceptron model described by Minsky and Papert is taken (1969, [11]) and the treatment of the article [21] is followed to implement a perceptron of the same logic on a quantum circuit. This model of a quantum perceptron is finally used in the third chapter and realized through a software development kit called Qiskit, provided by IBM Research. Here the steps followed for the implementation of the learning procedure will be explained and the results obtained will be analyzed. The process in which the machine is trained in this discussion is realized with a classical algorithm, and the quantum circuits have been run on a classical simulator, which behaves as an ideal quantum computer.

Chapter 1

Classical models

In this chapter at first it will be given an introduction of our knowledge about how biological neurons work (for more details see [15] and [23]). Consequently, the two principal artificial classical models for reproducing neural systems (neural networks) will be taken into consideration: in chronological order of implementation the McCulloch and Pitts model and the Rosenblatt classical perceptron. The former is more linear and simple, bit it will be seen it can reproduce almost all interesting dynamics of this type of systems; the latter is thought to get closer to the real functioning of our brain, even if it is built on hypothetical assumptions.

1.1 Biological model

Nervous systems are composed of single building blocks connected with each other. Neurons receive a signal through branches called dendrides, which are directly connected to other cells through synapses, and give back a response through the axon, at most one per neuron. The part of the cell that elaborates the information received and so gives the output to the axon is the body cell. These four elements are the base components needed for a modelling of a neuron, which is shown in Figure 1.1.
It is assumed that neurons communicate with each other by electrical impulses, with the connections controlled by biochemical processes through synapses. The totality of the cell bodies, together with their reciprocal interconnections, constitutes a neural network. The input for it is given by sensory receptors, which can take signals both from the body or from sense organs. After the processing given to the input by the neural network, the stimuli are passed to effectors, which control the organism and its actions. The body and its actions are continuously controlled via feedback connection to the neural network, implementing a sort of a closed loop control system.
The neuron fires if all the inputs collected in a certain interval (about 0.25 ms) together make the potential of the cell body reach a threshold level. If the threshold of potential

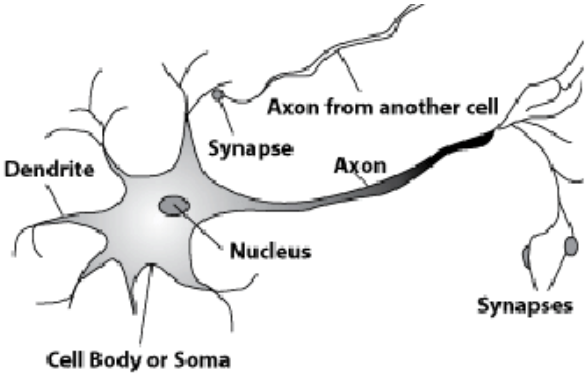

Figure 1.1: Schematization of a neuron ([3])

is reached, the condition for firing is respected and the neuron generates a pulse response (which is called action potential, Figure 1.2) that is sent to the axon. The two possible output states correspond to the neuron firing or not firing and this means the output is a binary signal. Since the input channels to neurons (dendrides) are connected by the appropriate links given by synapses to outputs of other neurons (axons), also inputs are binary signals. Therefore the activity of any cell can be represented by a boolean function f. In fact, if the number of dendrides of a neuron is n, the firing case denoted by 1 and the non-firing case by 0, the body cell acts as $f : \{0,1\}^{\otimes n} \to \{0,1\}$, which is the definition of a boolean function of n variables.

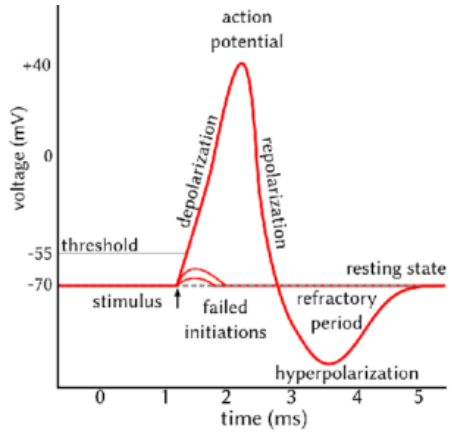

Figure 1.2: Action potential of a neuron ([8])

Input impulses can be excitatory or inhibitory. The first ones contributes positively to the activity of the neuron, inhibition instead is the prevention or termination of the activity of a group of neurons due to the antecedent or simultaneous one of a second group. Now the hypothesis is that there exist some synapses whose impulses inhibit the neuron,

which is instead stimulated by impulses through other synapses. In addition, some connections could be stronger than others and contribute more to the activity of the neuron. In artificial models this difference between synapses is modelled by associating to each input channel a real variable w, which is the weight with which the respective electrical signal is received by the neuron.
It has not been demonstrated yet if inhibitory signals are absolute or relative. While in the latter case a positive weight needs to be associated to the excitatory input channels and a negative one to the inhibitory ones, in the former if at least one of the inhibitory channels is active the neuron can't fire.
As said before, the neuron fires if the impulses collected in a certain period of time make the potential reach a threshold: this can be modeled by synchronous digital computation, simulating discrete time dividing it in intervals. Indeed, after an activation, the neuron can't be excited for a period called refractory period, which begins with the reaching of the threshold and ends when the neuron returns back to the resting state. In this this time the neuron is not able to receive signals in input. If we divide the time scale in intervals equal to this refractory period we can outline what happens assigning at the instant $k+1$ the firing of the neuron due to the input received at the instant k, and so on. Time units for modeling biological neurons can be taken of the order of a millisecond.
There are two aspects of biological neural networks that needs to be implemented in an artificial model. The former is that the antecedent activities of a region of the network could temporarily condition the response to a subsequent stimulation on the same part of the net. The latter concerns learning, which is a permanent change in the logic of the neural system, which means a change of the response due to a precise stimulation. Building an artificial network has not the claim to be an explanation of how biological neural systems work, but the aim is to try to reproduce their behaviour. Temporary storage is in fictitious models obtained by circular paths, with recursive functions in which output channels of a neuron are connected back to the input of the same neuron (an idea of this is explained in Section 1.2, for finite state machines). Long term memory and learning process are instead obtained by a change in the association of weights to input channels.

1.2 McCulloch and Pitts model

One of the first neuron models was proposed by McCulloch and Pitts in 1943 [9]. A McCulloch-Pitts unit takes n excitatory inputs x_1, x_2,..., x_n, and m inhibitory ones y_1, y_2,..., y_m. The following hypothesis are taken:

- output and inputs of neurons are binary signals, by convention $x_i, y_i \in \{0, 1\}$;
- a certain number of synapses must be excited in a certain fixed interval of time in order to excite a neuron, independently on the previous activity of any part of the

net;

- the only significant delay is synaptic delay, which means that approximately no time is taken for the computation;

- inhibition is absolute;

- no weights are assigned to input channels;

- the structure of the net (connections and functions computed by neural units) does not change with time.

Since inhibition is absolute, if at least one of y_i is activated, the output is 0. Otherwise the sum $x = x_1 + x_2 + ... + x_n$ is computed and if $x_1 \geq \theta$, which is the threshold value, the computation gives 1, if $x < \theta$ it returns 0.

Proposition 1.1 *Any finite boolean function can be realized by McCulloch-Pitts neural networks.*

It is possible to show that any boolean function can be written using the set of gates constituted by AND, OR and NOT, and that there are gates, such as NAND and NOR, which by themselves perform the same task. To justify Proposition 1.1, in Figure 1.3, 1.4, 1.5 it will be shown that McCulloch-Pitts units are sufficient to build the set of gates AND, OR and NOT [15].

Proposition 1.2 *Network built with McCulloch-Pitts units (with absolute inhibition) are equivalent to networks with relative inhibition.*

Proof. ([15]) It is easy to see how to implement absolute inhibition with relative inhibitory channels. Take a unit with n excitatory channels, a threshold θ , an integer $m > \theta$ which is the nearest integer to θ. To build an absolute inhibition it's sufficient to divide the desired input signal in $n-m+1$ wires and connect them to the unit as relative inhibitory channels. In this way if the input is equal to one the neuron will never fire, since in case all excitatory signals are taken equal to 1, one has $n - (n - m + 1) = m - 1 < \theta$.
On the contrary to obtain a unit with threshold θ and relative inhibition with units that have only one absolute inhibitory channel, build the following system. At first take a unit with threshold θ, n excitatory channels and one absolute inhibitory one. Then take a second unit without inhibitory channels with threshold $\theta + 1$. Finally use an OR gate that takes as inputs the outputs of the two previous units. A positive signal in output has to be obtained in two cases. The former coincides with the sum of excitatory inputs greater or equal than $\theta + 1$ with the inhibitory channel activated, the latter with the sum greater or equal than θ with the inhibitory channel turned off. Using OR gate the

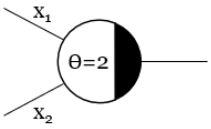

Figure 1.3: AND gate: for $x_1 = 1$ and $x_2 = 1$, then $x_1 + x_2 \geq \theta = 2$ and the output signal is equal to 1. In the cases in which at least one between x_1 and x_2 is equal to 0, the threshold is not reached and the output is 0.

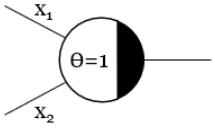

Figure 1.4: OR gate: to reach the threshold 1 it is sufficient that at least one between x_1 and x_2 is equal to 1. The only case in which the output is 0 is given by $x_1 = 0$ and $x_2 = 0$.

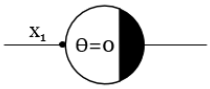

Figure 1.5: NOT gate: the black dot in the input channel denotes an absolute inhibitory channel. For this reason if $x_1 = 1$ the output that occurs is 0, while if $x_1 = 0$ the threshold $\theta = 0$ is reached and the unit returns 1.

system returns back a positive signal for both the desired cases.

For what concerns long term and temporary memory, only one of them can be realized using McCulloch-Pitts units. Long term memory can't be implemented since in this model there are no weights assigned to input channels; while temporary memory can be obtained organizing connections in the right way, building recurrent networks. This is true even if one of the hypothesis taken is that the action of a neuron singularly does not depend on the previous activity of any part of the net. To see what this means, take the example of finite automata. Finite automata are systems which have a finite number of input and output channels and a finite number of internal states. The system turns from one internal state to another and gives back a response (through the output) also taking trace of precedent activity. The simplest one is the binary scaler (Figure 1.6). It is possible to analyze the behaviour of the machine consequently to the receiving of an input. The important feature to notice is that the output depends both from the input and the precedent state of the automata. The response to the input received at time t is produced at time $t + 1$, and the transition $t \to t + 1$ constitutes an iteration. If one assumes the initial state $t = 0$ to be $x = 0$, $y = 0$, the following loop, where arrows

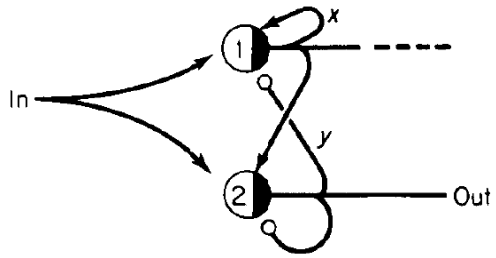

Figure 1.6: Binary scaler ([10]). Each unit takes as inputs an external channel (denoted as *In*) and the two outputs of the same units. The white dot denote it is an absolute inhibitory channel, while the values 1 and 2 are the two thresholds.

denote an iteration, is obtained if the input is kept equal to 1:

$$x \qquad 0 \to 1 \to 1 \to 0 \tag{1.1}$$

$$y \qquad 0 \to 0 \to 1 \to 0 \tag{1.2}$$

Notice that the state $x = 0$, $y = 1$ never occurs, in fact if the unit with threshold 2 is activated, it means also the one with threshold 1 has to, since they have the same inputs. If instead the input is equal to 0, beginning from all the three possible states, the following values are obtained.

$$x \qquad 1 \to 0 \qquad 1 \to 1 \qquad 0 \to 0 \tag{1.3}$$

$$y \qquad 1 \to 0 \qquad 0 \to 0 \qquad 0 \to 0 \tag{1.4}$$

It is evident that the machine gives 0 as result if input channel is off, while it can give both 0 and 1 if the channel is on depending on the previous state of the system.
A network with these properties is clearly a finite state machine, but on the contrary, one can show also that given a finite state machine, a network of McCulloch-Pitts units that reproduces it can be built.

Proposition 1.3 *Every finite state machine can be simulated by networks of McCulloch-Pitts units.*

Proof. Details can be found in [10],[15]; the scheme of the proof is shown in Figure 1.7. C_{ij} fires at time $t + 1$ when it receives positive signals from S_i and is in the state Q_j at time t. To each unit two output channels are attached: the first is connected to $Q(t + 1) = G(S(t), Q(t))$ and the second to $R(t + 1) = F(S(t), Q(t))$ in the respective connection boxes.
Suppose that at $t = 0$ S_i is given as input and the system is initialized in the state Q_j. Then the procedure ensures only one fiber per iteration is activated. In this way every possible finite state machine can be realized.

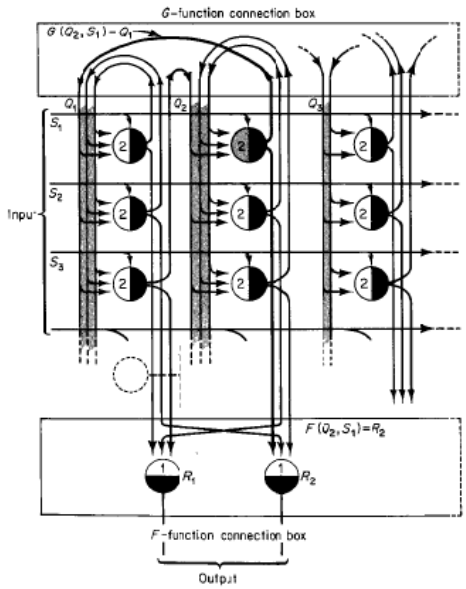

Figure 1.7: Generic scheme of a finite-state machine ([10]). Let's call the machine M, the m input channels $S_1, ..., S_m$, the output ones $R_1, ..., R_n$ and the states of the machine $Q_1, ..., Q_l$. In total the units are $m \cdot l$, they are labeled by C_{ij} and the threshold is set to 2. The functions $G(S(t), Q(t)) = Q(t+1)$ and $F(S(t), Q(t)) = R(t+1)$ are respectively the function that indicates the state of the machine at each iteration depending on the input and on the previous state and the function that associates the desired output to the same arguments.

1.3 The classical perceptron

The need of implementing a learning process leads to another model of neurons, the classical perceptron, which was implemented by F. Rosenblatt in 1958 ([17] and [18]). He was convinced that noise and randomness present in nervous systems were not inconvenient factors due to the approximated structure of the artificial model, but are inherent in the brain. If it's true of course the network can no longer be realized through boolean functions and it would rather be better to implement a model founded on probability theory, which will be adapted over the time under external stimuli. The assumptions made are the following:

- the connections of the nervous system are randomly chosen at birth, differ from one organism to another and are only subject to some genetic constraints;

- weights are associated to each channel and this allows the response to the same stimulus to change in time. The topology of the system instead, which consists in the connections between units, remains unaffected;

- after an exposure to a large number of similar stimuli the system will reinforce connections that have been in an active state for a long interval (increasing the absolute value of weights). This ensures common features that characterize similar inputs gain importance. In case for similar inputs the system gives a dissimilar answer, which means the system gives a wrong response, an external control can impose a deactivation of channels that lead to this result, weakening the respective connections. This process takes the name of learning procedure and has the aim to strengthen signals associated to the right answer.

The model taken into consideration below and drawn in Figure 1.8 tries to represent a photoperceptron, the type of neuron responding to visual stimuli. The visual stimuli that have to be identified as equal should correspond to the same output of the perceptron. The photoperceptron works as follows:

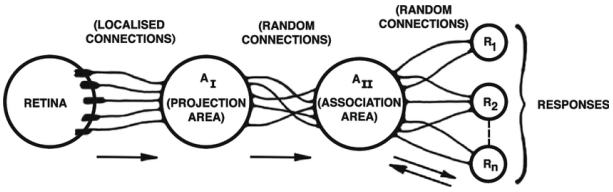

Figure 1.8: Photoperceptron scheme([18])

- Stimuli arrive on a retina of sensory units (S-points), which are assumed to have a binary response.

- Impulses are transmitted to an association area (A_I) through localized and deterministic connections. The idea is that each of these cells should analyze different small areas of the retina. Each A_I-unit is similar to McCulloch-Pitts units, with the difference that there is no absolute inhibition and that to every input channel there is associated a real weight. The S-points connected with a certain A_I-unit are called *origin points* of the unit. The cell fires if the threshold is reached and gives as output a binary signal.

- The connection between the projection(A_I) and the association area (A_{II}) are random and the units of A_{II} act the same as the ones of A_I. Both A_{II} and A_I-units have only one output channel. To the binary output of A_{II}-units there is associated a variable v, which can be any measurable characteristic of the output pulse, and is important for the memory of the system since it determines the change of weights of the respective channels. In fact, the change of weights, therefore the learning procedure, involves only channels that connect A_{II} with R-units, the ones which gives the final output.

- R-units take randomly chosen inputs in the A_{II} set and can be of two types: they can be activated when the mean value of input signals reaches the threshold θ_R or when the net value does. Since a stochastic procedure is used, usually the first is convenient because it is less affected by stochastic deviations. A-units transmitting signals to a certain response are called the *source-set* for that response.

- The connections between A_{II} and R-units are both forward and feedbacks. The feedbacks can be excitatory connections to the source-set cells or inhibitory feedback to the complement of the source-set. The latter is assumed because it leads to a more easily analyzing system. The responses in this type of networks are mutually exclusive. If the signals arriving to a specific response is more frequent or stronger than that one directed to the other R-units, it will tend to inhibit the alternative responses before their eventual activation.
  Each R-unit can then be an input for another perceptron.

1.3.1 Mathematical analysis of the perceptron

A photoperceptron without the projection area is taken into consideration, in which the stimuli go directly from the sensory units to the association area (A_{II}-units). To analyze the perceptron it is convenient to divide the activity of the neuron in two steps. The first is the *predominant phase*, in which A-units response is obtained but the R-units are inactive, and the the second is the *postdominant phase*, in which one of the possible R-unit is activated. The former has the aim to catalogue different input patterns and to reduce their complexity losing as little information as possible, the latter has the aim to evaluate together all these computed parts to give the final response.

For what concerns the *predominant phase*, the perceptrons are assumed to have a fixed threshold θ, x excitatory connections and y inhibitory connections; two variables are defined:

- P_a, the fraction of A-units activated by a certain given stimulus S_1, which is for a large retina equal to the probability that a given A-unit is activated by S_1;

- P_c, the probability that an A-unit which responds to a certain stimulus S_1 will respond also to another given stimulus S_2.

To have an efficient associative system both P_c and P_a have to be minimized and in addition P_a should have a constant value over stimuli that involve the activation of a different number of S-units.
From probability theory one can see that, assuming S_1 is a random stimulus, P_a is given by:

$$P_a(S_1) = \sum_{e=\theta}^{x} \sum_{i=0}^{\min(y,e-\theta)} P(e,i) \tag{1.5}$$

where

$$P(e,i) = \binom{x}{e} R_1^e (1-R_1)^{x-e} \binom{y}{i} R_1^i (1-R_1)^{y-i} \tag{1.6}$$

R_1 = fraction of S-points activated by S_1
x= number of total excitatory channels of the chosen A-unit
e = number of excitatory components received by the chosen A-unit from the stimulus S_1
y= number of total inhibitory channels of the chosen A-unit
i = number of inhibitory components received by the A-unit from the stimulus.

$P(e,i)$ is the probability that, given a precise number of inhibitory and excitatory signals in input, the threshold is reached; in Eq. (1.6) it is calculated by multiplying the probabilities of every channel to be activated or not with the number of possible permutations. In Eq. (1.5) instead, all the cases of $P(e,i)$ different from zero are summed.

Furthermore, for the definition of conditional probability, which is the probability of an event given that another has already occurred:

$$P_c = \frac{P_a(S_1, S_2)}{P_a(S_1)} \tag{1.7}$$

where $P_a(S_1, S_2)$ is the probability that a A-unit is simultaneously activated by two inputs S_1 and S_2. Therefore, if one assumes that the two stimuli have the same number of activated S-points, which implies $P_a(S_1) = P_a(S_2) = P_a$ and $R_1 = R_2 = R$, one gets

$$P_c = \frac{1}{P_a} \sum_{e=\theta}^{x} \sum_{i=0}^{\min(y, e-\theta)} \sum_{l_e=0}^{e} \sum_{l_i=0}^{i} \sum_{g_e=0}^{x-e} \sum_{g_i=0}^{y-i} P(e,i) \tag{1.8}$$

with

$$-l_e + l_i + g_e - g_i \geq 0$$

where

$$\begin{aligned} P(e,i) = &\binom{x}{e} R^e (1-R)^{x-e} \binom{y}{i} R^i (1-R)^{y-i} \binom{e}{l_e} L^{l_e} (1-L)^{e-l_e} \binom{i}{l_i} L^{l_i} (1-L)^{i-l_i} \\ &\binom{x-e}{g_e} G^{g_e} (1-G)^{x-e-g_e} \binom{y-i}{g_i} G^{g_i} (1-G)^{y-i-g_i}. \end{aligned} \tag{1.9}$$

where

$$R = \frac{r}{n}$$

$$L = \frac{l}{r}$$

$$G = \frac{g}{n - r}$$

n = total number of S-points
r = number of activated S-points by S_1 or S_2
l = total number of points activated by S_1 but not by S_2
g = total number of new points activated by S_2 but not by S_1
l_e = number of excitatory connections lost substituting S_1 with S_2
l_i = number of inhibitory connections lost substituting S_1 with S_2
g_e = the number of excitatory connections gained
g_i = the number of inhibitory connections gained.

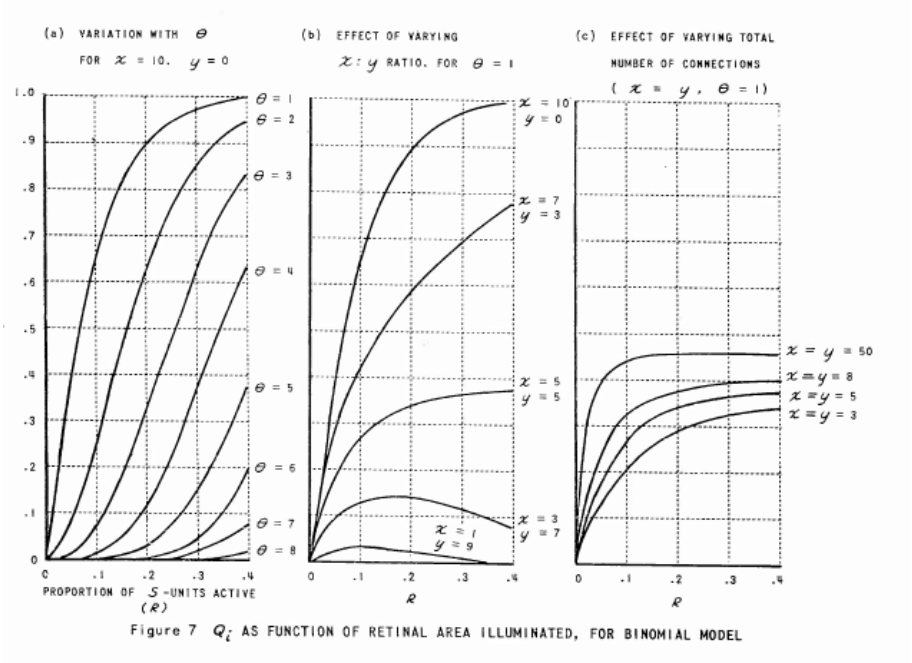

Figure 1.9: P_a as a function of illumination of the retina([16]), depending on the number of inhibitory and excitatory channels and on the threshold.

Analyzing the trend of P_a (Figure 1.9), it is possible to see that its value can be reduced by increasing the threshold θ of the A-units or by increasing the proportion of inhibitory connections. Furthermore, to reduce the variation in P_a for different stimuli, one can show this variation can be minimized for an equal number of inhibitory and excitatory input channels, as shown in Figure 1.9(c). If instead one analyzes P_c, the same trend depending on θ and on the number of inhibitory and excitatory channels is found. In

addition, one obtains that the overlap increases for large stimuli (with more S-units activated) and reduces for smaller ones. It is possible to obtain an optimal value of the threshold θ minimizing P_c.

Taking into account also the *postdominant phase*, the one involved in learning procedure, denote as $A_{S.R}$ the set of A-units activated by the stimulus S and and which transmit to the response unit R. The sets of A-units which responds to both stimuli are denoted as $A_{S_1,S_2.R}$. Since A-units have at most one output, they are connected only with one R-unit.
Consider two unrelated stimuli chosen at random, each of which activates the 50% of S-units, then assume R-units are two, denoted by R_1 and R_2. With these assumptions one obtains the following proportions of A-units activated:

$A_{S_1.R_1}$	0.25
$A_{S_1.R_2}$	0.25
$A_{S_2.R_1}$	0.25
$A_{S_2.R_2}$	0.25
$A_{S_1,S_2.R_1}$	0.125
$A_{S_1,S_2.R_2}$	0.125

Take into consideration P_c probability defined for the *predominant phase* and define the analog P_{cr} for the *postdominant phase*. Therefore the fraction of units which transmit to R_2 being activated by the stimulus S_1 that are activated also by the stimulus S_2 is

$$P_{cr} = \frac{A_{1,2.2}}{A_{1.2}} = 0.5. \tag{1.10}$$

If the stimulus S_1 for some reason leads at first to the response R_2, the set $A_{1.1}$ will be inhibited before having time to be enabled. This causes a raise in the value of v corresponding to the outputs of $A_{1.2}$ units, so that the response R_2 will be predominant in the future whenever the association units receive the stimulus S_1. If the stimulus S_2 is then received, a bias is present towards the response R_2 due to the association units belonging to the set $A_{1,2.2}$ (which implies $P_{cr} \neq 0$), and this could cause errors in the performance. However this bias is negligible if the overlap of the two stimuli is small, and will be further reduced by value-gain of v forcing the set to respond correctly to the stimulus S_2 (external control during learning process).
If on one hand allowing a P_c different from 0 allows an incredible greater number of possible input pattern to recognize, one has to take into account that this leads to a less precise performance in producing the response. This is due to the fact that the perceptron learning procedure, with which one can adapt the responses corresponding to all the input cases, after a training periods leads to a saturation of the system. Saturation of the system takes place when the information taken from the gain of new association

reinforcement balances with the information contained in the previous acquired associations lost by adjusting the same parameters (see Section 1.3.3).
A useful parameter for the capacity of the system after its saturation is the probability P_d, which is defined as the probability that a stimulus will retain the association to the correct response after the system has been satured. After the learning procedure there are two types of probability P_d. The first type consists in the probability that a pattern received during the learning process will be recognized correctly after the system has ended the procedure, the second is the probability that the system will recognize correctly a pattern not encountered during learning.

1.3.2 Minsky and Papert model

Minsky and Papert (1969) discussed in [11] a slightly different model, where the input patterns are projected in a retina of pixels with binary values and the response instead of computing a classification consists only in one unit. Therefore, the retina corresponds to the set of S-units and the association units are connected, with differently weighted channels, all to the same threshold θ unit. The response of the system consists in a binary value. A-units compute a set of local binary functions (or predicates), recognising local features in the S-points and the R-unit puts together all this information. The assumptions made for local predicates are realistic assumptions and differentiate perceptrons as:

- diameter limited perceptrons: the receptive field (the inputs) of each predicate has points of the retina contained in a limited diameter;

- perceptrons of limited order: the receptive field contains up to a certain number of points of the retina;

- stochastic perceptrons: the input points for every predicate are randomly chosen.

The operation of the system to divide the retina and analyze separately every subset and only then gather all the information loses in capacity of identifying global properties. One example is *connectedness*, which indicates if the figure given by the activated S-points can't be divided into two or more separated figures (Figure 1.11). Minsky and Papert in their book showed diameter limited perceptrons can't recognize it ([11]).
Analyzing the behaviour of this system, one finds that this perceptron is a computing threshold θ unit that takes n inputs $x_1, ..., x_n$, which can be real but also binary values, through n channels associated with the weights $w_1, ..., w_n$. One has that the threshold is reached and so the unit is equal to 1 when

$$\sum_{i=1}^{n} w_i x_i \geq \theta, \tag{1.11}$$

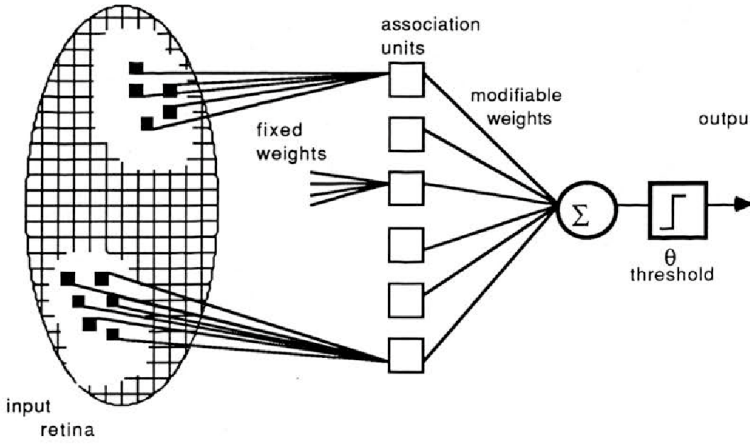

Figure 1.10: Minsky and Papert model([14])

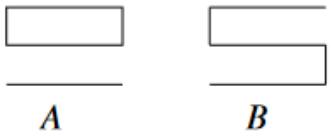

Figure 1.11: A, example of a non-connected figure; B, example of a connected figure([15]).

otherwise it is 0. The unit behaves the same also if the threshold is put to 0 and an inhibitory channel input of value $-\theta$ is added. In this way if we think input values as vectors in $\mathbb{R}_n$, Eq. (1.11) can be seen as the equation of a straight line that divides the manifold into two parts (see the following section). This two parts correspond to the two possible outputs.

1.3.3 Learning algorithm

As already explained, only weights associated to channels which connect A-units with R-units can change in time, through a process that takes the name of *learning procedure*. Take into consideration Minsky and Papert model of the previous section. To produce the desired response for each input pattern the weights $w_1, ..., w_n$ that connect each A-unit to the response have to be computed. This should be a process that takes information from experience.
Supervised learning algorithms take place when a system control process knows which is the correct answer for the set of input patterns involved in the learning procedure. This can be divided in corrective learning and reinforcement learning. The former is when

the only information the system can have is whether the answer given to an input by the current system is correct or not. For this reason the only possible correction that can be applied to the weights towards the right response has a standard and fixed real value. In the latter instead also the magnitude of the error committed by the current system in analyzing the input is known and this is used to compute the magnitude of the correction. Usually in this way the error is corrected only in a single step.
If $\boldsymbol{w} = (w_1, w_2, ..., w_n, -\theta)$ and $\boldsymbol{x} = (x_1, x_2, ..., x_n, 1)$, which are respectively the weight and input vectors, the request for activation is $\boldsymbol{w} \cdot \boldsymbol{x} \geq 0$. Assume both vectors can have real values, which is the general case.
Before implementing the algorithm a definition has to be given.

Definition 1.1 *Two sets of points A and B are called absolutely linearly separable if for every $x_1, x_2, ..., x_n \in A$ then $\sum_{i=1}^{n} w_i x_i > c$ and for every $x_1, x_2, ..., x_n \in B$ then $\sum_{i=1}^{n} w_i x_i < c$ for some real value of c.*

One can show that the property for two systems to be absolutely linearly separable or linearly separable (including also the equal sign in inequalities) is equivalent.
To see how to implement practically the algorithm for the learning process, two general sets of points in the n-dimensional input space A and B are taken. The n-dimensional input space is the space of possible inputs for the R-unit of the perceptron, which correspond to the outputs of the n A-units.

1. An initial vector $\boldsymbol{w_0}$ is generated randomly, t is set to 0;

2. a vector $\boldsymbol{x}$ is randomly taken from the set $A \cup B$,

   - if $\boldsymbol{x} \in A$ and $\boldsymbol{w_t x} > 0$, go back to (2);
   - if $\boldsymbol{x} \in A$ and $\boldsymbol{w_t x} \leq 0$, set $t = t + 1$ and $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} + \boldsymbol{x}$, then go back to (2);
   - if $\boldsymbol{x} \in B$ and $\boldsymbol{w_t x} < 0$, go back to (2);
   - if $\boldsymbol{x} \in B$ and $\boldsymbol{w_t x} \geq 0$, set $t = t + 1$ and $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \boldsymbol{x}$, then go back to (2).

The procedure is repeated taking vectors in the sets A and B until the system gives the right response for each of them. The following theorem ensures the convergence of this procedure ([15]).

Theorem 1.1 *If the sets A and B are finite and linearly separable (or equivalently absolutely linearly separable), the algorithm converges, which means it reaches a value for which the system always gives the right response.*

Proof. At first consider the set $A' = A \cup B^-$, where B^- is the set of the negated elements of B, obtained inverting the signs of the vectors components. In this way one can work with the same operations with all vectors involved in the learning procedure. Then

normalize vectors in A', operation that does not change the sign of $\boldsymbol{x} \cdot \boldsymbol{w_t}$. Since the sets are linearly separable, there exists a normalized solution $\boldsymbol{w}$ for the weight vector.
If all vectors are well classified it means the weight vector has reached a right value. Otherwise, at a certain time t a vector $\boldsymbol{a_i}$ has to be wrongly classified, then $\boldsymbol{w_{t+1}} = \boldsymbol{w_t} + \boldsymbol{a_i}$. To estimate the closeness of $\boldsymbol{w_{t+1}}$ to the solution of the problem, calculate

$$\cos\rho = \frac{\boldsymbol{w_{t+1}} \cdot \boldsymbol{w}}{\|\boldsymbol{w_{t+1}}\|}. \tag{1.12}$$

Furthermore,

$$\boldsymbol{w_{t+1}} \cdot \boldsymbol{w} = \boldsymbol{w_t} \cdot \boldsymbol{w} + \boldsymbol{a_i} \cdot \boldsymbol{w} \geq \boldsymbol{w_t} \cdot \boldsymbol{w} + \delta \geq \boldsymbol{w_0} \cdot \boldsymbol{w} + \delta(t+1), \tag{1.13}$$

where $\delta = \min(\boldsymbol{a_i} \cdot \boldsymbol{w})$, with $\boldsymbol{a_i} \in A'$, and the last step is obtained by iteration. For the denominator instead one can obtain

$$\|\boldsymbol{w_{t+1}}\|^2 = \|\boldsymbol{w_t}\|^2 + \|\boldsymbol{a_i}\|^2 + 2\boldsymbol{w_t} \cdot \boldsymbol{a_i} \leq \|\boldsymbol{w_t}\|^2 + \|\boldsymbol{a_i}\|^2 \leq \|\boldsymbol{w_t}\|^2 + 1 \leq \|\boldsymbol{w_0}\|^2 + t + 1, \tag{1.14}$$

for the normalization of $\boldsymbol{a_i}$ and because $\boldsymbol{w_t} \cdot \boldsymbol{a_i} \leq 0$ ($\boldsymbol{w_t}$ has to lead to the wrong response). Finally one obtains

$$\cos\rho \geq \frac{\boldsymbol{w_0} \cdot \boldsymbol{w} + \delta(t+1)}{\sqrt{\|\boldsymbol{w_0}\|^2 + t + 1}}. \tag{1.15}$$

As the right hand side grows as $\sqrt{t}$ and $\cos\rho \leq 1$, there will be a value of t for which this inequality will not be satisfied anymore. Therefore the iteration of the algorithm has to stop and $\boldsymbol{a_i}$ will be correctly associated to its response.
If the sets are not linearly separable, the algorithm does not converge and at a certain level it reaches the *saturation*, in which for every new correction the system loses the capacity of recognizing a pattern already learned during the process. In fact, if the two sets are not linearly separable, it means there does not exist a hyper-plane which divides them, each part corresponding respectively to the affirmative and negative response.

Chapter 2

Quantum model

In this second chapter it will be seen how to implement a quantum model for a perceptron starting from the classical models analyzed in the previous one. It is a very useful task because of the exponential growth in capacity of storing and processing information quantum systems have.

2.1 Introduction

Investigation is the last years has pointed out the convenience of using quantum computers for realizing artificial neural networks [20]. The advantage brought by quantum mechanics is the exponential growth in storage and recognition, due to quantum parallelism. A quantum memory register, which lives in a quantum state space M, can be in a superposition of states, each of which can be considered as an argument for a function f.
For exemple, take a quantum memory register constituted by n qubits, with their respective computational basis (see Appendix). The total state space is therefore $\mathcal{H}^n$, and it's m-dimensional, where $m = 2^n$. Define a function $f : \{0,1\}^{\otimes n} \to \{0,1\}$ that associates to every basis state a binary output. Computing the function on a state $|\psi\rangle \in \mathcal{H}^n$, with one iteration, for the linearity of quantum mechanics, means computing it for every basis state. This means while this operation in a classical computer has to be realized sequentially, on a quantum computer is instead processed in parallel. However, to extract the information about the function applied a measure has to be taken and if $|\psi\rangle$ is in a superposition of a huge number of quantum states, the probability of detecting the result on one of them in particular is small.

To build the quantum analog of the Rosenblatt perceptron model, consider the previous exemple of a quantum register of n qubits. Each of these qubits substitutes an output channel of the A-units (see Section 1.3) (in the classical case each of them is a bit of

information). In this model each qubit represents an individual neuron.

$$|\text{neural qubit}\rangle = rest|0\rangle + active|1\rangle \tag{2.1}$$

Rest and *active* are the probability amplitudes of being respectively inactive and active, and each neuron of the perceptron has therefore the possibility to be in a superposition of firing patterns.
The total quantum state lives in a m-dimensional space, and to each state of the computational basis a weight is associated. A perceptron with n input qubits is therefore equal to a classical perceptron with 2^n input bit channels.

The other aspect together with quantum parallelism concerns entanglement. States obtained by the product of all the qubit neurons in a certain superposition of rest and active states do not cover the whole space $\mathcal{H}^{\otimes n}$. Through unitary transformations applied to the set of qubits one can obtain also quantum entangled states (see Appendix), extending the domain to the whole $\mathcal{H}^{\otimes n}$.
The conceptual problem of implementing a perceptron on a quantum computer concerns the non-linear and dissipative dynamics of threshold functions in neural networks. This non-linearity is fundamental for the recognition of non-trivial patterns. Acting on superpositions of states with a threshold function breaks the linear and unitary evolution of quantum systems, which is not acceptable. Therefore, the break of linearity is allowed only after qubits are converted into classical bits. This happens if a measurement is taken, which leads to the collapse of the state onto one of the basis states.

In addition, as quantum computers are not available yet, quantum circuits has to be simulated, but even if it leads to a decrease of the computing speed, it overcomes the difficulties a real quantum computer would encounter due to irrelevant interactions with the real world. In fact, quantum theory is linear and transformation between states are realized by linear and unitary operators, which conserve probability. Quantum circuits are based on these transformations, but in the real world (for the interaction with the environment) it's very difficult to overcome dissipation (quantum analog to the classical irreversible loss of energy) and decoherence (destruction of the correlations of the pure quantum state), which have a non-linear and non-unitary dynamics[20].

2.2 Quantum model of a classical perceptron

In the following description a possible way of implementing a quantum perceptron inspired by the Minsky and Papert model is taken into account [21]. Weights and input vectors have components constituted by binary values (i_j are the input vector components and w_j the weights associated to every input basis vector), in particular for computations $x_j, w_j \in \{-1, 1\}$. A perceptron of n qubits therefore lives in

a 2^n-dimensional input pattern vector space. The computational basis is so given by $|j\rangle \in \{|00...00\rangle = |0\rangle^{\otimes N}, |00...01\rangle, |00...10\rangle, ..., |11...11\rangle\}$, with $j \in \{0, 1, ..., m-1\}$, where $m = 2^n$.

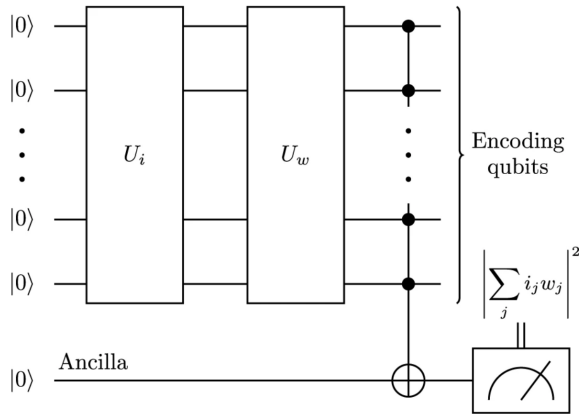

Figure 2.1: Scheme of the quantum circuit of a Minsky and Papert perceptron[21].

These special quantum states with vector components constituted only by values 1 and -1 are called REW states and can be written in the following way:

$$|\psi\rangle_{REW} = \frac{1}{\sqrt{m}} \sum_{j=0}^{2^n-1} (-1)^{g(j)} |j\rangle, \tag{2.2}$$

where $g : \{0,1\}^n \to \{0,1\}$ is a boolean function and it defines completely the state. Now two unitary gates have to be taken into consideration. They have to be unitary and to satisfy the following conditions. Denote the first as U_i, and impose

$$U_i|0\rangle^{\otimes n} = |\psi_i\rangle, \tag{2.3}$$

where

$$|\psi_i\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j\rangle. \tag{2.4}$$

The second instead, U_w, has to satisfy:

$$U_w|\psi_w\rangle = |1\rangle^{\otimes n} = |m-1\rangle, \tag{2.5}$$

where

$$|\psi_w\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} w_j |j\rangle. \tag{2.6}$$

Applying U_w to $|\psi_i\rangle$

$$U_w|\psi_i\rangle = \sum_{j=0}^{m-1} c_j|j\rangle \tag{2.7}$$

for some coefficients c_j. Therefore, if one considers that the base states are orthogonal and that U_w is unitary, from Eq. (2.5) and (2.7)

$$w_j i_j = m\langle\psi_w|\psi_i\rangle = m\langle\psi_w|U_w^\dagger U_w|\psi_i\rangle = m\langle m-1|\sum_{j=0}^{m-1} c_j|j\rangle = mc_{m-1}. \tag{2.8}$$

This means that up to a normalization factor the vector product is stored in a probability coefficient. The aim is to extract this information through measurements. In this way the linearity is broken as required by the threshold function. The measurement is made using an ancilla qubit (see Appendix) initially set to $|0\rangle_a$. With a C^{m-1}NOT (see Appendix) the following state is obtained:

$$\mathrm{C}^{m-1}\mathrm{NOT} : \sum_{j=0}^{m-1} c_j|j\rangle|0\rangle_a \rightarrow \sum_{j=0}^{m-2} c_j|j\rangle|0\rangle_a + c_{m-1}|m-1\rangle|1\rangle_a. \tag{2.9}$$

Measuring the ancilla qubit, the output 1 occurs with probability $|c_{m-1}|^2$. Repeating the procedure one obtains the value of the vector product by means of statistic data analysis. Once the measurement is made the information is classical and can be given as input to any non-linear function.
It is possible to notice that, since a probability is measured, both for $\vec{i}$ and $\vec{w}$ parallel and anti-parallel $c_{m-1} = 1$. For orthogonal vectors $c_{m-1} = 0$ and the ancilla qubit will always be in the state $|0\rangle_a$.

2.3 Realization of the quantum circuit

In the following sections two different realizations of the same perceptron model will be presented. The different steps to be performed for the simulation are the following:

- $U_i : |0\rangle^{\otimes n}|0\rangle_a \rightarrow \frac{1}{\sqrt{m}}\sum_{j=0}^{m-1} i_j|j\rangle|0\rangle_a$.
- $U_w : \frac{1}{\sqrt{m}}\sum_{j=0}^{m-1} i_j|j\rangle|0\rangle_a \rightarrow \sum_{j=0}^{m-1} c_j|j\rangle|0\rangle_a$.
- $\mathrm{C}^n\mathrm{NOT} : \sum_{j=0}^{m-1} c_j|j\rangle|0\rangle_a \rightarrow \sum_{j=0}^{m-2} c_j|j\rangle|0\rangle_a + c_{m-1}|m-1\rangle|1\rangle_a$.

The difference will consist in the realization of U_i and U_w, but in both ways they will be unitary and they will satisfy the requested conditions.

2.3.1 Sign flip algorithm

It's important to find the most convenient and efficient way of implementing unitary operators U_i and U_w. The first way is to use sign flip blocks, that are operators defined in this way:

$$SF_{n,j'}|j\rangle = \begin{cases} -|j\rangle \text{ if } j = j' \\ |j\rangle \text{ if } j \neq j' \end{cases} \tag{2.10}$$

It is evident these operators are unitary and that $\mathrm{C}^N\mathrm{Z}$ gate is a particular case of $SF_{n,j}$, with $j = m - 1$.

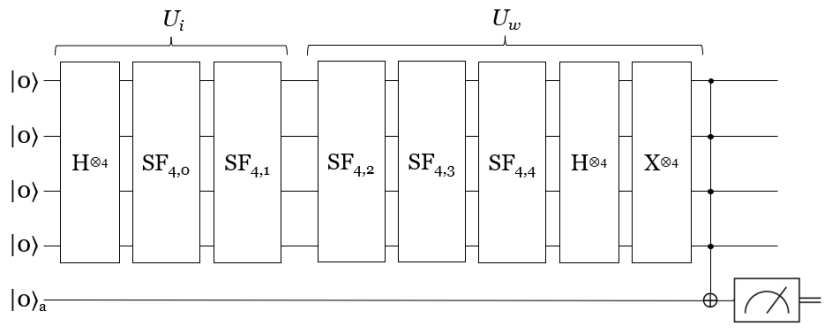

Figure 2.2: In this image the scheme of a four qubit perceptron with sign flip blocks is realized. The symbols $H^{\otimes 4}$ and $X^{\otimes 4}$ means 4 single operators are applied to the single qubits, and the last qubit $|0\rangle_a$ is the ancilla. In this exemple the input vector has weights $i_0, i_1 = -1$ and $i_j = 1$ for $j = 2, ..., 15$, while for the weight vector $w_2, w_3, w_4 = -1$ and $w_j = 1$ for all the other components. The last operation performed before the measurement is multicontrolled CNOT gate, where the target is the ancilla qubit (see Appendix).

In Figure 2.2, to build U_i, imagine to begin from the state $|0\rangle^{\otimes n}$ and apply the Hadamard gate to every qubit ($\mathrm{H}^{\otimes n}$) to obtain the transformation:

$$|0\rangle^{\otimes n} \to \sum_{j=0}^{m-1} \frac{i_j}{\sqrt{m}}|j\rangle \text{ with } i_j = 1 \ \forall j. \tag{2.11}$$

After that, $SF_{n,j'}$ can be used to obtain the desired weights i_j for the state $|\psi_i\rangle$ (Eq. 2.4).
Equivalently, to build U_w (Eq. 2.5), imagine to start from the state $|\psi_w\rangle$ (Eq. 2.6) and bring it to $\sum_{j=0}^{m-1} \frac{1}{\sqrt{m}}|j\rangle$ through sign flip blocks. After that, Hadamard gates are applied to every qubit, bringing the state to $|0\rangle^{\otimes n}$, followed by X-gates to finally find $|1\rangle^{\otimes n}$.
The final CNOT gate applied to the the five qubits (see Appendix) acts on the ancilla

as $|0\rangle_a \rightarrow |1\rangle_a$ in case the first four qubits are equal to 1. This is the reason why the final measurement made on the ancilla qubit gives information about the total quantum system.
For completeness $m/2$ independent sign flip blocks are required. Even if the invariance under a global -1 factor halves the number of necessary n-qubit operators to obtain any desired state, this number increases exponentially with n. It is therefore necessary to find a more efficient way of building U_i and U_w.

2.3.2 Hypergraph states algorithm

The second method involves instead a procedure called "hypergraph state generation subroutine"(HSGS).
To begin, a k-uniform hypergraph $g_k = \{V, E\}$ is a set of n vertices V with a set of edges E, where each edge connects exactly k vertices, and is called k-hyperedge. More generally, a hypergraph $g_{\leq n} = \{V, E\}$ is a set of n vertices V with a set E of hyper-edges of any order k, not necessarily uniform. Each edge is therefore a connection between a set of points.

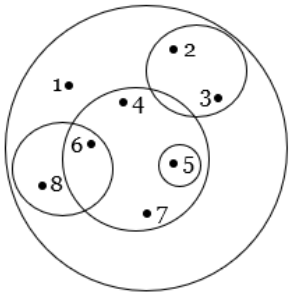

Figure 2.3: An exemple of an hypergraph, where each hyper-edge is represented by a circle including the desired vertices. The number of included vertices indicates the order of the edge.

It is possible to demonstrate that there is a one-to-one correspondence between the set $g_{\leq n}$ and REW states, defined in Section 2.2. To show this statement, introduce the notation for particular controlled-Z gates. Denote as $\mathrm{C}^{k-1}\mathrm{Z}_{i_1 i_2 \ldots i_k}$, with $k = 1, \ldots n$, controlled-Z gates that act on qubits $i_1, \ldots i_k$. Note that it is not specified which is the target and which are the control qubits, since permutation of indices does not change the action of the gate, which consists only in a change of sign of the basis states that have the chosen qubits set to 1. Then, assign to each vertex a qubit and initialize each qubit in the state $|+\rangle$ ($|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$). Wherever there is a k-hyperedge, perform a controlled-Z operation between the k connected qubits. Formally, if the qubits $i_1, i_2, \ldots i_k$

are connected, then perform the operation $\mathrm{C}^k\mathrm{Z}_{i_1 i_2 \ldots i_k}$.

$$g_{\leq n} \rightarrow \prod_{k=1}^{n} \prod_{\{i_1,\ldots i_k\} \in E} \mathrm{C}^k\mathrm{Z}_{i_1 i_2 \ldots i_k} |+\rangle^{\otimes n} \tag{2.12}$$

$\{i_1, \ldots i_k\} \in E$ means the k vertices are connected by a k-hyperedge. It is evident that for

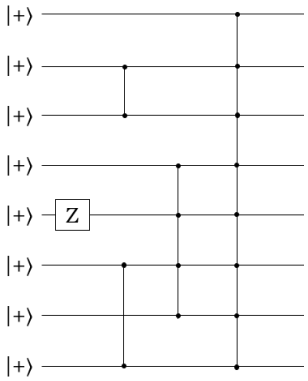

Figure 2.4: In this circuit the Z-gate and the other vertical lines are the $\mathrm{C}^k\mathrm{Z}$ gates applied in one-to-one correspondence with the hypergraph in Figure 2.3. From left to right, Z_5, $\mathrm{C}^1\mathrm{Z}_{23}$, $\mathrm{C}^1\mathrm{Z}_{68}$, $\mathrm{C}^3\mathrm{Z}_{4567}$ $\mathrm{C}^7\mathrm{Z}_{12345678}$ are applied to the initial state $|+\rangle^{\otimes n}$.

different hypergraphs, different REW states are obtained. Therefore with this procedure it has been shown that $g_{\leq n}$ is a subset of REW quantum states.
To show the converse use an operative approach, trying to associate to any of these quantum states a hypergraph. Start from a REW state $|\psi_{REW}\rangle$, and perform a procedure to finally obtain $|+\rangle^{\otimes n}$. At first erase all the minus signs of the states with one excitation (of the form $|0...01_j0...0\rangle$) applying Z_j gates. Note that in this way we might create unnecessary minus signs in front of states with more than one excitation. Then, apply $\mathrm{C}_1\mathrm{Z}$ gates in order to erase the negative signs in front of the components with two excitations (they could have been changed from the initial state due to the previous step). Note that this procedure will not change the sign of states with only one excitations taken into consideration before. Continue the procedure applying in the same way C^k until $k = n - 1$, erasing step by step the minus signs in front of the components of the computational basis. In general, at the step k, the signs in front of the states with up to k excitations will be erased. The set of gates that are needed to transform $|\psi_{REW}\rangle$ into $|+\rangle^{\otimes n}$ constitutes the corresponding hypergraph for the REW state under examination. Applying the same gates in the same order to $|+\rangle$ instead, $|\psi_{REW}\rangle$ is obtained. One can see that for every REW state there is a unique hypergraph associated in this way. The correspondence is therefore one-to-one. Even if the total number of gates to involve is almost the same ($O(2^n)$), the advantage of this algorithm comparing

it to sign flip consists in the fact that, while before all quantum gates involved n qubits, now at most one is a n-qubit gate. C^kZ with $k < n-1$ instead involves less qubits, and therefore some gates which involve different channels can be applied in parallel, with a consequent optimization in processing time. It is also possible to have an advantage in the number of quantum gates to use, since for example in actual superconducting quantum processors multi-qubit operations are not natively available and one has to decompose them into elementary operations. In addition, controlled-Z gates are unitary, as required (see Appendix).
The same perceptron model of the previous section will be described here with unitary operators U_i and U_w realized with the hypergraph states algorithm. The only basis state that can't change sign with CZ-gates is $|0\rangle^{\otimes n} = |0...0\rangle$. For this reason the total sign convention has to be taken such that the weight i_0 of the first vector basis is positive. For U_i operator, as for the sign flip implementation, at first the Hadamard gate has to be

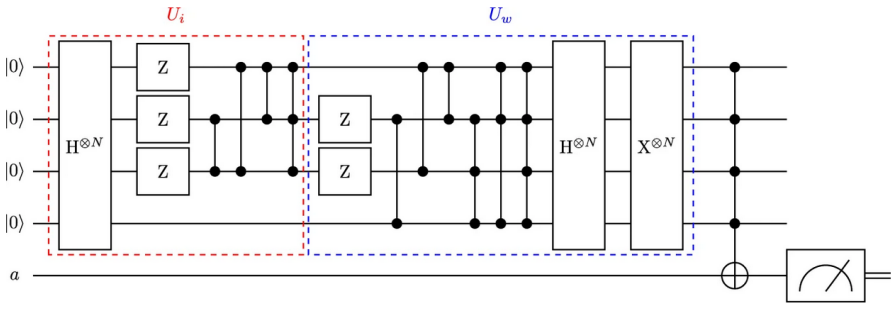

Figure 2.5: ([21]) HSGS realization of the perceptron model in Figure 2.2, with the same weight and input vectors. As explained above, the sign convention to adopt is the one with $i_0, i_1 = 1$ and $i_j = -1$ for $j = 2, ..., 15$, even if the vector to analyze was its negative counterpart. For the weight vector instead $w_2, w_3, w_4 = -1$ and $w_j = 1$ for all the other components.

applied to every qubit (Eq. (2.11)). Z_0, Z_1 and Z_2 set negative signs on the components of basis states with exactly one excitation corresponding to the first, the second or the third qubit. However they change the sign also to other non-desired components. In fact, after the first step, the basis states with exactly two excitations between the first three qubits have undergone two changes of sign, and therefore have positive components. C^1Z_{12}, C^1Z_{23} and C^1Z_{13} are necessary to fix them with correct minus signs. In this case, components i_{14} and i_{15} are affected by two changes of sign and are finally corrected by C^2Z_{123}. Equivalently, to build U_w (Eq. (2.5)), imagine to start from the state $|\psi_w\rangle$ (Eq. (2.6)) and bring it to $|+\rangle^{\otimes n}$ erasing step by step minus signs in the same way. After that, $H^{\otimes n}$ brings the state to $|0\rangle^{\otimes n}$ and $X^{\otimes n}$ finally to $|1\rangle^{\otimes n}$.

Chapter 3

Implementation of the artificial neuron

The model described in the previous chapter finds an important application in images recognition. Taking into consideration for example a 4-qubit circuit, a 16-dimensional Hilbert space is built, which can be represented by a 4×4 grid.

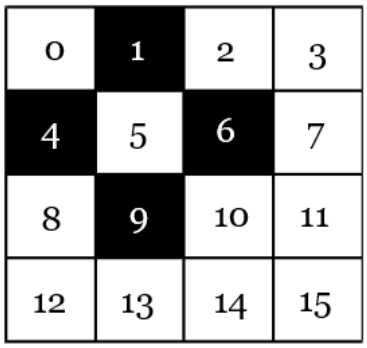

Figure 3.1: In the figure above the sixteen squares represent left-to-right and top-down the basis states of the Hilbert space of 4 qubits ($|00...00\rangle = |0\rangle^{\otimes N}, |00...01\rangle, |00...10\rangle, ..., |11...11\rangle\}$, denoted respectively by $i = 0, 1, ..., 15$). As seen in the previous chapter components are given by 1 or -1, represented here respectively by colours black and white. In the image one has the components equal to 1 for $i = 1, 4, 6, 9$ and to -1 for all the others.

In the following section the procedure and the results of the realization of a classical simulation of a circuit of the kind described in Chapter 2 will be analyzed. It will be thought in the logic of image recognition, and realized using the software development kit Qiskit. The purpose is to train the perceptron in order to recognize a precise image. The implementation has been made following the reference article [21].

3.1 Realization of the quantum perceptron model with Qiskit SDK

Qiskit is an open-source software development kit founded by IBM Research that can be used to construct quantum programs and run them on simulators or real quantum computers. The principal version of Qiskit uses mainly Python programming language. For what concerns the construction of the quantum circuit, instead of writing a code, there is on IBM quantum computing website ([2]) the possibility to graphically build the desired circuits through launching the application Quantum Composer. Using Python, instead, there are defined classes which permit to realize the quantum circuit and the state vector. One can at first set the number of qubits and classical bits. Then, there is a class which defines the circuit that takes qubits and bits as inputs, to which one can append the desired operators. The class for the state vector is instead the one which contains the information about the quantum state of the circuit. Its variables are always initialized with $|0\rangle^{\otimes n}$, where n is the number of qubits, and has to be changed through operators applied to the circuit. To run a variable of the class state vector on a defined quantum circuit, there is a function that updates the vector state if called, running the circuit on the specified device (real or ideal quantum computer).

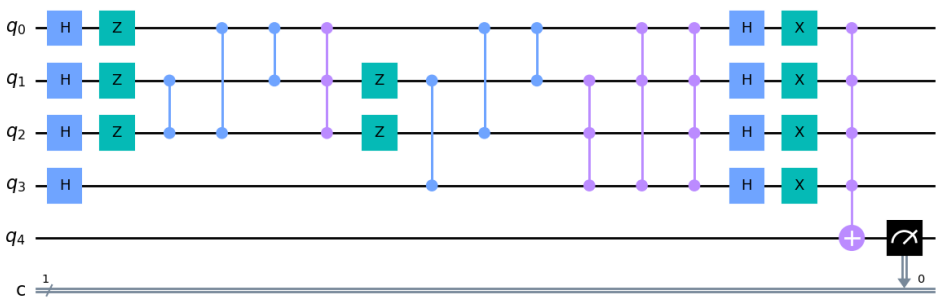

Figure 3.2: An example of realization of the circuit described in Figure 2.5 (of the previous chapter) with Qiskit.

The Python program that describes the desired circuit can then be run on a classical simulator that performs all unitary operations and measurements as specified by the code. Also noise that affects real computers can be simulated. All these features are present in the high-performance quantum computing simulator which is called Qiskit Aer, that has been used in this thesis. It provides interfaces to run quantum circuits with or without noise using multiple different simulation methods([1]).
If no additional parameters are specified when QasmSimulator is used, the system simulates the functioning of an ideal quantum computer, but as said before there are ways to introduce noise in the simulation. With the suitable class NoiseModel one can manually

add different type of errors on the desired qubits, or directly generate automatically a NoiseModel from the properties of real devices of the IBM quantum provider. To directly run the program on these real devices, IBM offers also access to their cloud-based quantum computers.

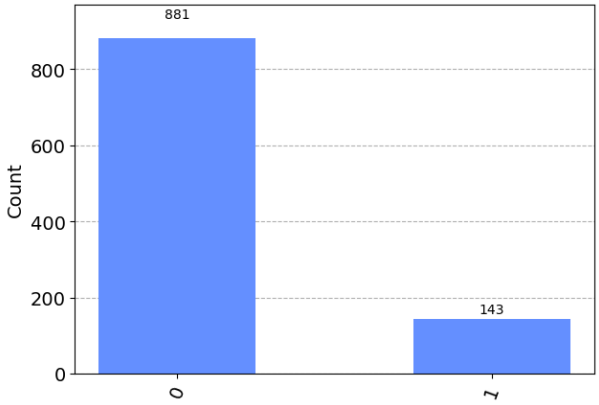

Figure 3.3: If the class QasmSimulator of Aer is used, after running the circuit in Figure 3.2 this histogram is obtained; in this case it acts as a perfect quantum computer without noise. It counts the occurrences in which the measurement of the ancilla qubit (fifth qubit in Figure 3.2) gives 1 and 0, after 1024 iterations. The results show that the scalar product of the input and weight vector is approximately 143/1024. In fact, as explained in the previous chapter, the scalar product of the two vectors corresponds to the occurrence of the state $|1\rangle$ in the measurement taken on the ancilla qubit. The obtained result is not exact because it derives from a classical statistic.

Figure 3.4: From left to right respectively the input and the weight vector of the circuit taken into consideration are represented with the notation adopted in Figure 3.1. The signs of the input vector are reversed in the circuit only because it is necessary to build it (for the HSGS algorithm) and this arrangement does not affect the result (see the previous chapter).

One of the most simple examples which clearly shows the differences between processing Qiskit programs on classic simulators of ideal quantum computers and on quantum real

computers consists in the realization of the circuit which builds the Bell state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ starting from the state $|00\rangle$, which is shown in Figure 3.5. These differences can be observed in Figure 3.6, where one can see in the real case there are outcomes that should not occur in a measurement of a Bell state. In fact, noise in real quantum systems leads to detect with a small rate also the states $|01\rangle$ and $|10\rangle$.

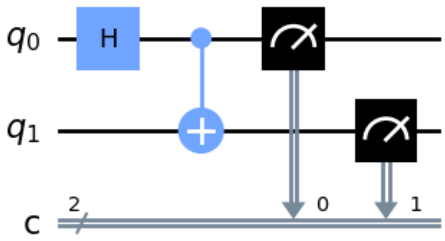

Figure 3.5: In this circuit one Hadamard gate is applied to q_0 and one CX gate to q_0 and q_1 (q_0 is the control and q_1 is the target). It acts as $|00\rangle \rightarrow \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. At the end a measurement is taken on both qubits.

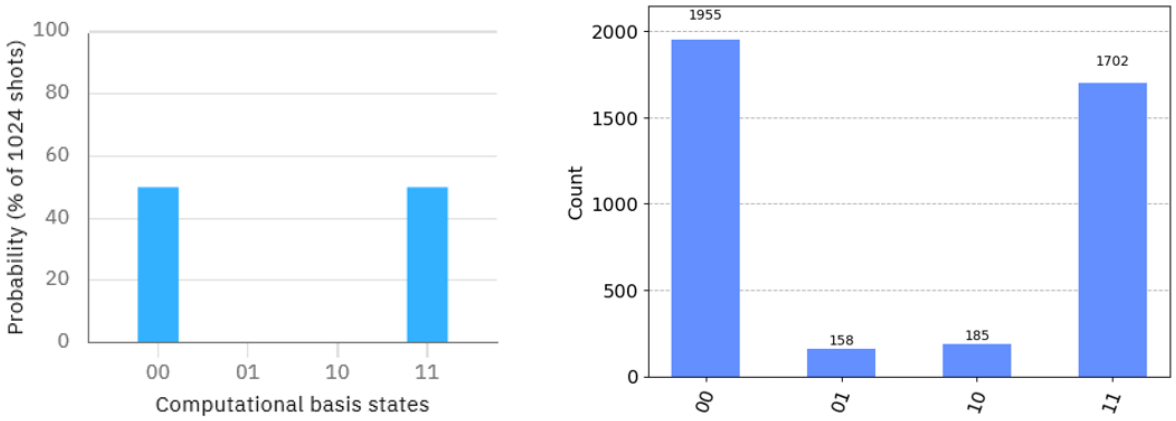

Figure 3.6: *On the left:* Histogram of the occurrences of measurements of the state in the circuit represented in Figure 3.5, using the classical simulator QasmSimulator, without noise. The histogram is the result of 1024 iterations. *On the right:* Histogram of the occurrences of measurements of the state in the circuit represented in Figure 3.5, using a real cloud-based IBM quantum computer. For this example the backend, which is the computer hardware, *ibmq_quito* has been used (see [2] for details). The histogram is again the result of 1024 iterations.

In [21], to analyze the efficiency of the quantum perceptron discussed, a statistic made on all the responses obtained by the machine with respect to all possible input patterns and all weight vectors is made. This is realized with 2-qubit quantum circuits, as the Hilbert space of the correspondent states is 4-dimensional and has therefore 2^4 vectors (if the components are restricted to have only binary values). The resulting grid has therefore 16×16 squares, each one corresponding to one possible input pattern and one possible weight. In [21] it is run both on a classical simulator and on a real quantum computer. The procedure carried out is calculating for every matching possibility of weight(w) and input(i) the response that the circuit as in Figure 3.2 would give. From algebraic calculation it turns out that the only positive response (which corresponds as explained in Chapter 2 to the absolute value of the scalar product between i and w greater than 0.5) should be obtained only for $i = w$ or $i = -w$. However the border is not so clear when the code is run on the real quantum computer, which shows how much the noise can alter the results obtained by unitary quantum dynamics. Figure 3.7 shows also the difference in using sign flip and HSGS algorithm, which means there can be a significant optimization and reduction of noise if one choose the best algorithm to build circuits. $\mathcal{D}$ is the root mean square of the absolute value of the scalar product between i and w, calculated taking as the mean value the ideal result represented in Figure 3.7 a). On the vertical axis is represented the square of the absolute value of the scalar product.

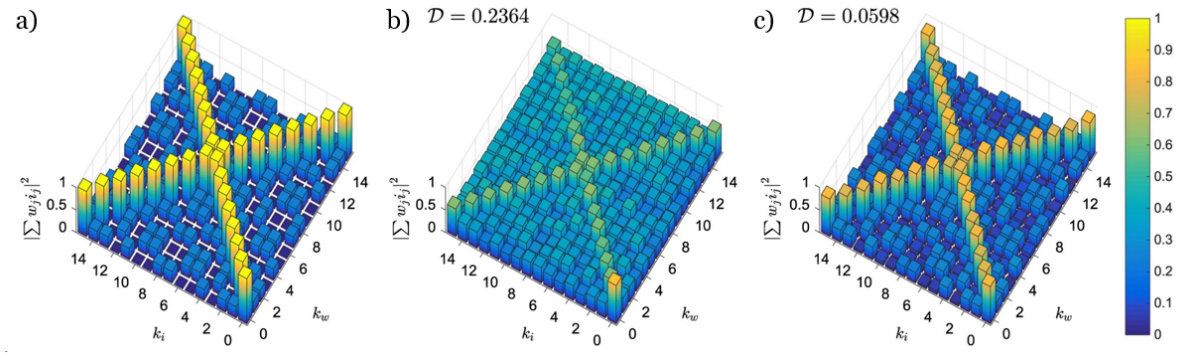

Figure 3.7: ([21]) *a*) Ideal outcome of statistic described in the paragraph above, simulated on a classical computer. *b*) Results from the Tenerife processor using multi-controlled sign flip blocks algorithm described in section 2.3.1. *c*) Results from the Tenerife processor using hypergraph states algorithm described in 2.3.2.

3.2 Learning procedure on a quantum perceptron

In this section will be explained a possible procedure to implement the training of an image recognition perceptron on a quantum computer. The results will be analyzed with a classical simulator and the model adopted will be the one explained in the previous chapter.
The first thing to notice is that for the learning procedure a huge quantity of input patterns have to be analyzed and the weight vector has to be updated each time a wrong response is given by the system (see 1.3.3). As the HSGS algorithm used to realize the perceptron requires that a circuit with different operators has to be constructed for each different input pattern and weight vector, this implies the circuit has to be built from zero at each iteration of the training.
In the case discussed below the pattern chosen to teach to the artificial neuron is the following:

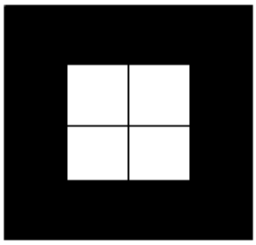

Figure 3.8: The figure represents the pattern that one wants the artificial neuron to learn. It corresponds, not taking into consideration the normalization factor, to the vector with the following components: $w = (1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1, 1)$.

3.2.1 Algorithm for the learning

In this section the logical steps realized in the Python program will be discussed, including the procedure to obtain the training set, which contains the information needed by the machine to learn the desired pattern.
The realization of circuits will be made possible by Qiskit, and in particular for simulating the running on a quantum computer the classical simulator QasmSimulator will be used.
First of all, the function $evaluate(weight, input)$ is defined, which takes the two vectors $weight$ and $input$ belonging to the Hilbert state space given by 4 qubits. This function gives out 0 if the absolute value of the scalar product between them is less than 0.5 and 1 if it's greater or equal to 0.5 (building the quantum circuit using HSGS algorithm). $Weight$ e $input$ can only have components constituted by $\pm\frac{1}{\sqrt{m}}$, where $m = 2^4$ (4 is the

number of qubits in addition to the ancilla) and it makes the vectors normalized to 1. Since *evaluate* gives back an absolute value (for how the quantum circuit is built), it gives 1 also in the case $weight = -input$.
Then, after the choice of a weight vector w from the same 4-qubit Hilbert space, a code has to be written to form the training set. The code used in this discussion selects randomly from the Hilbert space 3000 *input* vectors for which the function $evaluate(w, input)$ gives 0 as a result and 50 for which it gives 1 (with some restrictions that will be discussed later in this chapter). The code writes the training set constituted by all the *input* vectors on a file, together with the information about the output given by $evaluate(w, input)$.

The training procedure is the following:

- a *weight* vector is selected randomly in the Hilbert space;

- the following procedure is repeated 5000 times:

  – an *input* vector is randomly selected from the training set;

  – the function $evaluate(w, input)$ is called. If it returns back the value according to the correct one stored in the training set ($evaluate(w, input)$), nothing happens. Otherwise, if it returns a wrong response, the *weight* vector is updated in this way:

    1. $evaluate(weight, input)$ returns 1 but $evaluate(w, input)$ gives 0. In this case one goes through the components associated to all vector basis states (denoted by i), and selects the ones such that $weight_i = input_i$. Half of the components of the *weight* vector that satisfy this equation are randomly selected and changed of sign. The fraction l_n of them randomly selected in this case in which the response of the artificial neuron is positive even if the *input* vector in the training set is labelled as negative, takes the name of negative learning rate. In the artificial neuron implemented in this discussion l_n is taken equal to 0.5;
    2. $evaluate(weight, input)$ gives as a result 0 but $evaluate(w, input)$ gives 1. In this case *weight* vector components that have to be changed with a certain probability are the ones for which $weight_i \neq input_i$. A fraction l_p of components of the vector *weight* which satisfy this requirement are selected and changed of sign. l_p takes the name of positive learning rate and in the discussed case is taken equal to 0.5.

3.2.2 Remarks and results

The first aspect to notice is that this classical algorithm is written without taking account of the fact that the function *evaluate* does not distinguish between a vector and its

negative. This ambivalence implies that if one does not modify in some way the training set and the training procedure with this awareness, the *weight* vector stored in the memory of the artificial neuron does not converge, as it is constantly attracted both by w and $-w$.

In the graphics showed in the following pages the *iterations* of the training procedure are represented along the x-axis, while the y-axis indicates the $fidelity$, which is the scalar product between the vector *weight* that is updated during learning and w, the correct vector from which the training set has been built. Note that the $fidelity$ is therefore a real number and not an absolute value.

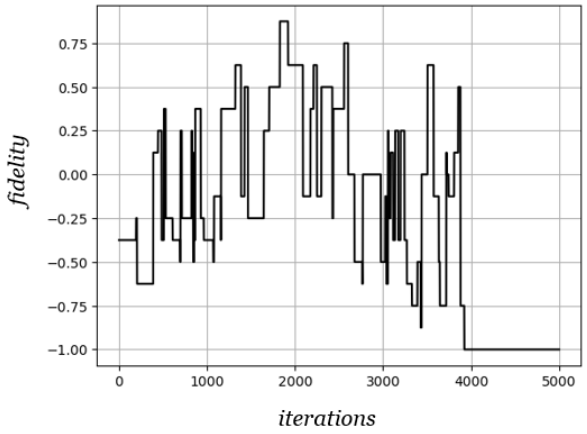

Figure 3.9: The graphic is obtained with one iteration of the learning procedure described in section 3.2.1, with no further requirements on the training set, that is uniformly distributed on the Bloch sphere.

In Figure 3.9 explained in the previous lines, the vector *weight* tends to approach also $-w$ and it converges to it after approximately 4000 iterations. One can also see that the curve has an oscillating trend between w and $-w$ before reaching one of them, which means it's attracted by both during the process. Analyzing the trend in Figure 3.10, which is an average of 100 trainings, one can see the curve does not converge to 1, but remains in a range around 0. This means the procedure makes confusion between the pattern w and its negative, converging half the time to the former and half to the latter.

One of the ways implemented to avoid the convergence to $-w$ is to build the training set only with vectors contained in one hemisphere, and in addition forcing the vector *weight* while it is being updated to remain in that hemisphere. It is necessary to choose an hemisphere which contains the vector w used to build the training set. This is obtained by imposing as a request for vectors in the training set to have the first component equal to 1, starting the procedure from a vector w that has this property and forcing

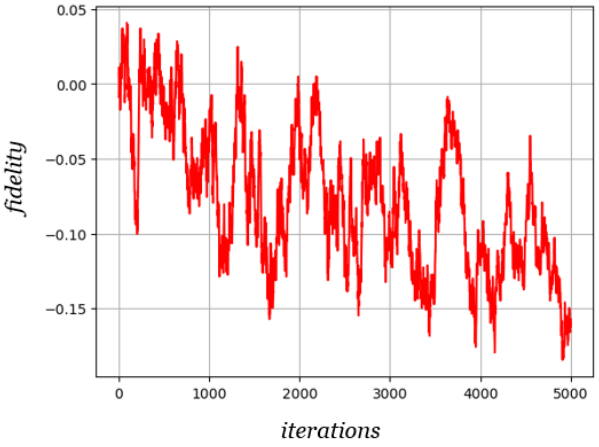

Figure 3.10: The graphic is obtained from an average after repeating 100 times the same training procedure of figure 3.9. The root mean square has been calculated and it has been found that going through the learning procedure it tends to increase and to approach 1. It is predictable since the single training procedure finishes usually with a $fidelity = \pm 1$ while the mean of the 100 training procedures has a value around 0.

it during the learning to keep this feature. In this way w, while being updated, never changes hemisphere. It is important to notice that keeping the vector *weight* in the desired hemisphere is an additional request one has to manually add during the learning procedure, because only changing the training set does not guarantee it.
In Figure 3.11 one can see the convergence of the $fidelity$ to 1 is reached approximately after 2000 iterations, with the value 0.8 exceeded after 1000. This means the algorithm with this corrections works as desired. After 4000 iterations the curve reaches exactly 1, which means in all the 100 training procedures the pattern w has been learned.

However the method of optimization of the training set used in Figures 3.11 and 3.12 avoids the convergence of the $fidelity$ to -1 but does not prevent the match between *input* vectors and the negative pattern $-w$. An explicit example is given by

$$input = (1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, -1, -1, -1),$$

which have the first component greater than 0, and

$$w = (1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1, 1),$$

which is the same of Figure 3.8. This is an example for which the vector *weight* stored in the artificial neuron could be attracted to $-w$. In fact, $evaluate(w, input) = 1$ because *input* is similar $-w$, and the algorithm then makes the vector *weight* approach $-w$.
To overcome this further problem, an alternative way of avoiding confusion between w

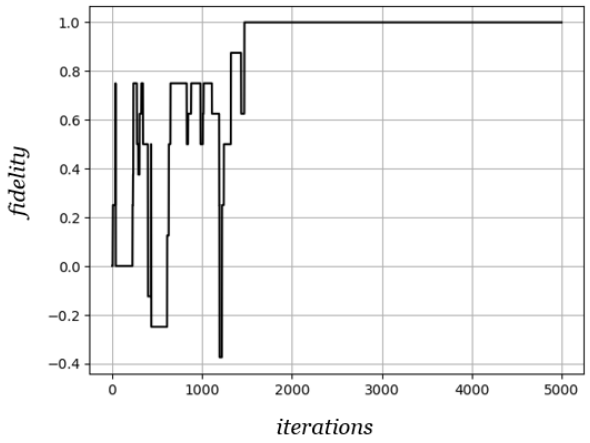

Figure 3.11: Graphic of a learning procedure with the training set contained in the hemisphere with the first component greater than 0 and the additional request for keeping *weight* in the same hemisphere. One can notice the convergence to 1 of the $fidelity$ is reached soon, even if there are some oscillations until approximately $iterations = 1500$. It is intrinsic in how the algorithm is build and there is no way to completely remove this oscillations, but at least one can try to eliminate the matching between *input* and $-w$, which in this case is still present (as explained in this section).

Figure 3.12: Graphic of the average of 100 training procedures equal to the one of Figure 3.11. One can calculate also the root mean square and it turns out than it starts from 0.25 at the beginning of the iterations and finishes with the value 0 after the curve has reached 1.

and its negative consists in selecting the vectors for the training set among those that have the number of positive components greater than the one of the negative ones. This

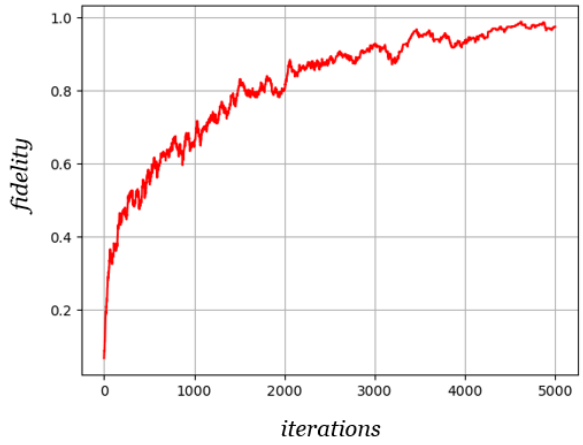

Figure 3.13: This is the graphic obtained building the training set distributed uniformly in the hemisphere with the first component greater than 0 without any changes in the training algorithm. The average is always computed after 100 training procedures. It is evident that comparing it with the graphic in Figure 3.12 the convergence of the curve to the value 1 is slower and it never reaches it. The root mean square of the $fidelity$ starts at the beginning of the procedure from the value 0.25 and finishes with 0.2 around the last *iterations*.

avoids confusion as the one described in the previous lines.
The problem with this training set is that the vectors which have more positive compo-

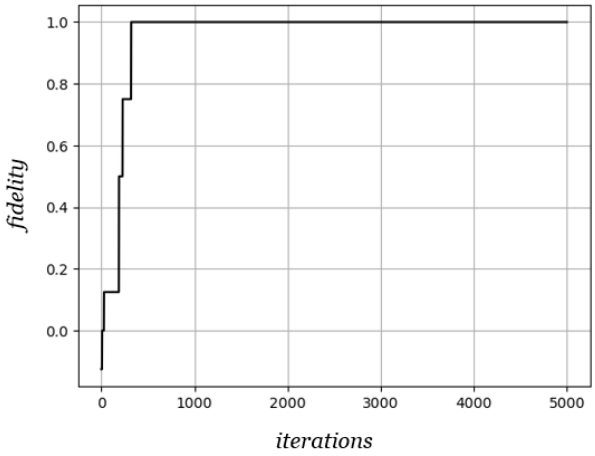

Figure 3.14: Example of a successful learning procedure using a training set with more positive components than the negative ones. The convergence seems fast, but to analyze correctly the trend one has to look at the average after more training procedures.

Figure 3.15: The graphic is again an average between 100 training procedures with the same choice of training set of Figure 3.14. It has also been found that the statistical errors have the same trend of the graphic in Figure 3.12.

nents than the negative ones constitute a half of the Bloch sphere but not a hemisphere. In fact, they are points distributed over all the Bloch sphere, and more operators are needed to flip components signs. This is a disadvantage in terms of operators involved and processing time.
The trend in Figure 3.15 is similar to that one of Figure 3.12, even if the convergence of the curve in the image in this case seems a little slower. The reason is probably because in Figure 3.12 there are 15 components to learn instead of 16 (the first is fixed). Nevertheless also in this case after almost 4000 iterations the 100% of the procedures end with $weight = w$, which means the pattern has always been learnt.

3.3 Conclusions

This discussion has shown the implementation of a quantum perceptron that, after an appropriate training period, is able to recognize the desired pattern. Once the pattern has been learnt by the perceptron, it remains stored in the memory (through the weight vector) and therefore the artificial neuron will give a positive response whenever it receives an input similar to it (in this case the one displayed in Figure 3.8). Some choices have been taken to optimize the learning and to avoid time delays in the convergence of the weight vector.
During the algorithm implementations two principal critical points had to be faced. They are both inherent in the quantum perceptron model described in Chapter 2. The first consists in the fact that the quantum circuit, which from a weight and an input vector gives back the response that indicates if they overlap or not, has to be built depending

on the vector themselves, because the operators have to be chosen according to vector components. This means that for each input and weight vector the circuit has to be constructed from zero and this leads to a significant time delay. The second concerns an intrinsic limit of quantum mechanics, since information has to be taken from measurements, in the form of probability amplitudes of detection. In fact, this leads to the loss of a phase factor, which in the case of the quantum perceptron implies that it does not distinguish a vector from its negative (w and $-w$, in the notation of 3.2.2). This is a problem because the training algorithm used in this discussion is a classical algorithm that takes into consideration the signs of the components, but it has been solved through an appropriate choice for the training set and some constraints in the algorithm (as discussed in 3.2.2).
The attempts made for the choice of the training set and for the constraints in the algorithm suggest the most performing algorithm and training set are the ones which trend is represented in Figure 3.12, where the first component of the *weight* vector is fixed and also the training set is chosen with this property.
Also the procedures used in Figures 3.13 and 3.15 perform well and avoid the weight vector to converge to $-w$. It is remarkable the fact that in cases of Figure 3.12 and 3.15 all procedures of the average taken finish with the right pattern stored in memory. This means the algorithm and the training set are correctly chosen.
In this discussion only the pattern in Figure 3.8 is taken into consideration for the learning, but it could be interesting to analyze the results of the learning procedure building the training set with a different vector w. With this change the same artificial neuron could lead to different responses by the neuron, such as a slower or faster convergence. Furthermore, a fundamental choice for the optimization of the learning procedure is the choice about how to build the training set from the selected pattern w. In this discussion only some possibilities have been examined, but in general the problem of finding the most suitable training set is not obvious and more attempts have to be made. Moreover, the statistics here have been taken after 100 learning procedures, for a problem of processing time, but to reduce statistical errors one could have increased this number (in the reference article [21] the average is made on 500 learning procedures).
The last comment is about running the code on a real quantum computer. In fact, the one implemented is a classical training algorithm thought in order to use a classical simulator and does not take into consideration the possible noise. In fact, using a real quantum computer some errors not compatible with the classical statistical ones could be made both in the calculation of the function *evaluate* or in general in the flipping of signs during the learning procedure. For this reason maybe the algorithm should be modified or noise corrections should be introduced.

Appendix: Introduction of quantum computing

Linear and unitary evolution of quantum systems

The connection to the mathematical formalism of quantum mechanics to the physical world is given by the postulates of quantum mechanics, which allow to give a physical interpretation to the quantum state and its evolution.

Postulate 3.1 *Every isolated physical system is associated to a Hilbert space called the state space and the system is described at every moment by a state vector* $|\psi\rangle \in \mathcal{H}$.

If it is assumed that every vector in the state space is a possible state, then the *superposition principle* holds. Furthermore, in every Hilbert space a inner product is defined, and it is assumed that the state is defined up to a normalization factor, which means $\langle\psi|\psi\rangle = 1$.

Postulate 3.2 *To every observable in classical mechanics there corresponds a linear, self-adjoint operator in quantum mechanics.*

This postulates ensures that the eigenvalues spectra consists of real numbers, and that eigenvectors constitute an orthonormal basis.

Postulate 3.3 *The time evolution of a closed quantum state is governed by a unitary transformation.*

$$|\psi(t)\rangle = U(t,0)|\psi_0\rangle, \tag{3.1}$$

where $|\psi_0\rangle = |\psi(0)\rangle$.
This is equivalent to say that the evolution of a closed quantum system is governed by the Schrödinger equation

$$i\hbar\frac{d|\psi(t)\rangle}{dt} = H|\psi(t)\rangle. \tag{3.2}$$

In fact, if one assumes the closed system evolution is described by the Schrödinger equation, from Eq. (3.1) and (3.2),

$$i\hbar\frac{dU(t+\Delta t,t)}{dt} = HU(t+\Delta t,t), \tag{3.3}$$

which means

$$U(t+\Delta t,t)=\exp\left[\frac{-iH\Delta t}{\hbar}\right]. \tag{3.4}$$

The evolution operator is therefore a unitary operator. On the other hand, if one assumes the evolution is given by a unitary operator U, takes the vector basis $|\psi_n\rangle$ of the space in which $|\psi(t)\rangle$ lives and the matrix representations of the operators, in the limit $t\to 0$

$$U_{nm}(t+\Delta t,t)=\delta_{nm}-\frac{i}{\hbar}tH_{nm}, \tag{3.5}$$

for some operator H, where $U_{nm}(t+\Delta t,t)=\langle\psi_n|U(t+\Delta t,t)|\psi_m\rangle$ and $H_{nm}=\langle\psi_n|H|\psi_m\rangle$. Since

$$U^*_{kn}U_{km}=(\delta_{kn}+\frac{i}{\hbar}tH^*_{kn})(\delta_{km}-\frac{i}{\hbar}tH_{km})=\delta_{nm}-\frac{i}{\hbar}tH_{nm}+\frac{i}{\hbar}tH^*_{mn}+O(t^2) \tag{3.6}$$

U_{nm} is unitary if H_{nm} is hermitian and if this happens H is self-adjoint (the Hilbert space is finite dimensional). One therefore has, for the completeness relation $\sum_k|\psi_k\rangle\langle\psi_k|=\mathbb{I}$,

$$\begin{aligned}\langle\psi_n|\psi(t+\Delta t)\rangle=&\langle\psi_n|U(t+\Delta t,t)|\psi(t)\rangle=\\ \sum_m U_{nm}\langle\psi_m|\psi(t)\rangle=&\langle\psi_n|\psi(t)\rangle-\frac{i}{\hbar}\Delta tH_{nm}\langle\psi_m|\psi(t)\rangle\Rightarrow\end{aligned} \tag{3.7}$$

$$\langle\psi_n|\psi(t+\Delta t)\rangle-\langle\psi_n|\psi(t)\rangle=-\frac{i}{\hbar}\Delta t\langle\psi_n|H|\psi_m\rangle\langle\psi_m|\psi(t)\rangle\Rightarrow \tag{3.8}$$

$$\frac{|\psi(t+\Delta t)\rangle-|\psi(t)\rangle}{\Delta t}=-\frac{i}{\hbar}\Delta tH|\psi_m\rangle\langle\psi_m|\psi(t)\rangle\Rightarrow \tag{3.9}$$

$$i\hbar\frac{d|\psi(t)\rangle}{dt}=H|\psi(t)\rangle \tag{3.10}$$

Therefore, for each unitary operator U, one can find for each instant t an hermitian operator H for which the system will satisfy the Schrödinger equation. This operator will be called the Hamiltonian, and its eigenvalues are the permitted values of energy for the system. The Schrödinger equation only applies to isolated systems, for which the hamiltonian does not depend on time, or to systems that interact weakly with the environment or in a way that doesn't break the quantum coherence (avoiding quantum collapse). Since the interaction with the real world does not respect this constraints, the evolution of quantum systems is usually not unitary, but stochastic and governed by probabilities.

Postulate 3.4 *Quantum measurements are described by sets of operator $\{M_n\}$ acting on the system state space, together with a set of possible outcomes $\{\lambda_m\}$. For each set of operators, that corresponds to an individual measurement, M_n must satisfy:*

$$\sum_n M_n^\dagger M_n=\mathbb{I} \tag{3.11}$$

The probability that the outcome λ_m occurs is

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle, \tag{3.12}$$

and the state after the measurement becomes

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}. \tag{3.13}$$

It can be shown that Eq. (3.11) ensures the sum of probabilities is equal to 1. It is important to notice that measurement opertors are not unitary unless there is only one possible outcome in the measurement. Different types of measurements will be discussed later in this chapter.

Postulate 3.5 *The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if $|\psi_i\rangle$ represents the state of the i-th component, the state of the composite system will be given by $|\psi_0\rangle\otimes|\psi_1\rangle\otimes...\otimes|\psi_n\rangle$, where n is the number of component physical systems.*

This postulate is suggested by superposition principle, which always holds in quantum mechanics, that guarantees the sum of two quantum states is again a quantum state. Postulte 3.5 allows to keep the linearity of quantum mechanics also in the composite physical system.

Qubit and multiple qubits

The qubit is the most basic quantum information unit. It is the quantum analog of the classical bit, which can take the values 0 and 1. However in quantum mechanics the superposition principle necessarily leads to take into account also all the states that can be written in the following way:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{3.14}$$

with $\alpha^2 + \beta^2 = 1$, where α and β are probability amplitudes.
In this way a 2-dimensional complex vector space is built and normalization can always be applied. For the construction of the Hilbert space $\mathcal{H}$ the states $|0\rangle$ and $|1\rangle$ constitute an orthonormal basis and are known as *computational basis states*. Eq. (3.14), apart from a modulo one phase factor, can be also rewritten as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{3.15}$$

This representation can be easily visualized by employing the so-called *Bloch sphere*, where $|0\rangle$ and $|1\rangle$ correspond respectively to the angles $\theta = 0$ and $\theta = \pi$ for every possible

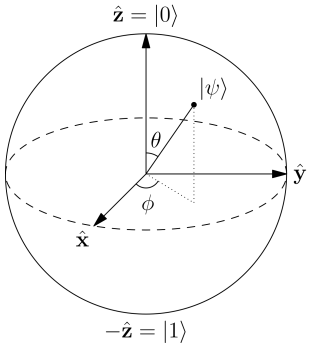

Figure 3.16: Bloch sphere: θ and ϕ, the parameters in Eq. (3.15), are respectively the polar and azimuthal angles of the sphere.

value of ϕ. An interesting observation follows from the infinite number of possible states contained in $\mathcal{H}$. However, the performance of a measurement in the computational basis leads to an irreversible change of the system and only two values can be obtained, while the remaining information is lost. The quantum qubit has therefore some hidden information that can be carried but can't be detected.
If one considers 2 qubits $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, computing a tensor product one finds the total state is

$$|\psi_1\rangle \otimes |\psi_2\rangle = \alpha_1\alpha_2|0\rangle_1 \otimes |0\rangle_2 + \alpha_1\beta_2|0\rangle_1 \otimes |1\rangle_2 + \beta_1\alpha_2|1\rangle_1 \otimes |0\rangle_2 + \beta_1\beta_2|1\rangle_1 \otimes |1\rangle_2. \quad (3.16)$$

The tensor product between the computational basis states is denoted simply as the set $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$, and constitutes a basis for a new extended space $\mathcal{H}^{\otimes 2}$. Again for superposition principle there are also some states that live in $\mathcal{H}^{\otimes 2}$ but can't be written in terms of tensor product between the two qubits. These states are said to be entangled, which means there exists a correlation between the qubits. The correlation can be total or partial. In the following four states, which take the name of *Bell states*, the correlation is total, which means that from a measurement of one of the two qubits, also the value of the other is known with probability 1.

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (3.17)$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \quad (3.18)$$

$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad (3.19)$$

$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \quad (3.20)$$

The procedure with the same considerations can be repeated in the case of n qubits, obtaining a 2^n-dimensional Hilbert space.
There are ways of quantifying the entanglement, they take the name of entanglement measures. For two-qubit quantum states, all possible entanglement measures turn out to be equivalent. One of this measures is the *tangle*, which is defined by:

$$|\psi\rangle \to \sqrt{|\langle\psi|Y\otimes Y|\psi^*\rangle|} \tag{3.21}$$

where

$$\begin{aligned} Y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \\ |\psi\rangle &= a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle, \\ |\psi^*\rangle &= a^*|00\rangle + b^*|01\rangle + c^*|10\rangle + d^*|11\rangle. \end{aligned} \tag{3.22}$$

$Y\otimes Y$ is the tensor product of the two Y-Pauli matrices, which is defined by:

$$Y\otimes Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \equiv \begin{bmatrix} 0\cdot\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & -i\cdot\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\ i\cdot\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & 0\cdot\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \tag{3.23}$$

Therefore from Eq. (3.22) and (3.23) one has

$$Y\otimes Y|\psi^*\rangle = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a^* \\ b^* \\ c^* \\ d^* \end{bmatrix} = \begin{bmatrix} -d^* \\ c^* \\ b^* \\ -a^* \end{bmatrix} \Rightarrow \tag{3.24}$$

$$\sqrt{|\langle\psi|Y\otimes Y|\psi^*\rangle|} = \sqrt{|2b^*c^* - 2a^*d^*|} \tag{3.25}$$

One can check this expression lives between 0 and 1 and is 0 for product states and 1 for the Bell states. At first take two qubits $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$. This means

$$|\psi\rangle \equiv |\psi_1\rangle\otimes|\psi_2\rangle = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle, \tag{3.26}$$

which with the notation used in (3.22) corresponds to

$$\begin{aligned} \alpha_1\alpha_2 &= a \\ \alpha_1\beta_2 &= b \\ \beta_1\alpha_2 &= c \\ \beta_1\beta_2 &= d. \end{aligned} \tag{3.27}$$

Substituting Eq. (3.27) in (3.25), one finds $\sqrt{|\langle\psi|Y\otimes Y|\psi^*\rangle|}=0$. If instead one considers Bell states, defined in (3.17)-(3.20), immediately finds $\sqrt{|\langle\beta_{ij}|Y\otimes Y|\beta_{ij}^*\rangle|}=1$ for $i=0,1$ and $j=0,1$.
If the total state has only a part that can be decomposed in a product of states, the *tangle* measure will be a value included in the range]0, 1[.

No-cloning theorem

There is an important difference between quantum and classical circuits, due to theoretical limits in quantum computing. In fact, while in classical computing data can be copied without any limit, in quantum circuits the copying operation is not possible at all. In fact, if we assume to begin with the states

$$|\psi\rangle\otimes|s\rangle \text{ and } |\phi\rangle\otimes|s\rangle \tag{3.28}$$

and we apply some unitary operation such that

$$|\psi\rangle\otimes|s\rangle\rightarrow U(|\psi\rangle\otimes|s\rangle)=|\psi\rangle\otimes|\psi\rangle \tag{3.29}$$

$$|\phi\rangle\otimes|s\rangle\rightarrow U(|\phi\rangle\otimes|s\rangle)=|\phi\rangle\otimes|\phi\rangle. \tag{3.30}$$

For the conservation of the scalar product using unitary operators, multiplying the two equations above and assuming $\langle s|s\rangle=1$,

$$\langle\psi|\phi\rangle=(\langle\psi|\phi\rangle)^2, \tag{3.31}$$

which holds only for null or orthogonal vectors.
This means in general it's not possible to produce an independent copy of a quantum state through unitary dynamics.

Single and multiple qubit gates, controlled and universal gates

Quantum circuits are the quantum analog of classical circuits and they carry qubits instead of bits. Each qubit is carried by a channel symbolized by a wire and information is transformed through quantum gates. Each wire does not necessarily correspond to a physical wire, but can represent for example the passage of time or the translation in space of a particle. Gates taken into account in quantum computing have to be linear and unitary, as a plausible assumption from the postulates of quantum mechanics. Actually, if one assumes quantum gates to be linear, unitarity follows from the request of the final

state to be normalized, for the conservation of probability. This is the only request, in fact all linear and unitary operators constitute a valid quantum gate.
An exception is constituted by measurements, which break unitary dynamics and turn quantum information into classical.
First of all, let's analyze the quantum NOT gate, acting on a qubit. Clearly, if we consider the computational basis, similarly to the classic case the NOT gate takes $|0\rangle$ into $|1\rangle$ and viceversa. In matrix representation,

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.32}$$

$$X|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{3.33}$$

Other important quantum qubit gates are the Z-gates

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.34}$$

$$Z|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} \tag{3.35}$$

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.36}$$

and the Hadamard gates

$$H|\psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha + \beta \\ \alpha - \beta \end{bmatrix}. \tag{3.37}$$

The Z-gate is the representation of the operator σ_z acting on the basis constituted by its two eigenvectors, $|0\rangle = |\uparrow\rangle$ (spin-up) and $|1\rangle = |\downarrow\rangle$ (spin-down), while the H-gate can be interpreted as a change of basis where the two new basis states are respectively $|0\rangle \to \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle \to \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.
Alternatively, considering the Bloch sphere, it can be shown that every single qubit gate can be written as a rotation and a reflection about axis. In particular, the Hadamard gate is a rotation of the basis states of $\frac{\pi}{2}$ about the $\hat{y}$ axis in the Bloch sphere followed by a rotation of π about the $\hat{x}$ axis.
For every single qubit gate one can build a respective controlled qubit gate. The simplest is a gate that has two qubits in input, the *control* qubit and the *target* qubit. The function of the control one is to able or disable the action of a single qubit gate U on the target. If the vector basis are chosen to be $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, in this order, where the first

qubit is the control and the second the target, this gates are represented by the matrix

$$U_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{bmatrix}. \tag{3.38}$$

Note that if the matrix U is unitary, the controlled gate obtained from it is unitary too. At first, take into consideration the controlled-NOT, or CNOT. In this case the gate applied to the target is the X gate.
The gate is represented by the matrix

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.39}$$

The behaviour of CNOT on the basis states is the following:

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |11\rangle \\ |11\rangle &\rightarrow |10\rangle. \end{aligned} \tag{3.40}$$

If one considers as output the second qubit after the transformation, classically the behaviour is that of the XOR gate. This explains the symbol $\otimes$ used in quantum circuits (Figure 3.17).

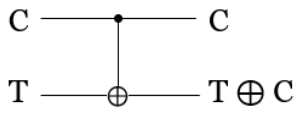

Figure 3.17: CNOT gate, where C indicates the control qubit, T is the target qubit and the symbol $\oplus$ is the classical XOR extended linearly to quantum qubits.

T and **C** in Figure 3.17 above are not written as quantum qubit states because the total state can be entangled and so it could not be possible to decompose it into the tensor product of two single states. An entangled state can be given as input but can be also obtained as output of the gate from a non-correlated initial state.
More precisely, there is a way to quantify the entangling power of a general two-qubit gate. The entangling power of a two-qubit gate U is defined as

$$EP(U) = \langle E(U|\psi_1\rangle \otimes \psi_2\rangle)\rangle_{\psi_1,\psi_2}, \tag{3.41}$$

where $E()$ indicates the *tangle* calculated on the state in brackets, and $\langle\rangle_{\psi_1,\psi_2}$ indicates the mean value averaged over the two qubits $|\psi_1\rangle$ and $|\psi_1\rangle$ sampled uniformly on the Bloch sphere. One can calculate that for CNOT gate $EP = \frac{2}{9}$. The requirement of conservation of probability, that translates in the request of gates to be unitary, is satisfied by CNOT gate. Note that this is due to the fact that CNOT is reversible, since from the output state one could rebuild the input. In fact, since the control qubit does not change with the transformation, it is known if the gate has flipped the target or not. On the contrary the classical XOR gate does not conserve the control bit and definitely loses the information. It is therefore an irreversible gate. In general, quantum unitary gates are always invertible, because the inverse of a unitary matrix exists and is again a unitary matrix, which therefore can constitute a quantum gate too.

If the operation to apply to the target is instead the Z operation, the controlled-Z qubit gate (CZ) is obtained.

$$U_{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{3.42}$$

The two qubit gates can be generalized to multiple control qubit gates, where an operation on the target qubit is applied only if all the control qubits are set to 1. In fact, if the total number of qubits is n, with $n-1$ control ones and one target, and the gate to apply to the latter is U, they are indicated by $\mathrm{C}^{n-1}\mathrm{U}$.
In general it is important to specify which qubit is the target and which the control ones, but for what concerns the case U=Z, $\mathrm{C}^{n-1}\mathrm{Z}$ acts simply inverting the sign of the component corresponding to the base state $|11...1\rangle$, and is therefore symmetric with respect to the n qubits.

The aim with quantum circuits, to be a valid alternative to classical ones, is to be able to reproduce any boolean function. It would be great to find a quantum gate which by itself could implement any desired circuit. In classical computation this task is carried out by NAND or NOR gates, which are called universal gates. In quantum computing there is no two-qubit gate that can solve this problem, but there is a way to obtain this, increasing the number of input qubits.
Consider a three-qubit gate called Toffoli gate (Figure 3.18), that acts one a three-qubit state space. If the vector basis are chosen to be $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$,..., $|111\rangle$, in this order, where the last qubit is the target and the first two the control qubits, the gate is

represented by the matrix:

$$U_T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.43}$$

It is immediate to see this is a unitary gate. Its action on the basis states is the following:

$$\begin{aligned} |000\rangle &\rightarrow |000\rangle \\ |001\rangle &\rightarrow |001\rangle \\ |010\rangle &\rightarrow |010\rangle \\ |011\rangle &\rightarrow |011\rangle \\ |100\rangle &\rightarrow |100\rangle \\ |101\rangle &\rightarrow |101\rangle \\ |110\rangle &\rightarrow |111\rangle \\ |111\rangle &\rightarrow |110\rangle \end{aligned}. \tag{3.44}$$

One can note it is equal to the one of the CNOT gate with the addition of a control qubit (U_T=C^{2}NOT).

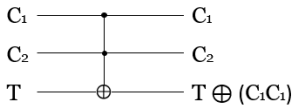

Figure 3.18: Toffoli, where C_1 and C_2 are the control qubits, T the target one. As for the CNOT gate, the action on any quantum state, included entangled ones, is a linear extension of the definition (3.44), which corresponds to the action of the matrix (3.43).

In the following it will be shown Toffoli gate constitutes a universal gate, which therefore can reproduce totally any classical boolean function.
Classical circuits are characterized by two main operations: *fanin* and *fanout*. *Fanin* is the classical operation in which wires are joined together in a logic gate, *fanout* is instead the splitting of a wire into more channels, producing therefore a certain number of copies. It is clear in general it's not directly possible in quantum circuits, because the first is an irreversible operation and the second is prohibited by the no-cloning theorem.

However multiple qubit gates allow to reproduce every classical irreversible gate (*fanin*) making it reversible, and allow also to obtain *fanout* (Figure 3.19 and Figure 3.20).

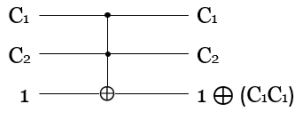

Figure 3.19: Toffoli gate: setting the target qubit to 1, one obtains as output an expression that corresponds exactly to the NAND gate. Since NAND is a universal gate, this means it can reproduce any *fanin* operation.

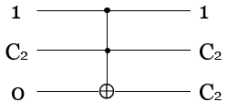

Figure 3.20: Toffoli gate: the first control qubit is set to 1 and the target to 0. In this way one obtains *fanout*. An important observation is that this does not violate the no-cloning theorem because the gate produces an entangled copy. In fact, if $\mathbf{C}_2$=$\alpha|0\rangle + \beta|1\rangle$, the Toffoli will act as: $\alpha|100\rangle + \beta|110\rangle \rightarrow \alpha|100\rangle + \beta|111\rangle = |1\rangle \otimes (\alpha|00\rangle + \beta|11\rangle)$.

For the implementation of *fanin* and *fanout*, one obtains that every classical circuit can be reproduced using only quantum unitary gates.
One can show ([4]) that Toffoli gate can be realized only using two-qubit CNOT and single qubit gates. In addition, CNOT and single qubit gates, besides being able to rebuild any possible classical circuit (which means they can completely reproduce boolean computation), one can show they can also reproduce any quantum computation ([12] for the demonstration).

Theorem 3.1 *Any logic quantum gate can be built from CNOT gates and single qubit gates.*

Other examples of multiple qubit gates used often in quantum computing are $\mathrm{H}^{\otimes N}$ and $\mathrm{X}^{\otimes N}$. They correspond to n single qubit gates applied to the n input qubits. Their matrices for the usual choice of basis states are the following block matrices:

$$H^{\otimes N} = \begin{bmatrix} \mathbf{H} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{H} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{H} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{bmatrix} \tag{3.45}$$

$$X^{\otimes N} = \begin{bmatrix} \mathbf{X} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{X} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{X} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{3.46}$$

Measurements and ancilla qubits

Different types of measurements can be applied, according to Postulate 3.4. The standard one is the projection-valued measure (PVM), but sometimes in order to extract the maximum information from the state another type is used, the positive-operator valued measure (POVM). In PVM measurements M_m operators are usually written as P_m and satisfy in addition to the requirements of the postulate:

- P_m are hermitian;
- P_m are positive semi-definite, which means that for every possible state $|\psi\rangle$ one has $\langle\psi|P_m|\psi\rangle \geq 0$;
- P_m are idempotent ($P_m^2 = P_m$)
- P_m are orthogonal ($P_m P_m = \delta_{mn}$).

P_j are therefore projection operators. Projectors on the basis states satisfy all the requirements listed above in addition to the ones of Postulate 3.4 and therefore constitute a PVM measure.
POVM measurements are instead a more generalized set of measures, which includes also the previous one. In fact, no more requirements are imposed to M_m, while a new operator is defined: $E_m = M_m^\dagger M_m$. This definition ensures E_m is hermitian and positive semi-definite. In fact $(M_m^\dagger M_m)^\dagger = M_m^\dagger M_m$ and $\langle\psi|M_m^\dagger M_m|\psi\rangle = (M_m|\psi\rangle)^\dagger M_m|\psi\rangle \geq 0$.
In the case of the projective-valued measure, from the itemized properties one has

$E_m = M_m^\dagger M_m = M_m = P_m$.
E_m are called POVM elements of the measurement. From Postulate 3.4, one has that the probability of the outcome m is given by $\langle\psi|E_m|\psi\rangle$. A possible way to implement one of this measures is also to choose directly the operators E_m (hermitian, semi-positive definite and such that $\sum_m E_m = \mathbb{I}$) and then rebuild $\{M_m\}$ setting $M_m \equiv \sqrt{E_m}$. For example, consider one has to distinguish a system that can live in two possible states: $|\psi_1\rangle = |0\rangle$ and $|\psi_2\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. As the two states are not orthogonal, there is not a projective-valued measurement that could determine with certainty the state of the system. However a POVM set of elements can be chosen in order to distinguish them sometimes, never making an error of mis-identification. Take three operators E_1, E_2, E_3, that corresponds to the three possible outcomes of the measurement $\lambda_1, \lambda_2, \lambda_3$, as the following:

- $E_1 \equiv \frac{\sqrt{2}}{1+\sqrt{2}}|1\rangle\langle 1|$
- $E_2 \equiv \frac{1}{(1+\sqrt{2})\sqrt{2}}(|0\rangle - |1\rangle)(\langle 0| - \langle 1|)$
- $E_3 \equiv \mathbb{I} - E_1 - E_2$.

It is easy to see they are hermitian, they satisfy $\sum_m E_m = \mathbb{I}$ and are positive semi-definite for $|\psi_1\rangle$ and $|\psi_2\rangle$. The first and the second operators are chosen to be respectively orthogonal to $|\psi_1\rangle$ and $|\psi_2\rangle$. In this way the probability of obtaining the outcome λ_1 if the state is $|\psi_1\rangle$ is given by $\langle\psi_1|E_1|\psi_1\rangle$=0, while the probability that λ_2 occurs if the state is $|\psi_2\rangle$ is given by $\langle\psi_2|E_2|\psi_2\rangle$=0. Therefore the occurring of λ_1 implies the system is in the state $|\psi_2\rangle$, and viceversa. No information is obtained if the third outcome is found.

It is possible to show that every general measurement can be reduced to a projective measurement through an auxiliary system, called the *ancilla* system, and a unitary operator. To this aim, take a set of measurement operators M_m acting on a state space M and build an *ancilla* system A with a basis $|m\rangle$ in one-to-one correspondence with the possible outcomes of the measurement. In this way a composite system is obtained. Take a state of the form $|\psi\rangle|0\rangle$, where $|\psi\rangle \in M$ and $|0\rangle \in A$, and define a unitary operator U in this subspace W of $M \otimes A$ that creates an entangled state:

$$U|\psi\rangle|0\rangle \equiv \sum_m (M_m|\psi\rangle)|m\rangle. \tag{3.47}$$

One can verify this is a unitary operator through

$$\langle\phi|\langle 0|U^\dagger U|\psi\rangle|0\rangle = \sum_{m,m'} \langle\phi|M_m^\dagger M_m|\psi\rangle\langle m|m'\rangle = \sum_m \langle\phi|M_m^\dagger M_m|\psi\rangle = \langle\phi|\psi\rangle. \tag{3.48}$$

As single and multiple qubits form finite dimensional state spaces, it is immediate to extend U, obtaining again a unitary operator, to the whole space $M \otimes A$. In fact, choose two orthonormal basis respectively in $W^\perp$ and $U(W)^\perp$ and build $U' : W^\perp \to U(W)^\perp$ such that it sends orthonormal basis states into orthonormal basis states. In this way the request of the conservation of the scalar product is satisfied. Then, build a PVM set of operators $P_m = \mathbb{I}_Q \otimes |m\rangle\langle m|$, and apply it to the state $|\psi\rangle|0\rangle$ after the unitary operator U. The probability that the outcome m occurs, from Postulate 3.4, is equal to

$$p(m) = \langle\psi|\langle 0|U^\dagger P_m U|\psi\rangle|0\rangle = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{3.49}$$

and after the measurement the state is found to be

$$\frac{P_m U|\psi\rangle|0\rangle}{\sqrt{\langle\psi|\langle 0|U^\dagger P_m U|\psi\rangle|0\rangle}} = \frac{M_m|\psi\rangle|m\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}. \tag{3.50}$$

In this way an equivalent set of measurement operators is found, and any general measurement can so be reduced to a projective-valued measure on an *ancilla* auxiliary system after the action of a unitary operator.

Bibliography

[1] In: Available on line. URL: https://qiskit.org/.

[2] In: Available on line. URL: https://quantum-computing.ibm.com/.

[3] A. Aboukarima, H. Elsoury, and M. Menyawi. "Artificial Neural Network Model for the Prediction of the Cotton Crop Leaf Area". In: *International Journal of Plant & Soil Science* 8 (Jan. 2015), pp. 1–13.

[4] D. P. DiVincenzo. "Quantum gates and circuits". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 261–276.

[5] V. Dorobantu. "The postulates of quantum mechanics". In: *arXiv preprint physics/0602145* (2006).

[6] M. Ghio et al. "Multipartite entanglement detection for hypergraph states". In: *Journal of Physics A: Mathematical and Theoretical* 51 (Mar. 2017).

[7] R. B. Griffiths. "Unitary Dynamics and Quantum Circuits". In: ().

[8] A. Kungl. "Robust learning algorithms for spiking and rate-based neural networks". PhD thesis. Jan. 2020.

[9] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[10] M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., 1967.

[11] M. L. Minsky and S. Papert. "Perceptrons". In: *Cambridge, MA: MIT Press* 6 (1969), pp. 318–362.

[12] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[13] A. Novikov and Z. Ramil. *Phase shift and multi-controlled Z-type gates*. May 2022.

[14] O. Rodríguez and J. Miguel. "Historical sociology of neural network research". In: 1991.

[15] R. Rojas. *Neural Networks - A Systematic Introduction*. Springer, 1996.

[16] F. Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[17] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[18] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[19] M. Rossi et al. "Quantum hypergraph states". In: *New Journal of Physics* 15.11 (2013), p. 113022.

[20] M. Schuld, Sinayskiy I., and Petruccione F. "The quest for a Quantum Neural Network". In: *Quantum Information Processing* 13.11 (Aug. 2014), pp. 2567–2586.

[21] F. Tacchino et al. "An artificial neuron implemented on an actual quantum processor". In: *npj Quantum Information* 5.1 (2019), p. 26.

[22] Colin P Williams, Scott H Clearwater, et al. *Explorations in quantum computing*. Springer, 1998.

[23] J. Zurada. *Introduction to artificial neural systems*. West Publishing Co., 1992.