ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

Variational Quantum Splines: Moving Beyond Linearity for Quantum Activation Functions

CANDIDATE:
Matteo Antonio Inajetovic

SUPERVISOR:
Prof. Claudio Sartori

CO-SUPERVISOR:
Dott. Antonio Macaluso

Academic Year: 2021-22

Session 1st

Contents

List of Figures

List of Tables

Abstract

Activation functions within neural networks play a crucial role in Deep Learning since they allow to learn complex and non-trivial patterns in the data. However, the ability to approximate non-linear functions is a significant limitation when implementing neural networks in a quantum computer to solve typical machine learning tasks. The main burden lies in the unitarity constraint of quantum operators, which forbids non-linearity and poses a considerable obstacle to developing such non-linear functions in a quantum setting. Nevertheless, several attempts have been made to tackle the realization of the quantum activation function in the literature. Recently, the idea of the QSplines has been proposed to approximate a non-linear activation function by implementing the quantum version of the spline functions. Yet, QSplines suffers from various drawbacks. Firstly, the final function estimation requires a post-processing step; thus, the value of the activation function is not available directly as a quantum state. Secondly, QSplines need many error-corrected qubits and a very long quantum circuits to be executed. These constraints do not allow the adoption of the QSplines on near-term quantum devices and limit their generalization capabilities. This thesis aims to overcome these limitations by leveraging hybrid quantum-classical computation. In particular, a few different methods for Variational Quantum Splines are proposed and implemented, to pave the way for the development of complete quantum activation functions and unlock the full potential of quantum neural networks in the field of quantum machine learning.

Chapter 1

Introduction

The discoveries of quantum mechanics from the early decades of the twentieth century, combined with the power of the computational theories and computer science capable of changing the human life for decades, led Feynman [8] and Manin [20] to propose the concept of Quantum Computing as early as the 80s. More thoroughly, quantum physics phenomena like superposition, entanglement and interference have been exploited to give rise to a new computational approach which was based on the so called Qubits, rather than the classic binary bits. Qubits can assume not only values like 0 and 1, but also a combination of them, achieving a stronger computational power. Quantum Computation began attracting attentions thanks to quantum algorithms capable of solving problems, intractable by their classic counterparts. The most known algorithm, which represented an epochal turning point was the "Shor's Algorithm", able to solve integer factorization in polynomial time [35].

In the last decades, given the enormous success of AI and Machine Learning, many studies have been done to exploit the quantum computation power for Machine Learning, giving the birth to the Quantum Machine Learning (QML). The new QML algorithms based on NISQ (Noisy Intermediate-Scale Quantum) devices proved their efficiency in specific tasks, while the ones based on fault tolerant devices have yet to

wait until the the development of such great scale quantum computers.

This thesis focuses on the development of a methodology that can pave the way for the use of nonlinear activation functions in the field of QML. Indeed, in the literature there's a huge lack concerning quantum activation functions and it is well known how fundamental these nonlinear functions are for Machine Learning and AI. The main problem and obstacle to develop such functions lies in the unitarity and linearity nature of quantum operators, the bricks of quantum computing.

Macaluso et al [18] proposed a non-linear approximation method based on "Quantum Splines" to move towards quantum activation functions. However, the QSpline method had some limitations such as: exploitation of fault-tolerant algorithms, post-processing steps and lack of generalization.

The proposal of this thesis overcomes such limits: in the first place (Variational QSplines) it extends the QSplines method with a Variational Algorithm (NISQ-based) and eliminates the need to apply post-processing steps to obtain the result of the activation function estimation. Secondly (Generalized Variational QSplines), the methodology is expanded in a more generalized fashion with a new formulation, taking full advantage of the nowadays available quantum algorithms.

The thesis is divided as follows:

- Chapter 2: Introduction to Quantum Computation and Quantum Machine Learning.
- Chapter 3: Discussion about the related works and foundations of this thesis.
- Chapter 4: Divided in two parts for the Variational QSplines and the Generalized Variational QSplines methods, this section describes the methodologies and the implementations developed in this thesis work.

- Chapter 5: Discussion on the experiments, results and drawbacks obtained with the two proposed methods.
- Chapter 6: Conclusions

Chapter 2

Background

In this Chapter the basics of Quantum Computing will be explained, from the Postulates of Quantum Mechanics to Quantum Circuits and devices. Then, the chapter moves to Quantum Machine Learning, the field of this thesis project, by analyzing methods and models.

2.1 Quantum Computing

The history of quantum computing begins in the 1980s, when Feynman[8] and Manin [20] proposed to combine the discoveries of quantum mechanics, developed in the early decades of the twentieth century, with the theory of computation, which also brought innovation and progress through the studies of Turing, Von Neumann and many others. In the last thirty years, a lot of quantum algorithms have been created and many researches proved how, theoretically, Quantum Computers are able to solve certain problems intractable for classical computers. In parallel, there has also been strong technological progress in the development of these quantum computers, especially with regard to small devices that will be able to be used in specific contexts. However, most of quantum algorithms assume a perfect working quantum machine, that is something we will not have soon. Besides this, in the last years, there has been

tremendous progress in the experimental developments of quantum computers, with the possibility to access small machines that it is reasonable to think will be useful in future for specific cases.

2.1.1 Quantum Mechanics

In order to understand how the Quantum Computation works it is necessary to deal with the basics and the Four Postulates of the Quantum Mechanics. Before stating these crucial concepts, it is also important to introduce the mathematical notation, the Dirac or Bra-ket notation, commonly used to describe them. This notation is very useful when the vector space concerned is an Hilbert Space, like the one used to represent Quantum Systems.

In the Dirac notation one n-dimensional vector, the so called "ket", and its dual, the "bra", are written as:

$$v = |v\rangle = \begin{bmatrix} v_0 \\ v_1 \\ \dots \\ v_{n-1} \end{bmatrix} \quad \langle v| = v^\dagger = \begin{bmatrix} \overline{v_0} & \overline{v_1} & \dots & \overline{v}_{n-1} \end{bmatrix} \tag{2.1}$$

The "bra" is the "adjoint" of the "ket", a row vector made up of the complex conjugates of the elements of the "ket". In general, let's provide the complete definition of "adjoint" operators. Given a linear operator A on a Hilbert space V, it exists a unique operator $A^\dagger$, called "adjoint", s.t. for all $|v\rangle$ and $|w\rangle$ from V:

$$(|v\rangle, A|w\rangle) = (A^\dagger|v\rangle, |w\rangle). \tag{2.2}$$

An operator is called "Hermitian", when it is the "adjoint" of itself.

Now let's define some basic concepts of the Hilbert Space with the

Dirac notation which will be very useful dealing with Quantum Computation:

- Inner Product:

$$\langle a|b\rangle = a^{\dagger}b = \overline{a}_0 b_0 + \overline{a}_1 b_1 + \cdots + \overline{a}_{n-1} b_{n-1}. \tag{2.3}$$

- 2-Norm:

$$\| |v\rangle \|_2 = \sqrt{\langle v|v\rangle}. \tag{2.4}$$

- Tensor Product:

$$|a\rangle \otimes |b\rangle = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ \vdots \\ a_1 b_n \\ a_2 b_1 \\ \vdots \\ a_2 b_n \\ a_m b_1 \\ \vdots \\ a_m b_n \end{bmatrix} \tag{2.5}$$

Now, it is possible to state the four postulates of the Quantum Mechanics and their relation with Quantum Computation and Information [26].

Postulate 1

Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the "state space" of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.

The first postulate gives the basis for the definition of the fundamental Quantum Computation concept, the qubit. It is the simplest quantum mechanical system and has a two-dimensional state space. Given $|0\rangle$ and $|1\rangle$, forming an orthonormal basis for the state space, an arbitrary state vector in the state space can be written as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.6}$$

where α and β are complex numbers. The condition that $|\psi\rangle$ is a unit vector, $\langle\psi|\psi\rangle = 1$, is equivalent to $|\alpha|^2 + |\beta|^2 = 1$ and it's called normalization condition for state vectors.

Postulate 2

The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time $t2$ by a unitary operator U which depends only on the times t_1 and t_2,

$$|\psi'\rangle = U\,|\psi\rangle\,. \tag{2.7}$$

Concerning the Quantum Computing field, the second postulate introduces the concept of "quantum gate", the equivalent of the unitary operator, or unitary matrix. Furthermore, like the first postulate, it doesn't provide any information about the quantum system.

Postulate 3

Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability

that result m occurs is given by:

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2.8}$$

and the state of the system after the measurement is:

$$\frac{M_m\,|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}. \tag{2.9}$$

The measurement operators satisfy the completeness equation :

$$\sum_m M_m^\dagger M_m = I \tag{2.10}$$

and it expresses that the probabilities sum to one:

$$1 = \sum_m p(m) = \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle\,. \tag{2.11}$$

The third postulate describes the effects of measurement on quantum systems and thus, on quantum circuits. The possibility to measure an observable at the end of a quantum circuit is crucial to quantum computation and its utility.

Postulate 4

The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle\otimes|\psi_2\rangle\otimes\cdots\otimes|\psi_n\rangle$.

Finally, the fourth postulate shows the relation between state spaces of systems which are components of an overall state space.

Now, given the four Postulates, it is possible to describe three main properties of Quantum Computing: “superposition”, “entanglement” and

"interference".

Quantum superposition states that multiple quantum states can be superposed together returning another quantum state. Furthermore, every quantum state can be represented as a sum of distinct states. Dealing with quantum computing, this property shows how a quantum system exists in all its basis states at the same time and thus, allows to evaluate a function simultaneously on different inputs.

A state that can't be written as a product of states of its component systems, is called an entangled state. Furthermore, if the quantum state of a composite system cannot be described individually, the measurement of an observable from a sub-system determines simultaneously the value of the other sub-systems. Entanglement is a crucial for quantum computing applications like quantum teleportation and superdense coding.

Quantum interference derives from the dual wave-particle nature of quantum systems, but differs from classic interference because in quantum mechanics a wave function can interact with itself, is a complex function and the interference effect depends on the absolute value of the wave function squared rather then the amplitudes of two waves.

2.1.2 Qubits

As already mentioned through the first postulate, the qubit is the fundamental unit of Quantum Computing and Information. The main difference between qubits and classical bits relies on the fact that the qubit not only can have one of the two basis values, but also a superposition of them; in other words a linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.12}$$

where α and β are called "amplitudes" and are complex numbers.

According to the Third Postulate, the probability to measure $|0\rangle$ is $|\alpha|^2$ and the probability to measure $|1\rangle$ is $|\beta|^2$. Moreover, the state of a qubit is a unit-vector in the two dimensional complex vector space. This is due to the fact that, given the two possible states, the sum of their probabilities have to sum to one. The equation 2.12 can be also written as:

$$|\psi\rangle = e^{i\delta}(\cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle). \tag{2.13}$$

It is important to notice that, in 2.13, the term $e^{i\delta}$ has no observable physical consequences. Indeed, according to the Born Rule, the term $e^{i\delta}$ becomes equal to 1 dealing with the outcome of a measurement. For this reason:

$$\alpha = \cos\theta/2, \tag{2.14}$$

$$\beta = e^{i\phi}\sin\theta/2. \tag{2.15}$$

In 2.15 $e^{i\phi}$ corresponds to the so called "relative phase".

The two-dimensional complex vector space mentioned above is the so called Bloch Sphere 2.1.

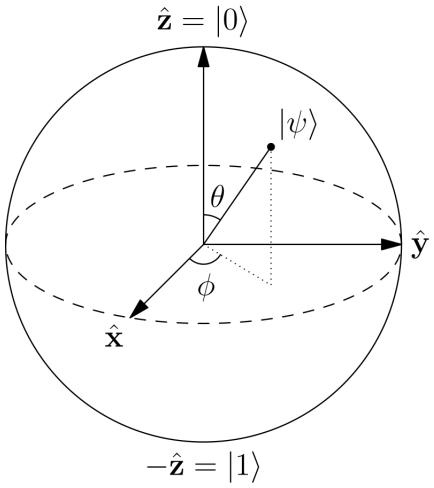


Figure 2.1: Bloch Sphere

When dealing with multiple qubits, the related state space grows by the power of 2; mathematically, given n qubits, the associated Hilbert Space will be 2^n-dimensional. For this reason, the state vector of a composite system made up of n qubits can be mathematically represented as explained by the fourth postulate of quantum mechanics. A simple way to understand the potential of the quantum computation with respect to the classical one, consists in considering the dimension of a state space generated by 200 qubits: 2^{200} is way bigger than the atoms in the universe.

Before going into the deepening of the Quantum Circuits and Quantum Devices, it is useful to introduce the basic quantum gates, the "bricks" of Quantum Computation.

2.1.3 Quantum Gates

In order to understand complex quantum circuits it is important to firstly start with the simplest units, the single qubit gates. Since a given qubit is represented as a vector, a single qubit gate (or mathematically, a unitary operator) corresponds to a 2×2 matrix. It is important to underline that is possible to apply only unitary operators to a single qubit. This fact derives from what explained with the first two postulates of Quantum Mechanics: the qubit state after the application of an arbitrary unitary operator must keep the normalization condition for state vectors.

Now let's define the most used and known single qubit gates:

- the Identity:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.16}$$

- the Pauli matrices X, Y, Z:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.17}$$

- the Hadamard gate:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.18}$$

The Pauli matrix X is the quantum equivalent of classic logic operator NOT. Indeed, given a quantum state $\psi = \alpha |0\rangle + \beta |1\rangle$ the X operator is able to switch its basis states:

$$X |\psi\rangle = X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \beta |0\rangle + \alpha |1\rangle . \tag{2.19}$$

Instead, the Pauli Y applies π rotations around the y-axis, while the Z gate applies a rotation of π around the z-axis. Let's see some examples of the application of the Hadamard gate to some qubit basis states and its ability to create superposition:

$$H |0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle , \tag{2.20}$$

$$H |1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle . \tag{2.21}$$

With the Pauli matrices it is possible to create an other group of interesting matrices, the "rotation operators" about the $\hat{x}$, $\hat{y}$ and $\hat{z}$ axis:

- Rotation Operators R_x, R_y, R_z:

$$R_x = e^{-i\theta X/2} = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{2.22}$$

$$R_y = e^{-i\theta Y/2} = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{2.23}$$

$$R_z = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \tag{2.24}$$

Now let's explain some essential relations and properties to deal with more complex quantum gates:

Theorem 1 [26] An arbitrary single qubit unitary operator U can be expressed, given α, β, γ and δ, as:

$$U = e^{i\alpha} R_{\hat{n}}(\theta) = e^{i\alpha} R_{\hat{x}}(\beta) R_{\hat{y}}(\gamma) R_{\hat{z}}(\delta). \tag{2.25}$$

This expression is crucial to prove that:

Corollary 1 [26] Given a unitary operator U and the Pauli operator X, then exist unitary operators A, B, C on a single qubit such that $ABC = I$ and $U = e^{i\alpha} AXBXC$.

Just as for classical computation, also for the quantum one conditional operations are essential. A classical example consists in "if A then do B, else C". The most known controlled gate is the Controlled-NOT operation (also called CX):

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.26}$$

$$CNOT : |c\rangle |t\rangle \rightarrow |c\rangle |t \oplus c\rangle \tag{2.27}$$

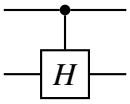
Trivially, from 2.27 and 2.26: if the control qubit c is set to $|1\rangle$ then the target qubit t is flipped, otherwise it doesn't change. The control

qubit instead never changes. Given the CNOT example, it is possible to extend the concept to a more general case.

An arbitrary Unitary Controlled Operation U, given a 2-qubit system with a control qubit c and a target qubit t, applies the following transformation:

$$C(U) : |c\rangle |t\rangle \rightarrow |c\rangle U^c |t\rangle \tag{2.28}$$

and, by means of circuits, it is represented as:



(2.29)

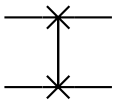
The controlled operation concept can also be extended to a multi qubit system thanks to Corollary 1. Given a system with $n+m$ qubits, a unitary operator U acting over m target qubits and n control qubits:

$$C^n(U) |x_1, .., x_n\rangle |\psi\rangle = |x_1, .., x_n\rangle U^{x_1,..,x_n} |\psi\rangle \tag{2.30}$$

The operator U is applied to the last m qubits (targets) if the first n (control) qubits are all equal to one; otherwise, nothing is done.

Other multi-qubit gates can be realized following the basic principles explained previously:

- the SWAP gate:



(2.31)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.32}$$

- the CZ gate:

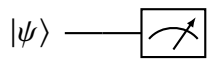
(2.33)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.34}$$

- For an exhaustive review on all the remaining multiple quantum gates see [26]

2.1.4 Measurement

Measurement is one of the most important concepts to study and create Quantum Circuits and derives from the third Postulate of Quantum Mechanics 2.8. It consists basically in transforming quantum information into classical one, in order to analyze and understand the output of the circuits.



(2.35)

Below are stated two crucial properties of quantum measurement:

- Measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit.
- Any unterminated quantum wires (qubits which are not measured) at the end of a quantum circuit may be assumed to be measured.

When we measure an observable on a quantum system, the final result is one eigenvalue of the observable. Thus, once we measure, the state vector of the system becomes one of the eigenvectors of the observable measured. To provide the output of a quantum cirucuit, we compute the

expectation value of a measurement. Thus, given the observale M, the expectation value will be:

$$E(M) = \sum_m m p_m = \sum_m m \langle\psi|P_m|\psi\rangle = \langle\psi|M|\psi\rangle . \tag{2.36}$$

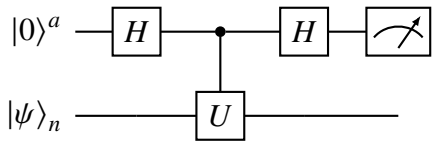
It is important to notice that the equalities above derive by the projective measurements properties (more insights in [26]) and the third Postulate.

In practice, while a classic bit can be accessed directly, for a qubit it is possible to get only a portion of the information it can encode. Furthermore, as already told through the first and the third postulates, we measure only the probability to get the state $|0\rangle$, $|\alpha|^2$, or the probability to get the state $|1\rangle$, $|\beta|^2$. An important typology of single qubit measurement is the so called "measurement in the computational basis". It has two outcomes depending on the two quantum operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. Both operators are Hermitian and together satisfy the completeness equation. Given $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probability to get 0 as outcome will be:

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \langle\psi|M_0|\psi\rangle = |\alpha|^2 \tag{2.37}$$

2.1.5 Quantum Circuits

Quantum circuits are models for quantum computation and the computation itself consists in a sequence of gates and measurements acting over the qubits of the system. In the Quantum circuits representation each line correspond to a qubit, and moving from left to right along the horizontal line we have the evolution of the quantum state in time. Let's now describe a classic and important circuit used in this thesis project, the Hadamard Test [2.38].



(2.38)

This circuit, given a unitary operator U acting over the space of the quantum state $|\psi\rangle$, computes the real part of the expected value $\langle\psi|U|\psi\rangle$. The first Hadamard Gate on the ancilla qubit(the one indexed with a) creates a superposition and consequently, the quantum state of the system becomes:

$$|\phi_1\rangle_{n+1} = (H \otimes I^n)\,|0\rangle\,|\psi\rangle_n = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\psi\rangle\,. \tag{2.39}$$

Secondly, as discussed previously, the operator U acts on the target qubit controlled by the ancilla qubit:

$$|\phi_2\rangle_{n+1} = CU\,|\phi_1\rangle_{n+1} = \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle_n + |1\rangle \otimes U\,|\psi\rangle_n). \tag{2.40}$$

Then another Hadamard gate is applied to the ancilla qubit:

$$|\phi_3\rangle_{n+1} = (H\otimes I^n)\,|\phi_2\rangle_{n+1} = \frac{1}{\sqrt{2}}(|0\rangle\otimes(I+U)\,|\psi\rangle_n + |1\rangle\otimes(I-U)\,|\psi\rangle_n) \tag{2.41}$$

. Lastly, by measurement, the probabilities to get the states $|0\rangle$ and $|1\rangle$ are respectively:

$$p(|0\rangle) = \frac{1}{4}\,\langle\psi|(I + U^\dagger)(I + U)|\psi\rangle \tag{2.42}$$

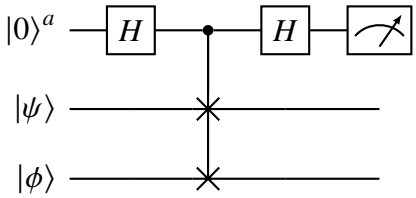
$$p(|1\rangle) = \frac{1}{4}\,\langle\psi|(I - U^\dagger)(I - U)|\psi\rangle \tag{2.43}$$

.

Finally, the expected value will be the difference of the two probabilities, $\mathbf{Re}\ \langle\psi|U|\psi\rangle$.

Another important circuit is the Swap Test 2.44, a building block for

quantum algorithms for inference [33] that can be seen as a particular case of the Hadamard Test.



(2.44)

This circuits behaves like the Hadamard test: in this case the controlled operation CU, is replaced by a Fredkin (also known as controlled SWAP) gate controlled by the ancilla qubit and acting over two quantum registers $|\psi\rangle$ and $|\phi\rangle$.

$$CSWAP\frac{1}{\sqrt{2}}(|0,\psi,\phi\rangle + |1,\psi,\phi\rangle) = \frac{1}{\sqrt{2}}(|0,\psi,\phi\rangle + |1,\phi,\psi\rangle) \tag{2.45}$$

Following the evolution of the quantum states through the circuit, as done for the Hadamard test, we get the probability to have the state $|0\rangle$:

$$p(|0\rangle) = \frac{1}{2} + \frac{1}{2}|\langle\phi|\psi\rangle|^2. \tag{2.46}$$

As it is possible to forecast from 2.46, with the Swap Test routine it is also possible to extract the dot-product between two quantum states, but at a cost of classical post-processing steps.

2.1.6 Quantum Devices

Now that all the basics of quantum computing and information have been explained it is also essential to underline some technological aspects related to quantum devices, focusing on long-term fault-tolerant quantum computers and near-term ones.

Fault-Tolerant Quantum Devices

Fault-tolerant devices are able to deliver uninterrupted service, even in the case of the failure of one or more components.

In order to build a quantum fault tolerant device, besides the number of qubits, it's important to guarantee their quality, accuracy and preserve their isolation while controlling the quantum system itself. Unfortunately quantum decoherence [37] represents a huge problem for the creation of quantum computers, since the nowadays technologies have still troubles in keeping both isolation and control for quantum devices. Therefore, large scale quantum computers, producing an amount of noise larger than the signals and thus, without a concrete error-correction, are not yet a reality.

NISQ Devices

As aforementioned, large scale fault tolerant quantum devices are not available nowadays, but a strong progress in the development of small-scale ones have been achieved. Indeed, the Noisy Intermediate-Scale Quantum (NISQ) [30] devices have become increasingly important for real world-application and studies. Obviously those devices as far from being an alternative to their classic counterparts and at the same time they cannot even execute many theorized quantum algorithms. Nevertheless, the NISQ devices are very important to face many real-world applications and also to test models which will be hopefully outstanding and relevant with future technologies. In the next section, one of the most promising and interesting fields of Quantum Computing, as well as scope of this thesis project, will be introduced.

2.2 Quantum Machine Learning

In this section, as the title suggests, the Quantum Machine Learning (QML) field is discussed, starting from the Classical Machine Learning and the Hybrid Variational algorithms concepts, to the nowadays frontiers of QML.

2.2.1 Introduction

There are two main approaches for Quantum Machine Learning, which can be differentiated depending on the kind of devices that is exploited for the task: NISQ [5] and faul-tolerant [32], both briefly described in the previous section. The main difference between the two approaches is that NISQ-based QML algorithms already have an immediate impact on real-world applications [13, 7], while fault-tolerant QML algorithms are more traditional and suffer from the same limitations of their technology base.

The Quantum Machine Learning goal is to combine the efficiency of quantum computing, which proved its information processing power in many cases of study [26], with the Machine Learning data processing, already known for its success in a huge amount of everyday tasks and researches, like image and speech recognition, pattern identification or strategy optimisation. Even if the quantum nature of the quantum computing algorithms makes them less clear and interpretable than their classical counterparts by means of information accessibility, many researchers believe that, once the physical technologies will allow to implement it, the quantum computing power will be able to revolutionize the intelligent data processing field [34]. Going practically, nowadays there are two main ways to merge quantum computing and machine learning: one consists in the application of well-known machine learning models to

improve quantum computing theory, while the other involves the adaptation of classic algorithms to quantum computers. The actual thesis work proposal lies in the latter branch since it deals with the implementation of quantum non-linear activation functions, one of the most important elements for Deep Learning and Neural Networks models. For this purpose, in the next subsection, the basic notions of Machine Learning and Neural Networks will be explained.

2.2.2 Classic Machine Learning

Machine Learning (ML) is a really wide sub-field of Artificial Intelligence, made up of a huge amount of methods and featured by as many applications. The goal of ML is to develop algorithms s.t. computers become able to learn patterns from a set of data by training them and without being explicitly programmed. It is mainly divided in three typologies:

- Supervised Learning - defined by the usage of labelled datasets and trained algorithms able to predict the outcomes (the labels) from the inputs (Examples: Classification, Regression).
- Unsupervised Learning - based on unlabelled datasets, its trained algorithms identify patterns from the input data (Examples: Clustering, Dimensionality Reduction).
- Reinforcement Learning - the trained algorithms are able to learn the optimal behavior in an environment in order to maximize a defined reward (Examples: QLearning).

Neural Networks(NNs) are ML computational models slightly inspired by biological neural networks. Although successful in many practical applications, the power of neural networks cannot be fully exploited. For example, the Universal Approximation Theorem [11], according to which neural networks can estimate any continuous bounded function,

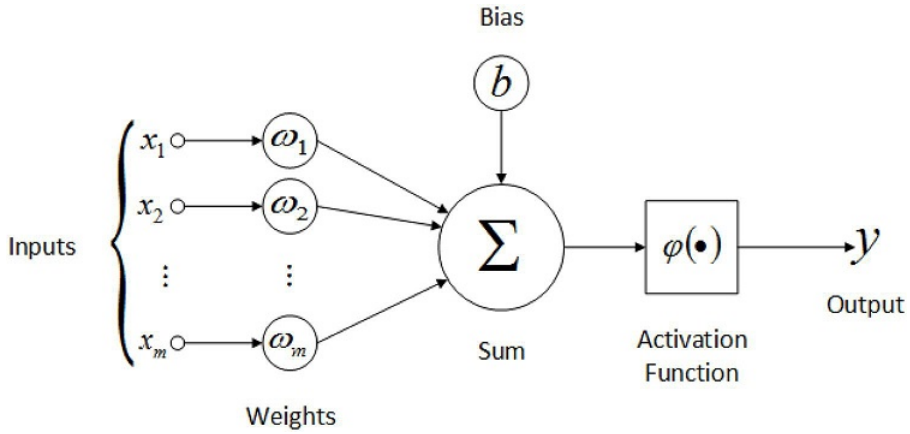


Figure 2.2: Artificial Neuron. Here φ is the activation function instead of σ.

is never exploited in practice due to the unfeasibility of training models with an exponentially large number of neurons in the hidden layer.

The basic unit of a Neural Network is the single neuron [24] 2.2, whose output y is computed as the sum of the weighted inputs and the bias, passed through the so called activation function σ:

$$y = f(x, W) = \sigma(\sum_{i=1}^{N} x_i w_i + b_i). \tag{2.47}$$

It is important to notice that here f is not the desired function to estimate with a ML model, it is just the function describing the single neuron. In most cases activation functions are non linear and thus able to adapt with different kind of data and scenarios, to perform better predictions.

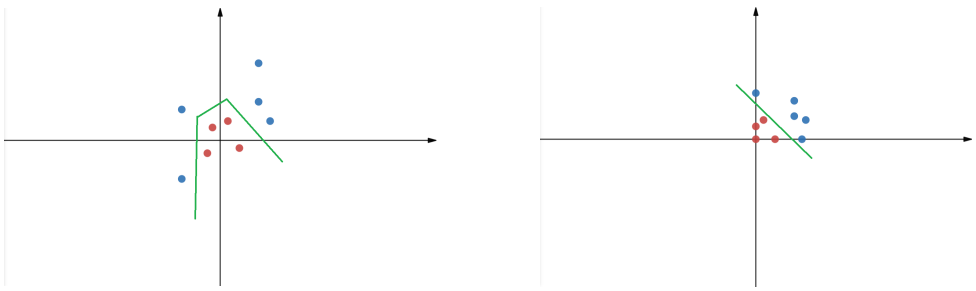


Figure 2.3: On the left a non linearly separable data distribution. On the right, the application of a neuron with a ReLU activation function over the same data in the so-called hidden space.

From the figure 2.3 it is possible to see a simplified example to explain the utility of non-linear activation functions. On the left we have non-linearly distributed data. With a linear activation function is not possible to linearly separate the red from the blue points. By applying a non-linear activation function, like the "ReLU" function 2.48 (or Rectified Linear Unit), we are able to move data points in a different space where they become linearly separable (like in the image on the right 2.3).

$$\phi(x) = max(0, x). \tag{2.48}$$

A more precise and deeper explanation about this topic (the XOR learning case) and Deep Learning in general can be found in [9].

Beyond the ReLU, there are many other non linear activation functions. Below we report the three remaining functions we used in this thesis work and from 2.4 it is possible to visualize them:

- The Sigmoid:
$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{2.49}$$
- The Hyperbolic Tangent or "Tanh":
$$\phi(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.50}$$
- The The Exponential Linear Unit or "Elu":
$$\phi(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{2.51}$$

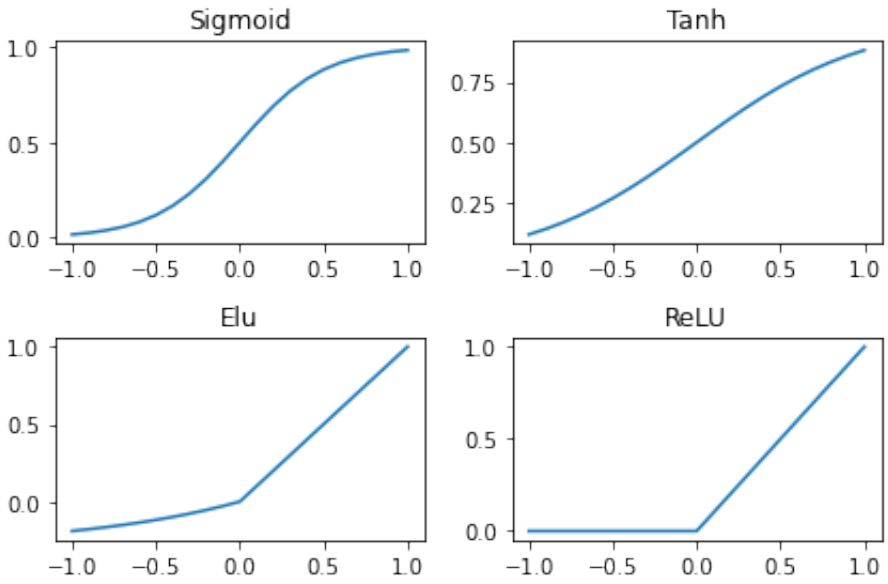


Figure 2.4: The four activation function stuided in this thesis project.

2.2.3 Hybrid Algorithms

Algorithms exploiting both classical and quantum computation are called hybrid algorithms [6] and have demonstrated a huge success in the Quantum Computation field. In particular, they showed their strength with QML by giving the chance to use NISQ devices instead of fault-tolerant ones and thus, become significant in real scenarios. Among these we can mention some very known algorithms like for example the Variational Quantum Eigensolver [28], used for chemistry applications, and hybrid algorithms for quantum simulations [14].

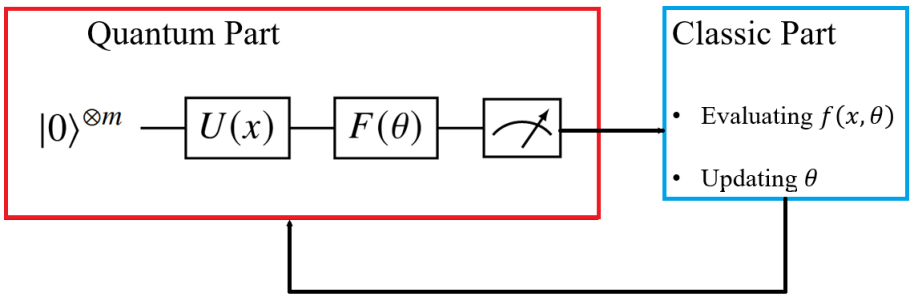


Figure 2.5: Hybrid quantum-classical approach for supervised learning

In general, as 2.5 shows, the classical part of these hybrid systems is exploited for the pre/post processing computations, evaluations and learning procedures, while the quantum part can be related to the "state preparation" (which will be discussed later), functions estimations and quantum measurements.

2.2.4 Supervised QML

Schuld and Petruccione [33] offer a clear and complete study about the Supervised Quantum Machine Learning field and it's the main reference of this sub-section. Additionally, also Benedetti et al. [2] gives a deep review about the Parametrized Quantum Circuits (PQC) for machine learning models, their typology and structure. We will focus now on the supervised models, used to tackle tasks like classification and regression. Given a dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ and a function $f : X \rightarrow Y$ able to map every $x^{(i)} \epsilon X$ to its related target $y^{(i)} \epsilon Y$, the goal of a supervised learning algorithm is to find the optimal parameter set θ^* s.t.:

$$\theta^* = \text{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} L(f(x^{(i)}, \theta), y^{(i)}) + R(\theta) \tag{2.52}$$

where L is the loss function to measure the error of the f estimation and R the regularization penalty function for undesired parameters values.

Hybrid quantum-classical algorithms for supervised learning follow the following schema:

- The Quantum circuit preparates the input data $x^{(i)}$ encoding it in a quantum state $\left|x^{(i)}\right\rangle$ through a quantum operator and computes the learning model's output $f(x^{(i)}, \theta)$.
- The Classic part of the algorithm evaluates the error through a cost function and updates the parameters θ depending on the error.

The most common approach to design the quantum part consists

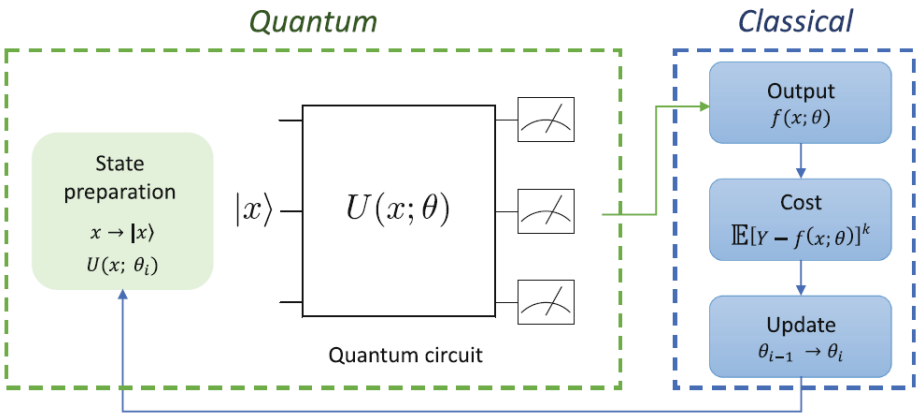
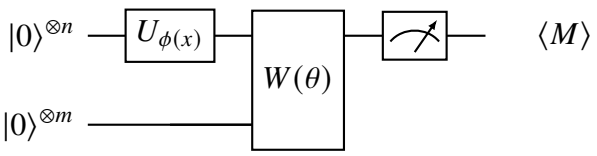


Figure 2.6: Image from [16]

basically in a first part, concerning the data encoding circuit able to translate the classical data X in quantum information and a second part to estimate the output of the function f. The two most known ways to tackle this task are the Variational Quantum Models(VQM) and the Quantum Kernel Estimation (QKE).

Variational Quantum Models

As aforementioned, for this kind of PQCs we have a data encoding circuit $U_{\phi(x)}$ able to encode x in a quantum state $|x\rangle$ and the Variational part $W(\theta)$, also called Ansatz, used to compute the observables to be measured, in order to obtain the model's output.



(2.53)

By increasing the number of qubits the Hilbert Space of the system increases exponentially and thus, the complexity of the gradient estimation to update the parameters grows. Indeed, with VQMs we have the so called Barren Plateaus problem: more the qubits used in the system,

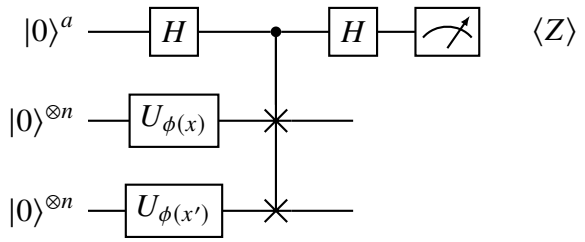
lower the probability to have a non-zero gradient along any direction [23].

Quantum Kernel Estimation

The idea behind QKE, or Kernel Methods, is to use a similarity function, or kernel, $K(x, x')$ in order to compute an inner product between two point x and x' in a feature space different from the one the two points belong to. The data encoding routine U_ϕ can be seen as a feature map which project the two data points to an higher order space vector where they can be more easy to classify thanks to a more explicit representation. So given a kernel $K(x, x')$ s.t.:

$$f(x, w) = \sum_{i=1}^{N} w_i K(x, x') \tag{2.54}$$

the goal of the QKE is to learn, classically, the set of weights w_i such that f produces correct forecasts. By means of quantum circuits the QKE is implemented through the already explained swap test routine:



(2.55)

where:

$$\langle Z\rangle = |\, \langle 0|U^\dagger_{\phi(x')} U_{\phi(x)}|0\rangle \,|^2. \tag{2.56}$$

By measuring the ancilla qubit $|0\rangle^a$ of this swap test routine, it is possible to evaluate the kernel product and thus, through a classic optimization part to find those w which minimize the error of the model. With respect to Variational Circuits, exploiting Kernel methods there is no need to deal with barren plateaus or to design Ansatzs [31].

2.2.5 State Preparation

$$|0\rangle^{\otimes n} \text{ —} \boxed{S(x)} \text{—} \quad |x\rangle \tag{2.57}$$

In the previous sub-section we already mentioned data encoding circuits without investigate them further. The state preparation process, namely the development of quantum circuits able to encode classic data in quantum states, is a crucial part of QML, as we saw with Variational Models and Kernel Methods. There are many ways to "prepare a state", but here we will discuss the two most known, namely Basis Encoding and Amplitude Encoding [33].

Basis Encoding

The Basis Encoding routine associates to a classic binary string a computational basis state of a n-qubit system. In this way, each classic bit is related to a qubit and consequently it is possible to perform quantum computation in parallel on all possible bit sequences thanks to the superposition property. The amplitude associated to each basis state doesn't encode information but gives the the probability to get the result of the computation for the measurement. For this reason, if we use this state preparation routine, the objective of the algorithm is to have as higher amplitude the one related to the basis state encoding the solution.

Given classic data in binary form, $(b_1, .., b_d)$ s.t. $b_i \epsilon \{0, 1\}$, the equivalent quantum state obtained through a basis encoding routine will be:

$$|x\rangle = |b_1, .., b_d\rangle\,, \tag{2.58}$$

where the number of qubits n_q is equal to d. For example, a classic bit string 001 will be encoded as $|001\rangle$.

Amplitude Encoding

Amplitude encoding, differently from the basis encoding routine, associates the required classical information to the amplitudes of the quantum state of the system and there are many ways to realize it. Given a classic data vector $x \epsilon R^{2^n}$ s.t. $\|x\|^2 = 1$, it can be represented with the following quantum state obtained with an amplitude encoding routine:

$$|x\rangle = \sum_{i=1}^{2^n} x_i |i\rangle . \tag{2.59}$$

Concerning instead a classical matrix $A \epsilon \mathbb{C}^{2^n \times 2^m}$ with entries a_{ij} s.t. $\sum_{ij} |a_{ij}|^2 = 1$, it can be encoded as follows:

$$|\psi_A\rangle = \sum_{i=1}^{2^n} \sum_{j=1}^{2^m} a_{ij} |i\rangle |j\rangle \tag{2.60}$$

An important feature of this state preparation typology is that, since the result of the computation of an algorithm can be encoded in one specific amplitude of the quantum system, the number of measurements scales with the number of the amplitudes differently from the basis encoding, where we have to measure all the qubits to get the desired result. One important example of amplitude encoding, as well as the one used for this thesis methods' implementation, is the Mottonen State Preparation [25]: its routine consists in the reverse operation of mapping the desired quantum state $|\psi\rangle$ to $|0\rangle^{\otimes n}$ through a quantum circuits featured by multiple controlled rotations.

2.2.6 Quantum Neural Networks

Mangini et al.[19] provide a comprehensive review of the actual attempts to reproduce, in a Quantum computing fashion, already existing Neural

Network models. In this subsection the QNN concept and some models will be mentioned. Comparing classical Neural Networks and PQCs, there can be found similarities: in both models we have a sequential information processing through parametrized layers and an iterative classical optimization procedure. For this reason it is possible to formulate a Quantum Neural Network as a PQC with multiple repetitions of layers:

$$U_{QNN}(\theta) = \prod_{i=L}^{1} U_i(\theta_i)W_i = U_L(\theta_L)W_L...U_1(\theta_1)W_1 \tag{2.61}$$

where $U(\theta)$ are variational gates, L is the number of layers and W_i are fixed quantum operations. In order to develop such QNNs an equivalent of the well known classic learning Universal Approximation Theorem [11] is needed. Many studies have been done in order to reproduce, in a quantum way, already existing NN models: concerning generative models there's a quantum formulation of the famous GANs [38], the already discussed Kernel methods, a quantum version for the Single Layer Perceptron [16], for Ensembles [17] and many others.

Another crucial topic for the development of QNNs is linked to the Activation Functions, fundamental feature of NNs neurons to deal with non linearly separable data. With Quantum Circuits, due to their unitary nature, is not possible to reproduce non linear functions and thus, neither the desired activation functions. The following chapter will discuss an attempt to create those functions with Quantum Circuits and lays the foundations of this thesis project contribution as well as mention the most important related works.

Chapter 3

Related Works

3.1 Fault-Tolerant Quantum Splines

In this section the Quantum Splines [18] work will be revised, since it is the starting point for the main contribution of this thesis project. The goal of the QSpline project was to tackle the Non-Linear Approximation problem through Quantum Splines, to overcome the linearity constraint linked to the unitarity of quantum systems and finally, to realize a quantum circuit able to generate Quantum Activation Functions. Before going into details of the QSpline idea, it is necessary to mention Regression Splines [12] and the HHL algorithm [10].

3.1.1 Regression Splines

Regression splines [12] are basis functions that can be interpreted as a combination of polynomials and step functions and can be used, just like other regression methods, to model non-linearities and fit the data. Given a set of input data X and a set of outputs Y let's recall the classic regression problem:

$$Y = f(X, \beta) + \epsilon, \tag{3.1}$$

where the goal is to find the function f that better fits the data, with a certain error ϵ. The Regression Spline approach involves the division of the range of X into K regions and, for each of these regions, a polynomial function have to fit the data and satisfy some smoothing conditions at the boundaries, the so called *knots*.

As already told, the regression splines are basis functions, which can be understood as a way to augment the input feature X through some transformations h_m and thus, allow to use linear models over the transformed input $h_m(X)$.

$$f(X, \beta) = \sum_{m=1}^{M} \beta_m h_m(X) \tag{3.2}$$

In particular, a B-spline [4] (B comes from basis) is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition. Any spline function S_d of degree d, on a given set of knots can be expressed as a linear combination of B-splines $B_{i,d}$ of that degree:

$$S_d(x) = \sum_i \alpha_i B_{i,d}(x). \tag{3.3}$$

Given a function f, its support is a subset of its domain containing elements not mapped to zero.

In the case of regression b-splines, the basis function will be a truncated power basis function:

$$h(x, \xi) = (x - \xi)_+^j = \begin{cases} (x - \xi)^j, \text{if } x > \xi \\ 0, \text{otherwise} \end{cases} \tag{3.4}$$

where ξ is the knot and j the degree.

The regression splines are used for the QSpline work to approximate a non-linear function y given a matrix S, containing the basis expansion

of the inputs $b(x_i)$, and the related spline coefficients β:

$$y = S\beta \tag{3.5}$$

where:

$$y_i = \sum_k \beta_k b_k(x_i). \tag{3.6}$$

So, given the matrix S and the vector y, to compute the spline coefficients a Quantum Linear Problem Solver is needed.

3.1.2 HHL

The Harrow, Hassidim and LLoyd (HHL) algorithm [10] is a quantum algorithm able to solve a linear problem of equations $Ax = b$: given an hermitian matrix A and a vector b, it is able to find x in polynomial time.

Given a NxN hermitian matrix A, it can be decomposed as follows:

$$A = \sum_{j=0}^{N-1} \lambda_j \left|u_j\right\rangle \left\langle u_j\right| \tag{3.7}$$

,

where $\left|u_j\right\rangle$ is the eigenvector of A; if A is invertible then:

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} \left|u_j\right\rangle \left\langle u_j\right| \tag{3.8}$$

,

$$|b\rangle = \sum_{j=0}^{N-1} b_j \left|u_j\right\rangle \tag{3.9}$$

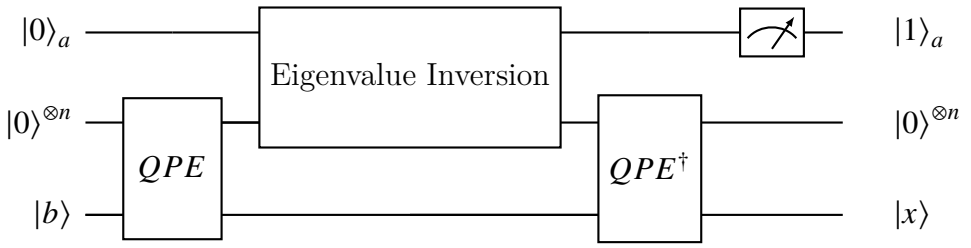
.

Given 3.8 and 3.9 the solution vector will be:

$$|x\rangle = A\,|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j \left|u_j\right\rangle \tag{3.10}$$

.

Considering that is possible to prepare the state $|b\rangle$, the solution $|x\rangle$ can be found by applying a unitary transformation e^{-iAt}. Given the mathematical background described above, let's explain now the main flow of the HHL algorithm. The Quantum Circuit for the algorithm 3.11 is made up of three registers: one for the binary representation of the eigenvalues of A, one for the solution $|x\rangle$ and one for the ancilla.



(3.11)

Then the following main steps are executed:

- Loading the data. $|b\rangle$ is encoded through a state preparation routine.
- Quantum Phase Estimation QPE [26]. Considering the representation of the unitary operator U:

$$U = e^{iAt} := \sum_{j=0}^{N-1} e^{i\lambda_j t} \left|u_j\right\rangle \left\langle u_j\right| \tag{3.12}$$

, then the QPE allows the transformation below:

$$|0\rangle \otimes |b\rangle \otimes |0\rangle \xrightarrow{QPE} \sum_{j=0}^{N-1} b_j \left|\lambda_j\right\rangle \left|u_j\right\rangle_n \otimes |0\rangle \tag{3.13}$$

- Eigenvalue inversion. This procedure applies a conditioned rotation by exploiting the auxiliary qubit:

$$\sum_{j=0}^{N-1} b_j \left|\lambda_j\right\rangle \left|u_j\right\rangle_n \otimes |0\rangle \rightarrow \sum_{j=0}^{N-1} b_j \left|\lambda_j\right\rangle \left|u_j\right\rangle_n \left(\sqrt{1-\frac{C^2}{\lambda_j^2}}\,|0\rangle + \frac{C}{\lambda_j}\,|1\rangle\right) \tag{3.14}$$

- $QPE^{\dagger}$. Application of the inverse of the QPE to isolate the solution and obtain the state:

$$\sum_{j=0}^{N-1} b_j \left|u_j\right\rangle_n \left(\sqrt{1-\frac{C^2}{\lambda_j^2}}\,|0\rangle + \frac{C}{\lambda_j}\,|1\rangle\right) \tag{3.15}$$

- Measuring (Rejection Sampling). Finally, measuring the ancilla, if we obtain $|1\rangle$ the result will be inversely proportional to λ_j and the state will correspond to the solution of the system.

However, the HHL has a strong limitation [1]: even if it seems to be promising with future large-scale computers, with nowadays NISQ computers it can be executed only with small-scale problems (2×2). Now that the preliminary concepts have been explained, namely the Regression Splines and the HHL algorithm, in the following section the heart of the QSpline project will be uncovered.

3.1.3 QSplines

As already mentioned at the beginning of this section, the aim of the QSplines is to move beyond linearity and succeed in approximating non-linear functions (e.g. Activation Functions for Neural Networks) through Quantum Splines. In this subsection the main steps of this work will be described.

In order to represent the non-linear function it is used a B-Spline regression and the goal is to find the optimal set of parameters, the set of

spline coefficients, to minimize the Residual Sum of Squares. The process to find those parameters can be seen as a ridge regression problem where observed variables are augmented with polynomials.

Since the polynomial basis function used have degree equal to 1, in each interval bounded by each subsequent pair of knots the data is fitted by a line. The vector $Y = \{y_1, .., y_K\}$, containing the values of the non-linear function to approximate, will be related to the inputs $X = \{x_1, .., x_K\}$ through the following linear system:

$$Y_{1\times K} = S_{K\times K}\beta_{1\times K} \implies \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_K \end{bmatrix} = \begin{bmatrix} S_1 & 0 & \dots & 0 \\ 0 & S_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & S_K \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_K \end{bmatrix}. \tag{3.16}$$

where S contains K S_k diagonal matrices representing the B-Spline basis expansion of the inputs $[x_{k,0}, x_{k,1}]$ and β is the vector of the splines coefficients. To find the solution of this linear system, the spline coefficients, the HHL algorithm is used.

$$S_k |\beta_k\rangle = |y_k\rangle \xrightarrow{HHL} |\beta_k\rangle \approx S_k^{-1} |y_k\rangle \tag{3.17}$$

Afterwords, the spline coefficients have to interact with the inputs to produce the estimates of Y. For this purpose, the swap-test routine is exploited:

$$|\beta_k\rangle \left|x_{i,k}\right\rangle |0\rangle \xrightarrow{swap-test} |a\rangle |b\rangle \left|f_{i,k}\right\rangle \tag{3.18}$$

.

By measuring the amplitudes of the ancilla qubit of the swap-test quantum circuit 2.44, we can find the estimate $\hat{Y}$ of the non-linear function evaluated in X. Due to the nature of the swap-test routine, to retrieve the final estimation the following post-processing procedure is

needed:

$$\left|f_{i,k}\right\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.19}$$

$$\alpha = \frac{1}{2} + \frac{|\hat{y}_{i,k}|^2}{2}. \tag{3.20}$$

Results

In the following table we have the Residual Sum of Squares (RSS) of both hybrid and full quantum methods with respect to the true activation functions studied: Sigmoid, Tanh, ReLU and Elu. In the hybrid case the product to compute $\hat{y}_{i,k}$ is done classically, while for the full quantum case instead through the swap test.

Activation function	QSplines (hybrid)	QSplines (full quantum)
Sigmoid	0.01	.75
Tanh	0.06	1.12
ReLU	0.14	8.16
Elu	0.12	7.06

Table 3.1: RSS scores. Fault-Tolerant QSplines

The results for the estimations of hybrid and full quantum QSplines are illustrated respectively in Fig. 3.1 and in Fig. 3.2, both taken from [18].

Discussion

The QSpline approach explained, suffers from the drawbacks and the limitations of the two quantum procedures applied: the inability of the HHL to deal with large linear systems on NISQ devices and the post processing process to extract the dot product after the swap-test. Indeed, if the goal is to realize a quantum activation function, then the values of $\hat{Y}$ should come directly from the quantum state and not through a classical post processing step. Furthermore, the proposed QSplines require ad-hoc

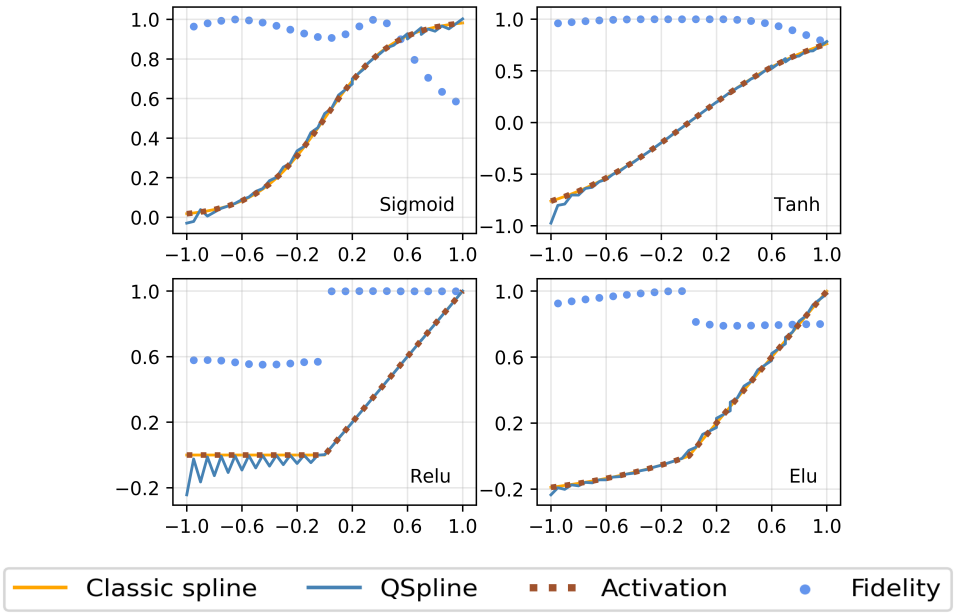


Figure 3.1: Hybrid QSpline.

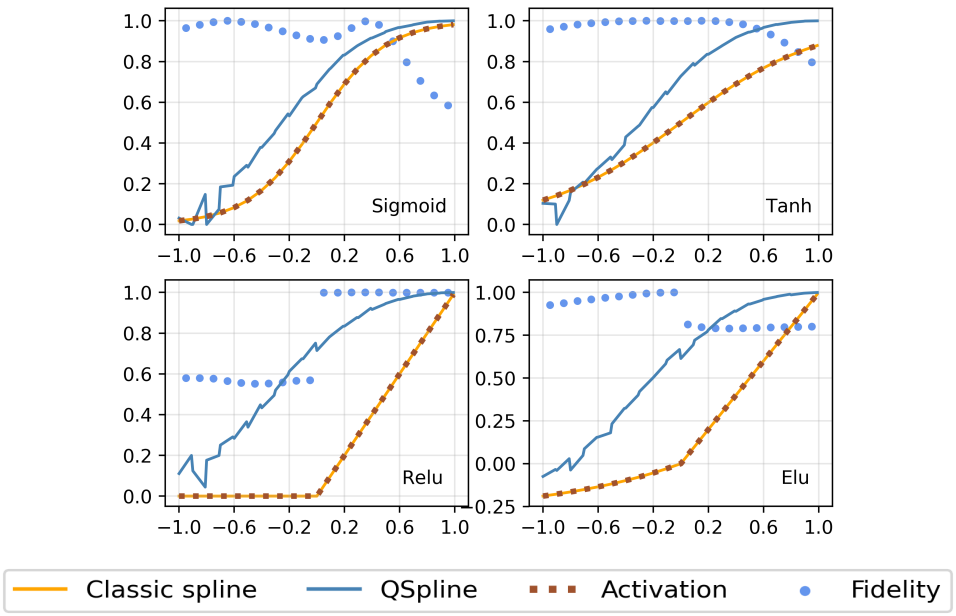


Figure 3.2: Full QSpline.

formulation for the basis expansion matrix and cannot be generalized easily.

3.2 VQLS - Variational Quantum Linear Solver

.

The most part of nowadays quantum algorithms for linear problems solving, just like the HHL [10], cannot be implemented on NISQ devices, due to the depth of their circuits. Prieto et al.[3], with their Variational Quantum Linear Solver(VQLS) algorithm, proposed a method to solve linear systems of equations with a variational hybrid quantum-classical approach, executable on near-term quantum computers. The VQLS algorithm, given a matrix A and a vector b is able to prepare a state $|x\rangle$ such that:

$$A|x\rangle \propto |b\rangle \tag{3.21}$$

where:

$$|x\rangle = \frac{\sum_i x_i |i\rangle}{\| \sum_i x_i |i\rangle \|_2}, \tag{3.22}$$

$$|b\rangle = \frac{\sum_i b_i |i\rangle}{\| \sum_i b_i |i\rangle \|_2}. \tag{3.23}$$

In order to understand the VQLS algorithm and how it is able to prepare a state linked to the solution of a linear problem, it is necessary to discuss its main flow.

Starting with the inputs of the algorithm, the matrix A is written as a linear combination of unitary matrices A_l:

$$A = \sum_{l=0}^{L} A_l c_l \tag{3.24}$$

where c_l are complex numbers. Then the vector b is prepared through a proper sequence of gates U, such that $U|0\rangle = |b\rangle \propto b$.

Another crucial part of the algorithm is the Variational Circuit, or

Ansatz, able to prepare the solution $|x\rangle$:

$$V(\theta)\,|0\rangle = |x(\theta)\rangle\,. \tag{3.25}$$

From here is already possible to broadly perceive how the hybrid quantum-classical approach works: the classical part of the computation optimizes a cost function by adjusting the θ parameters of the variational circuit.

A reasonable cost function for this algorithm should measure the overlap between $|\psi\rangle = A\,|x(\theta)\rangle$ and $|b\rangle$. In other words the state prepared by the Ansatz, the desired solution, "multiplied" by A, should match with the state $|b\rangle$. Mathematically the global cost function should follow the relation:

$$C_G = 1 - |\,\langle b|\Psi\rangle\,|^2 \tag{3.26}$$

where $|\Psi\rangle = |\psi\rangle\,/\sqrt{\langle\psi|\psi\rangle}$. Equivalently $|\Psi\rangle$ should match $|b\rangle$:

$$\Psi = \frac{A\,|x(\theta)\rangle}{\langle x(\theta)|A^\dagger A|x(\theta)\rangle} \approx |b\rangle\,. \tag{3.27}$$

Since this global function has gradients that vanish exponentially with the number of qubits [3], a local version of it was proposed. Starting from the explicit version of 3.26,

$$C_G = 1 - \frac{\sum_{l,l'} c_l c_{l'}^*\,\langle 0|V^\dagger A_{l'}^\dagger U|0\rangle\;\langle 0|U^\dagger A_l V|0\rangle}{\sum_{l,l'} c_l c_{l'}^*\,\langle 0|V^\dagger A_{l'}^\dagger A_l V|0\rangle} \tag{3.28}$$

if we substitue the projector $|0\rangle\langle 0|$ from 3.28 with $\frac{1}{2} - \frac{1}{2n}\sum_{j=0}^{n-1} Z_j$, a local version of 3.26 is obtained:

$$C_L = \frac{1}{2} - \frac{1}{2n}\frac{\sum_{j=0}^{n-1}\sum_{l,l'} c_l c_{l'}^*\,\langle 0|V^\dagger A_{l'}^\dagger U Z_j U^\dagger A_l V|0\rangle}{\sum_{l,l'} c_l c_{l'}^*\,\langle 0|V^\dagger A_{l'}^\dagger A_l V|0\rangle}. \tag{3.29}$$

This new formulation of C is called local in the sense that the numerator of the cost function expression can be computed "qubit-by-qubit" (due to the sum of over j) and not globally over the whole system.

The last expression can be rewritten as:

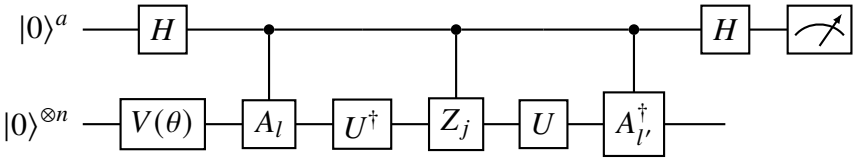
$$C_L = \frac{1}{2} - \frac{1}{2n} \frac{\sum_{j=0}^{n-1} \sum_{l,l'} c_l c_{l'}^* \mu_{l,l',j}}{\sum_{l,l'} c_l c_{l'}^* \mu_{l,l',-1}} \tag{3.30}$$

where

$$\mu_{l,l',j} = \langle 0|V^\dagger A_{l'}^\dagger U Z_j U^\dagger A_l V|0\rangle \,. \tag{3.31}$$

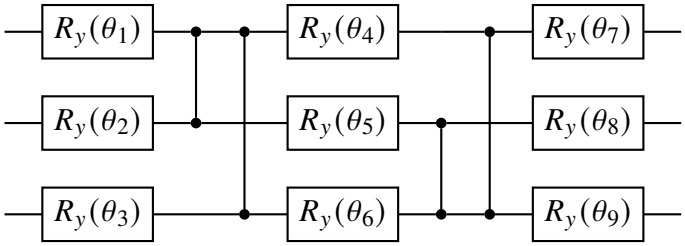
$\mu_{l,l',-1}$ stands for the same expression 3.31 where Z_j is replaced with the identity and thus, $\mu_{l,l',-1}$ corresponds to $\langle 0|V^\dagger A_{l'}^\dagger A_l V|0\rangle$.

Thanks to this version of the local cost function, it is possible to compute it through the following Hadamard Test:



(3.32)

So, this Local Hadamard test is used to compute $\mu_{l,l',j}$ for all the combinations of the variables l, l' and j to compute the total expected value of the cost function. It is important to notice how the circuit changes depending on l, l': each time there can be different gates representing different unitary matrices A_l. The variational circuit $V(\theta)$ proposed by the VQLS paper is a Fixed-structure layered Hardware-Eficient Ansatz. Furthermore, for each layer there are controlled-Z gates working on alternating pairs of neighboring qubits and each one of them is preceded and followed by a qubit rotation around the y-axis. From 3.33 it is possible to see an example of the proposed Ansatz, in the case of $n = 3$ qubits:



(3.33)

where $\theta_1, .., \theta_9$ are the variational parameters to optimize through the VQLS.

Given the value of the cost function, the classical part of the hybrid approach has to minimize it by changing the parameters of the Ansatz. The paper [27] gives an exhaustive analysis about the possible optimizers, depending on the number of qubits and the level of noise in the simulation (State vector, QASM or QASM+Noise).

Once the cost function is minimized, we get the optimal parameters θ_{opt} and then the Ansatz is able to prepare the state $\left|x(\theta_{opt})\right\rangle$:

$$V(\theta_{opt})\,|0\rangle = |x\rangle = \frac{x}{\|x\|_2}. \tag{3.34}$$

3.3 Quantum Activation Functions for Quantum Neural Networks

In a recent paper proposed by Maronese et al. [22] the same problem of this thesis project was tackled with a completely different approach. They developed a Quantum Perceptron through a n-to-2^n encoding model featured by inputs, weights and bias belonging to the real interval from -1 to 1. The algorithm proposed is able to produce activation functions exploiting only reversible operations and an iterative computation of all the powers of the inner product up to an order d. The estimation of the activation function is done, given those powers, following its d_{th} order Taylor series. One drawback of this approach consists in the dependence of the desired Activation Function's number of qubits on the related Quantum Perceptron and its number of inputs: for each input they need one qubit. Furthermore, the quantum circuit able to produce an Activation Function's output requires $n + d$ qubits, where n depends on the inputs and d on the order of the polynomial. Therefore, given 1 input for the SLP, the final circuit should have at least $d+1$ qubits and all the experiments with real Activation Functions they made have at least $d = 3$ and at most $d = 10$, requiring a lot of qubits. This thesis work proposes a method that, as well as being shallower by means of number of gates and circuit depth (once that the Ansatz has been optimized), doesn't depend on the SLP number of inputs and is able to produce an Activation Function's output with 3 qubit.

Now that we have resumed all the Quantum Machine Learning fundamentals and all the related topics and works, in the next chapter the contributions of this thesis work will be discussed.

3.4 Contribution

The contribution of this dissertation is divided into two parts. The first part proposes the QSplines[18] in the context of hybrid quantum-classical computation and reinterprets it via a variational approach without post-processing steps. The second and most important part proposes a generalized version of QSplines, with a novel basis expansion matrix formulation, making a step forward in the research of quantum activation functions. Importantly, the advantages of the proposed methodologies are two fold. Firstly, the proposed VQSplines overcome the unitarity constraint of quantum computation using near-term quantum devices. Secondly, VQSplines fill the gap for quantum activation function in the context of MAQA framework [16, 15], which aims to leverage the theoretical property of a classical neural network to be a Universal approximator and deliver its quantum counterpart.

Chapter 4

Variational Quantum Splines

Divided in two parts, this chapter contains the methodologies and implementations of this thesis project contributions.

4.1 A Variational algorithm for QSplines

In this section a Variational approach to the QSplines [18], named Vartional QSplines (VQSplines), will be described. The motivation behind this work relies upon the drawbacks about the HHL algorithm [10] and the swap test analyzed in the previous chapter 2.1.5. The goal is to realize a a full quantum circuit, through an hybrid quantum-classical procedure, able to estimate a non-linear function. For this purpose, just like the QSplines method, the final circuit will be made up of a first part computing the spline coefficients and a second one able to prepare a state containing the inner-product which should represent the non-linear function estimation.

4.1.1 Methodology

Let's recall the Linear System of equations proposed by Macaluso et al.[18] and showed in the previous chapter:

$$Y_{1\times K} = S_{K\times K}\beta_{1\times K} \implies \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_K \end{bmatrix} = \begin{bmatrix} S_1 & 0 & \dots & 0 \\ 0 & S_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & S_K \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_K \end{bmatrix}. \tag{4.1}$$

The problem can be solved exploiting B-Splines and Diagonal Block Matrix properties by decomposing it in K $2x2$ linear systems given a set of knots T s.t. $dim(T) = K$. Subsequently, the pseudocode procedure 1 used to decompose the problem in the K S_k matrices (given the inputs set X, the estimations set Y and the knots set T) is called "SubSystemsMatrices(X, Y, T)".

In 4.1 each element of Y is a couple of values of the non linear function ($y_k = [y_{k,0}, y_{k,1}]$), S_k represents the basis expansion of the inputs $x_{k,0}, x_{k,1}$ [4.2] and β_k is the couple of the spline coefficients related to the k_{th} sub-problem. In this approach only polynomials of degree 1 are used for the basis expansion.

$$S_k = \begin{bmatrix} 1 & x_{k,0} \\ 1 & x_{k,1} \end{bmatrix} \tag{4.2}$$

Thus, each time we have to solve the following linear sub-system:

$$y_k = S_k\beta_k = \begin{bmatrix} y_{k,0} \\ y_{k,1} \end{bmatrix} = \begin{bmatrix} 1 & x_{k,0} \\ 1 & x_{k,1} \end{bmatrix} \begin{bmatrix} \beta_{k,0} \\ \beta_{k,1} \end{bmatrix} = \begin{bmatrix} \beta_{k,0} + \beta_{k,1} * x_{k,0} \\ \beta_{k,0} + \beta_{k,1} * x_{k,1} \end{bmatrix}. \tag{4.3}$$

A first limitation can be already forecasted: for each couple of subsequents y_i and y_{i+1} from Y, due to the composition of the problem $y_{j,0}$ is equal to $y_{i,1}$. Reasonably, once we'll employ quantum circuits to solve

separated sub-systems and estimate separately each couple $\hat{y}_k$, the above mentioned condition will be difficult to satisfy, but let's continue the explanation of the method.

To solve these linear systems, the VQLS algorithm is adopted rather than the HHL. The former, with respect to the latter, can be implemented on NISQ devices even for large systems and is also faster. As already mentioned in the Chapter 3 (3.22, 3.23), the vector y_k, in order to solve the linear system, has to be normalized (4.4), and so will be the solution vector β_k.

$$y'_k = y_k / \|y_k\|_2 \tag{4.4}$$

The VQLS algorithm is able to compute the weights for its Ansatz and uses them to produce the solution of the linear system β_k, the desired B-spline coefficients:

$$S_k \left|\beta'_k\right\rangle = \left|y'_k\right\rangle \xRightarrow{VQLS} \left|\beta'_k\right\rangle = \beta'_{k,0} \left|0\right\rangle + \beta'_{k,1} \left|1\right\rangle \approx S_k^{-1} \left|y'_k\right\rangle \tag{4.5}$$

Afterwards, the coefficients β_k have to interact with the x_k point to produce the desired output y_k 4.3. According to [21] it is possible to compute the inner product between the two quantum states:

$$\left|\beta'_k\right\rangle = A \left|0\right\rangle_n , \tag{4.6}$$

$$\left|x'_k\right\rangle = B \left|0\right\rangle_n . \tag{4.7}$$

A and B are the amplitude encoding routines for the coefficients β and the vector x' respectively; x' stands for the normalized vector, used to apply the inner product. The equation 4.8 describes the final state of the inner product quantum circuit, while 4.9 shows the relation between the amplitude in the state $|0\rangle$ and the desired product.

$$B^\dagger A \left|0\right\rangle_n = a_o \left|0\right\rangle_n + \sum_{i=1}^{N-1} a_i \left|i\right\rangle_n \tag{4.8}$$

$$\hat{y}_k = a_0 = \left\langle x'_k \middle| \beta'_k \right\rangle = \left\langle x'_k \right| BB^\dagger \left| \beta'_k \right\rangle = \left\langle 0 \right| B^\dagger A \left| 0 \right\rangle \tag{4.9}$$

It is important to underline that this kind of inner-product, w.r.t to the swap test, encodes the final value directly in the amplitude of the state $|0\rangle$ and doesn't need the post-processing step used by [18] and discussed in section 2.1.5. Finally, by computing all the single estimates $\hat{y}_k$ we get the final set of estimations $\hat{Y}$. After this general description of the proposed method, summarized in the algorithm 1, let's now move to its practical implementation.

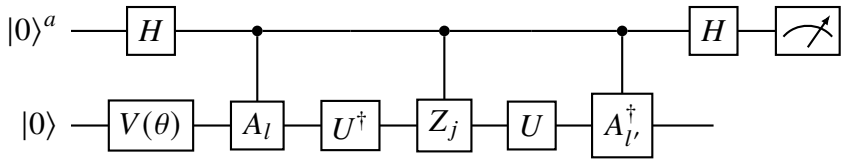
Algorithm 1 VQSplines pseudocode. I: maximum number of iterations. X fixed inputs. Y classic outputs. T knots set. ϵ is a lower bound for the variation of the cost function: if for a certain number of steps $\Delta C_L < \epsilon$ the cost function is in its minima and thus, following the "or" condition, the optimization procedure is stopped.

$S_{K\times K} \leftarrow$ SubSystemsMatrices(X, Y, T)
for k in K do
  $\theta_k \leftarrow$ Initialize()
  $|y_k\rangle \leftarrow$ Mottonen(y_k)
  while $i < I \vee \Delta C_L < \epsilon$ do
    $C_L(\theta_k) \leftarrow$ HadamardTest$(\theta_k, S_k, |y_k\rangle)$
    $\theta_k \leftarrow$ Update(θ_k)
  end while
  $|\beta_k\rangle \leftarrow$ Ansatz(θ_k)
  β.append(β_k)
end for
for x_k, β_k in X, β do
  for i in $len(x_k)$ do
    $x'_{k,i} \leftarrow [1, x_{k,i}]$
    $\hat{y}_{k,i} \leftarrow$ InnerProduct$(x'_{k,i}, \beta_k)$
    $\hat{Y}$.append$(\hat{y}_{k,i})$
  end for
end for
return $\hat{Y}$

4.1.2 Implementation

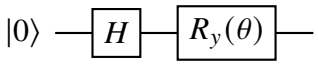
In this sub section the Quantum Circuits and the process exploited to implement the Variational QSplines are described. The K sub-problems described previously can be solved with only one qubit with the VQLS, since we need only one qubit to encode a bidimensional vector like y_k and β_k.

The VQLS algorithm used to solve the aforementioned 2×2 systems is based on a simple Hadamard test (pseudocode 1: "HadamardTest(θ,S_k,y_k)"):



(4.10)

- $V(\theta)$ is the following Variational Circuit or Ansatz (pseudocode 1: "Ansatz(θ)"):



(4.11)

  So, there is one weight for each sub-system.

- A_l are the gates used to encode the information related to the S_k matrix 3.24. In particular, the matrix S_k can be decomposed by means of a linear combination of unitary matrices:

$$S_k = \begin{bmatrix} 1 & a \\ 1 & b \end{bmatrix} = \sum_{l=0}^{3} A_l c_l = Ic_0 + Xc_1 + Zc_2 + R_y(3\pi)c_3 \tag{4.12}$$

  where I, X and Z are the known Pauli Matrices and R_y is the y-rotation matrix. The coefficients of the linear combination are computed as follows:

$$c_0 = (b+1)/2 \tag{4.13}$$

$$c_1 = (a+1)/2 \tag{4.14}$$

$$c_2 = (1 - b)/2 \tag{4.15}$$

$$c_3 = (a - 1)/2 \tag{4.16}$$

- U is the amplitude encoding routine used to encode y'_k to $\left|y'_k\right\rangle$, the Mottonen State Preparation [25] (pseudocode 1: "Mottonen(y_k)"):

$$U\left|0\right\rangle = \left|y'_k\right\rangle \tag{4.17}$$

.

Following the VQLS schema, the circuit (4.10) computes the expected real value of the cost function used to optimize the weights θ for the Ansatz(4.11). As a results of the VQLS, we obtain a quantum state whose amplitudes encode the solution of the linear system (4.18).

$$V(\theta_{k,opt})\left|0\right\rangle = \left|\beta'_k\right\rangle = \beta'_{k,0}\left|0\right\rangle + \beta'_{k,1}\left|1\right\rangle \tag{4.18}$$

The optimization part of this work, relying on the classic computation part of the Hybrid Approach behind Variational Models, is carried out with the COBYLA optimizer [29], suggested by [27] for VQLS state vector simulations.

Combining the VQLS and the quantum Inner Product, there's no need to measure the coefficients from the Ansatz and then encode them through the routine A (4.9) for the inner product: the optimization part of the VQLS algorithm returns the weights for the Ansatz, thus the latter can be used directly instead of the amplitude encoding routine.

To implement the Inner product (pseudocode 1: "InnerProduct(x,β)"), the equation 4.9 will change. Looking at the last part of the 4.3 expression, the inner product will be computed between $\beta'_k = [\beta'_{k,0}, \beta'_{k,1}]$ and

$x'_{k,i}$, which is $[1, x_{k,i}]$ normalized(4.19).

$$x'_{k,i} = \left[1 \; x_{k,i}\right] / \|[1 \; x_{k,i}]\|_2 \tag{4.19}$$

So the Inner Product circuit will be the following one:

$$|0\rangle \; \boxed{V(\theta_{k,opt})} \; \boxed{B^\dagger} \; a\,|0\rangle + b\,|1\rangle \tag{4.20}$$

$$a = \langle 0|\, B^\dagger V(\theta_{k,opt})\, |0\rangle = \left\langle x'_{k,i} \middle| \beta'_{k,i} \right\rangle \tag{4.21}$$

Thus, the amplitude a is associated to the estimation of the $y_{k,i}$ value:

$$\hat{y}_{k,i} = \left\langle x'_{k,i} \middle| \beta'_{k,i} \right\rangle \approx \beta'_{k,0} + \beta'_{k,1} * x'_{k,i} \tag{4.22}$$

So, this Variational QSpline method can be summed up by looking at the schema below:

$$\begin{aligned}
&|0\rangle \; \boxed{V(\theta_{1,opt})} \; \boxed{B_1^\dagger} \quad \hat{y}_1\,|0\rangle + .. \\
&\qquad\qquad \vdots \\
&|0\rangle \; \boxed{V(\theta_{k,opt})} \; \boxed{B_k^\dagger} \quad \hat{y}_k\,|0\rangle + .. \\
&\qquad\qquad \vdots \\
&|0\rangle \; \boxed{V(\theta_{K,opt})} \; \boxed{B_K^\dagger} \quad \hat{y}_K\,|0\rangle + ..
\end{aligned} \tag{4.23}$$

The final quantum state $\left|\hat{y}_{k,i}\right\rangle$, as already mentioned, suffers from two kind of normalizations: one introduced to solve the linear system (4.4) and one used to apply the inner product (4.19). The former causes an error for the coefficients β_k and the latter over the final product. Thus, the real final value is related to the approximation through the following

Method	P.D.	Linear Problem Solver	Quantum Product	Post Processing
QSplines [18]	✓	HHL [10]	Swap Test 2.44	✓
VQSplines	✓	VQLS [3]	Inner Product [21]	×

Table 4.1: Comparison between the QSplines and Variational QSplines (VQS) approaches. (P.D. stands for Problem Decomposition)

expression:

$$y_{k,i} \approx \|y_{k,i}\|_2 \|[1\ x_{k,i}]\|_2 \hat{y}_{k,i} \tag{4.24}$$

Before going into the second part of this thesis work, it is important to notice one important thing concerning the encoding of the vector $x'_{k,i}$. If the value of $x_{k,i}$ is negative than $\left|x'_{k,i}\right\rangle$ will be encoded through complex amplitudes and final result of the inner product a (4.21) will be imaginary. For this reason, the inner product circuit is modified when dealing with points belonging to negative part of the axis $\hat{x}$: by adding a $R_z(\pi)$ gate in the 4.20 circuit the imaginary amplitude a becomes real.

Due to the construction of the problem, the final estimation result will suffer, in addition to the normalizations penalties, from the initial decomposition of the matrix S: for each k_{th} problem the method will fit a line between two points without a general comprehension of the $\hat{Y}$ estimation problem. The effect of this penalty will be shown and discussed in the Experiments Chapter (5). From the table 4.1 it is possible to compare the QSplines approach with the proposed Variational QSplines. In the next section we provide a more general method to estimate a non-linear function, without the Linear Problem decomposition and executing the VQLS only once, instead of K times.

4.2 Generalized Variational QSplines

While the first section of this chapter was focused on the variational implementation of the QSplines[18] work, this section explains a generalized way to tackle the same problem facing directly the linear system for the spline coefficients estimation, without decomposing it in K subproblems and suitable for every number of qubits. As already done for the first part of the contributions, the proposed approach is shown through methodology and implementation.

4.2.1 Methodology

Let's recall the De Boor [4] recursive B-Spline definition and the related basis expansion, with knots list $T = [\xi_1, .., \xi_i, \xi_{i+1}, .., \xi_T]$:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } \xi_i \leq x < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.25}$$

$$B_{i,d}(x) = \omega_{i,d} B_{i,d-1}(x) + (1 - \omega_{i+1,d}(x)) B_{i+1,d-1}(x) \tag{4.26}$$

where:

$$\omega_{i,d}(x) = \begin{cases} \frac{x-\xi_i}{\xi_{i+d}-\xi_i} & \text{if } \xi_{i+d} \neq \xi_i \\ 0 & \text{otherwise.} \end{cases} \tag{4.27}$$

Given this definition, as already explained in the Regression Spline section from the Chapter 3, a non linear function f can be computed through its output estimates $Y = \{y_1, .., y_K\}$ given its inputs $X = \{x_1, .., x_K\}$:

$$Y = f(X, \beta) = S(X)\beta \tag{4.28}$$

where S is the matrix containing the B-Spline basis expansion of the inputs X and $\beta = \{\beta_1, .., \beta_K\}$ are the spline coefficients.

Now the Linear System describing the relation between the estimates

of the activation function Y, the matrix S and the spline coefficients β will be the following one:

$$Y_{1\times K} = S_{K\times K}\beta_{1\times K} \implies \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix} = \begin{bmatrix} B_{0,d}(x_1) & \dots & B_{l,d}(x_1) \\ B_{0,d}(x_2) & \dots & B_{l,d}(x_2) \\ \vdots & \ddots & \vdots \\ B_{0,d}(x_K) & \dots & B_{l,d}(x_K) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix} \tag{4.29}$$

where $l = dim(T) - d - 1$, d is the degree of the Bspline and T the set of knots. With $d = 1$, l will be equal to $dim(T) - 2$.

In order to apply this formalization to our QSpline problem and thus, to apply the VQLS algorithm, the basis expansion matrix S has to be square and non singular.

Therefore the length of T has to be suited to the number of coefficients and to the inputs X: $K = l = dim(T) - 2$. By computing all the $B_{i,d=1}(x)$ functions for all inputs x belonging to the interval $[0, 1]$, w.r.t. a proper T set, the linear system will be:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \dots \\ y_{k-1} \\ y_K \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1-x_2 & x_2 & \dots & \dots & 0 \\ \dots & 0 & 1-x_3 & x_3 & \dots & \dots \\ \dots & \dots & 0 & 1-x_4 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & 1-x_{K-1} & x_{K-1} \\ 0 & 0 & \dots & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \dots \\ \beta_{K-1} \\ \beta_K \end{bmatrix} \tag{4.30}$$

The S matrix is computed considering that the inputs X belong to an interval from 0 to 1, thus $x_1 = 0$. Following the De Boor B-Spline definition, the last element in the S matrix diagonal should be zero and the matrix non-square and thus, non-invertible. For this reason, the last

element is set to one necessarily. The pseudocode procedure 2 to build the S matrix is called "BasisExpansionMatrix". This 4.30 system can be therefore solved thorugh the VQLS algorithm, obtaining a quantum state whose amplitudes encode the B-spline coefficients β_i 4.31.

$$|\beta\rangle = VQLS(S, Y) = \sum_{i=1}^{K} \beta_i |i\rangle . \tag{4.31}$$

Then, given this new linear system representation, the $\hat{Y} = \{\hat{y}_1, .., \hat{y}_K\}$ estimates can be mathematically derived as follows:

$$\hat{y}_i = \beta_i(1 - x_i) + \beta_{i+1}x_i. \tag{4.32}$$

and computed as usual with the Inner product proposed by [21] between the quantum states encoding the rows of the S matrix and the quantum state $|\beta\rangle$.

The final $\hat{y}_i$ estimates could be also encoded, given a unitary operator P acting over $|\beta\rangle$, by the amplitudes of the following quantum state:

$$\left|\hat{Y}\right\rangle = P|\beta\rangle = \sum_{i=1}^{K-1} (\beta_i(1 - x_i) + \beta_{i+1}x_i) |i\rangle + \beta_K |K\rangle , \tag{4.33}$$

so a possible future work could consist in the development of such operator P. From 2 it is possible to sum up the Generalized Variartional QSplines approach.

Until now, the goal of this thesis contribution was to produce a quantum non-linear function by computing its estimates $\hat{Y} = \{\hat{y}_1, .., \hat{y}_K\}$ from a fixed set of inputs $X = \{x_1, .., x_K\}$. Now, exploiting what has been done so far, the objective is to produce a generic single output y_j of a non-linear function f s.t. $y_j = f(x_j)$. The construction of the S matrix allows to do that by considering which knots interval the input belongs to: each S row is built considering the input x_i B-spline basis expansion w.r.t. the knots set for each x_i from X. Therefore, given another point x_j,

Algorithm 2 GVQSplines pseudocode. I: maximum number of iterations. R rows of the S matrix. X fixed inputs. Y classic outputs. T knots set. ϵ is a lower bound for the variation of the cost function: if for a certain number of steps $\Delta C_L < \epsilon$ the cost function is in its minima and thus, following the "or" condition, the optimization procedure is stopped.

```
S ← BasisExpansionMatrix(X, Y, T)
θ ← Initialize()
|Y⟩ ← Mottonen(Y)
while i < I ∨ ΔC_L < ε do
    C_L(θ) ← HadamardTest(θ,S,|Y⟩)
    θ ← Update(θ)
end while
|β⟩ ← Ansatz(θ)
for r in R do
    |r⟩ ← Mottonen(r)
    ŷ_k ← InnerProduct(r,β)
    Ŷ.append(ŷ_k)
end for
return Ŷ
```

its correspondent row will have the same structure of the row related to the point x_i from X lying in the same knots interval. Assuming a generic input x_j encoded in the amplitude of one qubit, it is possible to extend it, through a mapping quantum circuit M, to its Basis expansions "row" $[0, .., 1 - x_j, x_j, .., 0]$:

$$\left|x_j\right\rangle = (1 - x_j)\left|0\right\rangle + x_j\left|1\right\rangle \xRightarrow{M} \left|x'_j\right\rangle_n = (1 - x_j)\left|s\right\rangle + x_j\left|s + 1\right\rangle \tag{4.34}$$

where s and $s+1$ are the basis state corresponding to the two subsequent non-zero elements of the S rows. Let's make an example with 3 qubits. Given a point x_j belonging to the same knots interval of x_2, and the latter row representation:

$$[0, 1 - x_2, x_2, 0, 0, 0, 0, 0], \tag{4.35}$$

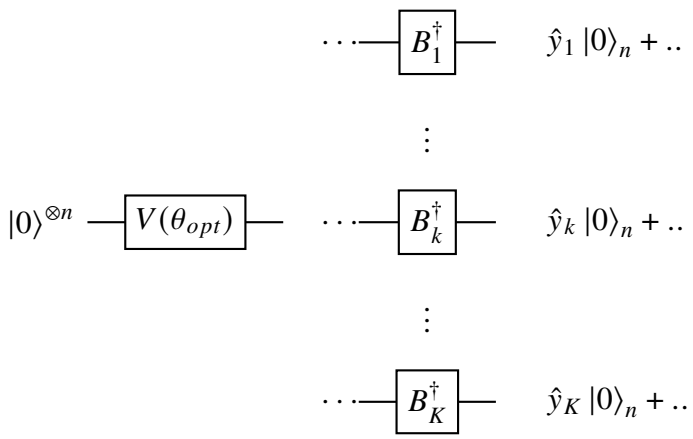
the circuit M will map x_j to:

$$[0, 1 - x_j, x_j, 0, 0, 0, 0, 0] \tag{4.36}$$

where $s = 1$ and thus,$|s\rangle = |001\rangle$ and $|s+1\rangle = |010\rangle$. Then, in order to obtain $\hat{y}_j$ we have to apply the dot-product between the row and the β coefficients computed through the S matrix and the VQLS. Summing up, the proposed Generalized Variational QSplines method is able from one side to produce a non-linear function f by means of its fixed output estimates $\hat{Y}$ and, on the other side, to return $f(x)$ given a generic input x.

4.2.2 Implementation

Differently from the Variational QSplines approach 4.23 here we don't need to apply K times the VQLS and use consequently K Ansatzs: now we optimize and use just one Ansatz able to prepare alone a quantum state whose amplitudes encode all the spline coefficients. Subsequently this state is used to compute the inner product with the rows of the matrix S (encoded through the routines B_i) and return the $\hat{y}_i$ estimates describing $\hat{Y}$.



(4.37)

Now let's discuss the actual implementation of the VQLS algorithm, this time more incisive than previously. Indeed, as already told, it is able to solve a linear system larger than 2×2 on NISQ devices, differently from the HHL. Exploiting this new methodology, given a number of knots such that the matrix for the basis expansion is a $2^n \times 2^n$ matrix, the VQLS algorithm can be applied with a number of qubits equal to n. Therefore the Hadamard test 4.10 has $n+1$ qubits: n to exploit the operators ($V(\theta)$,A and U) and one for the ancilla qubit. The variational circuits choice follows the proposals from [3] and [27], while the A_l gates coefficients are computed similarly. The S matrix can be seen as a Diagonal Block matrix, where each S_k 2×2 matrix can be decomposed as:

$$S_k = \begin{bmatrix} 1-a & a \\ 0 & 1-b \end{bmatrix} = \sum_{l=0}^{3} A_l c_{k,l} = I c_{k,0} + X c_{k,1} + Z c_{k,2} + R_y(3\pi) c_{k,3} \quad (4.38)$$

where the coefficients of the linear combination are computed as follows:

$$c_0 = 1 - a/2 - b/2 \quad (4.39)$$

$$c_1 = a/2 \quad (4.40)$$

$$c_2 = (b-a)/2 \quad (4.41)$$

$$c_3 = a/2 \quad (4.42)$$

In this method the S matrix is decomposed only to compute regularly the unitary gates coefficients, to represent the full matrix by means of quantum circuits and apply the VQLS, not to apply the algorithm K times to sub-matrices as done before. Moreover, the operator U is still realized exploiting the Mottonen State Preparation [25]. The only difference w.r.t. the first approach is that now U encodes the full vector

Y, rather than its portions. Nevertheless the normalization of Y is still required and the resulting penalty will be discussed in the next chapter 5. With this new formulation, we're able to solve only one linear system and encode all the spline coefficients in a unique quantum state by using one Ansatz $V(\theta)$ 4.31.

Concerning the estimation of the y values 4.32, we don't use one quantum circuit to encode them in the amplitudes of a unique quantum state (as described in 4.33). However the same Inner Product circuit used before is exploited: the product to obtain $\hat{y}_i$ is done row-by-column acting over the full matrix, preparing, as done before, the β vector with the current Ansatz and the row of the matrix with the Mottonen State Preparation (4.37). From the Table 4.2 we have a summary concerning the QSplines [18] and the two proposed methods. With the VQSplines we have overcome the post-processing step required by the swap test and tackled the problem with a Variational Algorithm, namely the VQLS. Moving then to the GVQSplines, the VQLS properties are exploited, through a new matrix espansion matrix formulation, to avoid the Linear System decomposition. It is important to underline that, before discussing the results of this approach, the new methodology is consistent to every linear $K \times K$ system choice satisfying the following relation:

$$K = 2^n = dim(T) - 2 \tag{4.43}$$

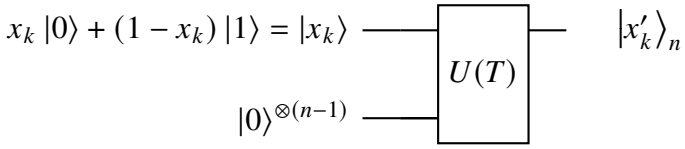
where n is the number of qubits and T a proper knots set.

Method	P.D.	Linear Problem Solver	Quantum Product	Post Processing
QSplines [18]	✓	HHL [10]	Swap Test 2.44	✓
VQSplines4.1	✓	VQLS [3]	Inner Product [21]	×
GVQSplines	×	VQLS [3]	Inner Product [21]	×

Table 4.2: Comparison between the QSplines, the Variational QSPlines (VQSplines) and the Generalized Variational QSplines (GVQSplines) approaches. (P.D. stands for Problem Decomposition)

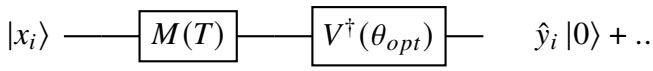
Quantum B-Spline Expansion

The implementation of the mapping circuit, used to map an input x_i to its basis expansion row form and multiply it by the β vector, consists basically in a simple sequence of quantum gates U (X,CNOT, CSWAP..) s.t.:



(4.44)

where n is the number of qubits used for the application of the method and $|x'_k\rangle$ is a n dimensional quantum state which amplitudes (K) encode the elements of the relative row of the S matrix. The sequence U depends on T and, in particular, on which interval $[\xi_i, \xi_{i+1}]$ contains the value of the input x_k, since, due to the construction of the model, the knots intervals determine the basis expansion of the inputs. As usual, according to [21], the output $\hat{y}_k$ will be encoded by the quantum state at the end of the following circuit:



(4.45)

where M is the mapping circuit 4.44 depending on the set T. The only difference w.r.t. the other quantum product circuits used in this methodology 4.37 is that here is the Ansatz to be transposed rather than the gate encoding the row.

Chapter 5

Experiments

While in the previous chapter were shown the two contributions of this thesis, namely the Variational QSplines 4.1 and the Generalized Variational QSplines 4.2, this chapter describes the experiments made and the practical results obtained with the proposed methods.

5.1 Experimental Settings

Before getting into the experiments, the methodologies are implemented throguh Pennylane 1(0.20.0 version), a cross-platform Python library for quantum machine learning. Secondly, more known python libraries such as scipy, numpy and pandas are also exploited.

1https://pennylane.ai/

5.2 Variational QSplines Part

In this section the Variational QSplines methodology explained previously will be applied to some classic activation functions used in Neural Networks. Many experimental details will be shown, different performances will be compared to vary intervals' size and thus, to vary number of steps used to compute the activation functions. Subsequently the consequences of the normalizations required (4.4,4.19) are investigated.

5.2.1 Baseline

The VQSplines method was tested with four activation functions (sigmoid, tanh, elu and relu), 1 qubit and 20 knots.

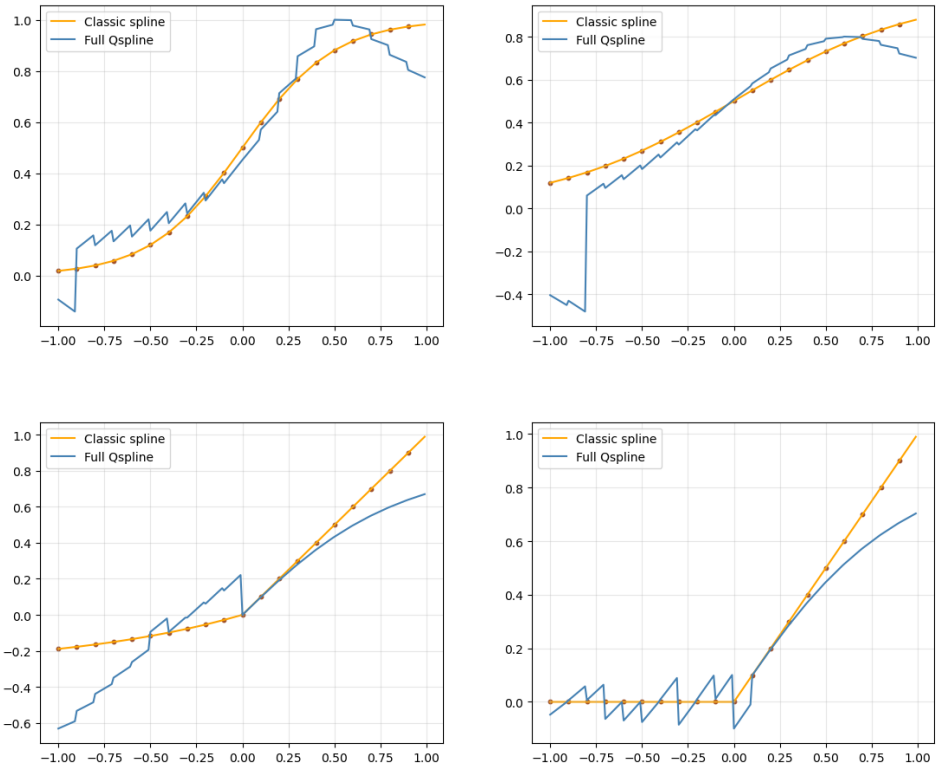


Figure 5.1: Number of knots: 20. Optimizer: COBYLA. Top-left: Sigmoid, Top-right: Tanh, Bottom-left: Elu, Bottom-right: Relu

From 5.1 it is evident how the four activation functions are likely and,

concerning the sigmoid and the tanh in particular, estimated better than the fault-tolerant QSplines [18] full quantum 3.2. In the following table is shown a comparison by means of Residual Sum Of Squares (RSS) between the fault-tolerant QSplines and the proposed Variational QSplines.

Activation function	QSplines (full quantum)	VQSplines
Sigmoid	.75	.32
Tanh	1.12	1.53
ReLU	8.16	.38
Elu	7.06	1.35

Table 5.1: RSS scores. Fault-Tolerant QSplines and VQSplines. For the VQSplines there's no hybrid version.

5.2.2 Number of Knots

Secondly, the numer of knots was doubled achieving a smoother estimation 5.2.

However, some errors at the boundaries are still present as well as poor predictions for $y_k < 0$ in the elu case. In the latter case, the norm of $y_k = [y_{k,0}, y_{k,1}]$ is always very low and thus, following 4.4, increases the error of the estimation. The error on the boundaries will be explained in the following sub-sections.

5.2.3 Condition Number

Looking at the boundaries the estimation performances are worse and unsatisfactory, but this problem can be easily explained. One important thing to notice when dealing with linear problems of equations, is the condition number of the matrix to invert. According to [36], given a linear system $b = Ax$, the "condition number" $\kappa(A)$ provides a measure of how much the matrix A is ill-conditioned:

$$\kappa(A) = \|A\|\|A^{-1}\| = \frac{\mu_{max}}{\mu_{min}} \geq 1 \tag{5.1}$$

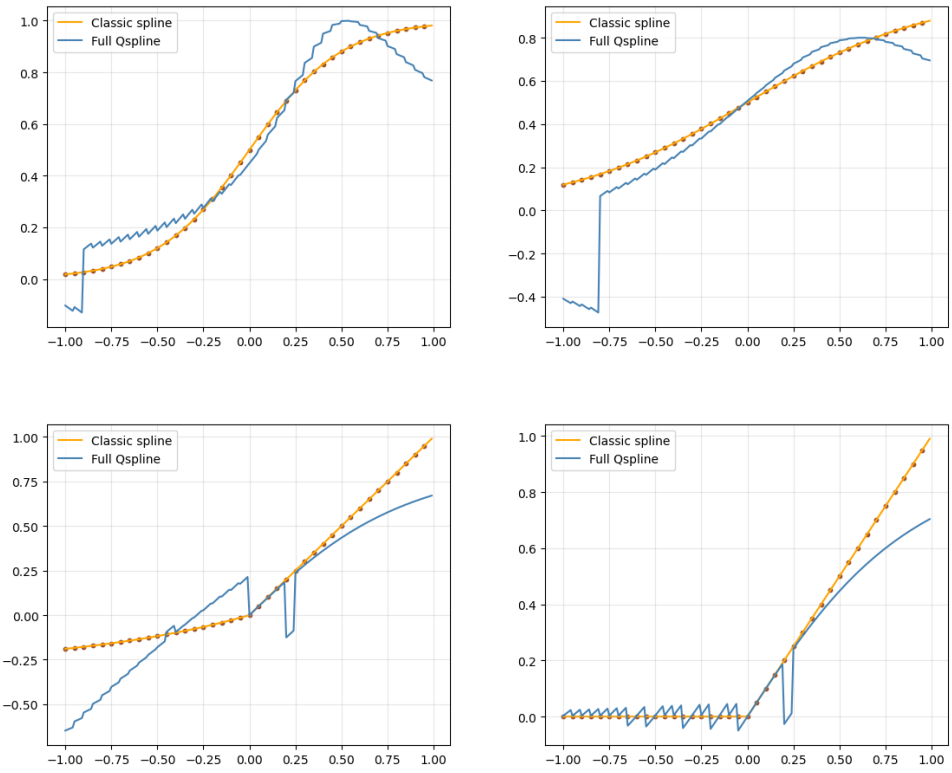


Figure 5.2: FULL VQSplines. Number of knots: 40. Optimizer: COBYLA. Top-left: Sigmoid, Top-right: Tanh, Bottom-left: Elu, Bottom-right: Relu

where the second equality from 5.1 is valid if the norm used is the L^2 norm and μ_{max} and μ_{min} are the maximal and minimal singular values of A respectively [36]. Larger the condition number of the matrix A, higher will be the probability to obtain numerical errors solving the linear system, since ill-conditioned systems converge slowly. This can be easily explained considering the following example from [36]:

$$A = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}, \tag{5.2}$$

if $a \longrightarrow 1$ then the condition number $\kappa(A) \longrightarrow +\infty$ and A is ill-conditioned. Indeed in the case $a = 1$, the system cannot be solved since A has $rank(A) = 1$, hence it is not invertible. For those reasons, closer the

matrix A to the non-invertibility condition, slower will be the procedure to invert it and thus, to solve the linear system $Ax = b$. By looking at 5.3 the subsystems where the condition number is higher are the same where the estimated function is worse: at the boundaries of the functions (5.1,5.2).

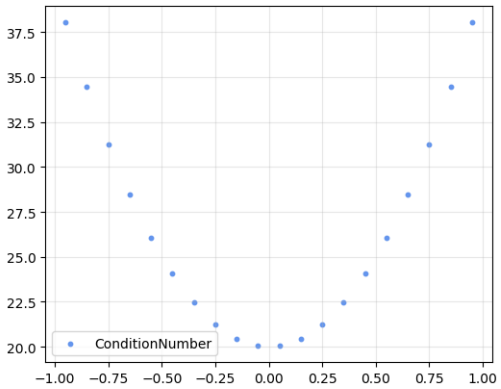


Figure 5.3: Condition number. VQSplines $S_{K\times K}$ matrix . The x-axis corresponds to the x_k inputs and the related S_k matrix, while the y-axis to the condition number.

By changing the optimizer of the VQLS algorithm from "COBYLA"to "BFGS", the error on the boundaries seems to be a little bit mitigated 5.4.

5.2.4 The ReLU case

Originally, the ReLU function was featured by unstable estimations for $x < 0$ 5.5. Indeed in those cases the associated sub system 4.3 becomes the following homogeneous system:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & x_{k,0} \\ 1 & x_{k,1} \end{bmatrix} \begin{bmatrix} \beta_{k,0} \\ \beta_{k,1} \end{bmatrix}. \tag{5.3}$$

Due to the VQLS, $y_k = [y_{k,0}, y_{k,1}] = [0, 0]$ is normalized and therefore becomes $[1/\sqrt{2}, 1/\sqrt{2}]$. Consequently, the quantum circuit returns as

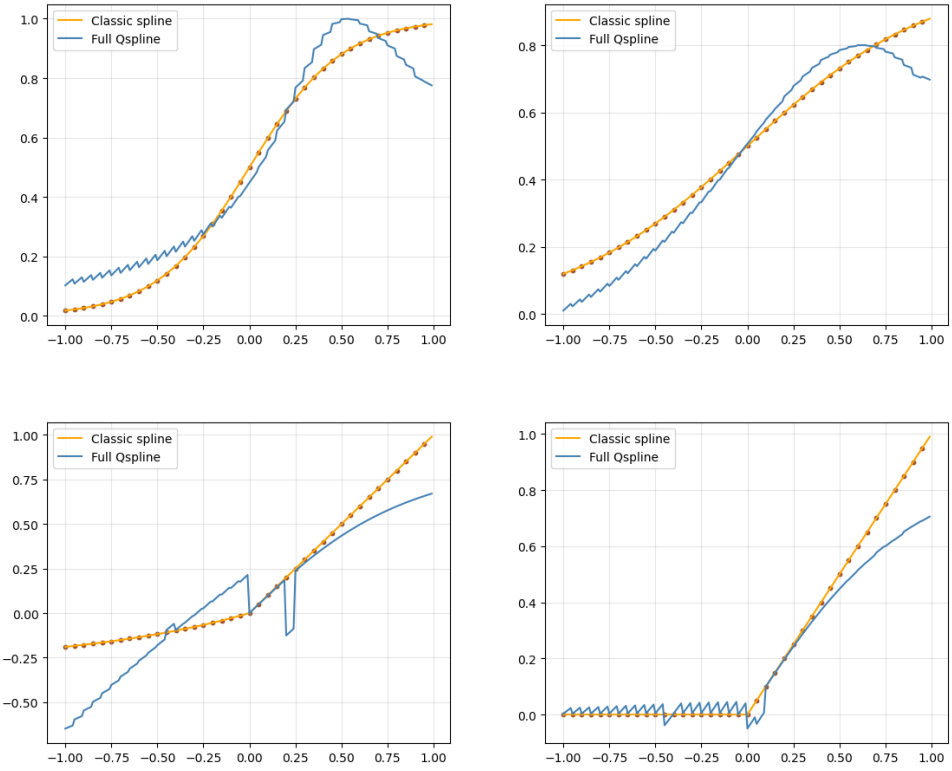


Figure 5.4: VQSplines. Number of knots: 40. Optimizer: BFGS. Top-left: Sigmoid, Top-right: Tanh, Bottom-left: Elu, Bottom-right: Relu

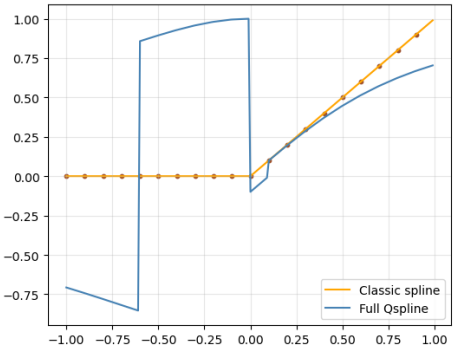


Figure 5.5: ReLu with $y_k = [0, 0]$ for $x < 0$

solution $\hat{\beta}_k \approx [-1, 0]$ instead of $[0, 0]$. For this reason, in all previous experiments, all the $y_k = [0, 0]$ were set to $y_k = [10^{-5}, 10^{-4}]$ achieving the already shown quantum ReLU function 5.1. With the Generalized Variational QSplines this problem disappears.

5.2.5 VQLS and Product Normalization cases

As we have discussed in the previous chapter about the k_{th} subsystem, the effects of the two normalizations applied in the method are analyzed. By considering the case when both $y_{k,0}$ and $y_{k,1}$ are close to zero, the norm term $\|y_k\|_2$ becomes really significant in the final prediction estimation (4.24). Indeed the more the norm is far from 1, heavier will be the impact of the normalization on the estimation. This can be verified by looking at the activation functions plots. This error is reduced where the norm is closer to one and where the curve is "more linear". The same reasoning can be done for the second source of normalization 4.19 given by the inner product. When $\|[1, x_i]\|_2$ becomes larger than 1 this penalty decreases the value of the estimation $\hat{y}_k$ w.r.t. the true y_k: from $x_i = .5$ onwards, the activation functions manifest a decreasing (or dumping) pattern w.r.t. the desired output. Following 4.4 if we multiply the outcome of the inner product 4.9 by the norm of the original y_k, the error from the VQLS normalization decreases significantly. In 5.6 we can see the ReLU example, just to visualize the VQLS normalization penaly effect.

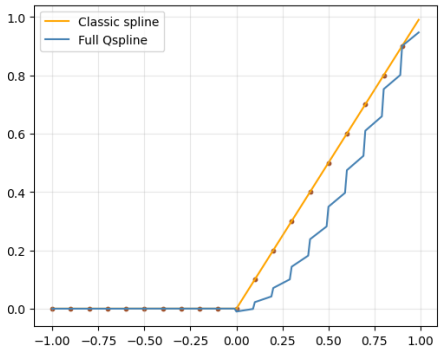


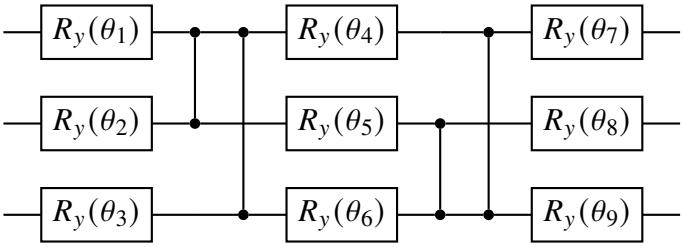
Figure 5.6: ReLU estimation "denormalized".

As aforementioned in the Variational QSplines section 4.1, the decomposition of the S matrix in 2×2 sub-matrices, combined with the normalizations required, lead the method to produce final estimates with

a “zig-zag” pattern. As we will see in the next section, this behaviour disappears with the Genralized Variational QSplines method.

5.3 Generalized Variational QSplines

In this section, the experiments done and the results obtained with the Generalized Variational QSplines method are analyzed. As done for the Variational QSplines case, four activation functions are tested. In the first part the VQLS β coefficients computation are evaluated. It is called “hybrid approach” since in this preliminary part the inner product to produce the $\hat{Y}$ is done classically. Subsequently, in the second and third parts the full quantum approach (where both the spline coefficients and the inner product are computed in a quantum fashion) is investigated, without and with normalizing the estimates Y respectively. In all the following experiments we tackle 8×8 linear systems with 3 qubit quantum circuits, the COBYLA optimizer and the following Ansatz already shown in Chapter 2 which we report again here for clarity:



(5.4)

where $\theta_1, .., \theta_9$ are the variational parameters to optimize through the VQLS. The S matrix 4.30 is the same for all the activation functions tested and it is computed taking as T and X knots and inputs sets belonging to the interval $[0, 1]$. Each input x_i belongs to a knots interval $[\xi_i, \xi_{i+1}]$.

5.3.1 VQLS Evaluation - Hybrid Approach

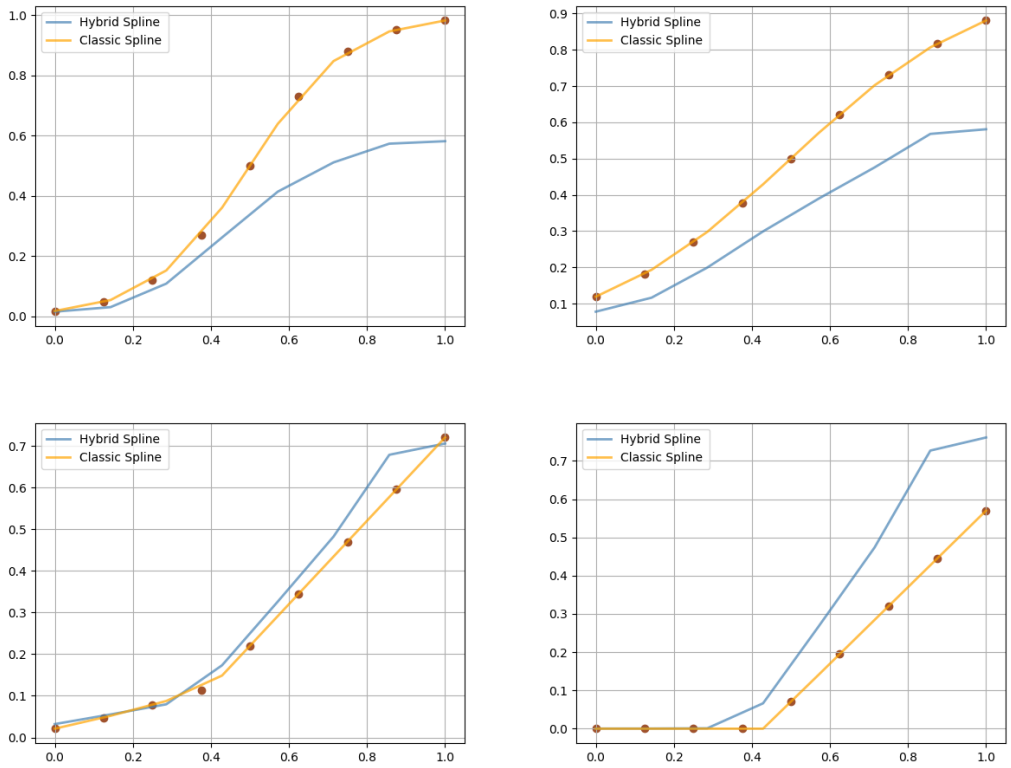


Figure 5.7: Hybrid Spline. The spline coefficients are computed in a quantum fashion thorugh VQLS, while the inner product is computed classically. Top-left: Sigmoid, Top-right: Tanh, Bottom-left: Elu, Bottom-right: Relu

In 5.7 we report a "hybrid" spline non linear function estimation, where the β spline coefficients are computed with the proposed basis expansion matrix S and the VQLS, while the "row-by-column" product is done classically. This intermediate result is shown to verify the goodness of the new methodology in calculating spline coefficients related to a nonlinear function. Indeed, the curves obtained with the quantum coefficients, even if scaled, are very promising. Nevertheless, the penalty effect of the normalization due to the VQLS usage, already explained in the previous sections, is preponderant. This time the normalization penalty is more effective because we are considering the full linear system and

the full vector of the estimation Y rather than K subsystems. Farther from 1 is $\|Y\|_2$, larger will be the error on the final estimation.

5.3.2 Full Quantum Approach

Following instead the complete methodology explained in the previous chapter 4.37, we apply the quantum inner product [21] to produce the final estimates $\hat{Y}$ achieving full quantum spline non linear functions 5.8.

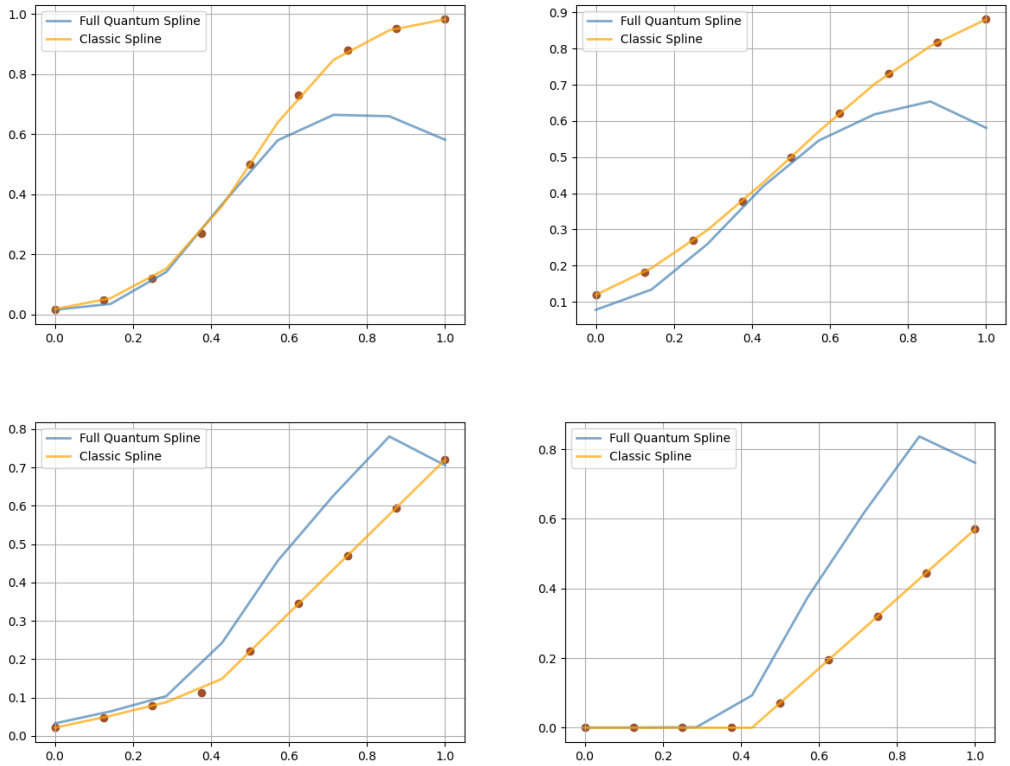


Figure 5.8: Full Quantum Spline. Both spline coefficients and the inner product are computed in a quantum fashion. Top-left: Sigmoid, Top-right: Tanh, Bottom-left: Elu, Bottom-right: Relu

Just like in the Variational QSpline method, the normalization required by the Inner Product circuit, this time used to encode the rows of S, affects again the final estimation. Obviously, it is less strong considering input values closer to 0, since the coefficients are very low. Instead, higher the inputs $x_i \epsilon X$ farther the final the estimation from the hybrid

curve, where the product is computed classically. The overestimation problem can be explained looking at the S matrix built with the current setup sets T and X: once we normalize the row to compute the quantum product, the values $1 - x_i, x_i$ on the new row become bigger than before. Instead, the first and last S rows norms are always 1 due to the construction of the matrix and thus, the first and the last estimations, $\hat{y}_0$ and $\hat{y}_K$ will be equal to the first and the last elements of the spline coefficients array β_0 and β_K. Summing up, except for the first and the last row, each "row-by-column" product will return an output estimate $\hat{y}_k$ bigger than the desired y_k. In the table below we can see the differences in the RSS scores w.r.t. the true activation functions of the hybrid and full quantum approaches:

Activation function	GVQSplines (hybrid)	GVQSplines (full quantum)
Sigmoid	.47	.28
Tanh	.26	.13
ReLU	.18	.38
Elu	.02	.12

Table 5.2: RSS scores. Hybrid GVQSplines and Full Quantum GVQSplines.

Instead in the following table a final comparison in terms of RSS w.r.t. the true activation functions between QSplines, Variational QSplines and Generalized Variational QSplines is shown.

Activation function	QSplines (full quantum)	VQSplines	GVQSplines (full quantum)
Sigmoid	.75	.32	.28
Tanh	1.12	1.53	.13
ReLU	8.16	.38	.38
Elu	7.06	1.35	.12

Table 5.3: RSS scores. Fault-Tolerant QSplines, VQSplines and GVQSplines. For the VQSplines there's no hybrid version.

5.3.3 Full Quantum and Hybrid Approaches - Normalized case

As already discussed previously, by considering normalized activation functions estimations Y s.t. $\|Y\|_2 = 1$, the final result should be more precise. Let's see what happens in this case where the normalization error due to the VQLS should disappear. For this case only sigmoid and tanh were reported, since elu and relu had already a Y set s.t. $\|Y\|_2 \approx 1$. From 5.9 it is possible to confirm and visualize the efficiency of the Generalized Variational QSpline methodology. The proposed S matrix fits the desired task, indeed the coefficients computed with the VQSL (solving the $Y = S\beta$ system) are related to $\hat{Y}$ estimations which very close to the classic ones Y.

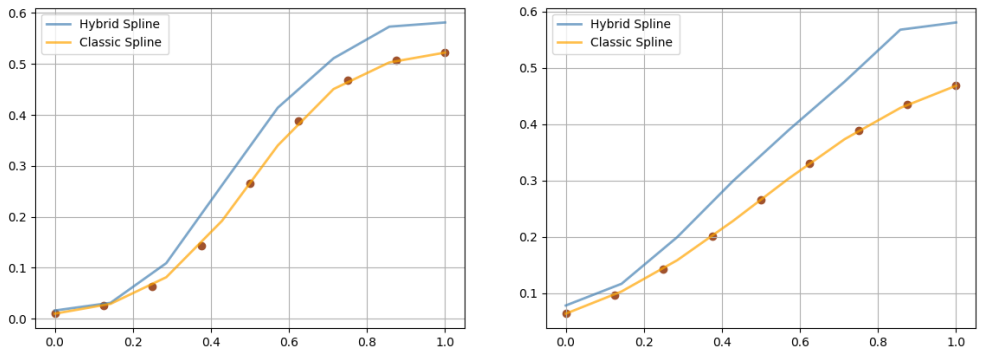


Figure 5.9: Scaled Hybrid Spline. Left: Sigmoid, Right: Tanh

Although the error over the β coefficients has been reduced by normalizing Y, the overestimation of the curves is still present for the hybrid approach 5.9 and even bigger for the full quantum one 5.10 due to the quantum product. Indeed, the latter leads always to an overestimation of the non-linear function for the full quantum approach w.r.t. the hybrid one. Indeed, the same reasoning done w.r.t. the S matrix's rows in the previous subsection is still valid here and explains the overestimation.

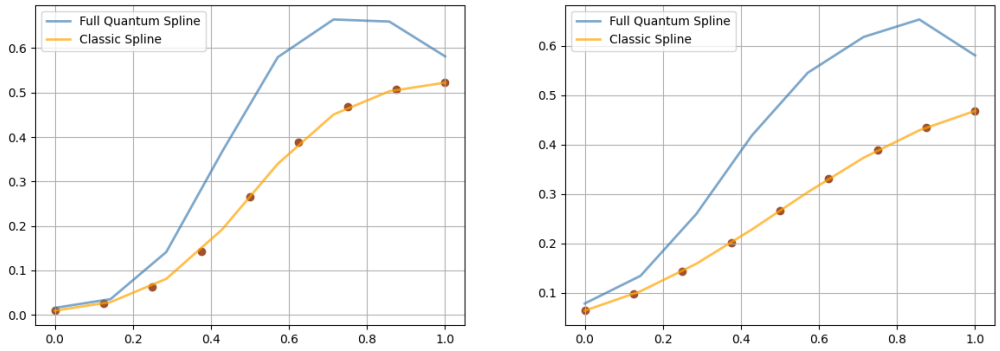


Figure 5.10: Scaled Full Quantum Spline. Left: Sigmoid, Right: Tanh

5.3.4 Quantum B-Spline Expansion

In the Generalized Variational QSplines section 4.2, a method to estimate the generic output of a non-linear activation function is explained. Basically, it follows the same procedure described to return the estimations $\hat{Y}$, indeed it exploits the same Ansatz optimized through the VQLS. The difference lies in the circuit encoding the inputs: for the estimation $\hat{Y}$ we used an amplitude encoding routine to encode the rows of the matrix S; here we need a mapping circuit that, given a generic input encoded in 1 qubit, returns its row representation as explained in the previous chapter. To test the mapping circuit realized to estimate the output of an activation function given a generic input encoded in one qubit, 10 equally distributed inputs (different from the inputs X and the knots T used to build S) from the interval $[0, 1]$ were provided to the circuit 4.45.

In 5.11 we have the sigmoid and elu cases: the outputs from generic inputs follow the behaviour of the curves obtained before with the fixed inputs X 5.10. One drawback of this method could be found by increasing the number of inputs 5.12. Indeed in 5.11 we had one generic input in each knots interval, but if we increase the number of knots in order to have more than one input for each interval the normalization leads to

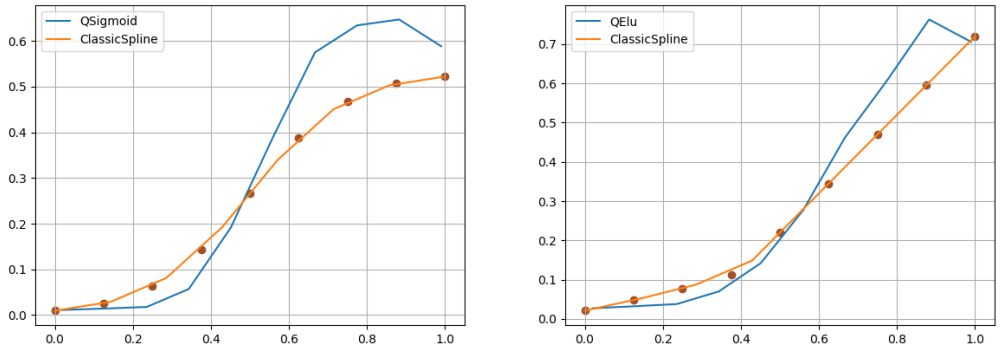


Figure 5.11: QSigmoid and QElu, given 10 generic inputs.

undesired decreasing patterns for the estimations $\hat{y}_k$ for $x_k > .6$.

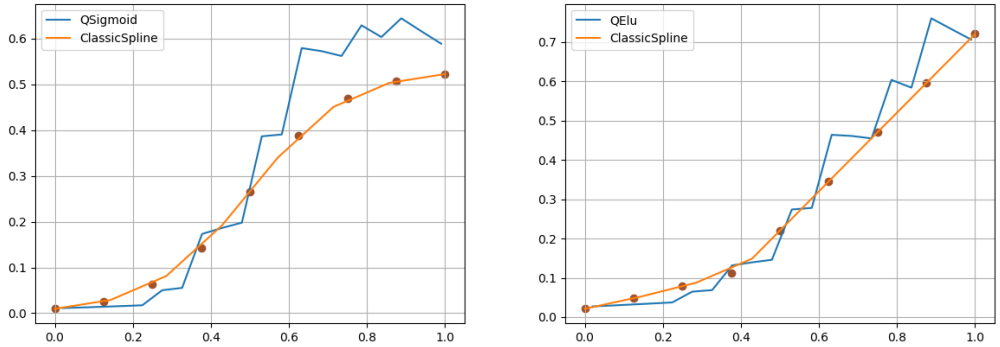


Figure 5.12: QSigmoid and QElu, given 20 generic inputs.

More practically, let's suppose to have two inputs x_k and x_j belonging to the same knots interval and such that $x_j > x_k > .6$. Due to the mapping circuit construction they will be related to the same row structure of the S matrix. However, since $\|1 - x_k, x_k\|_2 < \|1 - x_j, x_j\|_2$ the normalization penalty for the quantum product leads to achieve $\hat{y}_k > \hat{y}_j$ even if for the true values $y_k < y_j$. Obviously, concerning inputs lower than .5 we have the opposite pattern: $\|1 - x_k, x_k\|_2 > \|1 - x_j, x_j\|_2$ and thus, $\hat{y}_k < \hat{y}_j$. For our purposes this is tolerable (since we have monotonically increasing activation functions), but for an activation function with a decreasing pattern for $x_k < .6$ it is problematic as well as the case explained before. For these reasons, the mapping circuit can be changed to map all

the inputs belonging to the same knots interval directly to the same row representation, rather than keep the x_i row's structure and encode in the row representation the input values $1 - x_k, x_k$ as done actually. In this way, we could get the same estimation $\hat{y}_k$ for all the inputs belonging to same knots interval.

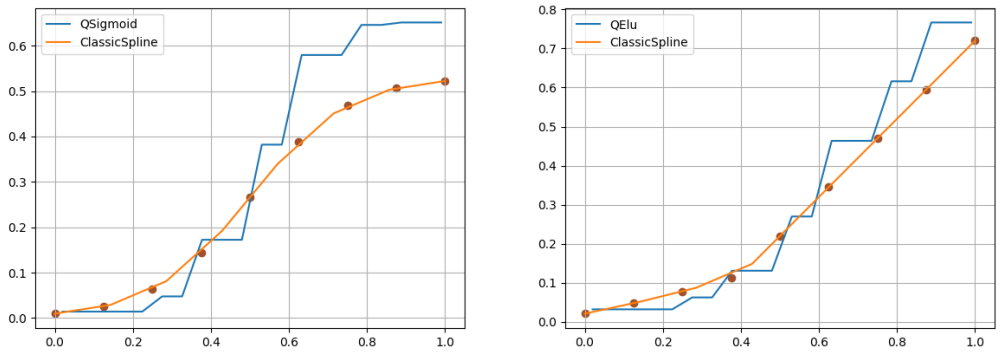


Figure 5.13: QSigmoid and QElu, where for each generic input, we encode its related knots interval rather than the input itself.

From 5.13 we can see an example of the desired circuit. A new formulation M has not yet been implemented, indeed the plots are computed by encoding the knots as input of the product instead of the input itself x_k. By developing a new mapping circuit $\boldsymbol{M}$ able to encode directly a generic input x_k to a fixed quantum state depending on the related knots interval together with an increased number of knots, we could get a very promising improvement.

5.3.5 Considerations

The optimization part of the VQLS, since now we have 3 qubits instead of 1 and the Ansatz is featured by 9 weights, is clearly slower. In order to reduce the number of necessary steps to minimize the cost function, all the weights are initialized to $-\pi$ rather than randomly, as done instead in the Variational QSPlines approach. Looking forward, by increasing the number of Knots, and according to 4.43 the number of qubits, the

final result should be more precise and fit better the desired activation function.

In this new approach we don't have anymore the "zig-zag" pattern due to the decomposition of the problem; moreover here we optimize only one Ansatz $V(\theta)$ able alone to produce all the spline coefficients, rather than many. Summing up, even if the normalization penalties are really significant, the shape of the final curve is very likely thanks to the proposed methodology. Therefore, the purpose of this thesis project, namely to achieve non linear functions through quantum circuits, has been quite well fulfilled.

Chapter 6

Conclusions

Quantum Machine Learning (QML) has recently attracted ever-increasing attention and promises to impact various applications by leveraging quantum computational power and novel algorithmic models, such as Variational Algorithms. However, although QML models offer several theoretical advantages with respect to their classical counterparts, the field is still in its infancy, and its practical benefits need further investigation.

This dissertation moves toward the adoption of QML algorithms to solve complex pattern recognition tasks. In particular, we showed that it is possible to circumvent the constraint of unitarity in quantum computation by presenting a more efficient version of the QSplines, whose implementation falls within the context of hybrid quantum-classical computation. In other words, the proposed method doesn't require error-corrected qubits and allows the approximation of non-linear functions using actual quantum technology. Furthermore, the same method can be employed as a subroutine to approximate non-linear activation functions in quantum neural networks, which is a compulsory stage for achieving universal approximation by means of the Quantum Single Layer Perceptron, as discussed in Macaluso et al.[16] in the context of the MAQA framework [15].

In practice, the contribution of the thesis is two folds. The first part

proposes the Variation Quantum Splines (VQSplines), a reformulation of the fault-tolerant QSplines[18] in a variational quantum setting that avoids the use of the HHL as a subroutine. The VQSplines have been successfully implemented using Pennylane to approximate non-linear activation functions typically adopted in classical Neural Networks. Additionally, in the second part of the thesis, the generalized method for VQSplines is discussed (GVQSplines 4.2). The benefit of this new formulation lies in the ability of GVQSplines to be completely independent of the specific structure of the spline matrix since it leverages de Boor [4] B-Spline as a classical baseline formulation. In fact, while the VQSplines method (as the fault-tolerant QSplines) requires decomposing the problem in K sub-problems and applying the algorithm K times, the GVQSplines instead, with a new basis expansion matrix formulation inspired by the B-Spline definition, avoids the linear system decomposition and allows to tackle the problem of the matrix inversion in an end-to-end manner, with one single linear system and a number of qubits which is related to the number of knots. For instance, with three qubits and only one execution of the VQLS algorithm, the GVQSplines can estimate non-linear functions better than the previous VQSplines, where only one qubit was used at the expense of K execution of the VQLS. The main drawback of the GVQSplines consists in the normalization conditions of quantum states and quantum algorithms, discussed in the chapter 4 and verified in the chapter 5. Despite that, both contributions consistently improved the previous fault-tolerant attempt to develop quantum activation functions. Moreover, this thesis work lays the foundation for a concrete application of a new model combining the already mentioned qSLP with quantum non-linear activation functions.

Bibliography

[1] S. Aaronson. Quantum machine learning algorithms: read the fine print.

[2] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini. Parameterized quantum circuits as machine learning models. Quantum Science and Technology, 4(4):043001, November 2019. DOI: 10.1088/2058-9565/ab4eb5. URL: https://doi.org/10.1088/2058-9565/ab4eb5.

[3] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles. Variational quantum linear solver, 2019. DOI: 10.48550/ARXIV.1909.05820. URL: https://arxiv.org/abs/1909.05820.

[4] C. d. Boor. A Practical Guide to Splines. Springer Verlag, New York, 1978.

[5] G. De Luca. A survey of nisq era hybrid quantum-classical machine learning research. Journal of Artificial Intelligence and Technology, 2(1):9–15, December 2021. DOI: 10.37965/jait.2021.12002. URL: https://ojs.istp-press.com/jait/article/view/60.

[6] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan. Hybrid quantum-classical algorithms and quantum error mitigation. Journal of the Physical Society of Japan, 90(3):032001, March 2021. DOI: 10.7566/jpsj.90.032001. URL: https://doi.org/10.7566%2Fjpsj.90.032001.

[7] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm, 2014. arXiv: 1411.4028 [quant-ph].

[8] R. P. Feynman. Simulating physics with computers. International journal of theoretical physics, 21(6/7):467–488, 1982.

[9] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[10] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. Physical Review Letters, 103(15), October 2009. DOI: 10.1103/physrevlett.103.150502. URL: https://doi.org/10.1103%2Fphysrevlett.103.150502.

[11] K. Hornik. Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2):251–257, 1991. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T. URL: https://www.sciencedirect.com/science/article/pii/089360809190009T.

[12] G. James, D. Witten, T. Hastie, and R. Tibshirani. An Introduction to Statistical Learning: With Applications in R. Springer Publishing Company, Incorporated, 2014. ISBN: 1461471370.

[13] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White. Towards quantum chemistry on a quantum computer. Nature Chemistry, 2(2):106–111, January 2010. DOI: 10.1038/nchem.483. URL: https://doi.org/10.1038%2Fnchem.483.

[14] Y. Li and S. C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. Phys. Rev. X, 7:021050, 2, June 2017. DOI: 10.1103/PhysRevX.7.021050. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.021050.

[15] A. Macaluso. A Novel Framework for Quantum Machine Learning. PhD thesis, alma, May 2021. URL: http://amsdottorato.unibo.it/9791/.

[16] A. Macaluso, L. Clissa, S. Lodi, and C. Sartori. A variational algorithm for quantum neural networks. In V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, editors, Computational Science – ICCS 2020, pages 591–604, Cham. Springer International Publishing, 2020. ISBN: 978-3-030-50433-5.

[17] A. Macaluso, L. Clissa, S. Lodi, and C. Sartori. Quantum ensemble for classification. CoRR, abs/2007.01028, 2020. arXiv: 2007.01028. URL: https://arxiv.org/abs/2007.01028.

[18] A. Macaluso, L. Clissa, S. Lodi, and C. Sartori. Quantum splines for non-linear approximations. In Proceedings of the 17th ACM International Conference on Computing Frontiers, CF '20, pages 249–252, Catania, Sicily, Italy. Association for Computing Machinery, 2020. ISBN: 9781450379564. DOI: 10.1145/3387902.3394032. URL: https://doi.org/10.1145/3387902.3394032.

[19] S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, and C. Macchiavello. Quantum computing models for artificial neural networks. Europhysics Letters, 134(1):10002, April 2021. DOI: 10.1209/0295-5075/134/10002. URL: https://doi.org/10.1209%2F0295-5075%2F134%2F10002.

[20] I. I. Manin. Vychislimoe i nevychislimoe / IU.I. Manin. Russian. "Sov. radio," Moskva, 1980, 127 p. :

[21] V. Markov, C. Stefanski, A. Rao, and C. Gonciulea. A generalized quantum inner product and applications to financial engineering, 2022. DOI: 10.48550/ARXIV.2201.09845. URL: https://arxiv.org/abs/2201.09845.

[22] M. Maronese, C. Destri, and E. Prati. Quantum activation functions for quantum neural networks, 2022. DOI: 10.48550/ARXIV.2201.03700. URL: https://arxiv.org/abs/2201.03700.

[23] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. Nature Communications, 9(1), November 2018. DOI: 10.1038/s41467-018-07090-4. URL: https://doi.org/10.1038%2Fs41467-018-07090-4.

[24] W. Mcculloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5:127–147, 1943.

[25] M. Mottonen and J. J. Vartiainen. Decompositions of general quantum gates, 2005. DOI: 10.48550/ARXIV.QUANT-PH/0504100. URL: https://arxiv.org/abs/quant-ph/0504100.

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

[27] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione. A comparison of various classical optimizers for a variational quantum linear solver. Quantum Information Processing, 20(6), June 2021. DOI: 10.1007/s11128-021-03140-x. URL: https://doi.org/10.1007%2Fs11128-021-03140-x.

[28] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. Love, A. Aspuru-Guzik, and J. O'Brien. A variational eigenvalue solver on a photonic quantum processor. English. Nature Communications, 5, July 2014. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[29] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Advances in Optimization and Numerical Analysis. 1994, pages 51–

67. DOI: 10.1007/978-94-015-8330-5_4. URL: https://app.dimensions.ai/details/publication/pub.1046127469.

[30] J. Preskill. Quantum computing in the NISQ era and beyond. Quantum, 2:79, August 2018. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331%2Fq-2018-08-06-79.

[31] M. Schuld. Supervised quantum machine learning models are kernel methods, 2021. DOI: 10.48550/ARXIV.2101.11020. URL: https://arxiv.org/abs/2101.11020.

[32] M. Schuld and F. Petruccione. Fault-tolerant quantum machine learning. In Machine Learning with Quantum Computers. Springer International Publishing, Cham, 2021, pages 247–272. ISBN: 978-3-030-83098-4. DOI: 10.1007/978-3-030-83098-4_7. URL: https://doi.org/10.1007/978-3-030-83098-4_7.

[33] M. Schuld and F. Petruccione. Supervised Learning with Quantum Computers. Springer Publishing Company, Incorporated, 1st edition, 2018. ISBN: 3319964232.

[34] M. Schuld, I. Sinayskiy, and F. Petruccione. An introduction to quantum machine learning. Contemporary Physics, 56(2):172–185, October 2014. DOI: 10.1080/00107514.2014.964942. URL: https://doi.org/10.1080%2F00107514.2014.964942.

[35] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5):1484–1509, October 1997. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137%2Fs0097539795293172.

[36] H. Sonnberger. Regression diagnostics: identifying influential data and sources of collinearity, by d. a. belsley, k. kuh and r. e. welsch. (john wiley & sons, new york, 1980, pp. xv + 292, isbn 0-471-05856-4, cloth $39.95. Journal of Applied Econometrics, 4(1):97–99, 1989. DOI: https://doi.org/10.1002/jae.3950040108. eprint: https:

//onlinelibrary.wiley.com/doi/pdf/10.1002/jae.3950040108. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.3950040108.

[37] H. D. Zeh. On the interpretation of measurement in quantum theory. Foundations of Physics, 1(1):69–76, 1970. DOI: 10.1007/BF00708656.

[38] C. Zoufal, A. Lucchi, and S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. npj Quantum Information, 5:1–9, 2019.