Gradu Amaierako Lana/Trabajo Fin de Grado

Fisikako Gradua/Grado en Física

Quantum Machine Learning

Cristian Romero García

Director:
Dr. Mikel Sanz
Codirector:
Prof. Enrique Solano

Universidad del País Vasco Euskal Herriko Unibertsitatea

ZTF-FCT
Zientzia eta Teknologia Fakultatea
Facultad de Ciencia y Tecnología

Department of Physical Chemistry
Faculty of Science and Technology
University of the Basque Country UPV/EHU

Leioa, September 2016

Contents

Chapter 1

Introduction and Objectives

In the age of Internet, information management and pattern recognition in large databases are fundamental labours. It is a cornerstone in most of the social, artistic, scientific and business activities. We need to transmit, analyze, compute and modify this information among other things. In consequence, the design of new tools to efficiently deal with this problems has become essential.

Among these tools, those that automatize in some way the management of information are proving very useful, due to the effectiveness they show with a minimum human intervention. The fields of machine learning and applied artificial intelligence design and study these algorithms, which learn how to process information automatically. Additionally, based on higher computational power or the limitation shown by current computer architectures, new ideas are currently being explored. Quantum computing is one of the most promising alternatives, and is receiving lot of attention and investments from different governments, companies, and institutions.

Recently, the new field of quantum machine learning has emerged, with the goal of combining machine learning and quantum computing. The objective is finding quantum implementations of machine learning algorithms which have the expected power of quantum computers and the flexibility and learning capabilities of machine learning algorithms.

In this work, we explain the basic ideas behind quantum machine learning, reviewing first the fundamental concepts to understand it. In the Chapter 2, we will introduce classical machine learning with some definitions and classifications, proposing a internal structure for the machine learning algorithms which we think could help in the understanding of their essential characteristics. Also, in the sections 2.3, we will present a brief discussion about what, in our opinion, are important aspects in the physical imple-

mentation of machine learning algorithms.

In the Chapter 3, basic notions of quantum information and quantum computing are provided, including as an example Grover's quantum search algorithm. Finally, in the Chapter 4, quantum machine learning is introduced, giving the fundamental ideas of a field with only a couple of years of existence, and focusing on the example of quantum neural networks. In the end of the chapter, proposals for implementing machine learning algorithms in open quantum systems are discussed.

The objectives of this work are:

- Understand what is machine learning, capturing the ideas behind the running mode of these algorithms, and getting the different classifications.
- Realize about the fundamental notions of quantum computing and understand the power and limitations of quantum computing devices.
- Capture the fundamentals of quantum machine learning, as well as some current approaches and examples.
- Analyze the characteristics required in a physical system which implements a machine learning algorithm. In the quantum realm, discuss the possibility of implementing quantum machine learning algorithms in open quantum systems.

Chapter 2

Classical Machine Learning

Machine learning is one of the more active fields in computer science. Due to the accessibility of internet and the current digital devices such as laptops, smartphones, digital cameras, etc, a massive volume of new data is continuously produced. Therefore, the amount of data which needs to be managed every day is huge, and the possibility of doing it manually turned out to be unfeasible long time ago. The situation affects not only our every day life, but also to some fields of science as astrophysics, biology, particle physics or psychology, in which huge volumes of experimental and observational data are continually collected and need to be analyzed. In all these cases, new approaches are needed to manage and analyze these data in an efficient and automatic way. The answer to this relevant question seems to relay on machine learning and artificial intelligence.

In this Chapter, basic notions of machine learning are introduced, giving some definitions and analyzing the most relevant approaches. First, some fundamental concepts about algorithms are introduced, emphasizing their importance. Then, the idea of machine learning is explained, providing different classifications of the algorithms and giving some examples which we find helpful to understand it. Finally, some ideas are shown related with the physical implementation of machine learning algorithms in non-universal computing machines, revisiting concepts such as memory and non-Markovianity.

2.1 The Concept of Algorithm

When we want to solve a problem, we need a way of solving it. We need a collection of instructions, which is known as an algorithm, that will help us finding the solution. In this context, an algorithm is a sequence of operations that we have to follow to get the solution of the problem. Let us illustrate this idea with a simple example. Let us imagine we have a list of

numbers which we want to sort in growing order. This problem, commonly known as a sorting problem, is very usual in computer science and there are plenty of algorithms to solve it [1]. Without going into details, two of these algorithms are explained to illustrate better the concept.

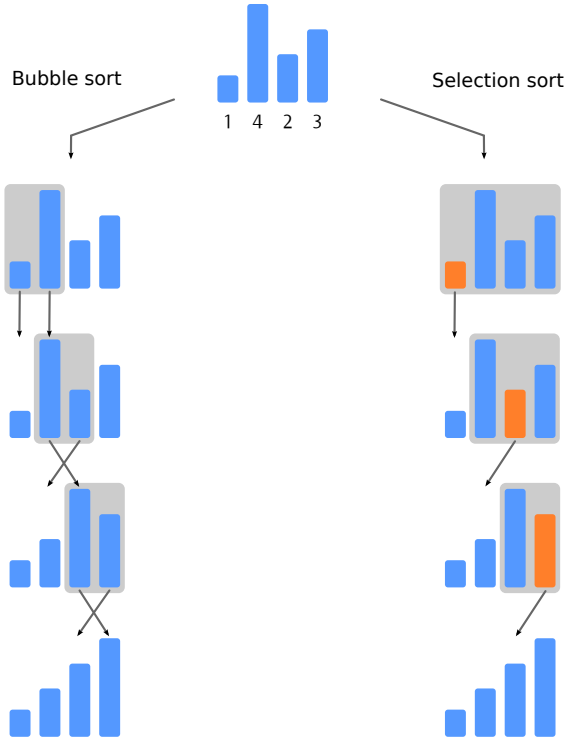

Figure 2.1: **Scheme of two different sorting algorithms.** This algorithms are known as Bubble sort and Selection sort and they solve the problem of sorting a list of numbers. In this example four numbers are ordered numerically. The first procedure compares adjacent numbers swapping those which are not in numerical order. The second algorithm finds the smallest number and moves it to the sorted section.

In the example of Fig 2.1, four integers have to be ordered numerically. This numbers are one, two, three and four, and are represented graphically with rectangles whose heights are proportional to the numbers. The problem is solved when each rectangle have a equal or higher one to his right.

The first algorithm, called *Bubble sort*, consists in comparing each pair of adjacent numbers of the list and swapping them if they are not in numerical order. This process is repeated until no more numbers are interchanged, which means that the list is in order. In the example of Fig 2.1, only one

run is needed to sort the numbers, but in general, if we have a list with N integers, the maximum number of steps is given by $\frac{N(N-1)}{2} \propto O(N^2)$.

The *Selection sort* algorithm works in a group of the total list of numbers, represented by a grey rectangle in the figure. Initially, the whole list is taken and the algorithm starts finding the smallest number. This is interchanged with the first of the group. As this number is already sorted, the size of the working group can be reduced, leaving out the ordered elements. The same process is repeated until the list is completely ordered. This algorithm shows a similar behavior, with a complexity given by $O(N^2)$.

When we find different algorithms to solve the same problem, as in this example, normally they need different resources. Some of them are faster, others need less physical memory, while others may give the correct solution only with certain probability. Finding or choosing an adequate algorithm is an important step in the resolution of a problem, since it determines how easy is going to be to solve the problem given some available resources.

Although the two procedures of the example are not the best, they are useful to understand the concept of algorithm. Note that we can use pencil and paper, an abacus, a digital computer, or any device as tools to solve a problem, but the algorithm is always essential, since this is the set of instructions which tell us how to use these tools. Therefore, we can say that it is the core of any problem solving process.

2.1.1 Elements of an Algorithm

The concept of algorithm and its definition have evolved during centuries, but it is not until the mid 20th century when a more formal approach is taken. Mathematicians like Kurt Gödel, Alan Turing and Alonzo Church analyzed the computing processes in a more theoretical manner, developing abstract models of computation which allow the analysis of algorithms independently of any physical implementation.

Figure 2.2 is the most basic diagram of an algorithm, reducing it to its essential elements. The algorithm is represented as an abstract entity which receives some input data, processes it, and gives an output. Any computational process can be represented in this way. Both input and output have to be included for being as general as possible. The input is the set of external data codifying the problem which we want to solve, and the output gives the answer or solution to the problems.

Another important aspect of this diagram is the internal state. Without it, we can think that algorithms are simply static mathematical functions

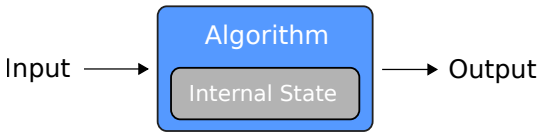

Figure 2.2: **Diagram of an algorithm.** Some of the representative parts of an algorithm are shown; the input, the output and the internal state.

that map inputs to outputs, but this turns out to be a very limited vision. When the concept of internal state is introduced, computation gets a more powerful meaning, since we can consider it a process in which the internal state adapts depending on the input, and gives different outputs for the same input due to the adaptability of the internal state.

In this sense, the computation is a evolution process of the internal state while solving a problem. In other words, we can consider an algorithm a dynamical entity interacting with the input data and giving an output which depends on it. It is noteworthy to mention that the existence of the internal state is independent of the input an the output, the same algorithm can thus give different outputs even receiving the same input. This is because the internal state when the input is given can be different and, in consequence, the evolution of the state and the corresponding output to.

2.2 Machine Learning Algorithms

Choosing an algorithm from a list like in the sorting example is not always possible. For most of the problems either we do not know how to create the algorithms, they are inaccurate, or they are really too resource consuming to being solved. For example, imagine that you want a computer program to identify a face in a picture, or the voice of a particular person. We are used to do these tasks everyday, but developing an algorithm to solve them in a computer is a hard problem, and they usually turn to be inefficient and inaccurate.

So what happens if we can not find an efficient or accurate algorithm to solve a specific problem? During years, the only option in this situation has been trying to find faster and more efficient algorithms. Nowadays, with the power of the computers and computing devices, these tools have become powerful enough to approach the developing and construction of algorithms in new ways. Machine learning is one of this approaches.

2.2.1 Definition and Explanation

Machine learning studies algorithms which learn and adapt from data [2, 3]. When it is said that an algorithm learns, it means that given the problem which we want to solve, the algorithm is able to modify himself based on some data to solve it more efficiently. These data are given to the algorithm as input, and it extracts from them the information which is relevant for solving the problem. The input data are commonly called experience and the information that the algorithm extracts in the process, knowledge, as shown in Fig. 2.3

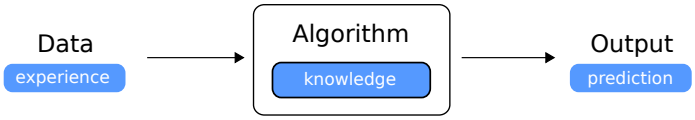

Figure 2.3: **Machine learning algorithm conceptual scheme.** This algorithms extract from the data the information needed to make future predictions. In some sense, the algorithm extracts knowledge from experience and uses it to make predictions.

We can say that machine learning algorithms extract, by learning from experience, the knowledge that is necessary to accomplish a task or solve some problem. This knowledge is used by the algorithm itself to give us an answer about unexperienced data. So there are two important concepts in these algorithms, the ability of generalizing from data by extracting the relevant information, and the use of this information to make predictions in other cases.

On a deeper analysis, machine learning algorithms can be understood as composed internally by two algorithms. We call them static and dynamic algorithms, and they are represented in Fig 2.4. The dynamic one receives the input data, processes it and gives an output in a similar way to the algorithm of Fig 2.2, but the difference now is that it can behave potentially in many different ways. The static algorithm is responsible of choosing this behavior depending of the problem which we want to solve. To achieve this, the static algorithm receives data from the dynamic one (input, output, internal state, etc.) and some extra learning data necessary to learn the problem.

In this analysis, we can say that learning consists in finding in the space of possible behaviors of the dynamic algorithm a behavior that executes a particular task better than the current behavior. The criterion used to evaluate how good a behavior for a specific task is, depends on the type of machine learning algorithm and on the learning data available.

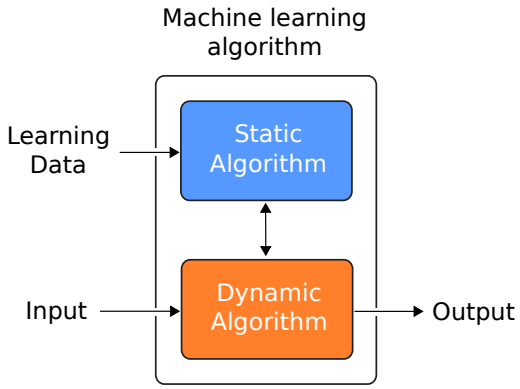

Figure 2.4: **Machine learning algorithm operation scheme.** Machine learning algorithms can be understood as internally composed of two algorithms. The dynamic algorithm have a wide range of possible behaviors and is the responsible of solving the desired problem, giving us the correct outputs for the inputs. The static algorithm fixes the behavior of the dynamic one based on the experienced data and some extra learning data.

When designing machine learning algorithms, there are two important things. On the one hand, the dynamic algorithm needs a sufficiently large space of possible behaviors, since this allows it to be more flexible and adapt easier to the problem. On the other hand, the static algorithm has to be designed to efficiently find an optimum behavior of the dynamic algorithm to solve a particular task given some learning data.

2.2.2 Classification of Machine Learning Algorithms

Although machine learning algorithms share some characteristics such as the ability of learning, it is a very wide discipline, and these algorithms can be classified in to groups based on different criteria. In general, this classification is not exclusive and the same algorithm can be included in more than one group at the same time. Even so, making this categories helps to capture the essential ideas of different algorithms and it allows us to explore better what machine learning is about.

The most common criterion employed in these classifications is based on the learning method, grouping algorithms depending on the type of learning data they need (see Fig 2.5). It consists of mainly three different groups:

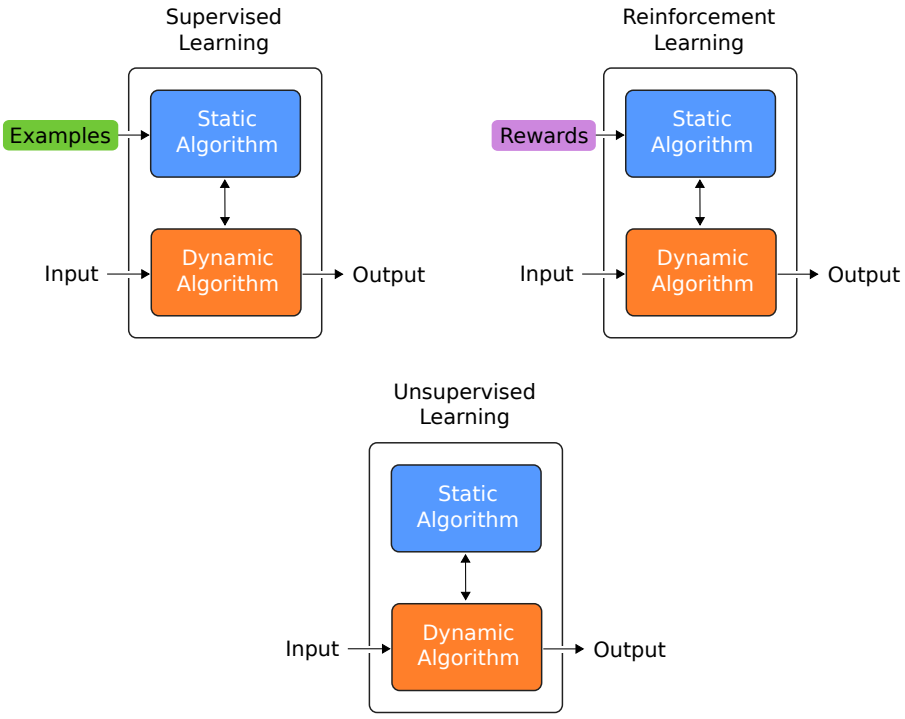

Figure 2.5: **Classification of machine learning algorithms.** Three different kinds of algorithm are shown, depending of the type of learning data they receive. In supervised learning, examples of some inputs and their desired outputs are received. Reinforcement learning works with rewards which quantifies how well is being the problem solved. Unsupervised learning does not receive any especial learning data.

Supervised Learning

Supervised learning algorithms use learning data to infer a mapping between inputs and outputs, and that mapping is used to make predictions on new and unlearned data. The training data is a set of examples, where each example consist in a input and his desired output. In this sense, it is similar to a problem of function fitting, although in the current case our purpose is not only finding a function but also using it for the predictions.

To make correct predictions in unseen data, the algorithm have to extracts the relevant information to attain this, rater than simply finding a behavior that maps correctly the learning inputs to outputs, but fails in the unseen data. In some way, the algorithm have to learn from the data rater than learning the data.

Analyzing a supervised algorithm with our model of two internal algorithms, we can say that to learn a correct mapping, the static algorithm has to be able to chose a behavior for the dynamic algorithm which reproduces in an acceptable way the input to output relation given by the learning data.

These algorithms can be adapted to solve a wide range of tasks, but they are commonly used in classification problems, due to their flexibility when different classes have to be learned. An example of one of this problems is depicted in Fig 2.6. We have a big data base of unlabeled images, all of them are either an image of an apple or an image of a banana. We need a procedure to automatically classify the pictures in these two groups. Thus, we are going to use a supervised learning algorithm.

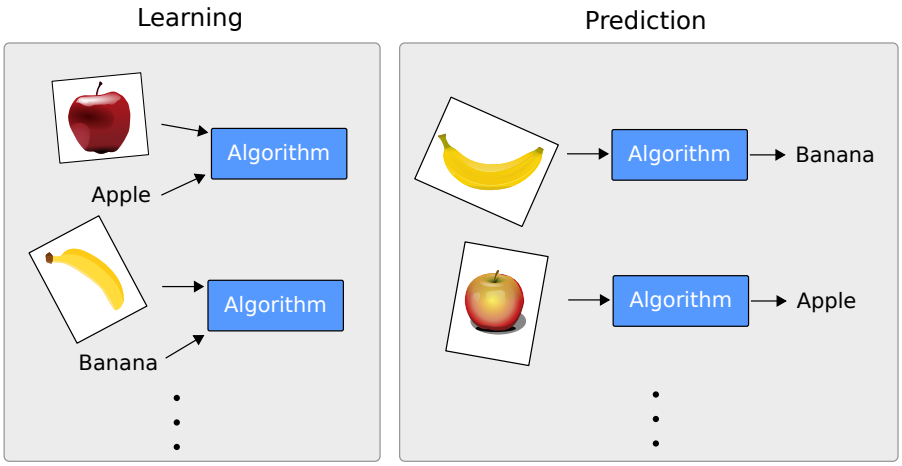

Figure 2.6: **Supervised learning algorithm example.** In the learning stage the algorithm receives images along with their corresponding label, depending on whether they are apples or bananas. In the prediction, unlearned images are given to the algorithm and it have to be able of classifying them automatically depending if a apple or a banana appears in the image.

First we train the algorithm by giving some of our images as input data and pointing out in the learning data the class to which they belong, this is, if they are apples or bananas. The images are represented by pixels, and this are going to be the input. In this learning stage, the algorithm learns to map the pixels of the input to the fruit they represent. When the algorithm has been trained with a reasonable quantity of images, we can see wether its predictions on the unlearned images are good. In this case, the algorithm can be used to classify the rest of images. If there are errors in the classification during the performance test, we can continue training the algorithm until the predictions are correct.

In this example, it is important to use images in the training which are representative of each of the groups, since the algorithm is going to learn faster the relevant features to classify fruits correctly.

Reinforcement Learning

These algorithms receive rewards as learning data. The reward is a quantity that tells to the algorithm how well is the task being solved. The goal of these algorithms is finding and using the strategy which gives the maximum long term reward. To get it, they have less information than the supervised learning algorithms, since now the algorithm does not receive the correct output for certain inputs.

So we can say that the approach in reinforcement learning consist in using the static algorithm to explore the space of possible behaviors of the dynamic algorithm and find those that increase the received long term reward.

Due to the nature of the learning by rewards, in these algorithms the learning and prediction are made in parallel. The algorithm have to take the correct output for certain input, and for this prediction is needed. Also, due to these outputs, new rewards are received, allowing the algorithm to learn. These two actions, prediction and learning, have to be made simultaneously to reach to a optimum behavior. So the reinforcement learning algorithms are online learning algorithms to, as is explained later.

Another aspect that differentiates these algorithms is the way in which the learning data is received. In the supervised algorithms, the input and his desired output are received at the same time. In the reinforcement algorithms, instead, the rewards can be received in any moment after the algorithm starts working. This means an extra challenge for the algorithm, because it have to conclude which parts of his behavior have been responsible of the rewards. In this way it can search for new behaviors that conserve these desirable parts and improve others.

An example of a situation where this type of algorithm might result practical is in the case of an algorithm which learns to play chess, represented in Fig. 2.7. The input in this situation is the current state of the chessboard, with the information of the position of all the pieces. The output is a prediction of the best move given the current state of the board. The way that the algorithm have of learning is through the rewards. The algorithm plays some matches against a player, and in each of the plays that the algorithm wins, it receives a positive reward. When it loses, instead, the reward is

negative.

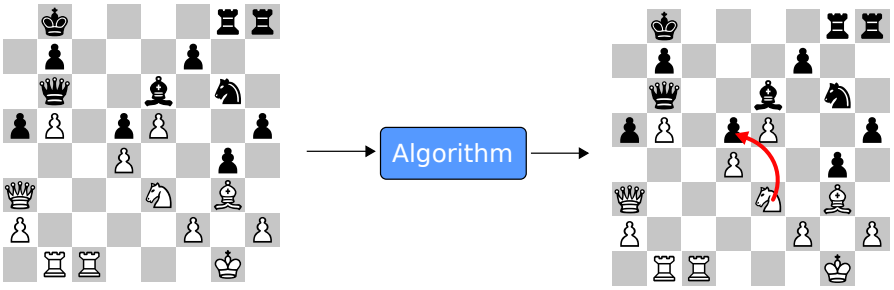

Figure 2.7: **Reinforcement learning example.** The algorithm receives the state of the chessboard and it makes a decision of his next move. The algorithm choses a strategy with the aim of maximizing his number of wins.

In this way, the algorithm should adapt itself in order to maximize the long-term reward. The behavior changes progressively, exploring different strategies which help to increase the received rewards. An important aspect of this example is that there is not always a correct behavior, since it depends on the player against whom you play. Even more, a particular player can change his strategy between games, which means that the optimal behavior also changes. A good reinforcement learning algorithm has to react fast to these situations.

Unsupervised Learning

Unsupervised algorithms do not receive any learning data. In this type of learning, patterns have to be found in the data, extracting the information about what is more often repeated. In some way, the behavior of the dynamic algorithm has to be selected with the aim of extracting the most characteristic features of the input. This is done by the static algorithm, as in the previous algorithms, but this time there is not learning data, so the criteria for selecting the best behavior must be codified inside the static algorithm.

Most of the criteria used are based on classifying the data in groups following some concept of similarity. These algorithms are named clustering algorithms an they can learn how to extract hidden information in the data, like categories and characteristics.

To see the difference between this and the rest of the mentioned algorithms, we can use it in the example of the supervised learning algorithm.

However, now there is not any example of classes to learn from. We only have all the images of apples and bananas. The unsupervised algorithm receives these inputs and have to be able to extract common characteristics of these pictures.

If the criteria used in the static algorithm is some kind of clustering, the algorithm is going to infer that there are two differentiated types of pictures, those in which appears an apple and those of a banana. It also can happen that the algorithm finds another possible classifications, clustering the pictures based on form, color, etc. This property makes unsupervised learning useful when hidden classes have to be discovered in big databases.

Other classification of the machine learning algorithms is based on the way the learning data is received, and gathers the algorithms in two different types:

Bach Learning

In the Bach learning algorithms, also known as offline learning algorithms, there are two differentiated stages in the running process of the algorithm. First, in the commonly called learning stage, the algorithm learns how to solve a particular task by employing the learning data. Then, in the prediction stage, the algorithm forecast the outcome for new data using the gained knowledge.

In the learning stage, the static algorithm fixes the behavior of the dynamic algorithm. In the prediction stage, instead, the dynamic algorithm make predictions using this behavior. Thus, when the learning have concluded, no more changes are made in the behavior of the dynamic algorithm. In this sense, it is like building a model that describes the input to output relation and then using this model to make predictions.

Online Learning

Unlike the bach learning, in the online learning algorithms, the learning and prediction stages are not well separated. These two phases are executed simultaneously. At the same time that the algorithm is making predictions, its behavior changes due to its progressive learning.

These algorithms are useful when there is not possibility of having the learning data from the beginning. In some situations the data comes sequentially, so the algorithm have to learn progressively at the same time that it makes predictions. Something similar happens when the received data depends on the previous predictions of the algorithm. In this situation, the

algorithm has to make a prediction by using the behavior previously learned, and in this way the learning process follows with the new data obtained.

If the criterion used in the classification of machine learning algorithms is the nature of the output data, other two groups can be differentiated:

Classification algorithms

Classification algorithms learn to classify the input data in a reduced set of classes. In this case, the output indicates to which of these classes belongs the current input. A classic example of this type of algorithms is the mail spam detector, in which the algorithm has to classify the input mails in to two categories, depending whether they are considered spam or not.

Regression algorithms

These algorithms are the machine learning equivalent of the regression analysis procedures, in which a function approximating the relation between some given inputs and outputs is estimated. In the machine learning case, the algorithm infers this function based on the data. Then, this function is used to predict the outcome for other inputs.

Since there are infinite numbers between any two real numbers, we need a infinite number of tags to identify any real quantity in a real interval. So real quantities can not be saved in a finite memory, and in consequence, in a computer. Real numbers have to be approximated.

With this last idea in mind, we can say that regression in machine learning is a classification process, since the algorithm is classifying the input into a very big group. Therefore, we can say that the main difference between classification and regression is due to the quantity of classes in which the inputs are classified.

2.2.3 Relevance and Applications

As already said, machine learning is a different approach to problem solving, and it turns out to be very useful in situations where big data quantities are available and need to be analyzed or classified. It is also practical when some flexibility is needed, since the same algorithm can learn how to solve different problems.

With the procedures of machine learning, problems that once were impossible, today can be solved with satisfactory results. This is why many different applications have emerged. It is used in different areas such us pattern recognition, voice recognition, computer vision, marketing, economics,

medical diagnosis, personalized recommendation systems, search engines, DNA sequences classification, etc. As they are so general and flexible, machine learning algorithms can be used in all these different and apparently unrelated areas.

Many researchers are working in the machine learning field to, find new algorithms which perform well in more situations, have a wider range of possible behaviors, need less resources for learning, and generalize better from experience.

One of the most active areas of research is known as deep machine learning or deep learning [2]. The idea of this field is to reach a higher level of abstraction than with normal machine learning algorithms. As previously explained, the learning process can be seen as an optimization problem to find the behavior which performs effectively in a particular task. As in most optimization problems, when the dimension of the space in which the problem has to be optimized grows, the computational power required must grow very fast.

This is the reason why only machine learning algorithms with a reduced behavior space have been used until few years ago, requiring a pre-processing stage in the data to reduce their dimension, and to extract only the features which are considered essential for the problem. Today, having more powerful computers, this first feature extraction stage is not so necessary and can be achieved directly by the machine learning algorithm automatically. Deep learning studies these algorithms.

2.2.4 Memory and non-Markovianity in Machine Learning

An interesting question which may arise when studying these algorithms is about the necessity of memory in a learning process. In the context of machine learning, learning refers to the ability of an algorithm of changing his behavior to perform better a task. To accomplish this, the external data is used, extracting from it the necessary knowledge and using it to make a forecast.

We can say that in a machine learning algorithm, predictions depends on the data received in the past. In this sense, those algorithms have memory, since their current behavior depends on the history of inputs. This memory emerges in the machine learning algorithm in its static algorithm, since it is the responsible of fixing the behavior of the dynamic algorithm depending on the data previously received.

Depending on the problem which the algorithm is solving, the time

length required for the memory can be very different. In some problems, a short-term memory is enough, since the knowledge is only used short after the learning. However, this is sometimes not sufficient, and a longer-term memory is needed.

To understand better the role of memory in the physical implementation of a machine learning algorithm, let us explain the concepts of Markovian and non-Markovian dynamics.

Let us start by introducing the concept of stochastic processes. These processes describe the system dynamics which evolves in a non-deterministic or stochastic way, which means that even knowing the current or past states of the system, the future state is random. However, we can study the evolution of the probability distribution of finding the system in a given state. Modeling these processes mathematically gives a useful tool to study the evolution of different real systems, such as Brownian motion, signals of audio and video, stock market, population processes, etc. It is important to note that deterministic processes can be thought as being stochastic, but without randomness.

Markov processes are a kind of stochastic processes, and receive this name because they satisfy the Markov property. A process shows Markovianity if the future probability distribution of the system can be determined just by knowing the current state and it is independent of the previous history of states. So the conditional probability function of the future state to the history of states depends only on the current state. It is commonly said that Markovian processes are "memoryless", since they do not remember their past.

Non-Markovian processes, as the name suggests, are processes that do not satisfy the Markov property. In the same way in which Markovian processes are called memoryless, the non-Markovian ones have properties that take us to think in them as processes with memory. This is because non-Markovianity implies that the evolution depends not only in the current state but in the history of states, so the behavior of the process depends of his past and in this sense it have memory.

In the classical case, due to the availability of powerful digital computers with universal computing capabilities, this discussion is unnecessary, because the memory can be simulated. In the quantum realm, instead, this universal computers do not exist yet, so problem specific quantum simulators are the only option. In this situation, and due to the presence of memory in a non-Markovian evolution, it seams a natural procedure implementing machine learning algorithms in physical quantum systems whose time evolution is

non-Markovian. This is going to be analyzed in Chapter 4.

2.3 Physical Implementation of Algorithms

While an abstract or schematic approach to the analysis of algorithms is very productive from a theoretical point of view, it does not help us in implementing these algorithms physically. More detailed explanations of the internal working mode of the algorithms can be given instead of simply analyzing their most essential characteristics as we have done, but even so, an exact implementation can not be concluded from there.

If we want to build a physical implementation of a particular algorithm, we need to translate the concepts that we have analyzed schematically into a more physical realm. As aforementioned in the previous section, this is not so relevant in the case of classical algorithms due to the existence of powerful universal classical computers. However, it will turn out to be key in the discussion of Chapter 4, since we are far from a universal quantum computer.

Looking at the diagram depicted in Fig 2.2, there is a translation of the used elements of the algorithm that seams to be natural to work with in to a physical framework, and it emerges if the algorithm is considered as a physical system. After this, the rest of the elements find a acceptable physical meaning. The input and the output can be seen as interactions between the system and his environment, and the internal state of the algorithm meets its analogous in the physical state of the system, which is represented by its degrees of freedom (Fig 2.8).

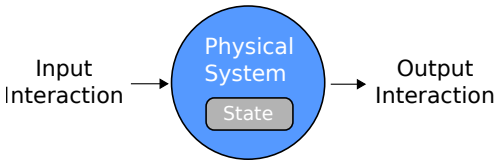

Figure 2.8: **Physical algorithm implementation scheme.** A basic translation of an algorithm to a physical system is shown. The internal state of the algorithm is represented in the physical state and the input-output are considered interactions between the system and the environment.

In this context, the computation performed by the algorithm is the dynamic evolution of this physical state. As the system interacts with the environment, this affects his evolution and, in this way, modifies the result. When the input and the output are treated as interactions, the input is the interaction by which the environment modifies the system, and the output,

on the contrary, is where the system is modifying the environment.

The distinction between input and output is subtle and not always clear in a physical context. In the framework of classical physics, this distinction can be done correctly, because the input is the interaction modifying the evolution of the system and the output is simply a measurement in this system. Classically this measurement does not modify the state of the system, something that is expected if we are considering it as an output. In the quantum realm, however, the measurements modifies the system, so it is not completely correct considering it only like an output and the situation is more complex.

In the design of a physical implementation of an algorithm there are some important issues. The physical system must be adequately chosen so that the abstract algorithm may be implemented and the information codified and retrieved straightforwardly. Also, the dynamics of the system have to be accurate representing the state transitions which are performed in the algorithm that we want to implement.

2.3.1 Physical Implementation of ML Algorithms

When an algorithm is implemented in a physical system, the problems are solved thanks to the evolution of the physical state of this system. In machine learning, the system has to be able to modify its behavior based on the data, which is equivalent to saying that it have to be able to modify its dynamics based on interactions with the environment. To explain this in a more clear way, we are going to separate this physical system in to two parts (see Fig 2.9) in a similar manner to what we have done with the machine learning algorithm in the Chapter 3.

These two parts gather two different degrees of freedom of the total system. We have called them fast and slow systems, for reasons which are explained below. The fast system is interacting with is environment in the input and the output, and is the part that make predictions. The slow system is interacting with the fast one, and it is responsible for fixing its dynamics. To perform this, it receives information through his interaction with the fast system.

It has been explained earlier that any learning algorithm shows some kind of memory. So if we want to physically implement a machine learning algorithm, we need a material way of having this memory. In the last description this is achieved with the separation done in the degrees of freedom. The fast and slow degrees of freedom of the total system are grouped in the fast and slow systems, respectively.

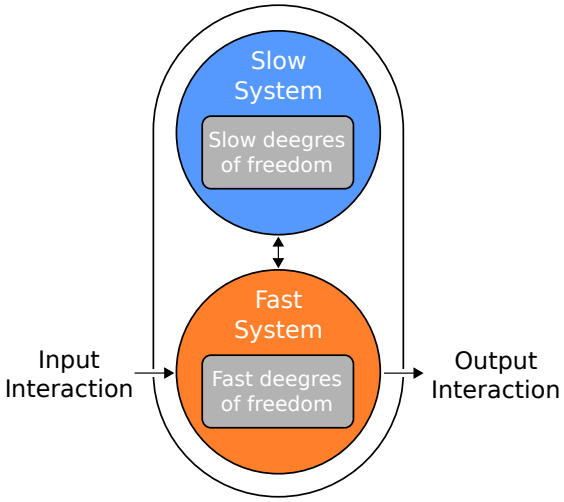

Figure 2.9: **Scheme of the physical implementation of Machine Learning algorithms.** We can separate the internal degrees of freedom of the physical system in two systems, depending if they are fast or slow. The fast degrees of freedom make the predictions and the slow ones retain the information needed to perform these predictions. Because these systems are interacting, they influence each other. The slow degrees of freedom affect the behavior of the fast ones, fixing their dynamics. In the other hand, the fast degrees of freedom affect the slow degrees permitting them to correct the information that they contain. In this mutual interaction is where emerges the ability of learning.

As the slow system contains the slow degrees of freedom, it can retain in the dynamics the information of the past data, and by means of the interaction with the fast degrees of freedom it can modify their dynamic. Therefore, the slow system is responsible of keeping the memory.

To be practical, the fast system have to be able of adopting a wide range of possible behaviors, which means that his corresponding degrees of freedom can show a wide range of different possible dynamics, in a similar way to what we have described earlier for the dynamic algorithm. Achieving this in the quantum realm, as is explained in Chapter 4, is an important challenge, since quantum mechanics are extremely linear, what difficult the design of rich and complex dynamics.

Chapter 3

Quantum Information and Quantum Computing

3.1 A Glimpse of Quantum Information

As in classical information theory, where the elemental concept is the bit, in quantum information there is a quantum equivalent called *qubit*. There, the Boolean states 0 and 1 are represented by a pair of normalized and orthogonal quantum states labeled as $|0\rangle$ and $|1\rangle$, respectively.

These states form a basis in which any qubit state can by represented as a superposition $\alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ satisfy the normalization condition. The possibility of being in a superposition brings to the qubit a continuous of possible states, compared with a classical bit. This difference open a new paradigm in information processing.

Another quantum property which also seems to be useful in quantum information is related with the fourth postulate of quantum mechanics. This says that the state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.

This postulate shows how to describe a quantum system consisting of a number of distinct physical systems, constructing a new Hilbert space from the spaces of each of the original systems. An astounding concept related to this postulate is the quantum entanglement. This phenomenon has received a lot of attention since the beginnings of quantum mechanics (*EPR paradox*) and nowadays continues being an active area of research. Let us try to resume the basic idea in the following example [4].

Let us consider a quantum system, consisting of two qubits. The most general state of one of these qubits is the superposition

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle , \tag{3.1}$$

were $|0\rangle$ and $|1\rangle$ are the two base vectors and $c_0, c_1 \in \mathbb{C}$. These two complex coefficients fix the quantum state of the qubit. Based on the fourth postulate, if we combine the two qubits an analyze the system as a whole, a possible state of the composed system can be

$$|\varphi\rangle = \frac{|0\rangle |0\rangle + |1\rangle |1\rangle}{\sqrt{2}} . \tag{3.2}$$

The notation $|u\rangle |v\rangle$ is a simplification of $|u\rangle \otimes |v\rangle$. This state is called a Bell state and shows the interesting property that there are no single qubit states $|a\rangle$ and $|b\rangle$ such that $|\varphi\rangle = |a\rangle |b\rangle$, a fact which is easily shown. We are going to suppose that this is possible, so

$$\begin{aligned} |\varphi\rangle &= |a\rangle |b\rangle \\ &= (\ a_0 |0\rangle + a_1 |1\rangle\ )(\ b_0 |0\rangle + b_1 |1\rangle\ ) \\ &= a_0 b_0 |0\rangle |0\rangle + a_1 b_1 |1\rangle |1\rangle + a_0 b_1 |0\rangle |1\rangle + a_1 b_0 |1\rangle |0\rangle\ . \end{aligned} \tag{3.3}$$

To obtain the Bell state, the coefficients have to be chosen to cancel the last two terms of the sum conserving the rest. However, this is clearly impossible, since this removes at least one of the desired terms. In this way, we have proved that the Bell state can not be obtained as the tensor product of the individual qubits states.

This implies that when these qubits are separated a large distance, they continue connected, even without the existence of any interaction between them. In this situation, it is said that the qubits are entangled, because it is impossible to separate the representation of the composite system into the qubits states, they are interconnected, and the measurements performed on one of the qubits affects the measurements made on the other. This property has important implications in quantum information theory.

3.2 Quantum Computing

The mechanical calculators of Leibniz, Pascal and Schickard, the mechanical computers of Charles Babbage, and also the first electric computers like the Z3 and the ENIAC, all of them where supported and explained by classical physics.

This changed with the appearance of the transistor and the semiconductor devices, which helped taking one more step in the process of miniaturization of the computing machines. These electronic elements were invented

thanks to the quantum theory, and they can not be understood without it. Since then, with the aim of increasing the performance and reducing the size of the computing devices, a lot of improvements have been made, and the quantum mechanics have been essential in all of these steps.

Currently, due to physical limitations in this miniaturization process and the necessity of more computational power, the paradigm is changing. Rather than using the quantum physics only in the design of the computing devices, as we have done until now, the idea is to codify also the information in quantum states, using the physical quantum properties to perform the computation.

One of the first ideas in this area come by Richard Feynman [5], who in 1982 observed that it seems impossible in general to simulate efficiently the time evolution of a quantum system in a classical computer. To completely describe a quantum system, the number of variables needed increases exponentially in relation with the number of particles. The reason relies on the exponential growing of the dimension of the Hilbert space of a composite system, which is the multiplication of the dimensions of his subsystems. So if we have N qubits, the Hilbert space that emerges in the interaction of these N qubits have a dimension of 2^N, which means that this is the number of variables needed to characterize the system.

Due to the size of the resulting Hilbert spaces, and consequently, to the huge computational power needed to update them, it is inefficient to simulate in general quantum systems in classical computers, even if the simulated system is composed of a reasonably small number of particles. In this situation, Feynman noted that a quantum computer can be used to simulate quantum systems efficiently, because all the things which make the computation hard classically, are natural in a quantum computer.

Around this idea, researchers suggested that maybe there are other problems which quantum computers can solve efficiently, apart from the quantum simulations. If in general a quantum system can not be efficiently simulated classically, it is very probable that some of the problems that this quantum system is able to solve are also inefficient to be solved classically. Following these ideas, quantum computing started as a new research field.

Quantum computing studies the computers and universal computational models which make use of quantum mechanic phenomena. It is about the implementation of algorithms that sustain their work in quantum properties. The theory of quantum mechanics have some aspects that are very different from their classical counterpart, and sometimes there is no classical counterpart at all, as in the cases of quantum entanglement and superposition.

These properties open new ways in the implementation of algorithms and computing devices. Additionally, they can help to understand better some aspects of the quantum mechanics which are still not well understood.

Nowadays, this field is still in his infancy and it has a long way ahead, but even so, there are reasons to believe that the computational power of these devices is larger than this of the conventional ones. Some advances have been made finding quantum algorithms which perform faster than the classical equivalents, such as the Shor's algorithm for factorizing numbers or the Grover's one for searching in an unsorted list. Successful technologies have emerged to implement these algorithms, such as trapped ions, superconducting circuits, nuclear magnetic resonance systems, photonics, etc.

3.2.1 Grover's Algorithm

Grover's algorithm is a quantum search algorithm [6]. The purpose is finding in a group of elements those that satisfy a particular condition. To do so, we have a black box, commonly called an oracle, with the ability of recognizing those elements that satisfy the condition we are finding for. If the size of the group is of N elements, classically we need to call the oracle $O(N)$ times to find all the elements that satisfy the condition, as we have seen in Section 2.1. Surprisingly, thanks to the properties of the quantum mechanics, Grover's algorithm is able to achieve the same result in $O(\sqrt{N})$ calls to the oracle. In some way, this algorithm makes different calls to the oracle at the same time, exploiting the parallel processing capabilities of quantum computing.

In this section, for the sake of clarity, a particular search problem is explained, specifically the case in which only one element satisfies the search condition. However, the procedure can be extended to more general situations.

Let us consider a search list consisting of N elements. In order to represent them, Grover's algorithm uses a Hilbert space of dimension N, which can be obtained with $n = log_2 N$ qubits. Each element of the list with index x is represented by a orthonormal base vector $|x\rangle$ in the state space of the qubits. The goal is identifying the index ω, whose element satisfies the search condition. To this end, the oracle is used, which is given by a unitary operation which has the following properties

$$\begin{cases} U_\omega |x\rangle = -|x\rangle , & \text{if } x = \omega \\ U_\omega |x\rangle = |x\rangle , & \text{if } x \neq \omega , \end{cases} \tag{3.4}$$

which can be equivalently expressed in a more compact manner by using the

identity operator I and a projector into the $|\omega\rangle$ ket, in the next expression:

$$U_\omega = I - 2\,|\omega\rangle\,\langle\omega| \ . \tag{3.5}$$

The algorithm consists in using the last oracle operator together with the Grover diffusion operators, defined as

$$U_s = 2\,|s\rangle\,\langle s| - I \ , \tag{3.6}$$

where $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle$. These two are applied as follows. First the state of the qubits is initialized to the $|s\rangle$ state. Then, the operators U_ω and U_s are applied in this order iteratively $r(N)$ times, whose expression is calculated later. Finally, after that, the system is measured, which is going to give the eigenvalue λ_ω with a high probability, from which the index ω can be concluded.

To describe better this procedure, let us analyze it in the subspace spanned by the vectors $|\omega\rangle$ and $|s'\rangle = \frac{1}{\sqrt{N-1}} \sum_{x\neq\omega} |x\rangle$, which represent the desired solution and a superposition of all the states which are not solutions, respectively. In this subspace, the initial state $|s\rangle$ is decomposed as

$$|s\rangle = \sqrt{\frac{1}{N}}\,|\omega\rangle + \sqrt{\frac{N-1}{N}}\,|s'\rangle \ , \tag{3.7}$$

and the operators U_ω and U_s are simple reflections in the two-dimensional plane about vectors $|s'\rangle$ and $|s\rangle$, respectively. So, when the two operators are applied one after the other, due to being reflections about different directions, the resultant operation is a rotation. It is more clearly depicted in the graphical representation in Fig 3.1, where the first Grover iteration step is shown.

To obtain the angle of rotation θ, we can calculate the angle between vectors $|s\rangle$ and $|s'\rangle$, knowing that it is half the rotation angle. From Eq. 3.7, it is concluded that

$$\sin\frac{\theta}{2} = \frac{1}{\sqrt{N}} \ , \tag{3.8}$$

from where the angle θ can be extracted in function of N.

The idea is repeating this iteration r times until our state vector is as close as possible to the solution $|\omega\rangle$. Therefore, r has to satisfy the following expression

$$r\theta + \frac{\theta}{2} \simeq \frac{\pi}{2} \ . \tag{3.9}$$

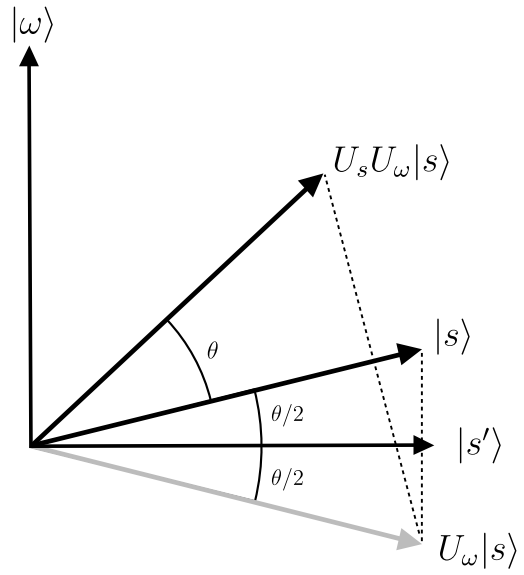

Figure 3.1: **Grover's iteration scheme.** The unitary operators U_w and U_s are applied consecutively to the state $|s\rangle$. Being both operators reflections about different directions, the resulting transformation is a rotation of angle θ.

When $N \gg 1$, Eq. 3.8 reduces to $\theta \simeq \frac{2}{\sqrt{N}}$, and the iteration number r takes the form:

$$r \simeq \frac{\pi}{4}\sqrt{N} \ . \tag{3.10}$$

Therefore, the number of iterations needed to find the element ω with the required precision is $O(\sqrt{N})$. Although the result of the measurement is probabilistic, when N is large, the angle of rotation of each iteration is small, allowing us to approach the state very close to the solution and ensure the correct result with a high probability.

3.2.2 Types of Quantum Computing

All the quantum computing devices share the use of quantum phenomena in their operation, but we can roughly distinguish two different types depending of their purpose. Those are quantum simulators and quantum computers.

Quantum Simulator

Quantum simulators are based on Feynman's ideas, whose aim is simulating a particular quantum model. These simulations result very useful in situations when it is difficult to measure directly the system or when extra

control is needed. These tools would be helpful to advance in research areas such as condensed-matter physics, high-energy physics, atomic physics, quantum chemistry, cosmology, etc. Due to the necessity of better understanding complex quantum systems, simulations are essential in these areas.

In quantum simulators, the state of the simulated system is mapped into the simulating system state, which belongs to a Hilbert space of the same dimension. To simulate the evolution of this system, the Hamiltonian acting on it has to be correctly engineered to evolve the state in a similar way to the simulated one. Finally, measurements are performed in the system, obtaining the desired data. Due to the probabilistic nature of the measurements, in general the process has to be repeated a number of times to extract statistics.

Quantum Computer

Quantum computers are designed as universal devices with the aim of solving more general problems beyond simulations. As aforementioned, quantum systems show properties which can be exploited in a variety of calculations.

The power of quantum computers relies on the superposition of quantum states, which allows the algorithms to run in superpositions of inputs, calculating the results for many different inputs at the same time. However, due to the collapse of the wave function after a measurement is performed, only one of these results can be obtained. To avoid this problem, algorithms have to be intelligently designed to give in the output the desired result with a high probability. Additionally, to exploit this parallelism, the output of the algorithm has to depend on all the results calculated in parallel.

3.2.3 Analog and Digital Quantum Computers

Quantum computing devices can be analog or digital depending on the method used to achieve the desired time evolution for the quantum state.

In the analog procedure, a quantum system is chosen in a way which allows us to solve a particular problem mapping the variables of the simulated system into the simulating system, and using his evolution to solve the problem. These implementations are problem dependent, since for each problem a quantum system which permits a correct mapping and a high level of control must be found. Therefore, the disadvantage of analog computing is that, as these devices are usually single purpose, finding a quantum system for each problem can be a difficult task. Although, analog computing devices can be very compact and efficient in resources, since they only have the essential parts to solve the problem.

The digital approach follows a modular design, having the quantum logic gates as the unit to construct the devices. These gates are similar to the classical logic gates, but they are applied on quantum states, which means that the operation is applied simultaneously to each element of the quantum superposition. These gates perform unitary transformations in the state of the system, and some types can be differentiated depending on the form of the unitary transformation that they perform. To achieve more complex computations, these quantum gates are added together, using the output of some of them as input of others.

This procedure of combining logic gates allows us to approximate any unitary transformation with few type of gates. Indeed, depending on the problem, the quantity of quantum gates required can be very large, resulting in inefficient implementations due to the resources needed.

Chapter 4

Quantum Machine Learning

Quantum machine learning aims to implement machine learning algorithms in quantum systems [7], by using the quantum properties such as superposition and entanglement to solve these problems efficiently.

The field of classical machine learning is receiving lot of attention and investments from the industry. Nowadays, due to the huge quantities of data with which we deal every day, new approaches are needed to automatically manage, organize and classify these data. Classical machine learning, which is a flexible and adaptable procedure, can recognize patterns efficiently, but some of these problems can not be efficiently solved by these algorithms. The companies whose labour consists in big databases management are aware of these limitations, and are very interested in new approaches to accomplish this. They have found in quantum machine learning one of these approaches.

Nevertheless, there is not a general theory to analyze and engineer new quantum machine learning algorithms, and there are additionally some unanswered related questions. One of the problems to be solved in quantum machine learning is the limitation present in the quantity of input data that the proposed implementations can handle. Although many-body quantum systems have a Hilbert space whose dimension increases exponentially in relation to the size of the system, permitting to store and manipulate a huge quantity of data, an important problem is to initialize accurately and efficiently this quantum state with the desired data. In machine learning this stage is essential, since learning a problem needs a lot of learning data.

Another important problem is to obtain quantum dynamics with memory which simultaneously conserves its quantum properties. As discussed in Section 2.2.4, memory is important in the implementation of machine learning algorithms in devices with non-universal computing capacities. This also holds in the quantum realm. Obtaining this memory in quantum dynamics

is even more difficult, due to the unitary evolution.

The aim of this Chapter is introducing some fundamental ideas about quantum machine learning. To this end, artificial neural networks are explained. This allows us to introduce the fundamentals of quantum neural networks. Finally, following the ideas about memory and non-Markovian dynamics, in the Section 4.2 open quantum systems are introduced, and we discuss the implementation of machine learning algorithms in non-Markovian open quantum systems.

4.1 Artificial Neural Networks

Artificial neural networks [2] are models inspired in biological neural systems. They are very common in machine learning due to their flexibility and the large data inputs which they can handle. Additionally, they are useful approximating a wide range of functions. The basic element of neural networks is the neuron, which has some input values x_i and a output value y (see Fig 4.1).

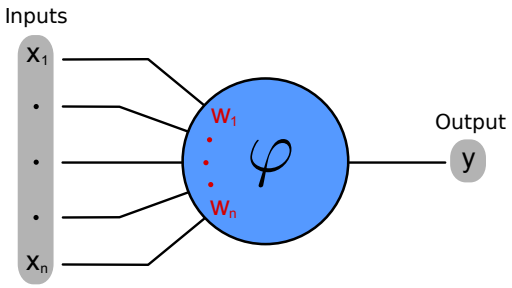

Figure 4.1: **Scheme of a artificial neuron.** It receives inputs x_i and gives a output y. To calculate the output a function φ is used which receives the sum of the inputs weighted by the w_i parameters.

A neuron can be understood as a function φ that gives an output depending on the sum of the inputs weighted by some w_i parameters,

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i\right). \tag{4.1}$$

When these neurons are grouped connecting the output of one layer to the inputs of other layers, a network is created, as shown in Fig. 4.2. Using this network allows us to approximate most of the functions by adjusting the weights of each neuron. There exist different procedures of modifying the weights, depending of the available information about the function which

we want to approximate. This flexibility allows the use of neural networks in different types of machine learning algorithms. The adjustment of these internal weights is better understood appealing to the dynamic and static algorithms explained in the first chapter. In this representation, the neural network is the dynamic algorithm and the responsible of adjusting his weights is the static algorithm.

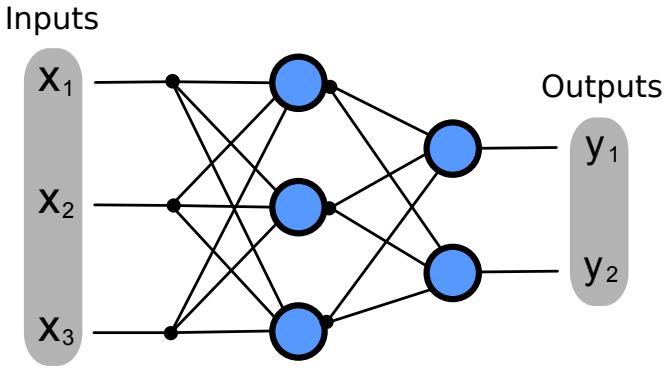

Figure 4.2: **Scheme of an artificial neural network.** A possible neuron network is shown, with three inputs and two outputs. It has two neuron layers. Using more layers permits approximating a wider range of functions, having the option of adjusting more weights.

The best-known type of neural network is based on the so-called perceptron neurons. Perceptrons perform a weighted sum of the inputs. If the resulting value is greater than a threshold δ, then the output is 1, otherwise the output is 0. So their corresponding φ function is a step function:

$$u = \sum_{i=1}^{n} w_i x_i \ , \qquad \varphi = \begin{cases} 1 & \text{if } u > \delta \ , \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The δ threshold should be adjusted in a similar manner as the weights. In order to do this, we only have to add an extra input in each neuron which always have the value -1, and hence, the threshold can be tunned as any other weight.

There are more different φ functions for the outputs, but even with these simple perceptron neurons, when grouping them correctly and adjusting their corresponding weights with a acceptable algorithm, it is possible to learn most of the classification problems.

4.1.1 Quantum Neural Networks

Quantum neural networks [8] are quantum versions of the artificial neural networks. Due to the promising power of the quantum computing devices and the flexibility of neural networks, there have been some recent attempts of combining both fields. It is expected that the improvement with respect to classical neural networks will be especially relevant in the learning stage, in which thanks to quantum entanglement the weights can be adjusted in paralell.

The most important challenge is combining the nonlinear dynamics of neural networks with the linear, unitary dynamics of quantum computing. Some proposals seem to obtain non-linear dynamics in the quantum system, using to this end measurements in the system or implementing the algorithm in an open quantum system [9]. Although no satisfactory procedure has been found yet to implement a quantum neural network containing all the interesting characteristics of neural and quantum computing together, it is a promising approach to fully-functional quantum machine learning algorithms.

4.2 Evolution of Open Quantum Systems

In quantum mechanics, the state of a system is represented by a vector in the corresponding Hilbert space. This vector is commonly called *ket* and his mathematical representation is $|\psi\rangle$. The ket is a complete representation, in the sense that it gives as all the possible information of a system.

In most of the cases, however, we do not know all this information. For these situations the quantum theory can be extended to describe systems whose state is not completely known. Suppose a quantum system is in one of a number of states $|\psi_i\rangle$, where i is an index, with corresponding probabilities p_i. The *density operator* for the system is defined by the equation

$$\rho \equiv \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \ . \tag{4.3}$$

This operator is known as density matrix, and all postulates of quantum mechanics can be reformulated in terms of it. This allows us to perform the analysis of systems about whose state we have some type of uncertainty. The density matrix becomes the most general representation of the state of a system.

Having introduced the density operator, we can differentiate two distinct types of states. When it is possible to represent the state as a ket $|\psi_i\rangle$ with pure certainty, it is said that the system is in a *pure state*. In this case, the density operator is simply $\rho = |\psi\rangle\langle\psi|$. Otherwise, if the last property can not be satisfied, the state is called a *mixed state*.

One of the most useful applications of the density operator formalism is the description of the subsystems of a composite quantum system. Let us assume that we have two subsystems A and B, whose composite state is described by the operator ρ^{AB}. The reduced density operator for the system A is defined by

$$\rho^A \equiv tr_B(\rho^{AB}) \ , \tag{4.4}$$

where tr_B is the partial trace over system B, defined as

$$tr_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \equiv |a_1\rangle\langle a_2| \, tr(|b_1\rangle\langle b_2|) \ . \tag{4.5}$$

Due to the reduced density operator, the measurements in the subsystem A can be handled in the same way as in the closed systems. It can be proven that the last definition of the partial trace is the only one which gives the correct measurement statistics for the subsystems.

Once we have this reduced density matrix, we may want to know its time evolution. This is essential when working with open quantum systems [10], because the dynamics of these systems is affected by their environments. Depending on the type of interaction between the subsystems or between the system and the environment, there are some different procedures and approximations.

A particularly useful method for describing this evolution is through the Lindblad master equation, which takes the form

$$\frac{d\rho}{dt} = -\frac{i}{\hbar}[H,\rho] + \sum_{k=1}^{M}\left(L_k\rho L_k^\dagger - \frac{1}{2}L_k^\dagger L_k\rho - \frac{1}{2}\rho L_k^\dagger L_k\right) \ . \tag{4.6}$$

H is the Hamiltonian of the system, ρ is the state of the system and the operators L_k are called Lindblad operators. The first term at the right of the equation is the responsible of the unitary evolution, and the other elements describe possible transitions that the system may undergo due to interactions with the environment. The Lindblad operators contain all the information about the system-environment interaction.

This master equation describes the non-unitary and Markovian time evolution of the reduced density matrix of the system. It is derived considering the unitary evolution of the composite system by the Liouville-von Neumann equation, in the approximation when the coupling between the system and the environment is weak compared to system energies (*Born approximation*) and the environment evolves at much faster timescales than the system (*Markov approximation*). The last assumption means that the environment has no memory and quickly relaxes to its steady state although being interacting with the system. This implies that the evolution of the density matrix is going to depend only in his current value.

In the last regime, there is no memory effect. In this Markovian approximation the system loses information to the environment during the evolution, and this information never comes back. So to have some type of memory in open quantum systems less strong assumptions are needed which permits a information return from the environment to the system, allowing the memory.

The most general known quantum master equation is Nakajima–Zwanzig's equation, which allows us the study of more general open quantum system, even those with non-Markovian dynamics. However, solving this equation is generally a very difficult task. A deeper research in open quantum systems is needed, to find simpler methods to describe open quantum system with memory.

4.2.1 ML in Open Quantum Systems

As mentioned in the Chapter 2, we are far from being able to implement universal quantum computers which can beat classical computers. The only practical alternatives in a short term are the single purpose quantum devices and hybrid systems, since they are simpler and more controllable. If we want to construct a machine learning algorithm in one of these devices, memory is required, since it is essential in learning and in consequence in machine learning.

The necessity of memory is problematic, since quantum devices commonly proposed for computation, which are small systems with unitary dynamics, do not show this desired memory. A option would be choosing a larger system with more complex dynamics which can show memory in a subsystem. However, this is also problematic, since describing and controlling one of these systems completely turns out to be extremely difficult. Also, when the size of the quantum systems grows, conserving the quantum properties without any quantum decoherence becomes really tricky.

A natural solution may be to implement the machine learning algorithm in a non-Markovian open quantum system using the environment as a resource. In this way, the memory can emerge naturally due to the non-Markovian dynamics of the system and we could expect a more robust behavior against decoherence due to the interaction with the environment.

More explicitly, from the point of view of our proposed machine learning algorithm physical separation, the open quantum system performs predictions and the environment provides the necessary memory to perform this predictions.

The challenge in this approach is to correctly design the system, the environment and their mutual interaction, with the aim of implementing a particular machine learning algorithm. A problem may be conserving the quantum properties in a dissipative open system which shows the desired memory. In order to show memory, the system has to be coupled to an environment, and it is usually a dissipative dynamics. Therefore, due to the quantum decoherence, the quantum properties disappear. With this in mind, finding a balance between memory and quantum phenomena becomes important.

Dynamics described by master equations do not seem to fit properly, since they are too cumbersome when one try to describe networks of the elements. To advance in this direction and yield practical results in quantum machine learning, a deeper knowledge about open quantum systems and their description is required.

Chapter 5

Conclusions

In this chapter we summarize the most important aspects learned in this work.

- The principal difference of the machine learning algorithms with regard to usual algorithms is the ability of learning based on the data. Unlike conventional algorithms, whose behavior is determined in the design, in machine learning, algorithms are adaptive and can learn how to solve different problems. The learning consists in automatically finding in a space of possible behaviors the one that solves a problem. To find it, received data is used.

- Memory is necessary in the implementation of machine learning algorithms in non-universal computing devices, due to the necessity of performing predictions which depend on history and past data. Although universal computers do not need this dynamic memory, since they can simulate it, a higher number of fully-controllable resources are needed, so in more direct physical implementations, memory seems essential.

- Open quantum systems prove to be useful to implement quantum machine learning algorithms.

  Until recent years, mostly closed quantum systems with unitary evolution were considered for quantum computing. Ideas as quantum error correction were devised with the aim of conserving in some way the properties of ideally closed systems. This procedure is limited for the enormous amount of resources required, and some algorithms like those of ML, demand efficient implementations of the dynamics.

  Therefore, open quantum systems started recently to be considered for quantum computing, due to the rich dynamics that they can show. Proposals of non-unitary dynamics based on measurements and feedback control have been made, but there is not a clear and general

approach yet. One of the most important challenges is finding the desired non unitary dynamics while retaining the quantum properties which make quantum computation more powerful than classical one.

More work is required for finding systematic methods of using non-Markovian dynamics to implement Quantum Machine Learning, having both memory and the power of quantum computing. Probably with a better comprehension of open quantum systems, novel ideas and proposals for an efficient implementation of ML algorithms can emerge.

Bibliography

[1] D. E. Knuth, *The art of computer programing* (Addison-Wesley, New Jersey, United States, 1973), Vol. 3, p. 105-139.

[2] E. Alpaydin, *Introduction to machine learning* (The MIT Press, Cambridge, UK, 2014), Ed. 3, Chap. 1 and 11.

[3] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning From Theory to Algorithms* (Cambridge University Press, Cambridge, UK, 2014), Chap. 1.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2000), Chap. 1 and 6.

[5] R. P. Feynman, *Simulating Physics with Computers*, Int. J. Theor. Phys. **21**, 467 (1982).

[6] C. Lavor, L.R.U. Manssur, and R. Portugal, *Grover's Algorithm: Quantum Database Search*. arXiv:quant-ph/0301079

[7] M. Schuld, I. Sinayskiy, and F. Petruccione, *An introduction to quantum machine learning*, Contemporary Physics **56**, 172 (2015).

[8] M. Schuld, I. Sinayskiy, and F. Petruccione, *The quest for a Quantum Neural Network*, Quantum Information Processing **13**, 2567 (2014).

[9] P. Pfeiffer, I. L. Egusquiza, M. Di Ventra, M. Sanz, and E. Solano, *Quantum Memristors*, Scientific Reports **6**, 29507 (2016).

[10] H. -P. Breuer and F. Petruccione, *The theory of open quantum systems* (Oxford university press, Oxford, UK, 2002), p. 109-139.