QUANTUM MACHINE LEARNING

A PRIMER FOR DATA SCIENTISTS

Leonardi Alchieri - 860624
Data Science
Università degli studi di Milano-Bicocca
l.alchieri2@campus.unimib.it

Davide Badalotti - 861354
Data Science
Università degli studi di Milano-Bicocca
d.badalotti@campus.unimib.it

Pietro Bonardi - 859505
Data Science
Università degli studi di Milano-Bicocca
p.bonardi2@campus.unimib.it

February 10, 2021

ABSTRACT

Throughout this work our aim has been that of describing, at best of current knowledge, the growing field of Quantum Machine Learning, mostly for a non-practical reader. Thus, the work proposes itself as a exposition of the most notable scientific literature about this subject.

In order to achieve this goal, the basics of computational logic, quantum mechanics and quantum logic were described. In particular, some specific elements and algorithms of quantum computing were introduced, in order to allow a better understanding of later QML techniques.

The main aspects of Quantum Machine Learning were analyzed, with detailed descriptions of some of the top-of-art algorithms, e.g. Quantum Natural Gradient and Quantum Support Vector Machines. Due to their interest in recent years, some introduction to Quantum Deep Learning, especially Quantum Neural Networks, was given, with in-depth description of some novel methods.

Contents

1 Introduction

The general development of the human kind, especially since the late 18th century, has gone in hand with the development of new physical and mathematical techniques. And, with them, the need for more and more power to do calculations, with the evolution from *human computers*, i.e. people whose job was literally to execute a large number of calculations, to the modern-day idea of a **digital computers**. [1]

The first modern computers, born form the ideas and inventions of Alan Turing, evolved, during the second half of the 20th century, into multi-purpose machines with capabilities not even imaginable in their beginnings. Since their logical conception as mechanical devices capable of doing simple operations on "memory slots", the idea behind modern computer has been that of the **bit**: a single piece of information that can be either *activated* (1) or *non activated* (0).

Starting as well from the end of World War 2, **quantum mechanics** established itself as one of the most successful physical theories. Ideas such as the *Uncertainty Principle* and the **superposition of physical states** allowed for new fascinating discoveries in many scientific areas.

As such, a somewhat "linear" evolution of scientific computation might be the introduction of quantum principles with computation logic and architectures. Indeed, since the first idea of a *quantum computer* by Nobel Physics Laureate Richard Feynman in 1982, theoretical and practical advancements have made one of the most interesting fields in today's Computer Science. [2] With the emergence of new algorithms and techniques, which would allow to solve complicated "classical" calculations in a fraction of its traditional time, the race for the first **general purpose quantum computer** is today at its height.

Given such premises in the evolution of Computer Science, the main goal of this work is to show how, from a theoretical point of view, many classical problems could be solved using **quantum logic and computation**. In particular, the focus will be towards the wide field of **Machine Learning**, a series of very different algorithm based on the idea of having the computer learns itself some its parameters and/or functions. Indeed, recent developments, even from a practical point of view, show how the use of **quantum** logic may either help speed up some classical calculations or allow to solve intractable problems.

In the first part of this paper an introduction to computer logic and quantum mechanics is given, in order to pose a knowledge basis. Quantum logic and computation is than described, before going into the heart of the work with descriptions of Quantum Machine Learning, its advantages, and its main algorithms analyzed today.

2 Logic & Computation

In this section we introduce logic and computation for the classical electronic computers. The latter use electric impulses to perform operations, and the abstraction above these electric signals are the **bits**, also known as binary digits. The aim of a computer (in general), is to mimic all the logic and arithmetic operations in order to deal with problems, and this is achieved through the use of transistors. The transistors are physical components made out of silicon, that can be arranged in order to perform simple logic operations. [3]

More in detail a bit is defined as:

$$x \in \{0, 1\} \tag{1}$$

It can takes only two possible values: "1" meaning high voltage, while on the other hand "0" meaning low voltage. Where high and low voltages are determined with respect to a prefixed threshold.

Instead of using the "0" and "1" notation, we will use an abstract symbol, mostly used in the field of **quantum computing**, $|0\rangle$ and $|1\rangle$ respectively. It is possible to associate with $|0\rangle$ and $|1\rangle$ two column vectors as follow:

$$|0\rangle \to \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle \to \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2}$$

This two vectors forms a basis for a space of 2 dimensions, and so are orthonormal.

2.1 Logic gates

A logic gate is a electric circuit composed of transistors, able to perform operations. The fundamental ones are "AND", "OR" and "NOT". All these work on bits and perform different logic on them. It is worth to notice, that from these operators it can be derived a series of other ones, listed in detail after.

Truth tables sum up operator logic, using $|x\rangle$ and $|y\rangle$ to represent bits. We shall thus show the logic for each operator.

$\lvert x\rangle$	$\lvert \bar{x}\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$

Table 1: NOT Operation truth table, uses only one input bit

$\lvert x\rangle$	$\lvert y\rangle$	$\lvert x\rangle \vee \lvert y\rangle$
$\lvert 0\rangle$	$\lvert 0\rangle$	$\lvert 0\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 1\rangle$	$\lvert 1\rangle$

Table 2: OR Operation, truth table

The **NOT operator** simply flips the bit state from one to another as illustrated in Table 1. The **OR operator**, as shown in Table 2, returns true only where **at least one** bit is equal to one, while the **AND operator**, Table 3, returns true only in the case where the two bits are true, so where $\lvert x\rangle$ **and** $\lvert y\rangle$ are equal to $\lvert 1\rangle$.

As stated previously from these operators one can derive other operators such as: the NOR (*not or*), NAND (*not and*) and XOR (*exclusive or*) defined as follow:

$$NOR = \overline{(\lvert x\rangle \vee \lvert y\rangle)} = \lvert \bar{x}\rangle \wedge \lvert \bar{y}\rangle \tag{3}$$

$$NAND = \overline{(\lvert x\rangle \wedge \lvert y\rangle)} = \lvert \bar{x}\rangle \vee \lvert \bar{y}\rangle \tag{4}$$

$$XOR = \lvert x \oplus y\rangle = \left\lvert (x \vee y) \wedge \overline{(x \wedge y)} \right\rangle \tag{5}$$

$$\tag{6}$$

The NOR and the NAND operators have the same truth tables of OR and AND respectively but conjugated. Meaning that, where there are $\lvert 0\rangle$s in the outputs of the OR and AND truth table, actually there are $\lvert 1\rangle$s and viceversa, as can be seen in Table 5.

Reversibility plays a relevant role in quantum computation, since, as we will see, the general computational process can be modeled with a **unitary operation** that is indeed reversible. It is worth to notice that the only reversible logical function, up to this point, is the NOT: each output arises from a unique input.

Another fundamental operation used in computer registers and more, is the **SWAP**. The SWAP operation is a two-bit reversible operations which exchange the values x and y of the two bits $\lvert x\rangle$ and $\lvert y\rangle$. The operator is defined as:

$$\mathbf{S}\lvert x\rangle\lvert y\rangle = \lvert y\rangle\lvert x\rangle \tag{7}$$

So if we consider in general n-bit state $\lvert x\rangle_n$, then we can define:

$$\mathbf{S}_{hk} = \mathbf{S}\lvert x\rangle_{n-1}, \dots, \lvert x\rangle_h, \dots, \lvert x\rangle_k, \dots, \lvert x\rangle_0 = \tag{8}$$

$$\lvert x\rangle_{n-1}, \dots, \lvert x\rangle_k, \dots, \lvert x\rangle_h, \dots, \lvert x\rangle_0 \tag{9}$$

The last two bits reversible operations we show is the **CNOT**, also know as control not. It is a pillar operator for quantum computation and it acts on a target bit according to the value of a control bit. By defining $\mathbf{C}_{h,k}$, if we suppose that h is the controller bit and k is the target bit, so CNOT flips the state of *k-th* bit if and only if the state of the *h-th* bit is equal to either $\lvert 1\rangle$ or $\lvert 0\rangle$.[4] In Table 6 are shown the CNOT gates in a 2-bit system.

We introduce a transformation named **Hadamart transformation** defined as:

$$\mathbf{H}\lvert x\rangle = \frac{\lvert 0\rangle + (-1)^x \lvert x\rangle}{\sqrt{2}} \tag{10}$$

The equation brings two informations:

- The case where $x = 0$:

$$\mathbf{H}\lvert 0\rangle = \frac{\lvert 0\rangle + \lvert x\rangle}{\sqrt{2}} \tag{11}$$

$\lvert x\rangle$	$\lvert y\rangle$	$\lvert x\rangle \wedge \lvert y\rangle$
$\lvert 0\rangle$	$\lvert 0\rangle$	$\lvert 0\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert 0\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$	$\lvert 0\rangle$
$\lvert 1\rangle$	$\lvert 1\rangle$	$\lvert 1\rangle$

Table 3: AND Operation truth tables, it uses two input bits

$\lvert x\rangle$	$\lvert y\rangle$	$\lvert x\rangle \oplus \lvert y\rangle$
$\lvert 0\rangle$	$\lvert 0\rangle$	$\lvert 0\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 1\rangle$	$\lvert 0\rangle$

Table 4: XOR Operation, uses two input bits

- The case where $x = 1$:

$$\mathbf{H}\lvert 1\rangle = \frac{\lvert 0\rangle - \lvert x\rangle}{\sqrt{2}} \tag{12}$$

This gate transforms a single bit into a **linear combination of them**.[3] From a classical point of view, this operator has no meaning, since a combination of bits does not represent anything. In Section 4 will be shown how, on the other hand, its use in quantum logic is fundamental.

2.2 Algorithm

A computer, in order to solve problems, needs a systematic procedure. The procedure is specified through a sequence of instructions and this is how an **algorithm** is defined. [5]

To ensure that the concept of algorithm correctly represents the effective method idea, it is necessary to introduce four properties. The first one is **finiteness**, namely that an algorithm has to be made out of a finite number of steps. The second is **definiteness**, i.e. the instructions have to belong to a prefixed and finite elementary instruction set. **Uniqueness** states that every instruction has to be uniquely interpretable. **Effectiveness**, the fourth and final one, means that a computer must be able to execute every instruction in a finite time.

2.3 Complexity

It is a difficult tasks to understand time complexity of an algorithm, since it depends on several factors, i.e. the processor, the programming language etc. So instead it is shown in this Section how the runtime of a function grows as the size of the inputs grows.

In order to express **running time complexity** in a systematic way, the **Big-O notation** is used. Using this, constant time can be express with $O(1)$, linear time with $O(n)$ and quadratic time with $O(n^2)$ where n is the input size. By definition, assuming two functions $f, g : A \subseteq \mathbb{R} \mapsto B \subseteq \mathbb{R}$, strictly positive, $x_0 \in A$ and a constant $M > 0$, we say:

$$f(x) \in O(g(x)),\, x \to \infty \iff \exists x_0 : |f(x)| < M|g(x)| \;\; \forall x > x_0 \tag{13}$$

Given the Big-O definition, we are able to know the running time complexity of algorithms. Actually complexity is known for the vast majority of them. In Table 7 are described a the Big-O runtime of a few simple algorithms.

$\lvert x\rangle$	$\lvert y\rangle$	$\lvert x\rangle\, NOR\, \lvert y\rangle$
$\lvert 0\rangle$	$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert 0\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$	$\lvert 0\rangle$
$\lvert 1\rangle$	$\lvert 1\rangle$	$\lvert 0\rangle$

$\lvert x\rangle$	$\lvert y\rangle$	$\lvert x\rangle\, NAND\, \lvert y\rangle$
$\lvert 0\rangle$	$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 1\rangle$	$\lvert 0\rangle$

Table 5: NOR and NAND operatos, truth tables

$\|x\rangle \|y\rangle$	C_{10}	C_{01}
$\|0\rangle \|0\rangle$	$\|0\rangle \|0\rangle$	$\|0\rangle \|0\rangle$
$\|0\rangle \|1\rangle$	$\|0\rangle \|1\rangle$	$\|1\rangle \|1\rangle$
$\|1\rangle \|0\rangle$	$\|1\rangle \|1\rangle$	$\|1\rangle \|0\rangle$
$\|1\rangle \|1\rangle$	$\|1\rangle \|0\rangle$	$\|0\rangle \|1\rangle$

Table 6: CNOT Operation

Big-O	Algorithm
$O(1)$	Determine if a number is even or odd
$O(n)$	Finding an item in an unsorted list or in an unsorted array
$O(n^2)$	Simple sorting algorithms, such as bubble sort

Table 7: Example of common algorithms running time complexity

Even if what described here is merely an idea, the same principles can be applied to quantum computation. Indeed, the use of the Big-O notation allows quite easily to compare the running time complexity of classical algorithm with their quantistic counterparts, as will be done throughout this work.

3 Quantum Mechanics: basic elements

In this section, some basic concepts of quantum mechanics are introduced. [3, 6]

Since quantum systems often behave very differently from classical systems, a novel representation is needed in their description. Understanding the meaning of this representation and its mechanisms allows the reader to comprehend the basics of quantum computation and Quantum Machine Learning (QML).

3.1 System, States, Measures

With the term *physical system*, we refer to any portion of the physical universe that can be described and analyzed. Classical systems are described by the laws of classical physics, an object falling from a cliff, an electric charge in space, etc.

Quantum systems, on the other hand, are described by the laws of quantum physics. Ensemble of atoms or particles are typical example of quantum systems. Actually, any event happening at atomic or subatomic scale can be described in term of a quantum system. Now, it is known that classical physical systems have a *deterministic nature*: given an initial state, the system will produce always the same output. For example, the behaviour of a moving object can be predicted given a certain set of initial parameters. More precisely, the result of any measurement on that object (position at a certain time, velocity, rotational speed) can be predicted in advance. On the other hand, quantum systems have an **intrinsic stochastic behaviour**. One cannot predict with certainty the result of a measure, but only its probability. In other words, a certain quantum measurement on a given system can have multiple outcomes, and each of these outcomes will manifest itself with a given probability, which can be predicted in advance. To represent this situation, the concept of quantum state is introduced. A **state** is an object that allows for the calculation of events probabilities, and contains all the information about a given system. To refer to a state, the **Dirac's notation** is used: a state of the system is associated with an object called *ket*, through the notation $|x\rangle$.

The simplest possible quantum system is called qubit, that has the following representation:

$$|\phi\rangle = a_+ |+\rangle + a_- |-\rangle \tag{14}$$

Where a_+ and a_- are complex numbers. When a measurement is realized on this system, it can only have one of two possible outcomes, which are represented by $|+\rangle$ and $|-\rangle$. Each of these outcomes has a probability of respectively $|a_+|^2$ and $|a_-|^2$. It is said that the system is a *superposition of two states*. The two states "coexists" inside the system, and one or the other can be revealed with a measurement. Mathematically, the state $|\phi\rangle$ is a vector in Hilbert Space $\mathcal{H}$, of which $|+\rangle$ and $|-\rangle$ are a basis. A practical example of this situation might be the following: the state $|\phi\rangle$ represent the position of a single, static particle. This particle can either be in two (fixed) position. As said, a quantum state (a ket) represent a physical system. In the case reported in (14), the state $|+\rangle$ represents the particle in the first position, while the state $|-\rangle$ represents the particle in the second position. The state $|\phi\rangle$, finally, represents the particle whose position is unknown. The particle is both in the first and second position, with given probabilities.

It is important to remark that the state $|\phi\rangle$ does not simply mean *since the position is unknown, it can be one or the other*, but physically represent a particle that is **both** in position one and two [1].

By implicitly assuming that the only possible outcomes of the measure are $|+\rangle$ and $|-\rangle$, it is possible to write:

$$|\phi\rangle = a_+ |+\rangle + a_- |-\rangle \longrightarrow \begin{pmatrix} a_+ \\ a_- \end{pmatrix} \tag{15}$$

In other words, the state is represented as a linear combination of the basis vectors of the Hilbert space $\mathcal{H}$.

$$|\phi\rangle = a_+ \begin{pmatrix} 1 \\ 0 \end{pmatrix} + a_- \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{16}$$

Where:

$$|+\rangle \longrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$|-\rangle \longrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Equivalently, another representation for states is defined: the *bra*.

$$\langle x| = |x\rangle^* \tag{17}$$

Where the symbol $(*)$ denotes the adjointness operation. Given $|\phi\rangle$ in (14):

$$\langle\phi| = a_+^* \langle+| + a_-^* \langle-| \longrightarrow (a_+^* \quad a_-^*) \tag{18}$$

It is then clear that the product between bra and ket is equivalent to the inner product between vectors:

$$\langle\phi|\phi\rangle = a_+^2 + a_-^2 \tag{19}$$

While the ket-bra product yields a matrix:

$$|\phi\rangle\langle\phi| = \begin{pmatrix} a^2+ & a_+a_-^* \\ a_-a_+^* & a_-^2 \end{pmatrix} \tag{20}$$

3.2 Operators

An operator is an object that, when applied to a quantum state, yields another quantum state:

$$A|\phi\rangle = |\phi'\rangle \tag{21}$$

For a two-level system, it's possible to write:

$$A|\phi\rangle = A(a_+|+\rangle + a_-|-\rangle) = a_+A|+\rangle + a_-A|-\rangle \tag{22}$$

So, in this case, an operator can be thought as a square matrix. In quantum mechanics, every observable (speed, momentum, position, etc.) is an operator. It it possible to obtain the **mean value of an observable O on a given state** $|\phi\rangle$ with the following expression:

$$\langle O\rangle = \langle\phi|O|\phi\rangle \tag{23}$$

It is important to remark that any information on a given quantum state must be obtained through a measurement. For this reason, in quantum computing and quantum machine learning the concepts of observable and measurement are very important, since they represent the only way to interact with a quantum system.

Another important concept in quantum mechanics is the *Wave Function collapse*: basically, any form of measurements on a quantum state affects it irreversibly, causing the collapse of the function. For example, supposing that a measurement on a two-level quantum state as in Equation 14 is made. The result of the measurement is $|+\rangle$. After the measurement, the state of the system will always be $|+\rangle$, as if the other component $|-\rangle$ ceased to exist. As stated by the Copenhagen interpretation of quantum mechanics: when a measurement is done on the system, its state changes, collapsing in the state that resulted by that measure. Any measurement, in conclusion, affects the system irreversibly.

[1]For a better comprehension of the phenomenon, see [7]

3.3 Density matrix

An alternative way to represent a quantum state is through its **density matrix**, defined as:

$$\rho_\phi = |\phi\rangle\langle\phi| \tag{24}$$

This form of representation is particularly useful, since, given an observable O, its mean value on a state $|\phi\rangle$ can be written as:

$$\langle O\rangle = \mathrm{Tr}(O\rho_\phi) \tag{25}$$

3.4 Pauli matrices

For a two level system, both operators and density matrix are 2×2 matrices. It is common to represent any 2×2 matrix as a linear combination of the following matrices:

$$I = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix} \qquad \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Where σ_n are called **Pauli matrices**. In other words, on a two level system, every observable and every density matrix (and so every state) can be represented as a linear combination of the Pauli Matrices and the identity. These operators, in other words, constitute a basis of the space of observables and states for two level systems, and will be very useful in the next sections.

3.5 Hamiltonians and Temporal evolution

Until now, when describing a quantum system, time dependency has not been accounted for. In other words, only static system have been considered. In quantum mechanics, quantum systems are usually subject to the action of time: they evolve. It is possible, then, to write:

$$|\phi(t)\rangle \tag{26}$$

Where the parameter t makes the time dependency explicit. As mentioned earlier, in order to modify a quantum state, an operator is needed. This holds also for time dependency. The temporal evolution operator is defined as:

$$S(t, t_0)\,|\phi(t_0)\rangle = |\phi(t)\rangle \tag{27}$$

It is also possible to show that this operator can be written, for infinitesimal time variations as:

$$S(t_0 + \varepsilon, t_0) = I - i\varepsilon\hat{H}(t_0) \tag{28}$$

In other words, the operator $S(t_0 + \varepsilon, t_0)$ shifts the state from time t_0 to time $t_0 + \varepsilon$.

The operator $\hat{H}(t)$ is the hamiltonian of the system. While the utility of such expression might not be clear at first sight, it is important to remark that hamiltonians are central concepts of mechanical physics in general [8]. Also, they are very important in quantum computing, inside the so-called *adiabatic model* [3]. It is possible to think of the hamiltonian as that operator which completely determines the temporal evolution of the system.

3.6 Entanglement of two qubit-states

It is possible to define a two qubit state from two single qubits, namely:

$$|\phi_A\rangle = \alpha_A\,|+_A\rangle + \beta_A\,|-_A\rangle \tag{29}$$

$$|\phi_B\rangle = \alpha_B\,|+_B\rangle + \beta_B\,|-_B\rangle \tag{30}$$

Then, the two qubit state can be defined as:

$$|\phi_A\,\phi_B\rangle = |\phi_A\rangle\,|\phi_B\rangle = |\phi_A\rangle \oplus |\phi_B\rangle \tag{31}$$

This is a first example of a *separable state*, i.e. a state that can be written as the tensor product between two different states. This is not always the case, for example:

$$|\phi_{AB}\rangle = \frac{|-_A\rangle\,|-_B\rangle + |+_A\rangle\,|+_B\rangle}{\sqrt{2}} \tag{32}$$

This state is not separable: the two subsystems are **entangled**.

Quantum entanglement is at the heart of quantum mechanics, and gives rise to some counter-intuitive phenomena that can be exploited in the fields of quantum information and computation. Supposing, for example, that A and B represent two different particles in the entangled state, Equation 32; one could imagine to perform a measurement on particle A, which gives result $|-_A\rangle$. From that moment on, any measurement on particle B will yield the result $|-_B\rangle$, because the state has "collapsed" in the first term of Equation 32. The two subsystems are entangled in the sense that a measurement on one of the two affects both irreversibly. In other words, measurement on the two subsystems are **perfectly correlated**. The utilization of this phenomenon in quantum computation is a very active area of research and development.

4 Quantum Logic

Using quantum machine paradigms in conjunctions with logical theories of computing lead to the development of **quantum computing**. This theory of computation allows for the definition of specific algorithms which, in many cases, can theoretically manage to achieve significant speedups, e.g. Shor's factoring algorithms is in polynomial time, while the best classical one is at best in sub-exponential time. [9]

While in classical computation the basic building block is the **bit**, a state which can either be 1 or 0, and thus carry a single information, its counterpart in quantum computation is the **qubit**, a two-level quantum state defined as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{33}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.[3] The bit states $\{|0\rangle, |1\rangle\}$ form a basis called **computational basis** in the 2-dimensional Hilbert space, and complex numbers α, β store the information: as opposed to the classical bit, it follows that, **potentially**, a qubit can store an **infinite amount of information.** From a physical point of view, a **measurement**, on the given computational basis, will give only one of the two possible outcomes from the **superposition state**, i.e. either $|1\rangle$ or $|0\rangle$, with probability $p(0) = |\alpha|^2$ and $p(1) = |\beta|^2$.

Two main models have been developed for quantum computation: by defining operations on single or multiple qubits, **quantum circuits** can be constructed; or by manipulating states at lower levels of abstraction, in the so-called **adiabiatic quantum computing**. Both models allow for one of the most interesting aspects of quantum computers: **quantum parallelism**, the main application of superposition of states, which allows in many cases greater problem speedups, as compared to a classical approach. Other models, such as *topological quantum computing* [10] and *one way quantum computing* [11] exist, but their description goes beyond the scope of the present work.

In the following pages, the main elements of quantum computing and quantum logic will be described from a theoretical point of view. An analysis of current and future real-world construction of quantum computers is not of interest in the present paper.

4.1 Qubits and the Bloch Sphere

As already mentioned, the most basic element of any quantum computing model is the **qubit**, as defined in Equation 33. While in itself a very useful representation, using the constraint given on the parameters,[12]

$$|\alpha|^2 + |\beta|^2 = 1 \tag{34}$$

a better representation can be obtained. Indeed, considering that Equation 34 defines a sphere in the $\alpha - \beta$ complex space, one can try to create some "spherical coordinates" in this space using the following parametrization:[2]

$$\alpha = \cos\frac{\theta}{2} \qquad \beta = e^{i\phi}\sin\frac{\theta}{2} \tag{35}$$

with $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$, obtaining:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle = \cos\frac{\theta}{2}\,|0\rangle + (\cos\phi + i\sin\phi)\sin\frac{\theta}{2}\,|1\rangle \tag{36}$$

With the given constraints, θ and ϕ define a point on a 3-dimensional sphere, called in this case a **Bloch Sphere**, which allows to give a **geometric representation to single qubit operations**. This can be seen better using the **density matrix representation**, for pure states, which can be demonstrated to be, for the given 2-dimensional Hilbert space:

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix} = \begin{pmatrix} \cos^2\frac{\theta}{2} & \frac{1}{2}e^{-i\phi}\sin\theta \\ \frac{1}{2}e^{i\phi}\sin\theta & \sin^2\frac{\theta}{2} \end{pmatrix} \tag{37}$$

[2]This is a somewhat ordinary mathematical application valid also in a classical 3D space.

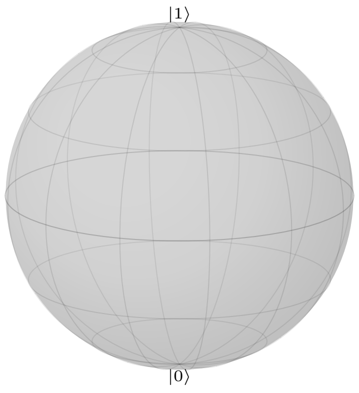


Figure 1: The Bloch Sphere, with the computational basis marked at their respective south and north pole.

It can be shown that any 2×2 complex Hermitian matrix can be expessed in terms of the identity matrix and a vector of Pauli matrices $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \sigma_3)$:

$$\rho = \frac{1}{2} \left(\mathbb{I} + \boldsymbol{s}\boldsymbol{\sigma} \right)$$

where the vector $\boldsymbol{s}$, called **Bloch vector**, can be written, for this case, as:

$$\boldsymbol{s} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\phi \end{pmatrix} \tag{38}$$

which is a 3-dimensional representation of a vector on a sphere.

In particular, the computational basis, in this 3-dimensional representation, represents the north and south pole, respectively for $|0\rangle$ and $|1\rangle$:

$$|0\rangle \Rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad |1\rangle \Rightarrow \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

The x-axis of the Bloch sphere is called the *diagonal basis*, and is given by the eigenvectors of the Pauli matrix σ_1, as defined in Section 3.4. Analogously, the y-axis, called the *circular basis*, is given by the eigenvectors of σ_2. And, since they represent the north and south pole, the *computational basis* corresponds to the z-axis. The further from the poles, which represent where the pure states intersect the classical behaviour, the more “quantum" the states are.

Using all stated here, it can be inferred that states with **multiple qubits** can be constructed easily, as *product spaces* of individual qubits:

$$|\psi\rangle_n = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle \tag{39}$$

with constraint $\sum_{x=0}^{2^n-1} |\alpha_x|^2 = 1$, where n represents the physical ways to encode the information, in a way similarly to the classical representation, as in Section 2. For example, in a 2-qubit system one can write:

$$|\psi\rangle_2 = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{40}$$

As one might expect, while for a single qubit it is straightforward to create a geometric representation as in the Bloch sphere, the same cannot be done in the presence of multiple qubit states.

The number of **probability amplitudes** can be shown to be 2^n for an n-qubit system, but which measurements can be performed put a limit to the amount of information that can be extracted from the system: this is called **Holevo's** bound, which states that n qubits can encode at most n states.[13]

4.2 Quantum Circuits

As described in Section 2, given a computer state, the *bit*, either 1 or 0, one can perform operations on it using **logic gates**, be it on a single bit, e.g. NOT, or multiple bits, e.g. OR. Since the **qubit** just introduced can be seen as the quantum-extension of the bit, and since one can also find the classical representations, as can be seen in Figure 1, **logic operations** can be applied to them as well.

Thus, generally, a **quantum logic gate** is defined as a transformation of an *input qubit* $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into an *output qubit* $|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$. Since the output is itself still a qubit, the condition $|\alpha'|^2 + |\beta'|^2 = 1$ must be satisfied. It can be shown that this condition is equivalent to declaring that a quantum logic gate is a **linear unitary transformation** of the form:

$$|\psi\rangle \Rightarrow |\psi'\rangle \hat{=} \hat{\mathcal{U}}|\psi\rangle \tag{41}$$

where $\hat{\mathcal{U}}^\top\hat{\mathcal{U}} = \hat{\mathcal{U}}\hat{\mathcal{U}}^\top = \mathbb{I}$. [12] Given this condition, since the transformation is unitary, it is also **reversible**.

From a practical point of view, in a similar fashion to classical logic, quantum gates can be represented symbolically using specific identifiers, as in Figure 2, with the line prepresenting the logical "wires" going in time from left to right.

The most basic quantum gates are the **single quantum gates**, i.e. transformations that are applied to only one qubit. The most elementary, and the only classical logic operation permitted due to its reversibility, it the NOT gate. [3] Similarly as in Section 2, where a bit changed to its opposite, the same happens for a quantum state, where the state $|0\rangle$ is changed into $|1\rangle$ and vice versa. Thus, for a general qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the result will be:

$$|\overline{\psi}\rangle = \alpha|1\rangle + \beta|0\rangle \tag{42}$$

Using the **matrix representation**, the operation is represented by:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_1$$

where σ_1 refers to the **first Pauli matrix**, as defined in Section 3.4.

In general, since single quantum gates are unitary transformations, their matrix representation will always be unitary: it means that they can be seen as a linear combination of *Pauli matrices*. The reverse is also true: any combination of *Pauli matrices*, i.e. any unitary transformation, acting on a qubit can be seen as a single-qubit gate. From a practical point of view, only a few gates are of great interest, but, due to the properties of quantum mechanics, some transformations of little interest in classical logic become quite relevant.

The **Hadamard transformation**, which in classical logic has no real meaning outside in usage with other transformations, can be defined in a similar fashion as in Section 2. In this case, writing Equation 10 as:

$$H = \frac{1}{\sqrt{2}}(\sigma_1 + \sigma_3) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{43}$$

it can be easily inferred that its application to a single-qubit state transforms it in the fashion as in Figure 3. Its most interesting usage, though, derives from applying the gate to a **single-bit** state $|x\rangle$, where either $x = 0$ or $x = 1$, and make it into a superposition of states. Using the Bloch sphere as reference, it allows to move from a state fixed on either the north and south pole to another position, allowing to move information from a classical point of view to a quantistic point of view. In Figure 4 is shown its application to a bit state and the definition of the quantum superposition from it. While classically it has no meaning, many interesting application could rise from it in the field of quantum computing. It can also be demonstrated that the Hadamard operator is **idempotent**, i.e. $H^2 = \mathbb{I}$.

A **phase shift gate** can be defined using the third Pauli matrix, σ_3. Consider the following expression:

$$\sigma_3|x\rangle = e^{-i\pi x}|x\rangle$$

where again $x \in \{0, 1\}$. Thus, one can define the **phase shift gate** as the **phase shift operator** as follow:

$$\begin{aligned} e^{-i\pi\sigma_3} &= \cos\phi\mathbb{I} - i\sin\phi\sigma_3 = \\ &= \begin{pmatrix} \cos\phi & 0 \\ 0 & \cos\phi \end{pmatrix} - \begin{pmatrix} i\sin\phi & 0 \\ 0 & -i\sin\phi \end{pmatrix} = \begin{pmatrix} e^{-i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix} = e^{-i\phi}\begin{pmatrix} 1 & 0 \\ 0 & e^{2i\phi} \end{pmatrix} \end{aligned} \tag{44}$$

This operator adds a **phase shift** of 2ϕ between the computational basis states. It should also be noted that the *global* phase shift $e^{-i\phi}$ can be dropped, as usual in quantum mechanics.[6] From the definition in Equation 44, using different

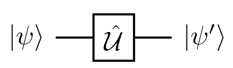


Figure 2: A general quantum gate representation. For quantum-specific gates, the most common representation is a box with a letter in it. Since the input and output are quantum states, the Dirac notation is left even for graphical representations.

$$\alpha|0\rangle + \beta|1\rangle \;-\!\!\!-\boxed{H}-\!\!\!-\; \tfrac{\alpha+\beta}{\sqrt{2}}|0\rangle + \tfrac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

Figure 3: Application of the Hadamard gate to a single-qubit state.

$$|x\rangle \;-\!\!\!-\boxed{H}-\!\!\!-\; \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{(-1)^x}{\sqrt{2}}|1\rangle$$

Figure 4: Application of the Hadamard gate to a single-bit state: creating a new superposition starting from a “pure" state.

ϕ, one can get different gates. The Z **gate** is obtained when considering $\phi = \frac{2}{\pi}$:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \sigma_3 \tag{45}$$

which leaves invariant the sign of $|0\rangle$ and changes to its inverse $|1\rangle$. If considering $\phi = \frac{\pi}{8}$, one obtaines the so-called T **gate**:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \tag{46}$$

From this, the famous **phase gate** $S = T^2$ is obtained, whose construction is justified in order to implement **universal fault-tolerant quantum computation**.[14]

It can be shown that, given the Bloch sphere representation of a single qubit, all single-qubit logic gates can be written as rotations along all of the axis of the 3-dimensional space.

As for **2-qubit** systems, since the use of NOR and NAND gates as in classical logic cannot be implemented, due to their non-reversibility, the prototypical gate for this case can be found in the **CNOT** gate. As described in Section 2, given a control qubit and a target qubit, it will change the value of the latter depending on the value of the former. Given a 2-qubit systems, two CNOT gates can be defined, as mentioned previously: C_{10} and C_{01}. From a practical point of view, and especially for this work, the former C_{10} is of greater interest and thus shall be sorely referred to as the CNOT gate.[3] In particular, this gate will change the outcome of the target gate if the control is $|1\rangle$. In matrix form, it can be written as:

$$CNOT = C_{10} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{47}$$

In Figure 5 is shown the representation of the CNOT effect, where the $\oplus$ operation is the classical XOR operator.

There is an interesting theoretical result connected to the *reversibility* of quantum gates. Since each operation is reversible, **no information is lost**. Thus, according to the classical Landauer's principle, which states that the minimum amount of energy required to change a bit is $k_B T \log 2$, with k_B the Boltzmann constant and T the temperature of the environment, quantum computation could be performed without loss of energy. [15] This theoretical result is still today disputed in literature.[16]

Other important gates are the SWAP gate, which can be created by combining 3 CNOT gates, and the Taffoli gate, which generalizes the CNOT gate in the case of 2 control qubits and one target qubit, and can allow quantum circuits to simulate irreversible classical circuits. If combining a CNOT with a Hadamard gate, one can generate the **Bell states** from the computational basis, which are *maximally entangled*.

4.3 Quantum Adiabatic Model

The adiabatic computing model is a different approach to calculations over qubits, as opposed to the quantum circuits. Generally, given a function over the qubit space $f : \{0,1\}^n \mapsto [0, \infty)$, one can state a minimization problem of the form $\min_{x \in \{0,1\}^n} f(x) = f(x_0) = f_0$, where f_0 is required to be unique. [17]

According to Farhi et al [18], the use of the **adiabatic theorem** can help solve this problem. Given an Hamiltonian H_0, as defined in Section 3.5, and considering *slow changes*, as to physically allow the system to adapt its configuration, one can find the ground state of a system of Hamiltonian H_1 via the following equation:

$$H(\lambda) = (1-\lambda)H_0 + \lambda H_1 \tag{48}$$

[3]According to Olivares [3], the C_{01} can be referred in literature as $\overline{\text{C}}$NOT, and its control value should be $|0\rangle$ in order to change.

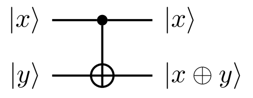


Figure 5: Gate representation of the CNOT gate. The qubit $|x\rangle$ is the **control qubit**, while $|y\rangle$ is the **target qubit**.

where $\lambda = \lambda(t) \in [0, 1]$ is a time-depentent parameter. If one defines the H_1 hamiltonian as follow:

$$H_1 = \sum_{x \in \{0,1\}^n} f(x) |x\rangle \langle x| \tag{49}$$

one can use the evolution of the system to obtain the minimum. Indeed, measuring the system in the computational basis at the end of the adiabatic process gives exactly x_0, since $H(\lambda)$ as in Equation 48 has reached its ground state.

Thus, the usage of adiabatic computing allows theoretically to solve optimization problems in a much faster way as compared to classical algorithms. Nevertheless, the computational complexity depends on the gap between the ground state and the first excited state, allowing at best a *quadratic speedup* for most algorithms, compared to their classical counterpart. It remains open whether quantum adiabatic computation can give an exponential speedup over classical algorithms or if a classical algorithm can simulate accurately enough a quantum adiabatic problem, rendering the usage of dedicated system unnecessary.[19]

As for its applications, it has been demonstrated that quantum adiabatic processes can run not only on specific hardware, but on any quantum computer. [20] As for Machine Learning applications, since many of its algorithms employ some sort of function optimization, its usage can be of great interest.

4.4 Quantum Parallelism

As stated previously, it can be shown that, while a single qubit could theoretically contain an infinite amount of information, realistically an n qubit system can represent at most n classical bits: there would appear that there is no real benefit in using quantum computation. Deutsch and Jozsa [21] showed that, by constructing an appropriate sequence of quantum gates, one can leverage some quantum properties and obtain the so-called **computation by quantum parallelism**: the process of measuring the same system with just one function, but obtain all possible results.

While a simplification for a 2-qubit system, consider the function $f : \{0, 1\} \mapsto \{0, 1\}$ and suppose that there is a 2-qubit system $|x, y\rangle$, where $x, y \in \{0, 1\}$. Suppose than exists a unitary transformation, U_f, that can transform the state $|x, y\rangle$, called the **data register**, into a new state $|x, y + f(x)\rangle$, called the **target register**. In particular, given $y = 0$, the transformation obtains $|x, f(x)\rangle$. The objective is to evaluate $f \; \forall x \in \{0, 1\}$, that is both $f(0)$ and $f(1)$. Classically, one would have to perform the operation two times, on two different bits, in order to get both results. By using a Hadamard transformation, as in Equation 10, on $|x\rangle$, a “new" data register can be obtained, which will *always* be, given either $x = 0$ or $x = 1$:

$$\frac{1}{\sqrt{2}} (|0\rangle \pm |1\rangle)$$

Notice that the Hadamard gate is applied only to the first qubit, since the second one is considered to be $|0\rangle$. In order to be more precise, the new register state would be:

$$\frac{1}{\sqrt{2}} (|0, 0\rangle \pm |1, 0\rangle)$$

Thus, applying the function U_f on this new register will give:

$$\frac{1}{\sqrt{2}} (|0, f(0)\rangle \pm |1, f(1)\rangle) \tag{50}$$

This result is obtained running the function U_f just one time, but gives both results. In Figure 6 is shown the quantum circuit for the 2-qubit system just described.

This pattern can be demostrated to generalize in an n-qubit system, which produces a superposition of 2^n states by using n gates.

Obviously, this analysis does not consider the difficulty, which shall not be described here, of exploiting this phenomenon without destroying the superposition state. Indeed, from a practical point of view, its implementation is still disputed. [22] [23]

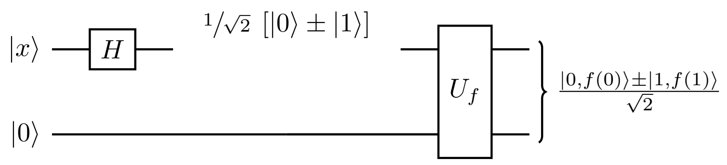

$|x\rangle$
H
$1/\sqrt{2}\ [|0\rangle \pm |1\rangle]$
$|0\rangle$
U_f
$\frac{|0,f(0)\rangle \pm |1,f(1)\rangle}{\sqrt{2}}$

Figure 6: The 2-qubit quantum parallelism operation, also called the **Hadamard transform**. Notice that the value of x does not matter, since the Hadamard gate moves it to the superposition state.

5 Quantum Computation

Inside this section, some basic quantum routines are illustrated. The importance of these routine in the field of quantum machine learning cannot be understated: any ML learning algorithms relies on algebraic operations such as vectors dot product, tensor products, matrix multiplications and addition, etc. It is then essential to implement efficient quantum equivalents of these operation, in order to handle data in the form of quantum states.

Other routines, such as Grover's Search, are not directly related to algebraic operations, but are equally frequent in most machine learning algorithm. These quantum *transpositions* of routines not only provide an efficient way to manipulate data in quantum form, but, in doing so, they often offer a rather significant speedup with respect to their classical counterpart. Illustrating how these speedups arise can be very useful to comprehend the logic behind quantum machine learning and quantum computing in general.

5.1 Quantum Dot Product

This routine allows to estimate the inner product of two complex vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ [24, 25]. First, it is necessary to embed the two vectors in quantum state $|x\rangle, |y\rangle$. This operation goes under the name of **quantum embedding**; many approaches to this operation have been proposed [26, 27], however, a discussion of the perks and drawbacks of each of these is out of the scope of this document. A more general and intuitive explanation of the need of this procedure is given in Section 5.3.1.

For this particular algorithm, it is assumed that:

$$|\mathbf{x}\rangle = \frac{1}{|\mathbf{x}|}\begin{pmatrix}\mathbf{x}\\ 0^n\\ 0^n\end{pmatrix} \qquad\qquad |\mathbf{y}\rangle = \frac{1}{|\mathbf{y}|}\begin{pmatrix}0^n\\ 0^n\\ \mathbf{y}\end{pmatrix} \tag{51}$$

It can be show that these states can be prepared if access to a QRAM is provided (see [28]). Now, the following state is prepared:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|x\rangle + |1\rangle|y\rangle) \tag{52}$$

Applying the Hadamard operator on the first cubit yields:

$$H|\psi\rangle = \frac{1}{2}\Big(|0\rangle(|x\rangle + |y\rangle) + |1\rangle(|x\rangle - |y\rangle)\Big) \tag{53}$$

It can be shown that the probability of measuring $|0\rangle$ is equal to $\frac{1}{2}(1 + \mathrm{Re}(\mathbf{x}\cdot\mathbf{y}))$. Now, applying a phase shift gate in the form:

$$P_{3\pi/2} = \begin{pmatrix}1 & 0\\ 0 & e^{i3\pi/2}\end{pmatrix} \tag{54}$$

to the state $|\psi\rangle$, and then the Hadamard transform to the first qubit, one can obtain:

$$\frac{1}{\sqrt{2}}(H|0\rangle|x\rangle - iH|1\rangle|y\rangle) \tag{55}$$

The probability of measuring $|0\rangle$ is now equal to: $\frac{1}{2}(1 + \mathrm{Im}(\mathbf{x}\cdot\mathbf{y}))$.

This procedure is repeated a constant number of times to get an estimate of the real and imaginary part of the inner product in question, by tracking the ratio of $|0\rangle$ outcomes to the total number of trials. This can be modelled by a binomial distribution specified by probability p, with numbers of trials m. The variance of the estimate of p is then $p(1-p)/m$. One is then able to compute an estimate of the product $\mathbf{x}\cdot\mathbf{y}$ with desired precision, simply by increasing the number of trials m.

5.2 Matrix Addition

In this section, a simple routine for matrix addition on quantum machines is presented. [24]. Many other linear algebra routines are currently available, e.g.matrix multiplication, Kroenecker sum, tensor product; however, it is easy to see that each of these routines relies on the same principles.

By understanding the principle behind quantum matrix addition, one should be able to grasp any other operation easily. The main idea behind these algorithms is the fact that any **Hermitian** matrix can be interpreted as a Hamiltonian operator, therefore Hamiltonian simulation techniques can be used to perform matrix operations.

So, without loss of generality, it's possible to assume two square matrices $A_1, A_2 \in \mathbb{C}^{N\times N}$ (any non square matrix can be padded by zeros). These two matrices need to be *embedded* inside Hermitian matrices. This can be easily achieved by defining:

$$X_3(A) = \begin{pmatrix} 0_{n\times n} & A \\ \overline{A} & 0_{n\times n} \end{pmatrix} \tag{56}$$

Where $\overline{A}$ is defined by: $A^*_{ij} = \overline{A_{ji}}$, where * denotes the conjugation. It's easy to see that $X_3(A)$ is hermitian by construction.

Now, given access to $e^{iX_3(A_1)t/n}$ and $e^{iX_3(A_2)t/n}$, it's possible to obtain an approximation of $e^{i(X_3(A_1)+X_3(A_2))t/n}$ with fixed error ε_1 in a total of $n = O(t^2/\varepsilon_1)$. Once an approximation of the exponential sum has been computed, the original sum can be retrieved.

Hamiltonian simulation methods have been widely studied and many approaches have been proposed over the years (See [29, 30, 31]) with increasing efficiency. Algebraic routines, for this reason, can rely on solid and widely studied techniques in their implementation.

5.3 Quantum Algorithms

5.3.1 Loading classical data, QRAM

Classical data must be inputted before being processed on a quantum computer. Now, it might be instructive to think that, for example, modern classical computers encode a floating point number in 32-bit, while the famous Google's quantum supremacy experiment used a total of 53 qubits [32]. This implies that modern state of the art quantum computers would not be able to perform a single floating point sum. In other words, *qubits are expensive*. There are, on the other hand, several ways to encode information in a qubit much more cheaply.

As shown in previous sections, a relative phase θ can be bijectively encoded in a qubit, for example by defining:

$$|\theta\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle) \tag{57}$$

This type of encoding allows for a much more efficient representation of data inside a qubit, and it's at the heart of embedding data in quantum states [27, 26].

An alternative example is the so-called **amplitude embedding**, where the entries of a complex vector $\mathbf{x} \in \mathbb{C}^n$ can be encoded in the amplitudes of a multi-qubit quantum state.

$$|\mathbf{x}\rangle = \tilde{K}\Big(\sum_i x_i |i\rangle\Big) \tag{58}$$

In this way, a vector of n entries can be encoded in a total on $\log_2 n$ qubits. Performing these kind of operation, though, is very expensive, and might cause severe operational slowdowns. This obstacle is known as the **input problem**, and significantly undermines the applicability of most big data quantum applications with the current technology.

Likewise, the equivalent **output problem**, where classical quantity need to be estimated from quantum states, brings the same amount of operational troubles. This problem might be partly be addressed by using a **quantum random access memory** (See [28]), but the expensiveness of such solution makes this approach unfeasible with the current hardware technology.

Ongoing work is needed to optimize quantum algorithms and to understand the sort of quantum computer needed to provide useful quantum alternatives to classical machine learning. Inside this work, the difficulties brought by the input and output problem will not be dealt with, and embedding and output phases will be treated as assumptions. It is important, in the end, to remark that these type of problems are not present when data is **natively quantum**, which opens space for the landscape of **quantum (enhanced) machine learning on quantum data** [33].

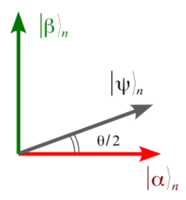


Figure 7: Initial State

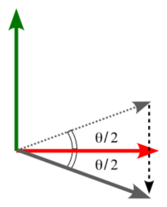


Figure 8: Oracle Fuction

5.3.2 Grover

The famous **Grover algorithm** was developed by Lov Grover in 1996 and it addresses the problem of finding an element in an unordered list. It uses qubits in superposition to make computation much faster than traditional algorithms on classical computers. [34]

Unstructured search is a common problem in computer science and, as seen in Table 7, for electronic computers, the running time complexity of finding an item in an unsorted list is $O(n)$. On the other hand, with Grover algorithm, an unstructured search can become quadratically faster, i.e. $O(\sqrt{n})$ [3].

Assuming a list $|0\rangle_n$ of n qubits, the **first step** of the **Grover algorithm** is to take this list and apply the Hadamart transformation to put qubits in superposition, as shown in Figure 7.

$$H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle_n \equiv |\psi\rangle_n \tag{59}$$

The **second step** is to use a **Oracle Function** $f(x)$, which takes in input qubits in superposition and returns 1 whether the input is the desired one, or 0 otherwise. So in the case where $f(x) = 1$, it flips the amplitude of the item, as shown in Figure 8, leaving the other qubits unchanged. This is known as **conditional phase shift**, which can be written as:

$$|x\rangle_n \xrightarrow{\mathcal{O}} (-1)^{f(x)}|x\rangle \tag{60}$$

Another way to represent Equation 60 can be:

$$2|0\rangle_n\langle 0| - \hat{\mathbb{I}} \tag{61}$$

In the third **third step**, a Hadamart transformation is performed again. Using this, one can define the **Grover operator** $\hat{G}$ as follow:

$$\hat{G} = \left[(2|\psi\rangle_n\langle\psi| - \hat{\mathbb{I}}) \otimes \hat{\mathbb{I}}\right]\hat{\mathcal{O}} \tag{62}$$

As shown in Figure 7, the $|\beta\rangle_n$ vector is the solution. The third step rotates $|\psi\rangle_n$ counterclockwise by an angle θ (Figure 9). So, from a geometric point of view, by repeating the second and third steps t times, the result will get closer and closer to $|\beta\rangle_n$.

As a matter of fact, it can be demonstrated that there is a best number $\mathcal{R}$ of Grover iterations, which bring the initial state $|\psi\rangle_n$ as nearer as possible to the state $|\beta\rangle_n$: further iterations would drive the state away form $|\beta\rangle_n$. The best number of iterations is obtained geometrically as stated in [3], and it is shown that $\mathcal{R} \sim O\left(\sqrt{N}\right)$. So as said early, it is quadratically faster than the classic algorithm that is $O(N)$.

5.3.3 HHL

The HHL algorithm is a fundamental and easy to understand routine, underpinning many quantum machine learning problems. The aim of the algorithm is to resolve the following problem: given a matrix $A \in \mathbb{C}^{N\times N}$ and a vector $\mathbf{b} \in \mathbb{C}^N$, find a vector $\mathbf{x} \in C^N$ such that $A\mathbf{x} = \mathbf{b}$. In other words, the objective is to find a state $|\mathbf{x}\rangle$ which minimizes:

$$|A\,|\mathbf{x}\rangle - |\mathbf{b}\rangle\,| \tag{63}$$

The algorithm is based on the following principles:

- The matrix A can be assumed to be hermitian without loss of generality, because it is possible to expand the space to make this true. For example, given a non-hermitian matrix A, one is able to define:

$$A_h = \begin{pmatrix} 0_{n\times n} & A \\ \overline{A} & 0_{n\times n} \end{pmatrix} \tag{64}$$

  Where $\overline{A}$ is defined as $\overline{A_{ij}} = A^*_{ji}$ where * denotes the conjugation. It is easy to check that A_h is hermitian.
- It is necessary to prepare the state $|\mathbf{b}\rangle$, starting from the classical vector $\mathbf{b}$. Again, as described in Section 5.1, this operation is possible if access to a QRAM is given [28].
- The eigenvalues of A can be computed; this process is possible thanks to a very common subroutine called **quantum phase estimation** (See [35]).
- $|\mathbf{b}\rangle$ can be expressed as $|\mathbf{b}\rangle = \sum_j b_j\,|E_j\rangle$, where $|E_j\rangle$ is an eigenvector of A with eigenvalue $\lambda_j \geq \Lambda$.

So, by applying phase estimation under A, and by applying a phase rotation of angle $\arcsin\left(\Lambda/\lambda_n\right)$ on an ancillary qubit and then uncomputing the phase estimation, it is possible to obtain:

$$\sum_n b_n\,|E_n\rangle \left(\frac{\Lambda}{\lambda_n}\,|1\rangle + \sqrt{1 - \frac{\Lambda^2}{\lambda_n^2}}\,|0\rangle\right) \tag{65}$$

From this state, a measurement is performed. If $|1\rangle$ is observed, the system will be in a state proportional to:

$$\sum_n \frac{b_n}{\lambda_n}\,|E_n\rangle = A^{-1}\,|\mathbf{b}\rangle = |\mathbf{x}\rangle \tag{66}$$

Then, a mechanical observable M is measured, and $F(\mathbf{x}) = \langle\mathbf{x}|M|\mathbf{x}\rangle$ is retrieved.

The best method known on classical computer to perform an equivalent operation requires $O(N\log N)$ steps, while the HHL algorithm takes $O((\log N)^2)$ steps.

There are some caveats to the HHL algorithm. First of all, finding the whole vector $\mathbf{x}$ requires $O(N)$ steps, one for each of his components. In order to avoid this problem, many generalizations of the HHL algorithm have been proposed, some of which allow the output to have a smaller dimension that the input. In doing so, they only provide features of the data, such as moments or the mean value. Furthermore, current cost estimates for the algorithm for practical problems are prohibitive. Recent works have proposed alternative ways to resolve the same problem. Despite being aimed at the same goal, these implementations differ greatly from the HHL algorithm, since they are inspired by adiabatic quantum computing [36]. While not providing remarkable efficiency gains, the main advantage of these proposals is the fact that no phase estimation or variable-time amplitude amplification is used, thus reducing the number of ancillary qubits greatly.

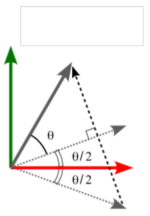


Figure 9: After the operator

5.3.4 qPCA

The objective of classical Principal Component Analysis is to find, given a set of *inputs* $\boldsymbol{X} \in m \times \mathbb{R}^n$, a lower-dimension representation, $\boldsymbol{C} \in m \times \mathbb{R}^l$, where $l < n$ and m is the number of "observations" given, losing as little information as possible. [37] Generally, this can be done by using an encoding function, $f : \mathbb{R}^n \rightarrow \mathbb{R}^l$ such that $f(\boldsymbol{X}_i) = \boldsymbol{C}_i,\ \forall i \in [1, m]$, and a decoding function such that $\boldsymbol{X}_i \approx g(f(\boldsymbol{X}_i))$. In **Principal Component Analysis** this is achieved constructing a matrix $\boldsymbol{D}$ such that $\boldsymbol{X}_i = \boldsymbol{D}\boldsymbol{C}_i$, with $\boldsymbol{D} \in \mathbb{R}^{n \times l}$, whose elements are given by the l eigenvectors corresponding to the largest eigenvalues of $\boldsymbol{X}$.

From a quantistic point of view, the **eigendecomposition** of a matrix, as just described to obtain the PCA matrix, corresponds to **simulating a Hamiltonian**. Given a Hamiltonian $H = \sum_{i=1}^{m} H_i$, the task is to simulate its exponentiation e^{iHt}, by applying a sequence of the type $e^{-iH_i t'}$, requiring at each step an error less then a given threshold ε. This idea was proposed in 2014 LLoyd et al. [38] based on the **Baker-Campbell-Hausdorff** formula, which allows to obtain the following approximation:

$$e^{iHt} \approx \left(e^{iH_1 t/n} \dots e^{iH_m t/n}\right)^n \tag{67}$$

It can be shown that, if each H_i acts only locally, i.e. if the matrix H is non-sparse, the right hand side of Equation 67 is much more efficient, with a possibly linear time of computation.

Unfortunately, a Hamiltonian describing the interaction of n qubits is usually **sparse**, since it has at most a constant number of non-zero entries. In order to avoid said problem, the solution is to select an arbitrary k integer such that the simulation will require $O(\log^* (n) t^{1+1/2k})$ access to the matrix entries of H, where $log^* n$ is the number of times the logarithm function is applied recursively before the result is ≤ 1. See Berry et al. [39] for more information regarding this theoretical passage, who also showed how sublinear time is not feasible.

Even when using the described method to access efficiently the matrix entries, in order to simulate the non-sparse Hamiltonian the best method, as described by Lloyd et al [38], is to use the higher order Trotter-Suzuki expansion, but by using a density matrix ρ as a Hamiltonian on another density matrix σ. It can be shown that this allows for a speedup from $O(n \log n)$ to $O(\log n)$, if used in conjunction with the **swap operator** S, as defined in Section 2. In order to get the final result, i.e. the eigenvectors, out of ρ, one can use a **phase estimation algorithm**[40] on $e^{i\rho t}$, which is the substitute for e^{iHt}. This leads to the following final result:

$$\sum_{i=1}^{n} r_i \left|\chi_i\right\rangle \left\langle\chi_i\right| \oplus \left|\hat{r}_i\right\rangle \left\langle\hat{r}_i\right| \tag{68}$$

where $|\chi_i\rangle$ are the **eigenvectors** of ρ, and thus the result desired, with $\hat{r}_i$ being the estimates of the corresponding eigenvalues.

As a recap, what is done is the follow:

- Define a Hamiltonian for the n-qubit system onto which to execute the qPCA algorithm.
- Construct a density matrix ρ as a substitute for H from the n-qubit system.
- Use the higher order Trotter-Suzuki expansion to simulate $e^{i\rho t} \approx \left(e^{i\rho_1 t/n} \dots e^{i\rho_m t/n}\right)^n$.
- Retrieve from the simulation of $e^{i\rho t}$ the desired eigenvectors, and an estimation of the eigenvalues, using a *phase estimation algorithm*.

It should be noted that this algorithm has been theoretically proposed in 2014 and, to our knowledge, there have been no real-world applications of it. Indeed, while some early quantum computing is possible today [41], many implementations are years away.

6 Quantum Machine Learning

6.1 Introduction[4]

Both machine learning and quantum computing are expected to play a role in how society deals with information in the future, so it becomes natural to ask how they could be combined [42].

Quantum machine learning is, in itself, a broad discipline. It investigates approaches and techniques that use synergies between machine learning and quantum information. For example, the study of classical machine learning for the

[4]This section is based on [42]

analysis of quantum measurements is a very broad research area. The main focus of this project, though, is to outline the possibilities given by the realm of quantum computing in the extremely wide landscape of machine learning. In other words, the main question is: how can quantum computing enhance the capabilities of intelligent data mining? Can quantum computers help to solve problem faster, can they learn from fewer data, or are they able to learn with a higher level of noise? While there are many ways to answer this questions, it is also important to outline the problems and obstacles in the development and applications of these techniques.

6.1.1 The rise of quantum machine learning

In recent years, there has been a growing body of literature with the objective of combining the disciplines of quantum information processing and machine learning.

Perhaps, the earliest notions were investigations into quantum models of neural networks starting in 1995. The hope of this works, mostly biologically inspired, was to find quantum-rooted explanations of how the brain works. In the early 2000s the question of statistical learning in quantum setting was discovered, but received only a small amount of attention. During these years, some workshops were organized and sporadic publications on the topic started to appear. The first monograph on the subject was published in 2014 by Peter Wittek with the title *Quantum machine learning - What quantum computing means to data mining* [12]. From that year onwards, interest in the topic increased and produced a rapidly growing body of literature covering numerous areas on the subject. Furthermore, combining a dynamic multi-billion dollar market with the still young and potentially profitable technology of quantum computing helped sparking a lot of interest in the industry.

6.1.2 Overview of the possible approaches

As previously mentioned, quantum machine learning is a broad research area. Aimeur, Brassard and Gambs introduced, in 2006 [43] a useful typology that distinguishes four approaches of combining machine learning and quantum computing, depending on how the data is generated and processed (Figure 10).

- The data can be generated by a quantum (Q) or classical system (C).
- The processing device can be quantum (Q) or classical (C)

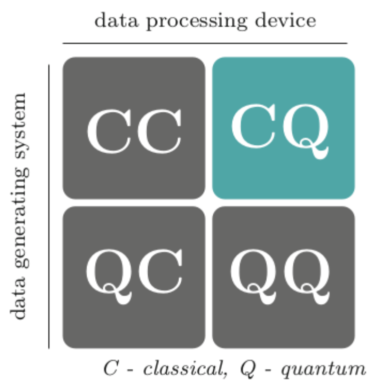


Figure 10: Four different approaches to quantum machine learning

The four areas defined by Figure 10 are:

- The case of classical data being processed classically (CC) is the conventional approach to machine learning. In the context of quantum machine learning it refers to machine learning based on methods borrowed from quantum research. An example is the application of tensor networks, (See [44]) which were originally developed for quantum many-body-systems, to network training. Also, many *quantum inspired* machine learning models exist, with various degrees of foundation on rigorous quantum theory. Wittek, in his monograph [12], describes this area as **quantum-like machine learning**, referring to all those models and optimization techniques inspired by quantum phenomena, such as entanglement and superposition (See [45], [46]).
- The case QC investigates how machine learning can help with quantum computing. For example, many machine learning models could be useful to determine the internal state of a quantum computer with as few measurements as possible [47].

- The CQ setting uses quantum computing to process classical datasets, and constitutes the main focus of this work. The data can be constituted of any kind of observations from classical systems, such as text, images, time-series. So, the main goal of the CQ approach is to design quantum algorithms for data mining. This could be done by translating classical models into the language of quantum algorithms, or, alternatively, to genuine new models derived from the principles of quantum computing. For both approaches, some notable examples will be illustrated.
- The last approach, QQ, looks at quantum data processed by quantum computers. This could mean two things: the data could be derived from measuring a quantum system in a physical experiment and feeding the value into a separate quantum device. On the other hand, a quantum computer might be used to simulate the dynamics of a quantum system. In this case, the input of a quantum ML model might be the state itself, in order to give the computer rapid access to all the information. Despite being very interesting, the QQ area might present some shady issues to unfold: does learning from quantum data produce different results from classical data? How can generation and analysis be effectively combined? This questions are still object of research and will not be treated here.

6.1.3 Quantum computing for machine learning

When designing quantum machine learning algorithms in the CQ setting, the possible approaches aim to combine the following strategies:

- The **translational approach** has the obective of translating classical models in the language of quantum mechanics, in the hope of harvesting algorithmic speedups. In other words, the goal is to reproduce the result of a given model, while transferring the whole or part of the computation to a quantum device, and keeping the resource cost as low as possible. In this case, learning does not pose a new problem here. Most of the algorithms, in fact, rely on speedups obtainable by common algebraic routines, such as matrix inversion, quantum search, resolving linear system of equations, etc. Quantum machine learning, in conclusion, is largely an application of quantum computing in this case.
- The **exploratory approach**, on the other hand, does not rely on a digital and universal quantum computer to implement quantum algorithm, but may use any system obeying the laws of quantum mechanics to derive and train a model that is suitable to learn data. The aim is not only to achieve consistent speedups, but to contribute to innovative methods to the machine learning community.

Aside from the approach used, any quantum machine learning model has a few aspects that are worth to point out:

- Data encoding is a crucial step of quantum machine learning with classical data, and defines the working principle of the algorithm. It is also, most of the times, the bottleneck of the algorithm.
- Many QML algorithms impose certain requirements on preprocessing, for example unitary input vectors.
- The result of any machine learning model is a measurement, since, as said, it is the only way to have an interaction with a quantum system.
- Quantum machine learning are often inspired by classical models.
- Quantum machine learning models must often be modified in order to be adapted to the functioning of quantum computers. For example, in a quantum setting, complex numbers are usually the natural choice of coefficients and the squared distance better suits the quantum formalism.

Another important aspect to point out is **how quantum computing can actually assist machine learning**. Until now, in fact, when describing quantum algorithms such as the HHL or the Grover search, the main focus has always been the so called *quantum speed-up*, a known aspect of quantum computing.

The research on quantum machine learning, however, does not focus solely on this aspect. There is, in particular, a total of four dimensions that deserve to be described when speaking about the advantages of QML over the classical one: *computational complexity, sample complexity, robustness to noise* and *model complexity*.

- *Computational complexity*, also referred to as *runtime complexity*, has been explored in the previous sections and, as previously said, is the most common form of advantage associated to the potential of QML and quantum computing in general. Of course, QML inherited this focus on runtime speedups from quantum computing, where algorithms such as Grover's Search have proven run-time bounds that suggest their higher efficiency in terms of asymptotic behaviour. This is the type of merit mainly looked after in the *translational approach*, and QML models that rely on this type of advantage (even simply by leveraging on one of these classic QC algorithms) might, in other words **provide a computational speedup over their classical counterparts**.

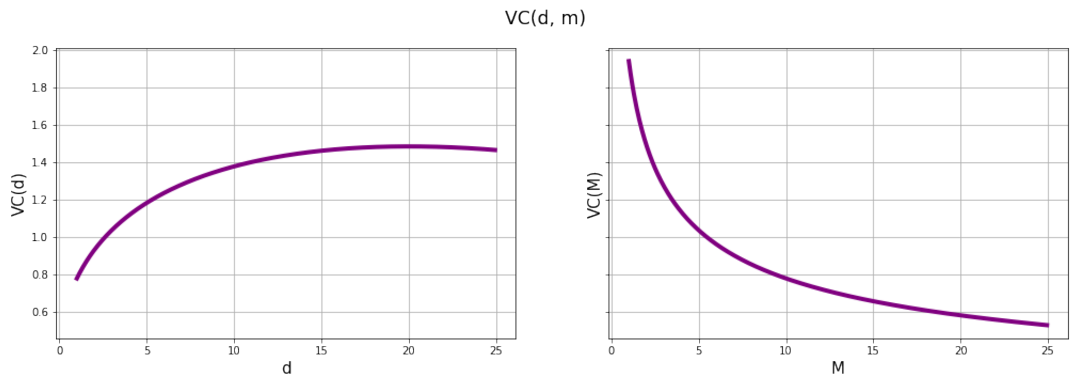


Figure 11: The Vapknik-Chervonekis term represented as a function of M and d separately. Default values used are $M = 10$, $\delta = 0.5$, $d = 1$. It is clear that a higher number of instances reduces the value of the upper bound of the generalization error, while, at constant M, a higher complexity (higher d) lifts the bound.

- *Sample complexity*, on the other hand, refers to the number of samples needed to generalise from data. In particular, the sample complexity of a machine learning algorithm refers to the number of samples that are required to learn a *concept* from a given *concept class*. In other words, how many instances are required to learn an acceptable approximation of the "law" underlying the data. Under this point of view, it has been shown that classical and quantum sample complexity are polinomially equivalent or, as stated in [48]:

  > [F]or any learning problem, if there is a quantum learning algorithm which uses polynomially many [samples] then there must also exist a classical learning algorithm which uses polinomially many [samples].

  In other words, no advantages are expected from quantum computing regarding the sample complexity.

- *Robustness to noise*. The term *noise* refer to corrupted instances. For example, in binary classification problem, the value of the class attribute y might be flipped with a probability of μ. It has been shown that increasing μ can make the problem classically unlearnable, while the quantum model is still able to learn by simply using more examples [49, 50]. Further evidence of this aspect is still under research, but these observation might be evidence that robustness against noise could be a fruitful avenue for further research.

- The term *model complexity* is a wide concept that refers to the flexibility, capacity, richness or expressive power of a model. Complex models offer a larger space of possible trained models but are much more prone to overfitting. Asking how quantum models can offer advantages in term of model complexity has two dimensions. First of all, the generalization error ε_{gen} of a model is bounded by Vapnik's term with probability δ:

  $$\varepsilon_{gen} \leq \varepsilon_{emp} + VC(M, d, \delta) \tag{69}$$

  Where ε_{emp} is the empirical error on the training set and VC is an expression that depends on the number of instances M, and the Vapknik-Chervonekis-dimension d (higher for more complex models). When training a machine learning model, one is interested to the ”slimmest" model that can effectively capture the pattern in the data.

  Starting from the equation (69), analyzing the VC term for quantum models could reveal some interesting comparisons to equivalent classes of classical models.

  On another side, quantum models could prove to be a useful ansatz in capturing patterns in certain datasets. An intuitive motivation of this aspect could be the following: machine learning models can recognize patterns inside the data. Once a pattern has been determined, a model is also able to produce predictions (following that same pattern), based on the data that it is fed into. Models can reproduce pattern that they can recognize, and, inversely, they can recognize pattern that they can reproduce. Recent research [51, 52] shows that quantum computers can sample from probability distributions that are exponentially difficult to sample from classically. If these distributions were to coincide with real world distributions, this would suggest the potential for significant advantage over classical models. One seemingly obvious case for the use of quantum models is when the system producing the data is a quantum system (QQ setting), but other applications are possible, as will be shown later.

6.1.4 A note on hardware - long-term and mid-term devices

The algorithms presented in this work assume, in most cases, the existence of a *universal, large scale, error corrected* quantum computer. *Universal* means that the computer can implement any unitary operation for the quantum system it is based on, and therefore any algorithm. *Large-Scale*, on the other hand, means that a high number of qubits are available. *Error-corrected* means that the results of the operations performed are exactly described by the theoretical equations of quantum theory, i.e. no noise is introduced when a transformation is performed.

With the actual state of technology, this assumption is far from true. Quantum computing is an emerging technology, and the actual generation of hardware is referred to as *noisy intermediate term devices*. So, intermediate term means that not every quantum computer is universal, many of them do not even aim at universality. Moreover, most of universal quantum computers are able to implement efficiently only a little number of gates. Also, the maximum number of qubits available on a real quantum computer is still rather small. As previously said, the famous computer of the Google's quantum supremacy experiment only had a hundred qubits. For this reason, intermediate term devices are also *small-scale*. Thidly, present quantum computers are *noisy*: the gates have a limited fidelity and precision and, furthermore, they are not equipped with mechanisms that allow them to correct errors. Hence, it is only possible to apply a small number of gates before the result of the calculation is too noisy to be useful.

At the current state of technology, the quantum community speaks of *intermediate term algorithms* if an order of 100 qubits and 1000 gates are used. An important research question is, for this reason, what approaches to quantum machine learning are actually possible in noisy, intermediate term devices.

6.1.5 Small and Big data

In section (5.3.1), the *embedding problem* has been briefly treated.

To recap, how the information is encoded inside the quantum states deeply affects the algorithms, both in execution-times and performance. Common approaches to data embedding were later illustrated. It has been also said that, even with technologies such as a QRAM, data embedding algorithms usually constitutes the bottleneck of many quantum machine learning procedures. The bottleneck of data encoding means that, besides a few special cases, quantum machine learning will not offer intermediate term solutions for big data processing.

Although the promise of quantum methods for big data sounds distant, there are plenty of real world application where data collection is expensive or data is limited by nature (biological experiments, specific data from reasoning). If quantum computing can show qualitative advantage, there will indeed be worthwile applications in the area of *small data*.

It is worth to note that some model, which are referred to as *hybrid schemes*, only process a subset of samples from the dataset at a time, thereby lifting the restrictions on the number of data samples. Notable examples of these approaches are the so-called **quantum kernels**, which will be presented in the next sections, and the hybrid gradient descent algorithm.

6.2 The power of data in quantum machine learning

An interesting result regarding the problem of model complexity has been obtained by Huang et al. in [53].

In particular, the work starts from the following common assumption, which has been explained in the previous section:

> If the model leverages a quantum circuit that is hard to sample from classically, then there is potential for a quantum advantage.

The authors then proceed to show that this picture is incomplete when some training data is provided. The provided data can elevate classical models to rival quantum models, even when the quantum circuit generating the data are hard to compute classically.

To support their claim, a flowchart for dissecting quantum advantage is presented, and a each case is simulated. The works focuses on *quantum kernels* and *projected quantum kernels*, since they provide provable guarantees and they are very flexible in the function the can learn. For example, recent advantage in theoretical machine learning show that training neural networks with large hidden layers is equivalent to training an ML model with a particular kernel, known as the neural tangent kernel [54, 55].

The research is focused on a supervised learning task with a collection of N samples $\{(x_i, y_i)\}$, where x_i is the input data and y_i is the label. Also x_i are assumed to be sampled independently from a data distribution $\mathcal{D}$, while the values of $y_i \in \mathbb{R}$ are generated by a quantum model as:

$$y_i = f(x_i) = \langle x_i | U_{QNN}^{\dagger} O U_{QNN}^{\dagger} | x_i \rangle \tag{70}$$

Where $|x_i\rangle$ is prepared using **amplitude encoding**, O is an observable and $U(\theta)_{QNN}$ is a unitary operator. Now, it is possible to show that the expression in 70 is hard to compute classically. However, training a classical model from data to learn this evolution could not be hard, in fact the expression can be written as:

$$\begin{aligned} f(x_i) &= \left(\sum_{k=1}^{p} x_i^{k*} \langle k| \right) U_{QNN}^{\dagger} O U_{QNN}^{\dagger} \left(\sum_{l=1}^{p} x_i^{l} |l\rangle \right) \\ &= \sum_{k=1}^{p} \sum_{l=1}^{p} B_{kl} x_i^{k*} x_i^{l} \end{aligned} \tag{71}$$

which is a quadratic function with p^2 coefficients. With at least $N \sim p^2$ training data, all the coefficients B_{kl} can be fitted and the function $f(x_i)$ predicted accurately.

This example uses an amplitude encoding and assumes to have access to data that trivializes exaclty fitting the model, but is sufficient to make a point. The more interesting case, though, occurs when different data encodings are provided and the number of data N is much inferior to the dimension of the model. As previosly stated, the work relies on *kernel methods*.

6.2.1 Classical - Quantum - Projected Kernels

A given kernel function corresponds to a nonlinear feature mapping $\phi(x)$ that maps x to a possibly infinite-dimensional feature space, such that $k(x_i, x_j) = \phi^{\dagger}(x_i)\phi(x_i)$. The *kernel tricks* allows the implementation of complicated feature maps $\phi(x)$ through the evaluation of relatively simple kernel functions k. In kernel ML based algorithms, the model can be expressed as $h(x) = w^{\dagger}\phi(x)$, where w is a vector in the feature space defined by the kernel. Also, usually a **kernel matrix** is defined as K:

$$K_{ij} = k(x_i, x_j) \tag{72}$$

In quantum mechanics, similarly, a kernel function can be defined using the native geometry of the quantum space $|x\rangle$. For example, a kernel function of the form:

$$k^Q(x_i, x_j) = |\langle x_i | x_j \rangle|^2 = \mathrm{Tr}(\rho(x_i)\rho(x_j)) \tag{73}$$

corresponds to a feature map $\phi(x_i) = \sum_{kl} x_i^{k*} x_i^{l} |k\rangle \otimes |l\rangle$. This particular kernel also has the ability to learn arbitrarily deep quantum neural networks U_{QNN} that measure any observable O.

Quantum kernels, as highlighted, suffer one major problem: when the dimension of the system increases, the dimensionality of the Hilbert space defined by the quantum kernel $\phi^Q(x_i)$ grows exponentially, causing the data to be far from each other and the kernel matrix K^Q to be close to the identity.

To circumvent this setback, a family of **projected quantum kernels** is proposed. These methods work by projecting the quantum states to an approximate classical representation, using the method of *classical shadows* [56]: this procedure allows to predict order M different functions of the state with high success probability with a total of order $\log M$ measurements. Also, the number of measurements is independent of the system size. As a matter of fact, the method of classical shadows allows to have an efficient classical representation of the states that does not depend on the system size. It is a good example of **hybrid method in QML**, along with the natural gradient descent.

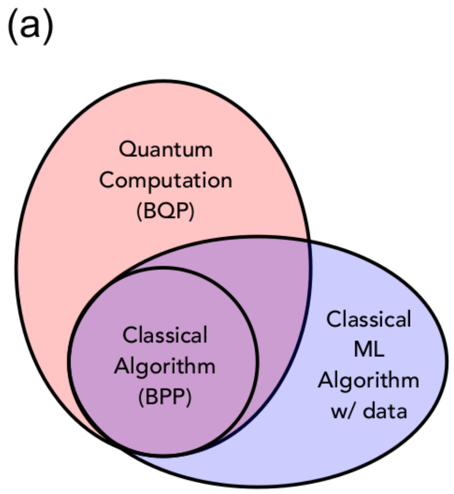
An example of projected quantum kernel is the following:

$$k^{PQ}(x_i, x_j) = \exp\left(-\gamma \sum_{k} ||\rho_k(x_i) - \rho_k(x_j)||_F^2 \right) \tag{74}$$

Where $\rho_k(x_i) = \mathrm{Tr}_{j \neq k}[\rho(x_i)]$ and $||\cdot||_F$ is the Frobenius Norm. Each density matrix can be calculated efficiently with the method of classical shadows.

6.2.2 Testing quantum advantage

As stated previously, the work proposed by *Huang et al.* aims at constructing a general framework for assessing the potential for quantum prediction advantage in a machine learning task. This framework is summarized in Figure 12.


(a)
Quantum Computation (BQP)
Classical Algorithm (BPP)
Classical ML Algorithm w/ data

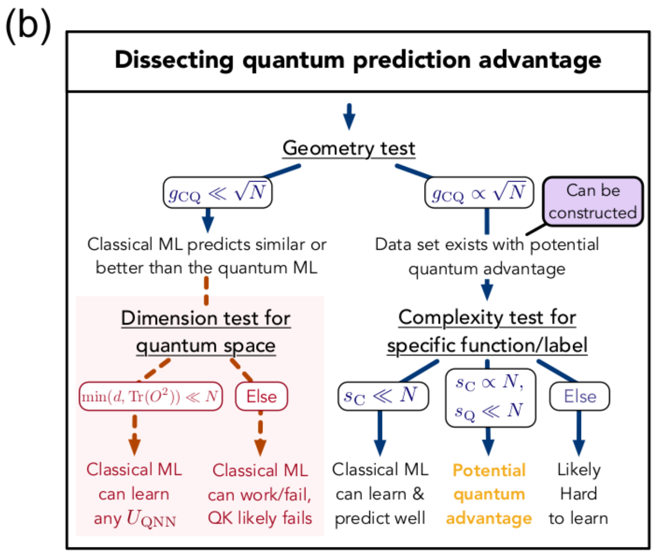

(b)
Dissecting quantum prediction advantage
Geometry test
$g_{CQ} \ll \sqrt{N}$
$g_{CQ} \propto \sqrt{N}$
Can be constructed
Classical ML predicts similar or better than the quantum ML
Data set exists with potential quantum advantage
Dimension test for quantum space
Complexity test for specific function/label
$\min(d, \mathrm{Tr}(O^2)) \ll N$
Else
$s_C \ll N$
$s_C \propto N$, $s_Q \ll N$
Else
Classical ML can learn any U_{QNN}
Classical ML can work/fail, QK likely fails
Classical ML can learn & predict well
Potential quantum advantage
Likely Hard to learn

Figure 12: From [53]. **(a)** shows the separation between between problem complexities that are created by the addition of data to a problem. Classical learning problems that can learn from data define a complexity class beyond classical computation (BPP), but is is still expected that quantum computation can efficiently solve problems that classical ML with data cannot. **(b)** Flowchart developed for understanding the potential quantum prediction advatage. Starting from N samples of data from a potentially infinite depth QNN are provided as input along with classical and quantum methods with associated kernels. A first geometric quantity g_{CQ} is evaluated from the data. This quantity, on it's own, measures the possibility of advantageous quantum/classical prediction separation without knowledge of the function to learn. To consider the function provided, a label/function specific test may be run using the model complexities defined s_C and s_Q

The model complexity coefficient s_K – As stated previosly, the framework is formulated to predict a quantum model $f(x)$ defined as in Equation 71. Supposing that N training examples $\{x_i, y_i = (x_i)\}$ are available and that a classical ML model $h(x) = w^\dagger \phi(x)$ is trained from this data using the kernel $k(x_i, x_j) = K_{ij} = \phi(x_i)^\dagger \phi(x_j)$, then the following error bound is valid:

$$\mathbb{E}_{x \sim \mathcal{D}} |h(x) - f(x)| \leq c \sqrt{\frac{s_K}{N}} \tag{75}$$

Where c is a constant and the coefficient s_K can be shown to be:

$$s_K = \sum_{i=1, j=1}^{N} (K^{-1})_{ij} \, \mathrm{Tr}\left(U_{QNN}^\dagger O U_{QNN} \rho(x_i) \right) \mathrm{Tr}\left(U_{QNN}^\dagger O U_{QNN} \rho(x_j) \right) \tag{76}$$

Which is equal to the **model complexity** of the trained function $h(x)$.[5] A smaller value of s_K implies better generalization to the new data sampled from $\mathcal{D}$, and thus less overfitting. Intuitively, s_K measures whether the closeness between x_i, x_j defined by $k(x_i, x_j)$ matches well with $f(x_i) \cdot f(x_j)$, recalling that a higher value of the kernel function implies closeness between the points. One problem might be the invertibility of K: if K is ill-conditioned, the coeffient s_K might be large, but this problem can be circumvented by applying regularization to the model (regularization method's original purpose is exactly this). First of all, it is crucial to highlight the dependency $\sim \frac{1}{\sqrt{N}}$ in Equation 75, reflecting *the power of data* to improve prediction performance. Second, given a dataset, it is possible to see if s_k is small with respect to N after training an ML model. In that case, the quantum model $f(x)$ can be predicted accurately even if $f(x)$ is hard to compute classically for any given x.

The geometric distance coefficients g_{12} and g_{CQ} – Given two different models, defined by K_1 and K_2, the potential advantage on the same data depends on the separation between s_{K_1} and s_{K_2}. This separation can be characterized by defining an asymmetric geometric difference that depends on the dataset, but is indipendent of the function values. This

[5]It is useful to remember that Equation 71 is equal to $\mathrm{Tr}\left(U_{QNN}^\dagger O U_{QNN} \rho(x_i) \right)$ for the properties of density matrices

quantity is defined by:

$$g_{12} = g(K_1 \| K_2) = \sqrt{\left\| \sqrt{K_2}(K_1)^{-1}\sqrt{K_2} \right\|_\infty} \tag{77}$$

Where $\| \cdot \|_\infty$ is the spectral norm of the resulting matrix. It is possible to show that:

$$s_{K_1} \leq g_{12}^2 s_{K_2} \tag{78}$$

Which implies that the prediction error bound for model 1 given in 75 is:

$$c\sqrt{\frac{s_{K_1}}{N}} \leq c g_{12}\sqrt{\frac{s_{K_2}}{N}} \tag{79}$$

Now, it is possible to consider the geometric difference $g_{CQ} = g(K_C \| K_Q)$ between a classic ML model with kernel K_C and a quantum ML model with kernel K_Q. If g_{CQ} is small, then $s_C \leq g_{CQ}^2 s_Q$. In other words, the classical model will always have a similar or better model complexity s_K compared to the quantum model. This implies that the prediction performance for the classical model will be competitive or better than the quantum model. This aspect is represented in the flowchart in Figure 12. On the other hand, if g_{CQ} is large, then it is possible to show that, using a dataset with $s_C = g_{CQ}^2 s_Q$, the quantum model exhibits superior prediction performance. A method to construct such dataset is given in [53].

The quantum model complexity s_K^Q – From the model complexity of the quantum data is possible to show that:

$$s_Q = \min(d, \mathrm{Tr}(O^2)) \tag{80}$$

Where $d = \mathrm{rank}(K^Q) \leq N$ and O is the observable chosen for the model. In practice, quantum kernel methods can learn any U_{QNN} when the dimension of the training set space d or $\mathrm{Tr}(O^2)$ are much lower the number of instances. Most observables, such as the pauli operators, will have exponentially large traces, so the central quantity is the dimension d.

The quantities presented and their relations are sufficient to explain all the possible cases in Figure 12(b). It is then important to remark that the only regime in which a quantum advantage is possible is for:

$$g_{CQ} \sim \sqrt{N}$$
$$s_C \sim N$$
$$s_Q \ll N$$

6.2.3 Numerical results

The framework is then tested up to a total of 30 qubits on engineered datasets, with respect to the prediction performance. The simulations are, to this day, the largest combined simulation and analysis in digital quantum machine learning. Results are reported in Figure 13.

First of all, it is easy to highlight the main problem affecting the original quantum kernels: the dimension d of the systems grows rather quickly with n, and the geometric difference becomes small, leading to the rather fast decrease in performance shown. On the contrary, projected quantum kernels are able to maintain a stable dimension d even when the system size grows. Second, it is important to remark the power of data in any ML setting, since by passing from $N = 100$ to $N = 600$ all the methods are improved. Thirdly, the predicted relationship between geometrical difference and performance holds for the simulation done. As it is easy to see, when g_{CQ} is small, the models have similar prediction accuracy, while a greater value of g causes the quantum model to perform better by a 20% term. It is possible to conjecture that the size of the margin implies this separation may even persist under moderate amounts of noise in a quantum device.

The work lays the foundation for understanding opportunities for quantum advantage in a learning setting, other than giving important insight on the problem of *model complexity* in quantum machine learning. As of today, this is the first empirical demostration of a large separation between classical and quantum machine learning models.

6.3 Quantum K-means

The K-means algorithm is probably the most well known unsupervised machine learning algorithm.[37] It is based on the idea that points in a space $C \subset M \times \mathbb{R}^N$ can be assigned a label (*cluster*) depending on the vicinity to some

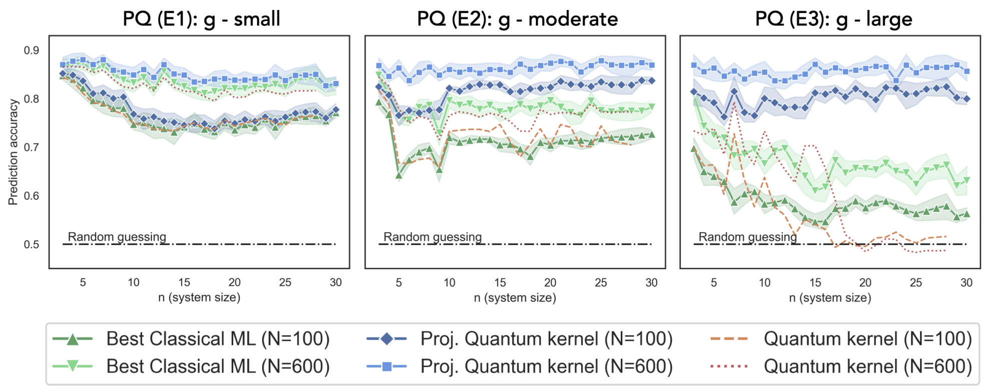


Figure 13: From [53]. A label function is engineered to match th geometric difference $g(C||PQ)$ between projected quantum kernels and classical approaches, demostrating a significant gap between quantum and the best classical model.

specified points, called **centroids** in this case. The algorithm starts from some initialized, be it randomly or otherwise, k *centroids*, called *K-means*; it then assigns to each point in the space a label according to its closest centroid, evaluated usually via *Euclidean distance*; and finally it recalculates the centroids as the *mean* of all points in a given cluster. Repeating the process iteratively, and by aiming at minimizing the distances inside each cluster, one can find a decent clustering solution for many problems, especially if the data tends to form “globural" clusters. It can be shown that, from a computational complexity, the time required at each step is $O(M^2N)$, where N is the number of dimensions, M is the number of given observations.

The most established algorithm for k-means clustering which can leverage quantum computations has been proposed in 2013 by Lloyd at el. [57] In its simplest form, the algorithm takes classical data and uses **Grover’s search** to find the closest centroid for each point. In this case, each point is still represented as a classical bit, and no inherent quantum characteristics are utilized. While, as already said, the classical algorithm takes in space $O(M^2N)$, the use of Grover’s search allows for a **speedup** to $O(M\log(MN))$. The addition of $O(M)$ at each step, for both the classical and quantistic one, is due to the need for reassignment of all points after the distances have been calculated.

This result can be further improved by removing the constraint to give the $O(M)$ labels after each step. This is accomplished by allowing to have **quantum superpositions** instead of classical vectors as results, via the use of the *adiabatic theorem* (Section 4.3). The construction of said quantum states and the corresponding algorithm is detailed hereafter.

1. Select k vectors with labels i_c. Remember that the k-means clustering algorithm requires to be given, as a hyperparameter, the number of clusters to be formed. These may be chosen at random or with some better criterion.
2. Start from the following state:

$$\frac{1}{\sqrt{Nk}}\sum_{c'=1}^{k}\sum_{j=1}^{M}|c'\rangle|j\rangle\left(\frac{1}{\sqrt{k}}\sum_{k=1}^{c}|c\rangle|i_c\rangle\right)^{\otimes d} \tag{81}$$

   where the d copies of the state $\frac{1}{\sqrt{k}}\sum_{k=1}^{c}|c\rangle|i_c\rangle$ allow to calculate the distance between each point and each centroid $(\|\boldsymbol{x}_j-\boldsymbol{x}_{i'}\|)^2$ (where i' was used in place of $i_{c'}$) using the *dot product*, as detailed in Section 5.1, via the state in Equation 81 with Equation 53. See appendix in Lloyd et al. [57] for a more detailed explanation of its application.
3. The result from the previous dot product gives the following **initial clustering**:

$$|\psi_1\rangle=\frac{1}{\sqrt{M}}\sum_{c=1}^{k}\sum_{j\in c}|c\rangle|j\rangle \tag{82}$$

   What is happening is that each observation j is associated with its cluster c; and all are joined together in a single superposed quantum state.

4. Using the d copies of the state in Equation 82 allows to construct **individual cluster states**, $\left|\phi_1^{(c)}\right\rangle = \frac{1}{M_c}\sum_{j\in c}|j\rangle$, and estimate the number of observations in each state, M_c.
5. Continue to the next re-clustering, assuming that the d copies of $|\psi_1\rangle$ are made available from the previous step.
6. Evaluate the distance between each cluster's mean and each other point:

$$\left\|\boldsymbol{x}_j - \frac{1}{M_c}\sum_{j'\in c}\boldsymbol{x}_{j'}\right\|^2 = \|\boldsymbol{x}_j - \overline{\boldsymbol{x}}_c\|^2 \tag{83}$$

   Evaluated as mentioned above.
7. Apply a phase $e^{-i\|\boldsymbol{x}_j-\overline{\boldsymbol{x}}_c\|^2\delta t}$ to each component of the initial state, i.e. $|c'\rangle\,|j\rangle$. Using the same logic as in Section 5.1, this is equivalent to applying the following Hamiltonian:

$$H_f = \sum_{c'=1}^{k}\sum_{j=1}^{M}\|\boldsymbol{x}_j - \overline{\boldsymbol{x}}_c\|^2\,|c'\rangle\langle c'|\otimes|j\rangle\langle j|\otimes\mathbb{I}^{\otimes d}$$

   This application is done then via the **adiabatic algorithm** (Section 4.3), which evolves the following state with Hamiltonian H_f:

$$\frac{1}{\sqrt{Mk}}\sum_{c'=1}^{k}\sum_{j=1}^{M}|c'\rangle\,|j\rangle\,|\psi_1\rangle^{\otimes d} \tag{84}$$

   considering as **initial Hamiltonian**:

$$H_b = 1 - |\phi\rangle\langle\phi|$$

   where $|\phi\rangle$ is state *superposition of all clusters*, i.e. $|\phi\rangle = {}^1/\sqrt{k}\sum_{c'=1}^{k}|c'\rangle$.
8. The result of the previous step, can be demonstrated [57], is:

$$\left(\frac{1}{\sqrt{M}}\sum_{c'=1}^{k}\sum_{j\in c'}|c'\rangle\,|j\rangle\right)|\psi_1\rangle^{\otimes d} = |\psi_2\rangle\,|\psi_1\rangle^{\otimes d} \tag{85}$$

   We have used the adiabatic theorem to assign states to clusters in the next step (ψ_2).
   Repeat this d times in order to create d copies. One can thus create a **superposition of cluster assignments at each step**.

It can be shown that, after a few iterations, which is what typically applies for the "classical" k-means, the final cluster result is:

$$|\chi\rangle = \frac{1}{\sqrt{M}}\sum_{c=1}^{k}\sum_{j\in c}|c\rangle\,|j\rangle \tag{86}$$

Thus, this state contains all of the final result in superposition, and can be sampled for information regarding each cluster's content. The best time estimate, as given by Lloyd et al.[57], which relies on the classical notion that the k-means algorithm converges in a few iterations, is $O(k\log(kMN))$. For a more detailed and systematic explanation of the just described algorithm, see also Kerenidis et al.[58]

It is known that the use of a good seed for the k-means can have a significant computational speedup, as can be seen with the famous `k-means++` algorithm, be it used classically or with Grover's search. [59] It can be shown that, using an mathematical construct similar to the general quantum k-means algorithm just described, i.e. using the adiabatic theorem, it is possible to determine, without a too high cost, a "good" starting set of centroids.

It can be shown that, using quantum-specific algorithms to solve linear systems of equations [40], it is possible to extend the results proposed above to **non-linear metrics**. This can be shown to give itself an exponential speedup as well as compared to the classical counterpart.

As mentioned throughout this paper for most of the detailed methods, given the complexity of achieving a *general purpose quantum computer*, and to our knowledge, there has been no practical test of the proposed k-means algorithm. Thus, the given exponential speedup as compared to a non-quantum algorithm has not been established experimentally, but it is, as of the time of writing, a **theoretical result**.

6.4 Quantum Natural Gradient

The Gradient Decescent is one of the most used algorithms in order to find optimum parameters for minimizing or maximizing a given function. In deep learning indeed, it is used to discover weights that, assigned to a network, minimize a loss function.

The iterative step of this algorithm is described as follow:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta) \tag{87}$$

Where $L(\theta)$ is the **loss function**, θ are the parameters and η is the *learning rate*. So this means that at each step t, the parameters θ_{t+1} are updated from the present parameters θ_t, through the computation of $-\eta\nabla L(\theta)$.

The problem with the above approach is that each optimization step is strongly connected to a **euclidean geometry** on the parameter space. The reason why it could be a problem lies behind the fact that the cost function might vary at a different rate with respect to each parameter. This can cause the optimization to struggle to find, or even miss, the minimum altogether. So if a change of coordinate system is performed $\theta \to \phi$, it is possible to find a parameter space where variations in $L(\phi)$ are similar across different parameters. Now the feasable region in which to perform gradient descent is nicer and more informative, leading to a faster convergence and avoiding local minima. [60]

In classical machine learning, the **natural gradient descent** avoids gradient descent in the parameters space. Instead it performs the gradient descent in the **distribution space**. Meaning that the parameters are changed based on how they affect the output distribution. So the optimization problem is considered as a probability distribution of possible output values given an input. In order to define "distance" between probability distribution, it is used the D_{KL}.[6]

The natural gradient descent is formally written as:

$$\theta_{t+1} = \theta_t - \eta F^{-1} \nabla L(\theta) \tag{88}$$

The fundamental element is F, known as the **Fisher information matrix**, and without going into detail, it is a metric that defines distance between two distributions and basically it transforms the steepest descent in the Euclidean parameter space, to the steepest descent in the distribution space. In other word it turns the standard gradient descent into the natural gradient descent. Empirically it is shown that this method works well while minimizing a cost function, as shown later.

Variational optimization of parametrized quantum circuits is an integral component for many hybrid quantum-classical algorithms. In this family of algorithms, natural gradient descent can perform evenly better. In the quantum field as stated in [61], the euclidean geometry is sub-optimal for optimization of quantum variational algorithms: instead of using Fisher information metric, the **Fubini-Study metric tensor**[7] is used. So the natural gradient descent can be translated in quantum field as:

$$\theta_{t+1} = \theta_t - \eta g^{+}(\theta) \nabla L(\theta) \tag{89}$$

The Fubini-Study metric "drawback" is that it cannot be evaluated on a quantum hardware so the **block-diagonal approximation** is computed. Luckily, as depicted in Figure 14, the block-diagonal approximation turns out to be advantageous over standard gradient descent.

6.5 qSVM

The **Support Vector Machine** (SVM) is probably one of the most famous **classification** *shallow* Machine Learning algorithm in use today. The main idea behind this method is to find some line (or hyperplane) to separate the space in distinct groups, using pre-labelled data. [62] Mathematically, given a set of **training instances** $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_M, y_M)\}$, where $\forall i \in [1, M],\ \boldsymbol{x}_i \in C \subset \mathbb{R}^N$ and a generic hyperplane $\boldsymbol{w}^\top \boldsymbol{x} - b = 0$, defined in $\mathbb{R}^N$, the algorithm tries to find the "best" one, i.e. the one which maximizes the hyperplane margins between the data and itself, $\left\| \frac{2}{\boldsymbol{w}} \right\|$. It can be shown that this corresponds, in the most simple and linear SVM, to the following problem:

$$\arg\min_{\boldsymbol{w}, b} \frac{1}{2} \|\boldsymbol{w}\|^2 \tag{90}$$

[6]Kullback–Leibler divergence or KL divergence is a measure of how far away two probability distributions are from each other.

[7]Fubini-Study metric, seen as the Fisher information matrix in quantum field

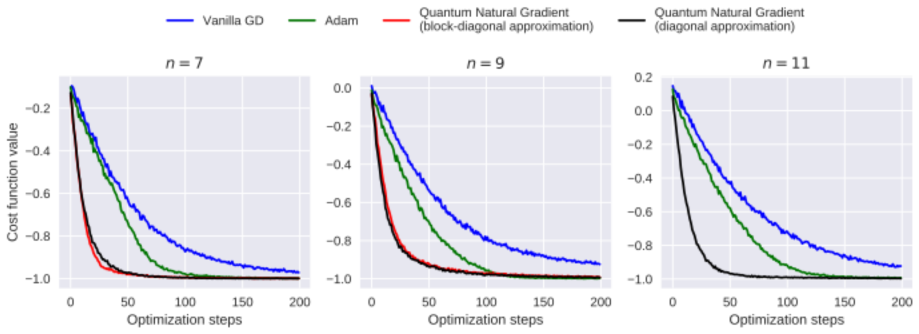


Figure 14: The cost function value for n = 7, 9, 11 qubits and l = 5 layers as a function of training iteration for four different optimization dynamics. 8192 shots (samples) are used per required expectation value during optimization.

with constraints $y_i\left(\boldsymbol{w}^\top \boldsymbol{x} - b\right) \geq 1$. It can be shown that solving this problem corresponds to solving its dual, which can be defined through the use of Lagrangian multipliers. The corresponding problem is thus:

$$\max_{\alpha_i} \sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \tag{91}$$

with constraints $\alpha_i \geq 0,\ \forall i$ and $\sum_{i=1}^{M} \alpha_i y_i = 0$. The parameters α_i arise from the Lagrangian constructions, and are related to the parameters $\boldsymbol{w}$ and b in Equation 90. For more information see Joachims [63].

Given the standard formulation as just described, some elements can be modified. In particular, the use of **soft margins**, i.e. allowing the algorithm a degree of missclassification, can be introduced, modifying the minimization problem to:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{M} \xi_i \tag{92}$$

In the dual formulation this corresponds to changing the constraint on the α_i from $\alpha_i \geq 0$ to $0 \leq \alpha_i \leq C$. Another modification to the “standard" linear SVM can be done in regard to the *shape* of the kernel. Indeed, instead of using a *linear hyperplane*, some transformation could be used, as follow (with soft margins):

$$y_i(\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - b) \geq 1 - \xi_i \tag{93}$$

where the **embedding function** ϕ allows for non-linear shapes. It can be shown that, in the dual formulation, this corresponds to changing the *inner product* $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ in Equation 91 with a **kernel function** $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The famous **kernel-trick** allows to skip the calculation of the embedding function and just evaluate the kernel function. A different formulation can be also given using a *ridge* regularization term, in the form:

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + \frac{\gamma}{2}\sum_{i=1}^{M} e_i^2 \tag{94}$$

with $y_i(\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - b) = 1 - e_i,\ \forall i = 1, \ldots, M$. Using the Lagrangian formulation, it can be shown that this problem corresponds to the following **least squares problem**:[64]

$$\begin{pmatrix} 0 & 1^\top \\ 1 & K + \gamma^{-1}\mathbb{I} \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{y} \end{pmatrix} \tag{95}$$

There have been proposed many different approaches for solving Support Vector Machine problems using *quantum computers*. Hereafter are presented some of the most established and well known ones. While others have been proposed, due to their novelty and, in some cases, very limited scientific consensus, their description has been omitted.

A method proposed by Anguita et al. [65] uses a variant of Grover's search in order to speedup the minimization problem. Indeed, whichever representation type of SVM is considered, be it as in Equation 91 or a more complicated formulation, classically the problem amounts to solving a **quadratic programming problem**, since a cost function to be maximized according to some constraint is given. Thus, the use of some quantum optimization, which may either speedup and/or allow to solve more complex problems, could be used. The algorithm by Anguita et al.[65] can be described as follow:

- Give the SVM model with discretized parameters, which is the case for most problems solved classically.
- Then prepare said parameters into some *qubits*.[8] They shall be stored into a quantum superposed state. The mentioned parameters are, using the above notation, α_i, ξ_i and e_i, depending on the particular problem considered.
- Use a quantum minimization algorithm. In particular, the method proposed by Dull et Holier [66] can be used. Without going into detail, it leverages a quantum exponential search algorithm many times, in k different runs. Indeed, it can be shown that, given all runs, where in each the algorithm converges after around $R \sim 22.5\sqrt{N}$ repetition, the probability of success is $P \geq 1 - \frac{1}{2^k}$.

This method's strength is given by its versatility: since the optimization requires just a search over the results space, any kind of SVM model can be presented, even non-convex ones, which classically would require non-exact methods.

Indeed, following a similar analysis, Denchev et al. [67] proposed the use of **adiabatic quantum optimization** to solve in cases when non-convex SVMs could arise, e.g. in the presence of noise in data. This can be achieved by writing the SVM problem into a **Quadratic Unconstrained Binary Optimization** (QUBO) problem, since it is suitable for adiabatic optimizations. This formulation, although, requires to evaluate the *kernel matrix* classically, which has a cost of $O(M^2N)$, where M is the number of observation in the training sample and N is the number of dimensions for each sample.

A completely quantum algorithm can be defined, with **exponential speedup**. As described by Wittek [12], this is based on the **least square formulation** given in Equation 95.

1. Define the matrix F, which comes from Equation 95:
$$F = \begin{pmatrix} 0 & 1^\top \\ 1 & K + \gamma^{-1}\mathbb{I} \end{pmatrix}$$
This matrix *need to be inverted*. In order to do this, one possible approach is to use the algorithm proposed by Harrow et al.[68]
2. Split the matrix given as $F = J + K_\lambda$, with:
$$J = \begin{pmatrix} 0 & 1^\top \\ 1 & 0 \end{pmatrix} \qquad K_\lambda = \begin{pmatrix} 0 & 0 \\ 0 & K + \gamma^{-1}\mathbb{I} \end{pmatrix}$$
where the matrix J is the *adjacency matrix of a star graph*. For more information on graph theory, see Jones [69].
3. Normalize F using its trace:
$$\hat{F} = \frac{F}{\text{tr}(F)} = \frac{F}{\text{tr}(K_\lambda)}$$
4. Calculate the exponential of $\hat{F}$, $e^{-i\hat{F}\Delta t}$. This can be done via **Lie's formula**[70]:
$$e^{-i\hat{F}\Delta t} = e^{\frac{-iJ\Delta t}{\text{tr}(K_\lambda)}} e^{\frac{-i\gamma^{-1}\mathbb{I}\delta t}{\text{tr}(K_\lambda)}} e^{\frac{-iK\delta t}{\text{tr}(K_\lambda)}} + O(\Delta t^2) \tag{96}$$
This can be done through *simulation*. As for the **sparse** matrices, i.e. J and $\gamma^{-1}\mathbb{I}$, the algorithm by Berry et al. [39] makes the calculation efficient. As for the **kernel matrix** K, which is dense, a process similar to the one used for the **qPCA** in Section 5.3.4 can be applied. But, since K is not surely Hermitian, using the *quantum dot product*, as described in Section 5.1, and with a QRAM (Section 5.3.1), the **normalized kernel matrix** can be obtained:
$$\hat{K} = \frac{K}{\text{tr}(K)} = \frac{1}{\text{tr}(K)} \sum_{i=1}^{M} \sum_{j=1}^{M} \langle \boldsymbol{x}_i | \boldsymbol{x}_j \rangle \, \|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\| \, |i\rangle \langle j| \tag{97}$$
which can be showed to be *Hermitian*, and thus a good candidate for the simulation.
5. In total, the evaluation of the **exponentiation**, in Equation 96, can be shown to be $O(\log M)$ in time.
6. The next phase is to run **quantum phase estimation**, in a manner again similar to Section 5.3.4, in order to get *eigenvalues* and *eigenvectors*. Note that this passage is all but trivial: for more information, see Harrow et al.[68]

[8]Naturally, this step is taken for granted here, but it's important to remark its non-triviality.

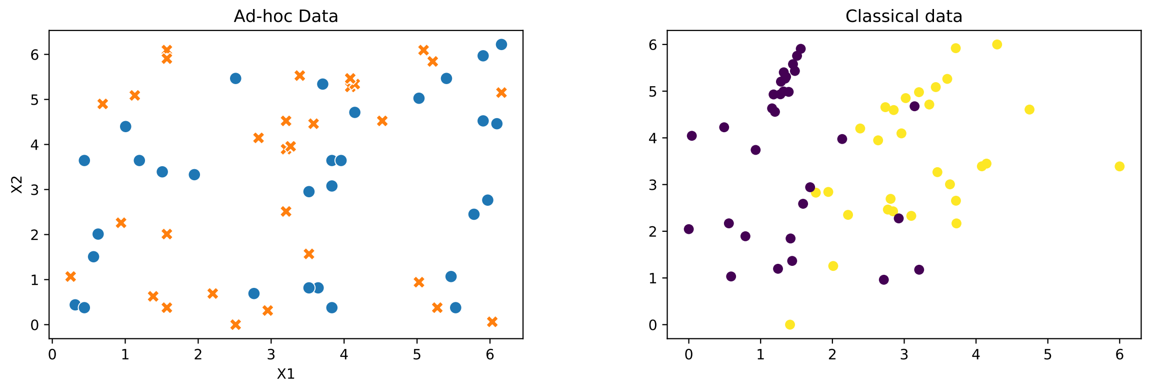


Figure 15: The two datasets used in the classification example. (Left) Quantum[9] dataset. (Right) Classical dataset.

7. Following Harrow et al.[68], in order to complete the *Least Squares* solution, the variable $\boldsymbol{y}$ in Equation 95 must be expressed in the **eigenbasis** of F.
8. Then, invert the eigenvectors and obtain the solution for the SVM $|b, \alpha\rangle$.

It can be shown [12] that the overall training time for this SVM is $\log(MN)$. Unfortunately, this method does not allow for complex kernel functions: while linear and polynomial ones are easy to implement, the same is not true for the *Radial Basis Function*, which is the most common kernel today. It should be kept in mind that, by using the *Least Squares* formulation of the SVM, the result tends to be less sparse, with possible overfitting scenarios.

While the presented methods for the use of quantum computing in SVMs are aimed mostly at **computational speedups**, another approach from quantum logic can arise: the use of **quantum feature spaces** to solve problems that classically are not solvable. Havlicek et al. [71] describe an approach to this problem, using a specific feature map. Without going into detail, the main idea is rather simple: for complex kernel functions, which might not be simulated reasonably using classical computers, one can map the data to a quantum feature space, run a quantum kernel function, and then re-project the data back to a classical feature space. Just for reference, the feature map used by Havlicek et al.[71] was generated by the following unitary transformation:

$$\mathcal{U}_{\phi}(\boldsymbol{x}) = U_{\phi(\boldsymbol{x})} H^{\otimes n} U_{\phi(\boldsymbol{x})} H^{\otimes n} \tag{98}$$

where H is the Hadamard gate and $U_{\phi(\boldsymbol{x})}$ is a diagonal gate dependant on a function ϕ which encodes the data; the particular implementation is not very important, since any diagonal unitary may be used. A similar approach, but described in a much general setting, in described in Section 6.2.1.

Thus, given the premise that quantum computing can be leveraged to calculate kernels that are classically either too hard, e.g. exponential in time or space, or impossible, hereafter will be shown a *working* **example** of such approach. In 2016, IBM Research released to the world a **5-qubit universal supercomputer**, accessible through the cloud. [72] While many advancements are still ahead as far as research on quantum computation is concerned, this early prototype, although small in computational size and noisy, has today a lot of potential applications.[73] In particular, given the possible approach to qSVM described above, it is possible to implement a version of it through the **IBM quntum experience** cloud platform, via the **qiskit** python library.[74] Thus, following the proposed example by Havlicek et al. [71], and the subsequent tutorial present on the IBM Quantum Experience platform [75], the code and results of the application to ad-hoc data of a **quantum-enhanced Support Vector Machine** is presented.

Two datasets were used. The first was prepared artificially in a manner similar to what described by Havlicek et al.[71] As for the understanding of this example, just consider that this dataset is 2-dimensional and divided into 2 groups, identified by as many labels. The second one was created artificially as well, but by using the `make_classification` function in the Python library `sklearn`; as for the former, the data was created in a 2 dimensional space and separated into 2 distinct classes, using some quantum-like simulation. In Figure 15 are depicted the two different datasets. As can be seen, while the latter and “classical" one presents a class division in space which is intuitive, the same cannot be said about the “ad-hoc quantum dataset".[9]

[9]Note that the dataset is not quantum. The label quantum is just used in order to better distinguish between the more “classical" one and the one prepared especially for a quantum-enhanced Support Vector Machine.

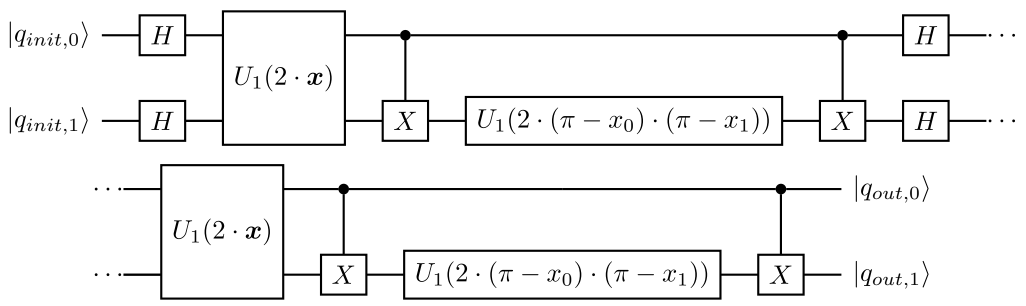


Figure 16: 2-qubit feature map gate used in the example given.

Ad-hoc Dataset		
SVM kernel	5-fold CV accuracy	std
ZZ feature map	1.0	0.0
rbf	0.45	0.15
linear	0.4	0.17
polynomial	0.52	0.06

Classical Dataset		
SVM kernel	5-fold CV accuracy	std
ZZ feature map	0.6	0.1
rbf	0.88	0.09
linear	0.85	0.1
polynomial	0.86	0.08

Table 8: (Left) Accuracy and its standard error calculated for all models on the *quantum ad-hoc dataset*. (Right) Accuracy and its standard deviation on the classical dataset generated from `sklearn`.

Both datasets were used to perform **5-fold cross-validation** on 4 different types of Support Vector Machines: one quantum-enhanced SVM, and 3 classical SVM. As for the quantum one, a feature map similar to what presented by Havlicek et al.[71] was used, and is presented in Figure 16.

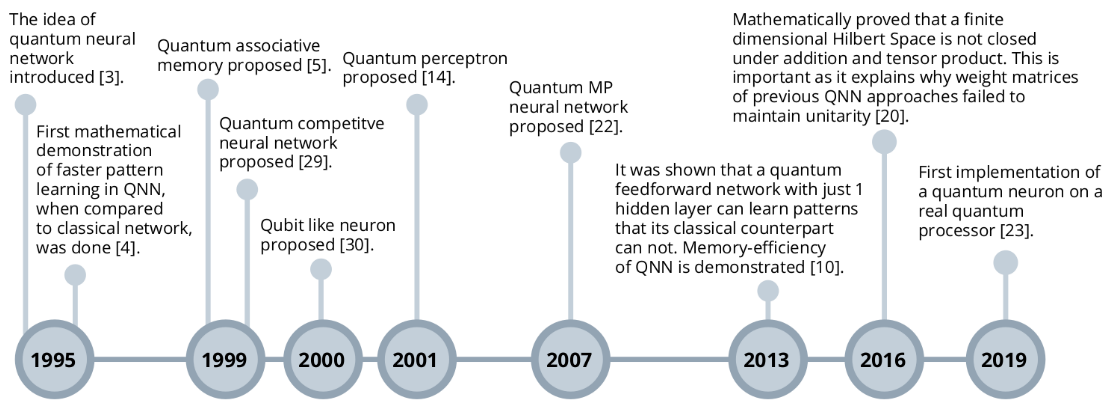
The results obtained are presented in Table 8. As one might expect, on the task designed specifically to be solved well by the **quantum-enhanced SVM**, this former performed the best. The interesting result, though, it how poorly traditional "classical" SVM kernels performed, i.e. either at or below 50% accuracy. While better "classical" models could be built, it is nonetheless striking how poorly the presented Support Vector Machines managed to fare. On the other, using the data prepared using the `sklearn` library, which is usually proposed for classical machine learning problems, the opposite happens: classical kernels perform quite well, and not too differently from one another, while the quantum-enhanced fares poorly. This is in line with what already mentioned in Section 6.2.2: while quantum computing can enable new and interesting solutions to problem hard to solve with classical computer, they are not universal problem solvers, and their application must be tailored to a specific scenario.

7 Quantum Deep Learning

7.1 Introduction to Quantum Neural Networks

Neural networks enjoy widespread success in both research and industry and, with the advent of quantum technology, it is a crucial challenge to design quantum neural networks for **fully quantum** learning tasks.

The sector of quantum neural networks, just like any other quantum machine learning field, is still in its exploratory phases, but in recent years has shown remarkable progress. Lately, even some application of quantum neural networks on real world problems have started to emerge [76, 77, 78]. In other words, some sectors are exploring the possibilities of quantum computing, and it might be possible that this kind of interest could be extremely beneficial to the development of these technologies. Despite the enthusiasm, there are still many challenges and open problems left for QNNs, namely, finding the correct generalization of the perceptron, architecture, optimization algorithm, loss function, etc.. The proposed models are numerous, each with its own advantages and downsides, and a general consensus on the *best* model has not been found yet. A very promising architecture has been developed by Beer et al. in the QQ setting, [79]. This model fully implements training and inference on a deep feed forward neural network, for the task of learning an unknown unitary.


The idea of quantum neural network introduced [3].
First mathematical demonstration of faster pattern learning in QNN, when compared to classical network, was done [4].
Quantum associative memory proposed [5].
Quantum competitve neural network proposed [29].
Qubit like neuron proposed [30].
Quantum perceptron proposed [14].
Quantum MP neural network proposed [22].
It was shown that a quantum feedforward network with just 1 hidden layer can learn patterns that its classical counterpart can not. Memory-efficiency of QNN is demonstrated [10].
Mathematically proved that a finite dimensional Hilbert Space is not closed under addition and tensor product. This is important as it explains why weight matrices of previous QNN approaches failed to maintain unitarity [20].
First implementation of a quantum neuron on a real quantum processor [23].
1995
1999
2000
2001
2007
2013
2016
2019
Figure 17: From [81]. Timeline of the main advancements in quantum neural network research. A brief overview of the models is available at the source document. (Citations refer to the source paper bibliography)

Other researchers have developed an equivalent architecture for **convolutional neural networks** in the quantum realm, for the tasks of quantum phase recognition and quantum error correction [80]. The proposed QCNN also allows for efficient training and implementation on realistic near term quantum devices.

In the CQ setting, which is the main interest of this work, studies have shown that learning is actually possible, and many fully quantum models have been proposed, but their actual implementation on realistic near term quantum devices has still a long way to go, mainly due to hardware problems. Also, the processes of optimization and learning in the CQ has presented some obstacles, as will be shown in the next sections.

First, a brief history of quantum neural network is given, based on [81].

7.2 History of quantum neural networks

Just like classical neural nets, the history of quantum networks is closely tied to neuroscience. Since it is speculated that the human brain is governed by the laws of quantum mechanics, a neural network built on quantum laws seemed, to some researchers, a more natural choice if the aim is to model the human brain accurately. S. C. Kak [82], was the first to present the idea of a quantum neural network in 1995, basically from a neuroscience point of view. In the same years, the computational efficiency of an hypothetic quantum neural network has been shown by Menner and Narayanan [83]. These first two researches established the basis for future works, and in the following 25 years, many quantum neural networks models have been proposed.

In the same way in which focus from neuroscience as faded with regard to the implementation of classical networks, the latest quantum NN models do not aim at *reproducing the human brain*, but only at learning efficient models from data. A common approach used in formulation of quantum neural networks is redesigning of classical components with the principles of quantum mechanics, similarly to what is done in other QML algorithms. There is one major difference though: neural networks, in fact, are usually highly non linear models, while in quantum mechanics operators that act on states are always linear.

Implementing a non-linear activation function is a major problem, and several attempts aim at resolving this problem, i.e. by using measurements, but measurements suffer from decoherence, and this solution is, generally, not considered adeguate. To this day, there are no proposals for the implementation of a non-linear quantum operator, and most quantum neural networks offload nonlinearities to classical computers, or make use of quantum kernels.

In Figure 17, a summary of the main advances in the field of quantum neural network is presented. It is possible to divide the timeline in two separate phases: from 1995 to 2007 the main focus of research has been on formulating several models. After 2007, the prime focus shifted towards implementation.

Recently, cloud-based services have allowed for the implementation of small-size models on actual quantum computer, and the first neuron has been developed on a real quantum processor [84]. In this work, two different approaches to feed forward networks are presented, the first is a **hybrid model**, in which quantum neural networks are formalized as **variational circuits**[42], the other is based on a work conducted by Farhi and Neven, where a model of QFFNN is presented, capable of classifying classical and quantum data [85] on near term processors. Variational circuits can also

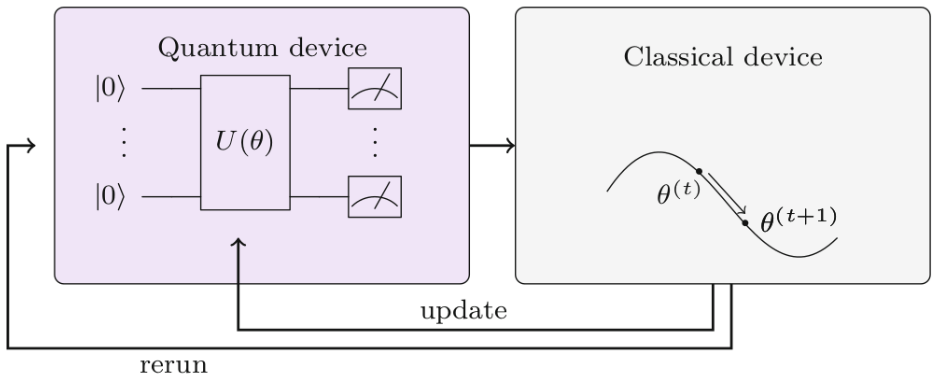


Figure 18: Scheme of an hybrid quantum-classical training algorithm for a variational circuit. The quantum device computes terms of an objective function, and subsequently use a classical device to compute better circuits parameters.

be used in the latter setting, in order to reduce memory requirements during training. For this reason, a description of hybrid variational circuits is given, based on [42].

7.3 Variational Circuits

With full-blown fault tolerant quantum computers still in the future, a particular class of hybrid classical-quantum algorithms has become popular to design near term applications for quantum devices. The basic idea behind hybrid training of variational algorithms is to use a quantum device to compute the value of an objective function $C(\theta)$ for a given set of classical parameters θ. A classical algorithm is then used to optimise over the parameters by making queries to the quantum device.

A basic structure of a variational circuit is shown in Figure 18. The quantum device implements a parametrised circuit $U(\theta)$ that prepares the state $U(\theta)\,|0\rangle = |\psi(\theta)\rangle$ which depends on a set of parameters θ. Measurements on the final states $|\psi(\theta)\rangle$ return estimates of the expectation values or the state of a certain qubit. A cost function $C(\theta)$ uses the expectation values to define how good θ is in a given problem context. The goal is to find values for θ that minimizes $C(\theta)$.

Now a few considerations about the state $|\psi(\theta)\rangle$: of course the state must be prepared by the circuit $U(\theta)$. The space of states that a given circuit is able to prepare is tipically much smaller than the space of all possible states, since the number of parameters required for such an implementation is quadratic in the dimension of the Hilbert Space, and becomes prohibitive with growing number of qubits. The advantage of this kind of approach lies in the fact that not all the objective functions can be efficiently computed classically, so, if a circuit $U(\theta)$ is not classically simulable, the overall training exhibits an **exponential speedup**.

There are some aspects of variational circuits that make them suitable for near-term quantum technologies, in particular:

- The dimension of the total circuit is only a fraction of a fully quantum algorithm. Thus, it is easier to maintain coherency when performing such operations.
- There are many possibilities for the circuit, which means that it can be designed based on the strength of the device. (The circuit can be arbitrarily deep, depending on the ability of the device to correct noise).
- The fact that the circuit can be learned can introduce robustness against systematic errors.
- Compared to other types of algorithms, given the iterative nature of the algorithm, noise does not constitute a major problem.

As of today, a great number of variational algorithms has been implemented. The most important are:

- **Eigensolvers**, whose aim is to find the ground state of quantum systems.
- **Approximate optimization algorithms**, to solve combinatorial optimization problems.
- Recently, cloud-based services have allowed for the implementation of small-size models on actual quantum computer, and the first neuron has been developed on a real quantum processor [84]., where is the expectation value of an observable O can be interpreted as the output of the classifier: $f(x;\theta) = \langle\psi(x;\theta)|O|\psi(x;\theta)\rangle$.
- **Numerical simulation of quantum many-body** density of probability estimates.

- **Deep neural networks**, as will be shown in the next section.

In order to optimize for the parameters, classical optimization algorithms are used, i.e. simplex method, numerical gradient methods, analytical gradient methods, etc.. Please refer to section 7.3.4 of [42] for an analytical explanation of gradients and derivatives in variational circuits.

In the next section, a particular decomposition of variational circuits is shown, where gates can be interpreted as linear layers in a neural networks.

7.3.1 Variational classifiers as neural networks

A linear single qubit gate can be defined as:

$$G(\alpha,\beta,\gamma) = \begin{pmatrix} e^{i\beta}\cos\alpha & e^{i\gamma}\sin\alpha \\ -e^{i\gamma}\sin\alpha & e^{i\beta}\sin\alpha \end{pmatrix} = G(u,v)\begin{pmatrix} z & v \\ -v^{*} & z^{*} \end{pmatrix} \tag{99}$$

Where $|z|^2 + |v|^2 = 1$. If the single qubit gate is applied to the ith qubit on a n qubit register, it can be written as:

$$G_{q_i} = \mathbf{1}_1 \otimes \cdots \otimes G(z,v) \otimes \cdots \otimes \mathbf{1}_n \tag{100}$$

Where $\mathbf{1}_i$ are 2×2 identity matrices. To make this elementary gate more universal, another gate is considered: $c_{q_j}G_{q_i}$ is a single qubit gate controlled by another qubit j. The only difference between the the two is the fact that the latter carries a *diagonal unit entry* in some rows and columns, so for example:

$$(c_{q_3}G_{q_2})_{ij} = \begin{cases} 1 & \text{if } i=j, (G_{q_2})_{ij} = 0 \\ (G_{q_2})_{ij} & \text{otherwise} \end{cases} \tag{101}$$

But other rules are applied for other control qubits.

As a product of matrices, a variational circuit of depth L:

$$U(\theta) = G(\theta_L)\ldots G(\theta_1) \tag{102}$$

can be understood as a sequence of linear layers of a neural network where the layer have constant size. The position of the qubit as well as the control qubit determine the architecture of each layer. In Figure 19 an example with $n = 3$ is provided.

To illustrate the meaning of Figure 19, consider the 3 qubit state as an 8-dimensional vector of amplitudes $(\alpha_1,\ldots,\alpha_8)$. The operators defined in Equation 101, 99 are actually 8×8 matrices whose coeffients reflect the structure exposed in the figure ($(G_{q_2})_{ij}$ is the connection between the ith input unit and jth output unit). Basically, the control qubit breaks the simmetric structure by replacing some connections with identity, thus carrying the same values from previous layers.

Currently, numerous investigations are carried about the representational power of these kinds of classifiers, and a major problem has recently been exposed.

Barren plateaus in QNN training landscapes It has been shown by McClean et. al. [86] that random circuits, as the one shown previously, do not constitute a suitable choice for variational circuits in a hybrid quantum-classical setting. The exponential dimension of the Hilbert space, in fact, and the gradient estimation complexity often bring the problem on a so called *barren plateau*: the probability that the gradient along any reasonable direction is non-zero to some fixed precision is exponentially small as a function of the number of qubits. Despite this discovery has shifted the attention from hybrid to fully quantum architectures, not all hope is lost here, as stated in [86]:

> Historically, vanishing gradients may have played a role in the early winter of deep neural networks. However, multiple techniques have been proposed to mitigate this problem, and the amount of training data and computational power available has grown substantially. One approach to avoid these landscapes in the quantum setting is to use structured initial guesses, such as those adopted in quantum simulation. Another possibility is to use pre-training segment-by-segment, which was an early success in the classical setting.

Notably, the model proposed by Beer et. al. [79] does not suffer from this problem, and the path toward a near term quantum deep neural network seems less steep in the QQ setting.

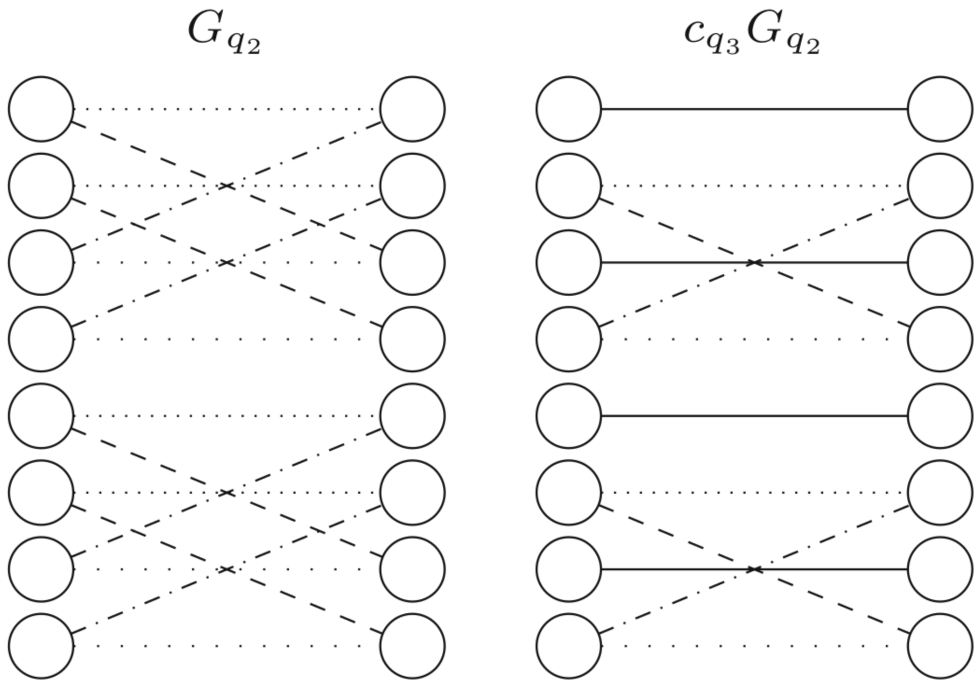


Figure 19: From [42], the single qubit gate and the controlled single cubit gate applied to a system of 3 qubits. The solid lines are identity connections of weight 1, while all the other line styles indicate varying weight parameters. To lines of the same stile indicate that the parameters are shared.

7.4 Fully-Quantum Feed Forward Neural Networks

A notable approach to fully quantum neural networks on near term processors has been developed by Farhi and Neven [85]. Other than developing a full model for supervised learning in quantum setting, the research shows through numerical simulation that learning is actually possible even for near term processors, by classifying downsampled images coming from the MNIST dataset [87]. Furthermore, a novel way for representing data in quantum superposition is presented.

7.4.1 Introduction and setting

The dataset used for theoretical formulation consists of strings in the form $z = z_1 z_2 \dots z_n$ where each z_i is a bit taking the value ± 1 and a binary label $l(z)$ chosen as ± 1. For simplicity, it is assumed that the dataset consists of 2^n strings in total. The quantum processor, on the other hand, can act on $n + 1$ qubits, n for strings and the last one that serves as a readout, ancillary qubits are ignored. Similarly to the neural network models based on variational circuits, the quantum processors implement unitary transformations on input states. Also, a toolbox of unitaries, determined experimentally (see [88]) is available, and defined as:

$$\{U_a(\theta)\}$$

Each unitary acts on a subset of qubits and depends on a continuos parameter θ. A set of L unitaries is defined as:

$$U(\boldsymbol{\theta}) = U_L(\theta_L) \dots U_1(\theta_1) \tag{103}$$

where:

$$\boldsymbol{\theta} = (\theta_L, \dots, \theta_1) \tag{104}$$

While states are, momentarily, defined as:

$$|z_1, z_2, \dots, z_n, 1\rangle \tag{105}$$

On the readout qubit, a Pauli operator is measured, referred to as Y_{n+1}. The result of the measurement is ± 1. The predicted label for a given string z is defined as:

$$\langle z, 1|U(\boldsymbol{\theta})^{\dagger} Y_{n+1} U(\boldsymbol{\theta})|z, 1\rangle \tag{106}$$

A schematic representation of the network is presented in Figure 20:

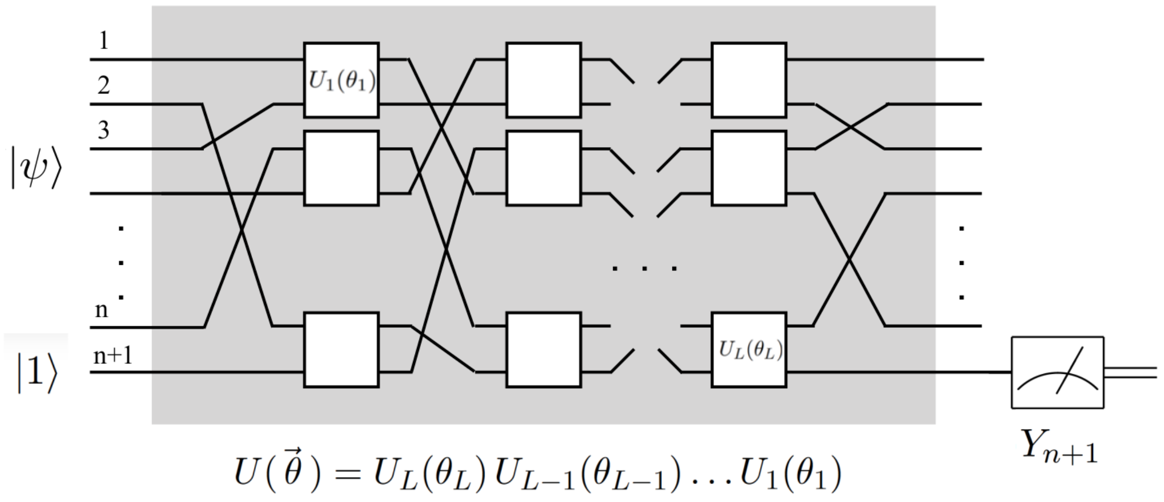

1
2
3
$|\psi\rangle$
n
n+1
$|1\rangle$
$U_1(\theta_1)$
$U_L(\theta_L)$
Y_{n+1}
$U(\vec{\theta}) = U_L(\theta_L)\, U_{L-1}(\theta_{L-1}) \dots U_1(\theta_1)$

Figure 20: A schematic representation of the quantum neural network proposed in [85]. An input state $|\psi\rangle$ is prepared and transformed with the subsequent application of single qubit gates. After the transformations, a Pauli operator is measured on the readout qubit. The average result of the measurements is the predicted label for the input data.

Lastly, the loss function is defined as:

$$\text{loss}(\boldsymbol{\theta}, z) = 1 - l(z)\, \langle z, 1|U(\boldsymbol{\theta})^{\dagger} Y_{n+1} U(\boldsymbol{\theta})|z, 1\rangle \tag{107}$$

which is linear in the margin. Similarly to a variational circuit, for every instance the loss is computed and the weights are modified accordingly with a stochastic gradient descent algorithm. Two procedures for calculating the gradient are presented.

7.4.2 Learning - Calculating the loss gradient

Once the sample loss is estimated, the objective is to calculate the loss gradient with respect to $\boldsymbol{\theta}$.

Simmetric difference method The first method is based on the fact that a function derivative can be obtained (with second order accuracy) by taking its simmetric difference:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon} + O(\varepsilon^2) \tag{108}$$

To achieve this, of course, the error for estimates of $f(x)$ must be at least $O(\varepsilon^3)$ accurate. The loss can be estimated to order ε^3 with of order $1/\varepsilon^6$ measurements. Each component of the gradient can be obtained with accuracy of order η by making of order $1/\eta^3$ measurements. This procedure must be repeated L times to get the full gradient.

Auxiliary circuit First, it can be supposed that each unitary in the network can be written in the form:

$$U_k(\theta_k) = \exp(i\theta_k \Sigma_k) \tag{109}$$

Where Σ_k is a generalized Pauli operator (i.e. a tensor product of Pauli operators). It can be shown that:

$$\frac{\mathrm{d}\text{loss}(\boldsymbol{\theta}, z)}{\mathrm{d}\theta_k} = 2\,\text{Im}\left(\langle z, 1|U_1^{\dagger} \dots U_L^{\dagger} Y_{n+1} U_L \dots U_{k+1}(\Sigma_k U_k) \dots U_1|z, 1\rangle \right) \tag{110}$$

That can be reexpressed in the following way:

$$\mathcal{U}(\boldsymbol{\theta}) = U_1^{\dagger} \dots U_L^{\dagger} Y_{n+1} U_L \dots U_{k+1}(\Sigma_k U_k) \dots U_1 \tag{111}$$

$$\frac{\mathrm{d}\text{loss}(\boldsymbol{\theta}, z)}{\mathrm{d}\theta_k} = 2\,\text{Im}\, \langle z, 1|\left(\mathcal{U}(\boldsymbol{\theta}) \right)|z, 1\rangle \tag{112}$$

Where $\mathcal{U}(\boldsymbol{\theta})$ can be viewed as a quantum circuit composed of $2L + 2$ unitaries. So, using an auxiliary qubit and a Hadamard transformation, it is possible to measure the value of the gradient. This methods avoids the numerical accuracy issue that comes with approximating the gradient as outlined in the previous paragraph. The cost is that an auxiliary qubit is needed, along an *auxiliary circuit* of depth $2L + 2$.

7.4.3 Data in quantum superposition

In the previous setting, the data was fed into the network one string at the time. It's possible to leverage on the quantum properties and present the data as *superposition of states*. To do this, the sample space can be divided in two batches, the first composed of the data labeled as $+1$ and the second composed of the remaining data:

$$|+1\rangle = N_{+} \sum_{z:l(z)=+1} e^{i\phi z} |z, 1\rangle \tag{113}$$

$$|-1\rangle = N_{-} \sum_{z:l(z)=-1} e^{i\phi z} |z, 1\rangle \tag{114}$$

where N_{+} and N_{-} are normalization factors and ϕ are set to zero WLOG, since no hints are available for a particular choice. Each of the two states just written can be viewed as a batch containing all of the samples with the same label. Now, it can be shown that the mean value of Y_{n+1} on the state $U(\boldsymbol{\theta})|+1\rangle$, is the expression:

$$\langle +1|U^{\dagger}(\boldsymbol{\theta})|+1\rangle \tag{115}$$

It can be shown that this expression is equal to the average over all samples with label $+1$ of the QNN's predicted label values. The same holds for the state $|-1\rangle$. So, in conclusion, the expression:

$$1 - \frac{1}{2}\Bigg(\langle +1|U^{\dagger}(\theta)Y_{n+1}U(\theta)|+1\rangle - \langle -1|U^{\dagger}(\theta)Y_{n+1}U(\theta)|-1\rangle \Bigg) \tag{116}$$

is the **empirical risk** of the whole sample space. Basically, if a set of parameters θ is found for which the expression yields 0, then the network can correctly classify all the examples in the training set. It has been found empirically that training the network with this yields better results on the downsampled MNIST dataset, as stated in the paper:

> we saw more than an order of magnitude improvement in the sample complexity required to get comparable generalization error on individual test samples.

In other words, it takes an order of magnitude more of samples while training from the individual vectors to achieve the same generalization power of a model trained with data in superposition.

7.4.4 Final remarks

The methodology shown by Farhi and Neven, despite constituting only an exploratory work, sets out a specific framework to build quantum neural networks that can be used for supervised learning. The work is limited by the dimension of the system, and would probably struggle with the previosly cited *barren plateaus* at increasing number of qubits. Also, the simulations are conducted with little to no guidance with regard to the set of unitaries to use during training. As previously stated, it constitutes an exploratory research whose purpose is to show that learning through a fully quantum neural network is actually possible. Lastly, this procedure can be classically simulated.

7.5 Quanvolutional Neural Networks

Convolutional Neural Networks are some of the most famous architectures used today for many different tasks, the foremost of which is *image recognition* tasks, given its conception inspired by the visual perception mechanism of living creatures. [89] Their name comes from the use of **convolutional layers** as their main backbone, which allows to create hierarchical layers of different representations of the input tensor (usually an image).[37] In recent years many advances in the study of Convolutional Networks have been great. See Gu et al. [90] for the many techniques and specialization of the convolutional layer introduced recently.

Given the advances in quantum computation and machine learning described throughout this paper, it is not striking the many proposals for the use of quantum advantages for convolutional neural networks have been introduced. As for other types of algorithms, the definition of such models can either rely on completely quantum systems, as proposed by Cong et al. [91], or on hybrid methods, as shown by Harderson et al. [92]. Given its more “practical" appeal, the descriptions of the latter will be given. As a warning, it should be noted that the techniques present in literature are all very recent, with at most 2 years since publication. Thus, not too much review has been done on them, and their results, especially the presented one, should still be taken with a “work in progress" approach.

Quanvolutional Neural Networks are simply an extension of classical Convolutional Neural Networks, where some convolutional layers are substituted by a **quanvolutional layer**. These layers are built very similarly to their classical counterpart: they are a stack of N filters, where each one produces a feature map by transforming, locally in space and fully in depth, the input data. The key difference is the use of **random quantum circuits** in order to transform data, which, according to Henderson et al. [92], could lead to an increased accuracy. It should be noted that the idea of using **random non linear features** is not new to machine learning, and has been already implemented for the development of specific Convolutional Neural Networks. [93] [94] Quanvolutional layers are designed in a manner very similar to their classical counterpart, and thus it is possible to stack many filters together and providing each layer with a specific configurational attribute. With this idea in mind, it is possible to create a traditional architecture, but with the presence of a quanvolutional layer instead of a classical convolutional one.

The quanvolutional layer is applied to an input tensor to produce a **feature map**, using local operations. But, while classically such operations where a simple *element-wise matrix multiplication*, the quantum counterpart uses locally a **quantum circuit** to transform the input data, in a manner similar to what described in Section 7.3. The methodology proposed by Henderson et al. [92] uses **random quantum circuits** in order to transform data. In should be noted, although, that the use of circuits with a particular structure can be theoretically applied, even though no knowledge on this regard has been established.

The general application of the quanvolutional layer can be defined as a function of a local subsection of the input, e.g. a 2×2 matrix of the input image, the encoding and decoding functions, and the quantum circuit itself:[92]

$$Q(u_x, e, q, d)$$

where u_x is the input local tensor, e the encoding function, which transforms the input tensor u_x into a quantum state, q the quantum circuit transformation and d the decoding function, which takes the output of the quantum circuits and executes a finite number of measurements to obtain the final result. Given the different choices of quantum circuit q, the computational complexity depends on it when given access to a *quantum computer*. But, given the recent advancements in the development of *quantum simulations* using classical computers, Henderson et al. [92] state that the proposed method could run, without the need for a **qRAM**, via one of said **quantum simulations**, but in longer time than classical convolutional layers[95]. On this regard, exact theoretical estimations were not given. Thus, without access to a true quantum computer, which might be able to theoretically evaluate the quantum circuit q faster than the $O(n^2)$ of a classical convolutional filter, the most interesting approach for this method are the different output feature maps. Nevertheless, as stated by Henderson et al. [92], no clear advantage for the use of QNNs over CNNs has been established to date.

Some results have been established by Henderson et al. [92], which compared the proposed architecture with 2 classical ones: a traditional CNN, and a Convolutional Neural Network where the first convolutional layer applies a random non-linear transformation, in a manner similar to the quanvolutional layer, but classically. Without going into detail of the single architectures, in Figure 21 are shown the results presented by Henderson et al. [92]. As can be seen, both models which use non-linear random transformation fare better than a traditional CNN on the same task, which in the given case was the classification of images from the MNIST dataset. Thus, from this result, one might argue that the real benefit is given by the addition of random elements to a convolutional network, be it classically or using a quantum simulation, as such was the approach used. It should be noted, though, that the quanvolutional layer used for this experiment was **not** trained with the rest of the network, but used some fixed parameters. It may be possible to train such layer as well, as suggested by Mari [96].

Nonetheless, given the possible interest and applicability of the given topic, a similar experiment to that proposed by Henderson et al. [92] is presented. Following the tutorial presented by Mari [96], a QNN architecture was simulated and trained over the MNIST dataset. This was achieved using the `Python` programming language with the `pennylane` framework for quantum computation, in addition to traditional libraries, e.g. `Keras` and `tensorflow` for Deep Learning.

Given the heavy computational cost for quantum simulations, not all of the MNIST dataset was used, as opposed to what was done by Henderson et al. [92] Thus, a train set of 700 samples and a validation set of 300 was selected. A simple pre-processing, with rescaling of the images and preparation for a traditional neural network, was performed. [97]

A similar network to that developed by Henderson et al. [92] was used to built 2 networks, instead of the 3 proposed in the aforementioned research. The models built were a conventional Convolutional Neural Network, with the following architecture:

$$\texttt{CONV1}(50) \to \texttt{CONV2}(64) \to \texttt{MAXPOOL}(2 \times 2) \to \texttt{FC1}(1024) \to \texttt{Dropout}(40\%) \to \texttt{FC}_{\text{output}}$$

and a **Quanvolutional Neural Networks**, which just adds a non-trainable quanvolutional layer before all of the layers in the “standard" CNN above. Thus, from another point of view, in the present experiment, and in a similar manner in the experiment presented by the aforementioned paper, the fixed quanvolutional layer acts just as a *feature extractor*

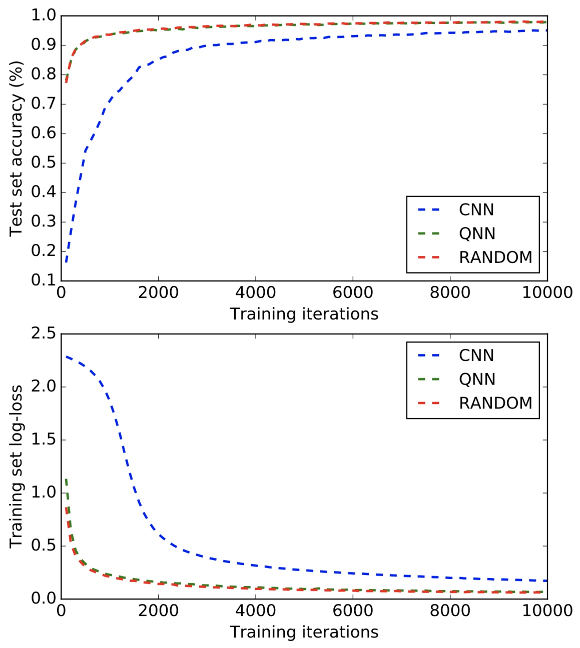


Figure 21: Accuracy and Loss accross training epochs for all models used. Images from Henderson et al. [92]

and creates a new “noisy" input data. The quantum layer was built with 4 filters, each operating some random encoding to a 2×2 region of the input data using some *Random Quantum Circuit*, defined implicitly by the `pennylane` quantum library. For completeness, and for interest as well, below is reported the `Python` code which defines the random quantum circuit that operates on each 2×2 window of the input data:

```
# This is the "decorator syntax".
# It is equivalent to writing qml.qnode(dev) = circuit(qml.qnode(dev))
#
@qml.qnode(dev)
def circuit(phi=None):
    # Encoding of 4 classical input values
    for j in range(4):
        qml.RY(np.pi * phi[j], wires=j)

    # Random quantum circuit
    RandomLayers(rand_params, wires=list(range(4)))

    # Measurement producing 4 classical output values
    return [qml.expval(qml.PauliZ(j)) for j in range(4)]
```

The results of the proposed experiment are shown in Figure 22. It can be seen that, overall, using the validation accuracy, no model seems to fare better than the other. The only noticible difference is given the much “prominent" oscillation of the QNN in the validation accuracy, compared to the standard CNN: this might be due to the injection of “noise", through the random quantum circuit operations, in the input data. While this result appears to be less interesting than the one shown in Figure 21 by Handerson et al. [92], this might due to different reasons, such as a larger train sample for the latter or some systematic error in the former. Another reason might be given by the different quantum simulation on which the 2 experiments run: while the proposed experiment used the `pennylane` library, Henderson et al. [92] used simulations via the `QxBranch Quantum Simulation System`, not available for public use. Which of the proposed explanations might have caused such discrepancies is not determined here.

In conclusion, while on paper a promising and interesting hybrid classical-quantum variation of the traditional Convolutional Neural Network, the given results show no quantum advantage. Nevertheless, it should be noted that all presented models used a fixed-parameter quanvolutional layer, and the implementation of a **natural gradient descent**

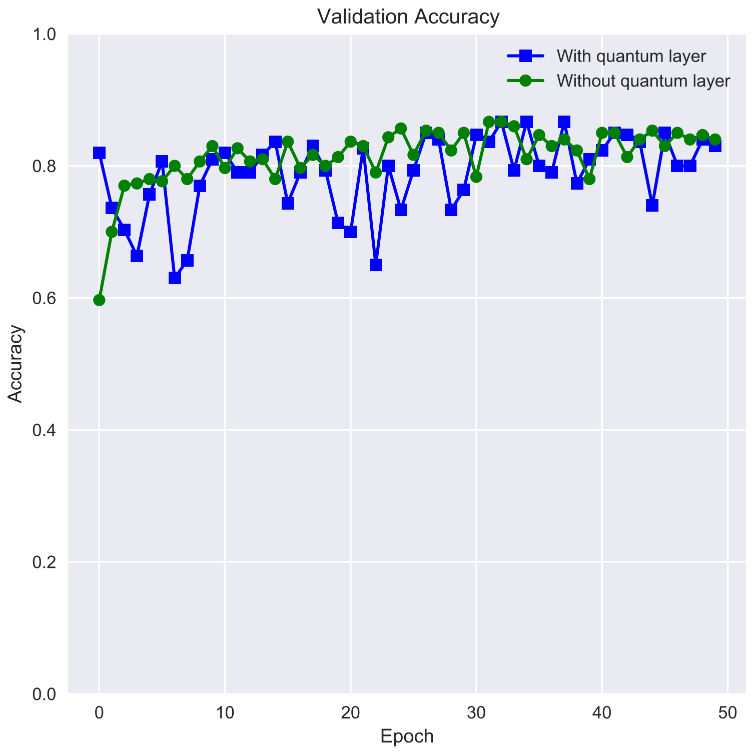


Figure 22: Validation Accuracy over all training epochs, for the 2 proposed models.

(Section 6.4), which is allowed under the `pennylane` package [96], for parameter update in the quanvolutional circuits might allow for a better usage of the proposed method. Such idea, at best of knowledge, has not yet been experimented.

References

[1] B Jack Copeland. The modern history of computing. 2000.

[2] Simon Bone and Matias Castro. A brief history of quantum computing. *Imperial College in London*, 1997.

[3] Stefano Olivares. *Lecture Notes on Quantum Computing*. January 2020.

[4] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[5] Massimiliano Giacomin Giovanni Guida. *Fondamenti di Informatica*. FrancoAngeli, 2015.

[6] Stefano Forte and Luca Rottoli. *Fisica quantistica*. Zanichelli, 2018.

[7] Richard P Feynman. *The Feynman Lectures on Physics Vol 3*. Narosa, 1965.

[8] Lev D. Landau and Evgenij M. Lifsits. *Fisica Teorica 1 - Meccanica*. Editori Riuniti, 2010.

[9] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[10] Ville Lahtinen and Jiannis K Pachos. A short introduction to topological quantum computation. *SciPost Physics*, 3(3), 2017.

[11] Robert Raussendorf and Hans Briegel. Computational model underlying the one-way quantum computer. *arXiv preprint quant-ph/0108067*, 2001.

[12] Wittek Peter. *Quantum Machine Learning*. May 2014.

[13] Alexander Semenovich Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973.

[14] Peter W Shor. Fault-tolerant quantum computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 56–65. IEEE, 1996.

[15] Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM journal of research and development*, 5(3):183–191, 1961.

[16] Stefanie Hilt, Saroosh Shabbir, Janet Anders, and Eric Lutz. Validity of landauer's principle in the quantum regime. *arXiv preprint arXiv:1004.1599*, 2018.

[17] David P DiVincenzo. *Quantum Information Processing: Lecture Notes of the 44th IFF Spring School 2013*. Forschungszentrum, 2013.

[18] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[19] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 279–287, 2001.

[20] Vivien M Kendon, Kae Nemoto, and William J Munro. Quantum analogue computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1924):3609–3620, 2010.

[21] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

[22] Marco Lanzagorta and Jeffrey Uhlmann. Is quantum parallelism real? In *Quantum Information and Computation VI*, volume 6976, page 69760W. International Society for Optics and Photonics, 2008.

[23] Yu I Ozhigov. Quantum parallelism may be limited. *arXiv preprint arXiv:1610.01089*, 2016.

[24] Liming Zhao, Zhikuan Zhao, Patrick Rebentrost, and Joseph Fitzsimons. Compiling basic linear algebra subroutines for quantum computers, 2019.

[25] Dickens James. Quantum computing algorithms for applied linear algebra. Aug 2019.

[26] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning, 2020.

[27] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 122(4), Feb 2019.

[28] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), Apr 2008.

[29] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information and Computation*, 12, 2012.

[30] Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, Sep 2018.

[31] Dong An, Di Fang, and Lin Lin. Time-dependent unbounded hamiltonian simulation with vector norm scaling, 2020.

[32] F. Arute, K. Arya, R. Babbush, and et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 2019.

[33] A. V. Uvarov, A. S. Kardashin, and J. D. Biamonte. Machine learning phase transitions with a quantum processor. *Physical Review A*, 102(1), Jul 2020.

[34] Lov K Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 53–62, 1998.

[35] Marco A. Rodríguez-García, Isaac Pérez Castillo, and P. Barberis-Blostein. Efficient qubit phase estimation using adaptive measurements, 2020.

[36] Yiğit Subaşı, Rolando D. Somma, and Davide Orsucci. Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Physical Review Letters*, 122(6), Feb 2019.

[37] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017.

[38] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[39] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[40] Seth Lloyd. Quantum algorithm for solving linear systems of equations. *APS*, 2010:D4–002, 2010.

[41] Adrian Cho. Google claims quantum computing milestone, 2019.

[42] Francesco Petruccione Maria Schuld. *Supervised learning with Quantum computers*. Springer, 2018.

[43] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. *Advances in Artificial Intelligence Lecture Notes in Computer Science*, page 431–442, 2006.

[44] E. Miles Stoudenmire and David J. Schwab. Supervised learning with quantum-inspired tensor networks, 2017.

[45] P. Bruza and R. Cole. Quantum logic of semantic space: An exploratory investigation of context effects in practical reasoning. 2005.

[46] Diederik Aerts and Marek Czachor. Quantum aspects of semantic analysis and symbolic artificial intelligence. *Journal of Physics A: Mathematical and General*, 37(12):L123–L132, Mar 2004.

[47] Scott Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3089–3114, Sep 2007.

[48] Rocco A. Servedio and Steven J. Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004.

[49] Andrew W. Cross, Graeme Smith, and John A. Smolin. Quantum learning robust against noise. , 92(1), July 2015.

[50] Nader Bshouty and Jeffrey Jackson. Learning dnf over the uniform distribution using a quantum example oracle. *SIAM J. Comput.*, 28:1136–1153, 03 1999.

[51] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, 2018.

[52] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, and et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[53] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean. Power of data in quantum machine learning, 2020.

[54] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020.

[55] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python, 2019.

[56] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, Jun 2020.

[57] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[58] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. In *Advances in Neural Information Processing Systems*, pages 4134–4144, 2019.

[59] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[60] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum natural gradient. *Quantum*, 4:269, May 2020.

[61] Aram Harrow and John Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms, 2019.

[62] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[63] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

[64] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[65] Davide Anguita, Sandro Ridella, Fabio Rivieccio, and Rodolfo Zunino. Quantum optimization for training support vector machines. *Neural Networks*, 16(5-6):763–770, 2003.

[66] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

[67] Vasil S Denchev, Nan Ding, SVN Vishwanathan, and Hartmut Neven. Robust classification with adiabatic quantum optimization. *arXiv preprint arXiv:1205.1148*, 2012.

[68] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[69] Owen Jones. Spectra of simple graphs. *Whitman College*, 13:1–20, 2013.

[70] Sophus Lie. Theorie der transformationsgruppen i. *Mathematische Annalen*, 16(4):441–528, 1880.

[71] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[72] IBM Research. The ibm quantum experience. https://www.research.ibm.com/quantum, 2016.

[73] Salonik Resch and Ulya R Karpuzcu. Quantum computing: an overview across the system stack. *arXiv preprint arXiv:1905.07240*, 2019.

[74] Andrew Cross. The ibm q experience and qiskit open-source quantum computing software. *APS*, 2018:L58–003, 2018.

[75] Qiskit Development Team. Quantum-enhanced support vector machines. https://qiskit.org/documentation/tutorials/machine_learning/01_qsvm_classification.html, 2017.

[76] Takayuki Sakuma. Application of deep quantum neural networks to finance, 2020.

[77] D. Yumin, M. Wu, and J. Zhang. Recognition of pneumonia image based on improved quantum neural network. *IEEE Access*, 8:224500–224512, 2020.

[78] Fei Li and Guobiao Xu. Quantum bp neural network for speech enhancement.

[79] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J. Osborne, Robert Salzmann, Daniel Scheiermann, and Ramona Wolf. Training deep quantum neural networks. *Nature Communications*, 11(1), 2020.

[80] Quantum convolutional neural networks. 2018.

[81] Saurav Sutradhar et al. Chakraborty Simantini, Tamal Das. An analytical review of quantum neural network models and relevant research. 2020.

[82] S. C. Kak. Quantum neural computing. 1995.

[83] A. Narayanan T. Menneer. Quantum inspired neural networks. 1995.

[84] Dario Gerace Francesco Tacchino, Chiara Macchiavello and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. 2019.

[85] Hartmut Neven Edward Farhi. Classification with quantum neural networks on near term processors, 2018.

[86] Jarrod R. Mcclean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1), 2018.

[87] The mnist database.

[88] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven. Quantum algorithms for fixed qubit architectures, 2017.

[89] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[90] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[91] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[92] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

[93] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[94] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.

[95] Cupjin Huang, Michael Newman, and Mario Szegedy. Explicit lower bounds on strong quantum simulation. *arXiv preprint arXiv:1804.10368*, 2018.

[96] Andrea Mari. Quanvolutional neural networks. `https://pennylane.ai/qml/demos/tutorial_quanvolution.html`, 2019.

[97] Siham Tabik, Daniel Peralta, Andres Herrera-Poyatos, and Francisco Herrera. A snapshot of image pre-processing for convolutional neural networks: case study of mnist. *International Journal of Computational Intelligence Systems*, 10(1):555–568, 2017.