eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Degree in Computer Engineering

Computer science

End of degree work

Quantum extreme learning machine for classification tasks

Author

Unai Sainz de la Maza Gamboa

2022

eman ta zabal zazu

Universidad del País Vasco | Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Degree in Computer Engineering

Computer science

End of degree work

Quantum extreme learning machine for classification tasks

Author

Unai Sainz de la Maza Gamboa

Director(s)

Jose A. Pascual Saiz

Summary

Quantum computing is one of the most researched areas in computer science and physics. However, current quantum computers are influenced by unwanted noise from environmental factors. Quantum Extreme Learning Machine (QELM) is a hybrid classical-quantum framework that is intended to take advantage of these complex and rich dynamics of noisy intermediate-scale quantum (NISQ) devices to improve learning capacity. This work explores the power of a contemporary gate-based QELM relative to current classical binary classification problem solvers. On the other hand, we have not limited ourselves to transferring the classical algorithm to its corresponding quantum algorithm but have investigated it more deeply, proposing several variations of it. We need to consider that the use of quantum simulators limits our work, which may lack rich enough dynamics to generate sufficiently complex output. Finally, other directions, such as exploring alternative quantum platforms as a quantum substrate and implementing them in real hardware, may be considered a potential focus for future research.

Contents

List of Figures

List of Tables

1. CHAPTER

Introduction

In recent years, an explosion of unconventional computing methods and systems has been occurring. These unconventional methods are mostly neuro-inspired, where the computational paradigm goes hand in hand with the design of a physical substrate trying to mimic the computational power of the human brain, [1].

One of those methods is the extreme learning machine (ELM), an algorithm proposed by [2] that is a particularisation of the reservoir computing (RC) framework applied to feed-forward neural networks (NN). This framework exploits the natural dynamics of input-driven randomly connected NN for information processing, not requiring gradient-based back-propagation to work. The only difference between RC and ELM is that the first one exploits the natural dynamics of the substrate as an internal memory of the past inputs, while ELM does not.

The main advantage of the ELM concept is its minimal requirements for learning (what is known as the training process), mainly because the learning process is made without an iterative tuning of the hidden nodes. Figure 1.1 shows the three layers of a typical ELM algorithm: an input layer, the hidden layer (or the substrate), and the output layer. This process can be summarised as follows: the information from the input is fed into the substrate, which works as a reservoir. The intermediate representation produced by the previous step is then used to create the desired output after optimisation of the output connections.

In ELM, one only needs to adjust the weights (connections) of the substrate with the output layer via, for example, linear regression. The rest of the weights are randomly

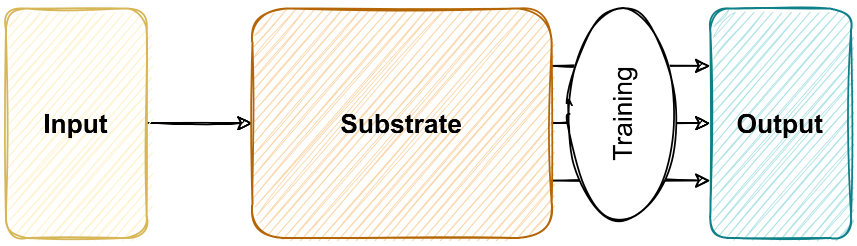


Figure 1.1: Basic ingredients of ELM

initialised and not optimised through the process. The bulk of the processing is offloaded to a fixed complex system, and desired input-output maps are achieved by adjusting how its state is post-processed, [3]. This allows the algorithm to avoid multiple iterations and local minimisation while maintaining good generalisation and fast training times.

The quantum counterpart of ELM is the quantum extreme learning machine, so instead of having a number of artificial hidden nodes (the substrate), we have a quantum system as a dynamical system with a number of qubits (the quantum substrate). This will be referred to in the following as QELM and is the main topic of this work. Notice that the same applies to RC, known as QRC, to its quantum counterpart.

Recent advances in quantum computing have accelerated efforts to develop new algorithms that can take advantage of this new information processing paradigm. In a similar line, QELM is an excellent candidate to demonstrate the potential of how rich quantum dynamics can enhance classical machine learning (ML) methods.

Quantum systems exhibit many degrees of freedom that can be exploited for QELM. Therefore, exploring extended quantum systems as substrates for ML in QELM represents a timely and potentially disruptive opportunity. The main idea is to take advantage of the quantum complexity as a substrate, more precisely, to use the complex dynamics and the noise of those NISQ devices. Also, the quantum substrate can offer a faster computation of the final state compared to the classical substrate.

Quantum noise, such as decoherence, is a decisive limiting factor in gate-based quantum computing, but in principle, QELM is well suited for current noisy-intermediate-scale-quantum (NISQ) systems. Because of that, some proposals have been made to exploit the natural dynamics of gate-based superconducting quantum processors as a reservoir [4].

The rest of the work is organised as follows. In Chapter 2, we explain the project's objectives. In Chapter 3, we briefly introduce quantum computing and fix our notation. In Chapter 5, we present ELM and its quantum counterpart, the QELM, dissecting the process of translating the classical algorithm to quantum computing. In Chapter 4, we study the state-of-the-art and some recent proposals made in the field. The performance of QELM in some standard datasets is underlined in Chapter 6. We end the work by discussing some conclusions and highlighting potential opportunities for this research line.

2. CHAPTER

The aims of the project

The project's primary goal is to explore the quantum counterpart of ELM (Extreme Learning Machine). To accomplish this task, we need to learn the fundamentals of quantum computing, understand them, and be able to use the quantum computer as a computational resource. Furthermore, we need to study the literature and familiarise ourselves with some quantum platforms to be used, in our case, the IBM-Q platform.

Secondly, to fully understand the ELM algorithm, an in-depth study of the method has to be carried out, exploring all the details related to the original algorithm, e.g., the ideas behind it, the methods, the possible improvements, and all those things that can be relevant in order to understand the algorithm altogether.

The third step is to learn about the RC (Reservoir Computing) framework and see how this has been adapted using a quantum-based reservoir. More specifically, see how this quantum-based reservoir has been used to adapt the ELM algorithm to its quantum counterpart. This has to be done studying the literature and the state-of-the-art.

The primary purpose of this project is to combine every knowledge acquired in a practical case study. More precisely, propose a new hybrid method that uses the ELM algorithm training strategy and quantum systems as a computational resource, i.e. as a reservoir, and then use it to solve classification problems.

Finally, to evaluate the value of the proposed model, we will need to choose a diverse and complete set of standard benchmarks, the correct performance metrics, and other classical models to compare against them.

The goals can be summarised as follows:

- Study the basic mathematical foundation required to understand the fundamentals of quantum computing.
- Complete comprehension of the ELM algorithm and the RC framework in order to translate them to its quantum counterpart.
- Propose a new QELM method using gate-based quantum circuits as a reservoir, and explore some variants of the proposal.
- Compare the performance of the proposed QELM with the classical ELM and other classical models using some standard datasets for classification problems.

3. CHAPTER

Introduction to Quantum Computing

Here, we present a less formal but hopefully more intuitive approach to explaining the basic theory necessary to understand and recognise the significance of this work.

3.1 The qubit

Classical computation is based on binary logic. Here the information is carried on systems called bits that can take two states, either 0 or 1.

On the other hand, Quantum computation relies upon a physical system called a qubit being in a state. The computational basis states are labelled as $|0\rangle$ and $|1\rangle$; these are the quantum analogues of the classical binary states 0 and 1.

The notation used before to label the states is known as *bra-ket* notation (or Dirac notation) [5] and is a helpful way of expressing vectors in quantum mechanics, such as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{3.1}$$

These are known as *kets*, and their Hermitian conjugates are known as *bras*,

$$|0\rangle^{\dagger} = \langle 0| \tag{3.2}$$

A critical difference between the quantum case and the classical one is that a qubit can exist in a continuum of states between $|0\rangle$ and $|1\rangle$; [6], this is known as superposition, which can be written as a linear combination of the computational basis states,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.3}$$

where $|0\rangle$, $|1\rangle$ are orthonormal basis states, which satisfy

$$\langle 0|0\rangle = 1, \langle 1|1\rangle = 1, \langle 0|1\rangle = 0, \langle 1|0\rangle = 0, \tag{3.4}$$

and $\alpha, \beta \in \mathbb{C}$ are complex probability amplitudes that satisfy $|\alpha|^2 + |\beta|^2 = 1$. Therefore, the general form of a single qubit state can be expressed as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \theta \in [0,\pi], \phi \in [0,2\pi] \tag{3.5}$$

where ϕ and θ are real numbers that define a point on a unit three-dimensional sphere, as shown in Figure 3.1. This is usually known as *Bloch sphere*, and transformations on a single qubit are seen as rotations on this sphere.

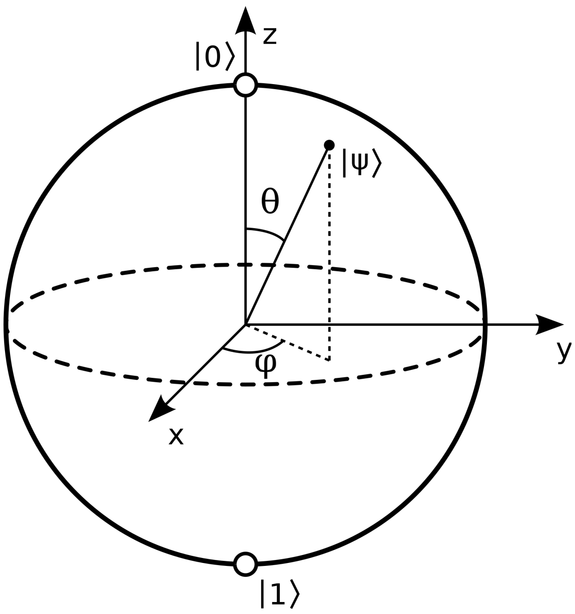


Figure 3.1: Bloch sphere representation of a qubit

Combining a *bra* and a *ket*, we can form an inner product,

$$\langle\psi|\psi\rangle \equiv \langle\psi|\psi\rangle = \|\psi\|^2 = 1, \tag{3.6}$$

and can also facilitate the outer product,

$$|\psi\rangle\langle\psi| = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}\begin{pmatrix}\alpha^* & \beta^*\end{pmatrix} = \begin{pmatrix}|\alpha|^2 & \alpha\beta^*\\ \beta\alpha^* & |\beta|^2\end{pmatrix}. \tag{3.7}$$

3.1.1 Multiple qubits

We can scale up the system, including additional qubits, so the quantum state of such unentangled qubits can be described by the tensor product of the individual quantum states. For example, if we have a system with two qubits, $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, they compose a system that can written as

$$\begin{aligned}
|\psi_1\rangle \otimes |\psi_2\rangle \equiv |\psi_1\rangle|\psi_2\rangle \equiv |\psi_1\psi_2\rangle && (3.8)\\
= (\alpha_1|0\rangle + \beta_1|1\rangle)(\alpha_2|0\rangle + \beta_2|1\rangle) && (3.9)\\
= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle && (3.10)\\
= \begin{pmatrix}\alpha_1\alpha_2\\ \alpha_1\beta_2\\ \beta_1\alpha_2\\ \beta_1\beta_2\end{pmatrix}. && (3.11)
\end{aligned}$$

where $\otimes$ is the Kronecker product.

This can be generalised to a system with n qubits, $|\psi_1\ldots\psi_n\rangle$, where the basis state $|x\rangle$ is located in the $(x+1)^{th}$ position of the vector that belongs to a particular vector space, namely the Hilbert space, that can be described as

$$\underbrace{\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{\text{n times}} \equiv (\mathbb{C}^2)^{\otimes n}. \tag{3.12}$$

So to describe a n qubit system, we need $2^n - 1$ numbers to describe it. This allows quantum systems to have more degrees of freedom compared to classical methods, e.g., in principle, a system with $n = 300$ qubits could perform more calculations at once than atoms in the observable universe.

3.1.2 Quantum gates

The classical gates are the basic building blocks of classical computers; similarly, on the quantum computer, we have the quantum gates. But comparing the classical setting, the quantum one is quite different because qubits can exist in superpositions of states.

Quantum gates are unitary transformations that are described by unitary matrices U, such that $U^\dagger U = I$. For example, the CX (controlled-NOT) gate, flips the second (target) qubit if the first (controlled) qubit is a $|1\rangle$. Thus, in this setting, the action can be represented on the computational basis by the matrix

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.13}$$

The CX gate has particular importance in quantum computation, mainly because it is the standard form to introduce entanglement into a quantum system.

3.1.3 Measurement

Quantum states are changed when we measure them, referred to as *wave function collapse*. More specifically, we measure the state in a orthonormal basis, for example the computational basis, after which it will collapse into the state $|0\rangle$ or $|1\rangle$, with probabilities $\langle 0|\psi\rangle = |\alpha|^2$ and $\langle 1|\psi\rangle = |\beta|^2$ respectively. After the measurement, the system will continue evolving from that collapsed state.

We can measure all the systems, collapsing the complete system, or we can measure a chunk of it, only collapsing the measured part and keeping the rest of the system in superposition.

We measure an observable, M, which is a Hermitian operator ($M^\dagger = M$) that has a spectral decomposition

$$M = \sum_m m P_m \tag{3.14}$$

where P_m is the projection operator onto the eigenspace of M with eigenvalue m. When measuring this observable for a state $|\psi\rangle$, the possible outcomes are given by the eigenvalues, m, for which the state will collapse to $\frac{P_m|\psi\rangle}{\langle\psi|P_m|\psi\rangle}$. Therefore, the probability of obtaining the measurement m is given by $\langle\psi|P_m|\psi\rangle$.

3.1.4 Expectation value

A measured observable can be seen as a discrete random variable, taking specific values (eigenvalues of M) with certain probabilities. We can define the expectation of this observable as

$$\begin{aligned}
\mathbb{E}(M) &\equiv \sum_m m\mathbb{P}(M = m) && (3.15)\\
&= \sum_m m\langle\psi|P_m|\psi\rangle && (3.16)\\
&= \langle\psi|\sum_m mP_m|\psi\rangle && (3.17)\\
&= \langle\psi|M|\psi\rangle\,. && (3.18)
\end{aligned}$$

This can also be written as

$$\langle M\rangle_{|\psi\rangle}\,. \tag{3.19}$$

However, in reality, we cannot measure this directly, and in practice, we will need to run the circuit many times (known as *shots*) and then calculate the average value. This can also be seen in how often the circuit should be evaluated (or "sampled") to estimate statistical values.

3.1.5 Quantum entanglement

Quantum entanglement is a particular feature related to quantum information and quantum computation. Because of this feature, some states in the Hilbert space cannot be written as a tensor product of multiple single qubit states. Instead, these states are *entangled*, mainly because the qubits have intrinsic influences over each other, i.e. they cannot be described independently.

One of the most used examples is the Bell pair, defined as

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{3.20}$$

cannot be written in a tensor product form like

$$(\alpha_1 |0\rangle + \beta_1 |1\rangle)(\alpha_2 |0\rangle + \beta_2 |1\rangle) \tag{3.21}$$

for any $\alpha_1, \beta_1, \alpha_2, \beta_2 \in \mathbb{C}$. When a single qubit of this state is measured, we immediately know the other qubit state.

3.1.6 Quantum circuits

As we want this work to be self-contained, here we explain the basics of quantum circuits to understand those shown in Section 5.2.

In a quantum circuit diagram, each solid depicts a qubit, or more generally, a qubit register. By convention, the top line is assigned to qubit 0, and the others are labelled sequentially. Operations are represented by quantum gates acting on one or more qubits, denoted as a box. For example, Figure 3.2 shows the Hadamard operation acting on a single qubit.

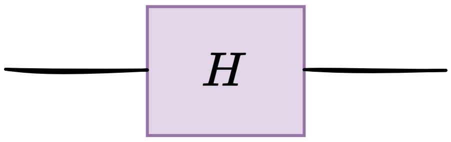


Figure 3.2: Hadamard gate applied to a single qubit

On those diagrams, time flows from left to right. Quantum gates are ordered in chronological order, with the left-most gate as the gate applied first to the qubits. For example, Figure 3.3 describes A, B, C operations applied to a single qubit, forming the unitary matrix $U = CBA$; notice that the convention for matrix multiplication obeys the opposite, with the right-most matrix applied first.

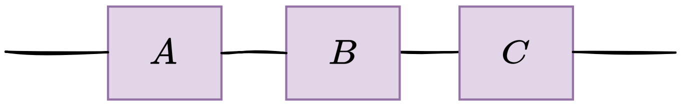


Figure 3.3: A sequence of three gates applied to a single qubit

Consider the unitary operation $CNOT_{01}(H \otimes 1)$; this gate sequence creates a maximally entangled two-qubit state:

$$CNOT_{01}(H \otimes 1) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{3.22}$$

Figure 3.4 shows the circuit for preparing this maximally entangled quantum state. Notice that the symbol behind the Hadamard gate represents the *CNOT* gate, where the black circle indicates the control qubit, and the cross indicates the target qubit.

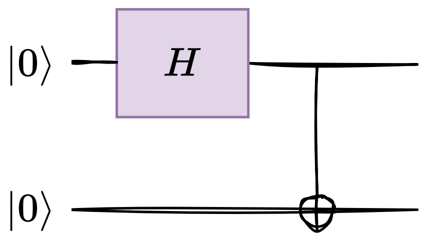


Figure 3.4: A circuit for preparing the maximally entangled quantum state from the initial state

A critical point about quantum circuits is that they need to have the same number of inputs and outputs. As all quantum operations, except measurement, are unitary and therefore reversible, if they did not have the same number of outputs as inputs, they would not be reversible and hence not unitary.

The remaining operation to visualise in quantum circuit diagrams is the measurement operator. As you can see in Figure 3.5, the measurement operation is denoted by a meter symbol. This operation takes a qubit register (solid line), measures it, and the output is saved in a classical information register (double solid lines).

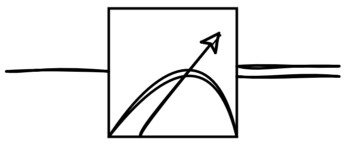


Figure 3.5: A measurement operator acting on a single qubit

A particular type of quantum circuit is the variational circuit. This circuit depends upon classical parameters. This allows us to convert classical information into quantum information; this is further explained in Section 5.2.1.

4. CHAPTER

State of the art

In this chapter, we study the current methods proposed for QELM and QRC. The reason to add QRC here is that it is the most studied method in the literature, and the recent advances are also valuable for QELM.

First, note that the research community uses both QELM and QRC terms interchangeably, sometimes referring to QELM as QRC for non-temporal tasks. Here, we only use QRC to refer to jobs applied to temporal tasks to maintain a consistent nomenclature.

As QRC and QELM are novel approaches to quantum neuromorphic computing, they are of great interest in the areas of machine learning and quantum computing. Following the initial proposal of quantum spin networks as reservoir substrates [7], there has been a variety of work exploring the possibilities that quantum mechanics can offer to this area of research. At the moment, the spin-based implementation is the most used quantum platform for QRC [8, 9, 10, 11, 12, 13, 14]. Also, for QELM, fermionic and bosonic setups have been proposed for light field phase estimation [15], and entanglement detection [16], which are typically challenging to extract from experimental setups. The use of QELM has also been reported in an NMR experiment [17].

Regarding the classical tasks, [18] proposes using quantum circuits as a substrate for classifying synthetically generated data. Additionally, the authors suggest that the feature space enhanced by quantum entangling operations is essential to obtain good performance in QELM [18]. On the other hand, [19] also uses quantum circuits, but in this case, to study the electronic ground and first excited states of two molecules. However, they are not limited to analysing the model's performance; they also study the complexity of the quantum circuits, using the majorization principle [20] as an indicator.

As far as know, there are, at present, no attempts to apply the QELM framework to the most studied classical tasks like image classification. Instead, recent proposals have applied QRC to this task [21, 22], treating images as temporal data, that is, decomposing them into a series of time steps and introducing each one of them into the quantum circuit at a time. For each time step, they measure the circuit using a technique called *mid-circuit measurement*[1]. Notice that this technique has not been implemented yet in all the circuit-based quantum hardware available. The same strategy is used in [4], where they apply QRC for classifying objects from time-series data generated from the sensor robot that grabs them.

QELM has also been proposed for quantum chemistry [23], being an example of a quantum task. Here, the first two excited molecular energies, transition dipole moment between the ground states, and the corresponding excited states are predicted from the ground-state wave function of the molecule.

[1]https://quantum-computing.ibm.com/lab/docs/iql/manage/systems/midcircuit-measurement/

5. CHAPTER

ELM and QELM

5.1 Extreme Learning Machine

Extreme Learning Machine is a learning algorithm that, unlike most other neural network learning algorithms, does not require gradient-based backpropagation to work. Instead, it uses Moore-Penrose's generalised inverse [24] to set the weights of the output layer, setting all the other connections randomly.

5.1.1 Formal description

For N distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in R^n$ are the features associated to the samples, and $y_i = [y_{i1}, y_{i2}, ..., y_{im}]^T \in R^m$ are the true labels of the samples, a standard single-hidden layer feed-forward network with L hidden nodes and activation function $g(x)$ is modelled as

$$f_L(x) = \sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = o_j, j = 1, ..., N \tag{5.1}$$

where $w_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the bias term of the ith hidden node, and $w_i \cdot x_j$ denotes the inner product of w_i and x_j.

This feedforward network can approximate the N samples with zero error such that $\sum_{j=1}^{L} \|o_j - y_j\| = 0$, so there exist β_i, w_i, and b_i such that

$$f_L(x) = \sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = y_j, j = 1, ..., N \tag{5.2}$$

This can be written as $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H}$ is the output matrix of the hidden layer and $\mathbf{T}$ is the target matrix for the training data. We can also write it as an optimisation objective such that

$$\|\mathbf{H}\hat{\beta} - \mathbf{T}\| = min_{\beta}\|\mathbf{H}\beta - \mathbf{T}\| \tag{5.3}$$

and the smallest norm least-squares solution of the above system is

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{5.4}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$. We use the generalised inverse because there could be issues calculating the inverse of the matrix if the matrix is singular. Note that constructing the pseudoinverse is equivalent to the usual inverse in nonsingular (thus invertible) matrices, and approximates well when the matrix is singular [25].

5.1.2 ELM algorithm

The whole process explained above can be summarised as a four-step algorithm that, given a training set $\phi = (x_i, y_i) | x_i \in R^n, y_i \in R^m, i = 1, ..., N$, and hidden node number L,

1. Randomly assign input weight w_i and bias b_i, for $i = 1, ..., L$.
2. Calculate the output matrix of the hidden layer $\mathbf{H}$.
3. Calculate the output weight $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{T} = [t_1, ..., T_N]^T$.
4. Use $\hat{\beta}$ to make a prediction on new data $\hat{\mathbf{T}} = \mathbf{H}\hat{\beta}$.

5.2 Quantum Extreme Learning Machine

The idea of a quantum extreme learning machine lies in replacing the complex classical dynamical system with a quantum one, using the Hilbert space, where quantum states live (a complex high-dimensional space) as an enhanced feature space of the input data. Nevertheless, post-processing the output in the same way as we did in the classical algorithm.

5.2.1 Data encoding

The first issue to address is introducing all the data into the quantum system. Data encoding is a crucial part of all quantum machine learning models, so the expressive power of various strategies has been studied before [26]. Approaches such as basis encoding, amplitude encoding, and many others have been studied in the recent literature [27, 28]. However, not all of these encoding strategies are suitable for our purpose. Here, we introduce the amplitude and angle encodings, two of the most widely used encoding schemes appropriate for our objective.

With *amplitude encoding*, the data are encoded in the amplitudes of a quantum state. A classical N-dimensional data point x is represented by the amplitudes of a n-qubit quantum state $|\psi_x\rangle$ as

$$|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle \tag{5.5}$$

where $N = 2^n$, x_i is the i-th element of x and $|i\rangle$ is the i-th computational basis state. Since classical information forms the amplitudes of a quantum state, the input must satisfy the normalisation condition $|x|^2 = 1$.

This method encodes 2^n inputs in n qubits, providing an exponential advantage in terms of storage. Unfortunately, preparing the superposition state requires a complicated circuit whose depth scales exponentially [29]. This may be a limitation in using amplitude encoding on larger quantum computers.

Angle encoding uses rotation gates to encode the data, so given an input feature vector $x = [x_1, ..., x_N]^T \in R^N$, the encoding can be described as

$$|x\rangle = U(x_i) |0\rangle^{\otimes n} \tag{5.6}$$

where U is a unitary operation parameterized by x_i such as rotation gates R_x, R_y, R_z. This encoding strategy encodes N features in n qubits with a constant depth quantum circuit, where $n \geq N$, so it is very efficient in terms of operations, only requiring a single qubit rotation to realise it [30], but is not optimal in the number of qubits used [31].

One of the advantages of using angle encoding is that the tensor product produces a number of nonlinear basis functions, exponential in the number of qubits [32]. Other classical approaches need to increase the depth of the circuit to obtain a more significant number of nonlinear basis functions and increase precision [33].

Despite not being the most efficient in terms of storage, with all the strengths shown above, we have decided to use angle encoding to encode our data into quantum circuits. We make an additional trick to increase the model's precision, modifying angle encoding to contain more information as we make the quantum system bigger. First, the data is encoded into the quantum system, all this data is scaled to $[0,1]$. Then, we duplicate and slightly modify the data, using the complete quantum system when performing angle encoding. For example, suppose we have a scaled x input vector and a system with n qubits. We can define the modified encoding scheme as

$$|x\rangle = \bigotimes_{i}^{n} i * x_{(i\%N)}, \tag{5.7}$$

where for each qubit, we multiply the qubit index $i \in [0,n]$ with the corresponding data point x chosen, using the modulus operation between the qubit index and the total number of features N.

The main idea behind this is to take advantage of the complete Bloch sphere, projecting every data point more dispersedly and achieving a precision increase when measuring. Notice that if we use the standard angle encoding with a data point $x \in [0,1]$, we only use one quadrant of the Bloch sphere, concentrating all the information in this small region and making the classification after the measurement less accurate.

Apart from our method, another possibility is to scale the data to $[0,2\pi]$ and encode it without data duplication. As we are also using the complete Bloch sphere, this method improves the performance; however, we see that our scheme still works better.

5.2.2 Quantum substrate

To check whether the idea of QELM is working, we keep the quantum circuit as simple as possible, not adding extra complexity but maintaining the main advantages of using a quantum system as a substrate.

As seen in Figure 5.1, the circuit is made up of three main parts; the first is the input encoding part explained in the previous section, and a cascade of CX gates forms the second to generate additional entanglement between qubits. Finally, since we cannot directly

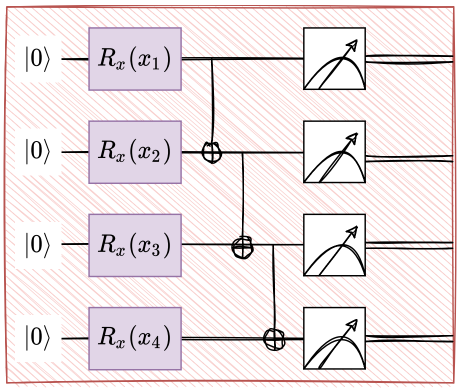


Figure 5.1: Quantum circuit example

access none of the information, a suitable observable M should be measured. This can be done for all the qubits or a subset.

Formally, this quantum circuit can be described as follows, having a n-qubit system, which is initialised to

$$|0\rangle^{\otimes n}. \tag{5.8}$$

We feed the input into the quantum system using the input encoding applied to the initial state

$$U(x)\,|0\rangle^{\otimes n}. \tag{5.9}$$

Denoting the cascade of gates $CX\ CX_{n-1,n}\dots CX_{2,3}CX_{1,2}$ as $\sigma(X)$ and the entire circuit action as $V(x)$, this will create the state

$$V(x)\,|0\rangle^{\otimes n}. \tag{5.10}$$

To extract the information, we select to average the Z observable defined by the matrix

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5.11}$$

in one qubit, with eigenvectors $|0\rangle, |1\rangle$ and eigenvalues $1, -1$, respectively. To calculate the average from the measurements, running and measuring the circuit $s >> 1$ times, with s known as shots, will approximate

$$\left\langle Z^{\otimes n} \right\rangle_{V(x)|0\rangle^{\otimes n}}. \tag{5.12}$$

From the measurements, we obtain a vector $h \in R^{1\times n}$ of the expectation values, where n

is the number of qubits in the system, seeing this h vector as the output of the quantum substrate.

5.2.3 Training strategies

The original idea of the classical ELM is to treat the whole training set all at once, passing every example through the dynamical system, i.e. through the single layer feed-forward network (SLFN). However, we cannot directly translate this strategy and use the quantum system as a quantum dynamical system, i.e., the quantum substrate.

Due to the input encoding strategy selected (restricted to N features, with $N < n$, where n is the number of qubits of the quantum system), and the size of the current quantum systems (with $n \leq 24$ qubits), we are limited in the number of examples that we can pass through the quantum substrate at a time. Taking this into account, we decide to treat each example individually, encoding N features for each instance.

As explained above, we get a vector $h \in R^{1\times n}$ of expectation values of Z observable from the quantum circuit; this is done for every example. Then we concatenate all of those vectors $h_i \in R^{1\times n}, i = 1,\ldots,m$, where m is the number of examples on the training set. So we construct a $H \in R^{n\times m}$ matrix concatenating all the intermediate representations. This is equivalent to treating all the examples of the training set at once, i.e., the classical method used in the original ELM.

After constructing the matrix H, we use the same procedure in the original ELM explained in Section 5.1.1. Still, instead of calculating the $\hat{\beta}$ weights using the H matrix we get from the classical feed-forward hidden layer, we use the one constructed using the intermediate representations calculated by the quantum substrate.

To summarise, we introduce each example through the quantum substrate; then, we get an intermediate representation for each sample using the expectation values; hereafter, we use those representations to construct the H matrix. Then we calculate the Moore-Penrose generalised inverse of this matrix, that is, $H^{\dagger}$, and use it to calculate the optimised $\hat{\beta}$ weights. Notice that those weights are then used on inference time, as we did with ELM. Finally, we show the entire process graphically in Figure 5.2.

Apart from the method we explain above, we explore other strategies to see the potential of different ideas and variations of the original algorithm.

The first variation we explore is to change the training algorithm, that is, to use another

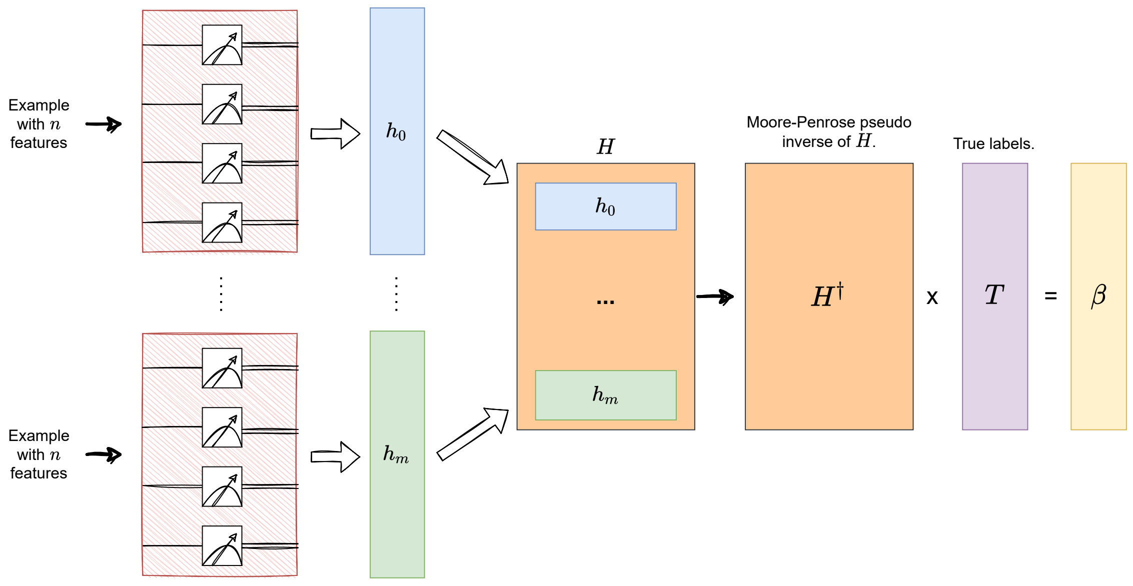

Example with n features
h_0
h_m
Example with n features
H
h_0
...
h_m
Moore-Penrose pseudo inverse of H.
$H^\dagger$
x
True labels.
T
=
β

Figure 5.2: QELM training illustration

optimisation algorithm to train the model, thus maintaining the quantum substrate as a computational resource. We choose the classical gradient descent algorithm for the optimisation algorithm, changing the learning process into an iterative one.

We also propose using an ensemble of quantum substrates, passing every example for each one of them and combining the outputs. In this case, we continue with the same training process, solving it as a one-step optimisation process.

However, we discard both ideas, the first one because we do not see any performance improvement, but we have seen worse performance and much slower training times. We think this is mainly because we eliminate the original efficient optimisation process, making it iterative and slow.

Although we have seen improvement in model accuracy for the ensemble of quantum substrate variation, training and inference times are unfeasible. For training the model with this method, we need to pass every example with N features through a quantum system with n qubits T times, where T is the number of substrates. This makes the process very costly in the number of operations and slows the model.

Notice that another option could be to replace the optimisation algorithm with a similar one. For example, the *Ridge Regression* method solves linear least squares but uses the *L2-norm regularisation* technique to reduce overfitting. However, we did not implement this variation because it is beyond the scope of this work.

6. CHAPTER

Results

Since quantum computers are currently more difficult to access than classical computers, all the experiments have been carried out using quantum simulators [34]. These simulators perform all the linear algebraic operations that mathematically describe quantum systems. This is very demanding in terms of computation, so to reduce this overhead, all the benchmarks are done using small and not very complex (in terms of the number of features) standard datasets for binary classification problems.

6.1 Experimental setup

6.1.1 Execution environment

We have explored a wide range of quantum simulators from different quantum software libraries. In the end, we chose a combination of two of them, namely Pennylane [1], by Xanadu, and Qiskit [2], by IBM. We use the first one to build the quantum circuits, making it possible to run them with most of the simulators and real hardware available. In particular, we use the ideal (noise-free) *qiskit.aer* simulator provided by Qiskit to run the circuits.

Apart from the noiseless simulations done using the Aer simulator, we also perform noisy simulations. More specifically, we use the *Noise Model* object provided by Qiskit, with

[1] `https://pennylane.ai/`
[2] `https://qiskit.org/`

which we can add the noise parameters of the real IBM quantum processors to our simulated quantum circuits. It should be noted that these noise emulations currently have their limitations and are not exact replications. All the simulations, both noiseless and noisy, have been executed on 1024 shots.

We build our noise model using the 16 qubits *ibmq_guadalupe* quantum computer noise parameters. That is, for depolarizing noise, an Avg. CNOT error of $p \approx 0.01$ and an Avg. Readout error of $p \approx 0.02$, and for thermal relaxation noise, an Avg. Amplitude damping $T1 \approx 93\mu s$, and an Avg. Phase damping $T2 \approx 95\mu s$.

On the other hand, even having tried to use real quantum hardware provided by *IBM* and *IonQ*, the restrictive access queue policies have made it unfeasible.

6.1.2 Datasets

For the datasets, we have selected both synthetically generated data using the *scikit-learn* library (SKL) and real data from the *UCI Machine Learning Repository*.

To aid the reader in judging the results, we will briefly describe the datasets using a short description of the problem, the number of examples, and the number of features. Also, notice that the dataset names use the format [UCI | SKL]_name, where the first part of the name describes the data source, and the second part is the name of the dataset.

- *SKL_make-circles*: produces Gaussian data with a spherical decision boundary for binary classification. There are 100 samples (points), and 2 features (x, y).
- *SKL_make-gaussian_quantiles*: divides a single Gaussian cluster into near-equal-size classes separated by concentric hyperspheres for binary classification. There are 100 samples (points), and 2 features (x, y).
- *SKL_make-moons*: produces two interleaving half circles for binary classification. There are 100 samples (points), and 2 features (x, y).
- *SKL_make-classification*: generate a random binary classification problem with some noise. There are 100 samples (points), and 2 features (x, y).
- *UCI_breast-cancer*: there are 569 samples (patients), and 30 features (thickness, cell size uniformity, etc.). The variable to predict is encoded as 2 (benign) and 4 (malignant).

- *UCI_banknote-authentication*: there are 1372 samples (images of banknotes), and 4 features (variance of the image, skewness, kurtosis, and entropy). The variable to predict is encoded as 0 (authentic) and 1 (forgery).
- *UCI_haberman-survival*: there are 306 samples (patients), and 3 features (age, year of operation, number of positive auxiliary nodes detected). The variable to predict is encoded as 0 (survived) and 1 (died).
- *UCI_early-diabetes*: there are 520 samples (questionnaires from patients), and 16 (age, sex, sudden weight loss, alopecia, etc.). The variable to predict is encoded as 1 (positive) and 0 (negative).

Name	**Examples**	**Features**
SKL_make-circles	100	2
SKL_make-gaussian_quantiles	100	2
SKL_make-moons	100	2
SKL_make-classification	100	2
UCI_breast-cancer	569	30
UCI_banknote-authentication	1372	3
UCI_haberman-survival	306	3
UCI_early-diabetes	520	16

Table 6.1: Summary of the datasets

6.1.3 Training and evaluation framework

Every dataset is divided into two subsets, namely training and test sets. We keep the 80% of the dataset for the train set and the remaining 20% for the test set. Note that these splits are made using the same random seed to maintain the reproducibility of the experiments.

We test the scalability of quantum systems using different numbers of qubits; more precisely, we select a set of various qubits depending on the number of features for each problem. However, the system with the highest number of qubits is still tiny to keep the experiments as close as possible to the actual hardware.

The problem we try to solve is a binary classification problem, but as some of the datasets are not balanced in the number of labels to predict, so we need to be careful with the metrics we use.

For balanced datasets, we have used the accuracy metric defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{6.1}$$

where *TP* are true positives, *TN* true negatives, *FP* false positives, and *FN* false negatives. On the other hand, for unbalanced datasets, as the accuracy can be misleading in these cases, we have decided to use a combination of three metrics, namely precision, recall, and F1 score. Those are defined as follows

$$Precision = \frac{TP}{TP + FP}, \tag{6.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{6.3}$$

$$F1\_score = \frac{2 * (Precision * Recall)}{Precision + Recall}. \tag{6.4}$$

In Section 6.2, we use precision, recall and f1 score to evaluate the models on unbalanced datasets. However, as the last one is defined as the harmonic mean of the other two, we will use the f1 score as the primary metric to measure the performance even though we present all three metrics.

In order to provide more reliable results, each training and evaluation process is performed 5 times, taking the mean and standard deviation of the computed metrics.

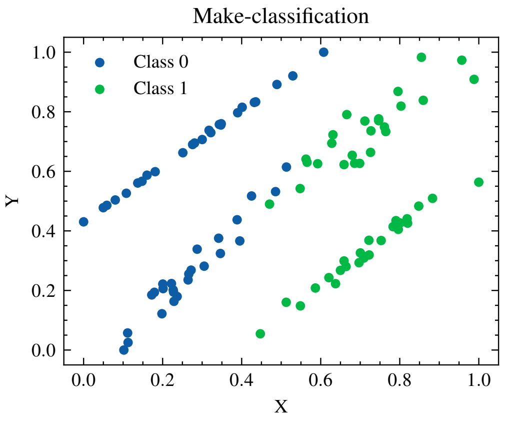
6.2 Analysis of the results

We first want to check if the model can perform well with linearly and non-linearly separable data, using four synthetically generated data from *scikit-learn* library. Then, as a sanity check, we run a simple linear model incapable of correctly classifying non-linear data, particularly a linear Support Vector Machine (SVM) [35].

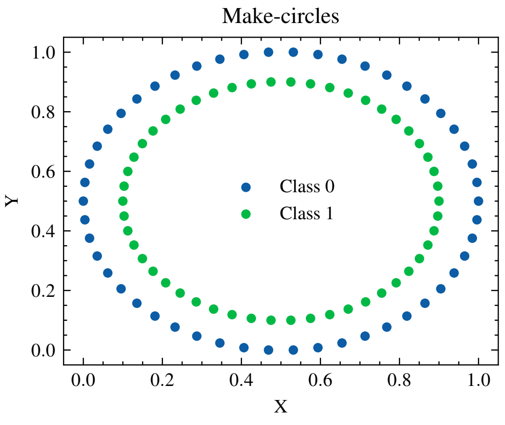
Once we have verified that our model is capable of performing well with linear and non-linear data, we test our model on the real datasets from *UCI Machine Learning Repository* against the ELM and a Random Forest Classifier (RF) [36].

6.2.1 Scikit-learn datasets

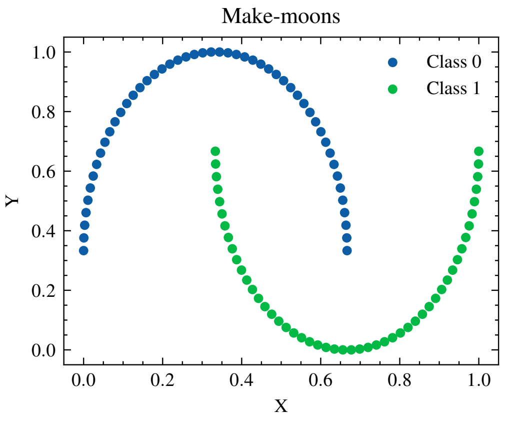
As shown in Figure 6.1, we have one linearly separable dataset, make-classification (a), and three non-linearly separable datasets, namely make-circles (b), make-moons (c), and make-gaussian_quantiles (d). All these datasets are balanced in the number of labels to predict; therefore, the use of accuracy is valid when measuring the model's performance.



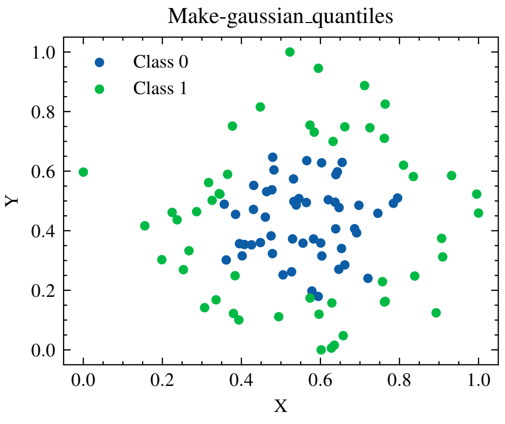
(a) Make-classification



(b) Make-circles



(c) Make-moons



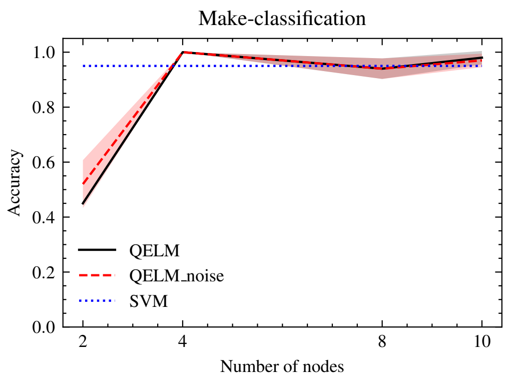
(d) Make-gaussian_quantiles

Figure 6.1: 2d visualization of the scikit-learn datasets

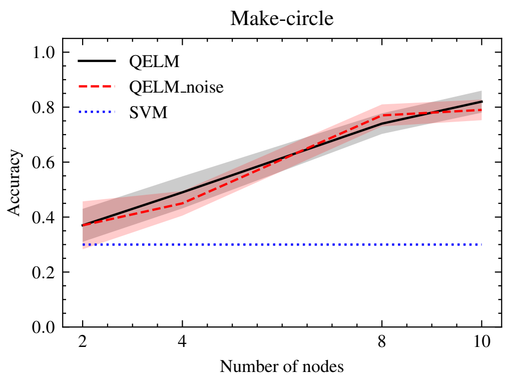
We start our analysis with the make-classification dataset, where, as can be seen in Figure 6.2 (a), for the system of 2 qubits the model performs worse than the linear SVM baseline. However, with a system of 4 qubits or more, the model matches or outperforms the linear baseline.

For the non-linearly separable datasets, our model outperforms the linear SVM on make-circle (b) and make-gaussian (d). Being QELM, capable of classifying correctly around the 80% of the test examples for both cases with a system of 8 qubits. On the other hand, SVM classifies around 40% of the examples correctly, that is, worse than a random model. In the case of the make-moons datasets, we get a weird result, with SVM classifying the test set perfectly and QELM going from the accuracy of 82.5% with two qubits to classifying perfectly all examples with eight qubits. The not expected extraordinary performance of the SVM may have been due to the small number of examples in the dataset and favourable train-test splits, where the linear model has been able to separate the data ideally with a line.

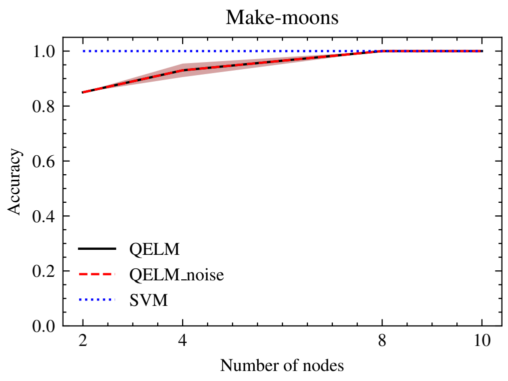
We do not see any significant influence of the noisy simulations, being very similar in performance to the noiseless ones.



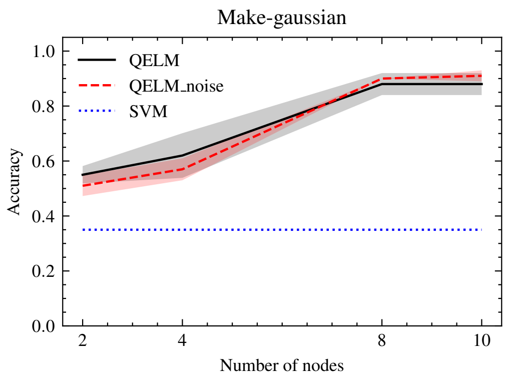
(a) Make-classification



(b) Make-circles



(c) Make-moons



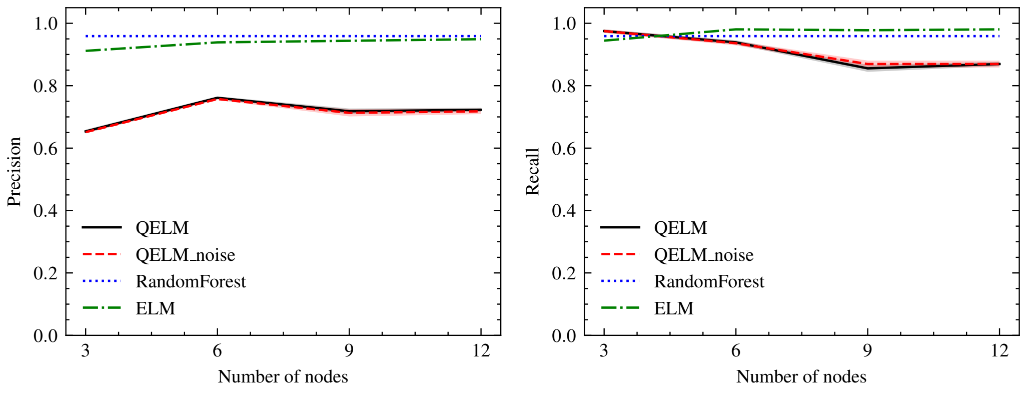
(d) Make-gaussian_quantiles

Figure 6.2: Accuracy score for scikit-learn datasets

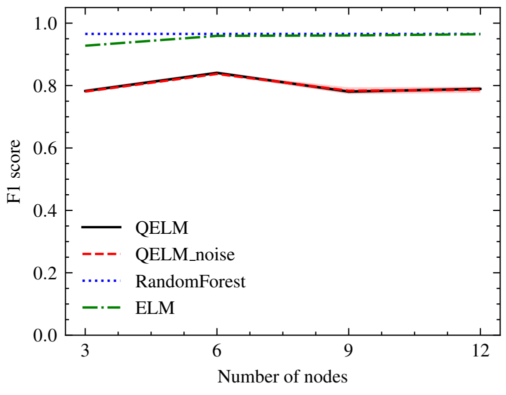
6.2.2 UCI breastcancer

As this dataset contains 30 features, and with the encoding strategy we chose, we would need 30 qubits to get the information into the quantum system; we reduce the dimensionality with Principal Component Analysis (PCA) to three features. We use the same number of qubits for the smallest system, scaling it up to twelve qubits for the biggest one.

As shown in Figure 6.3, our model performs pretty well, getting around 80% of F1 score, but not even matching the classical models, with both Random Forest Classifier and ELM getting around 97% of F1 score. However, we get similar results to other works like [37], where they use other quantum methods like quantum-kernel support vector machine (qKSVM) and quantum distance classifier (qDS) with a higher number of features and real quantum hardware.



(a) Precision **(b)** Recall

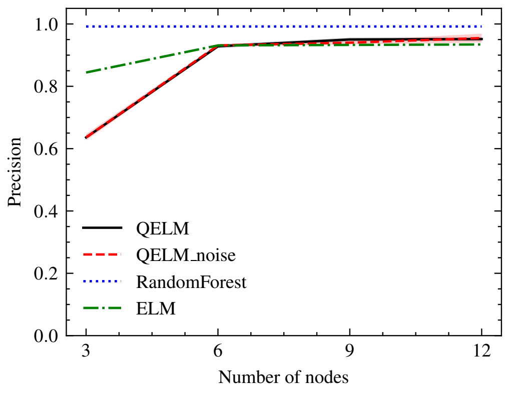


(c) F1 score

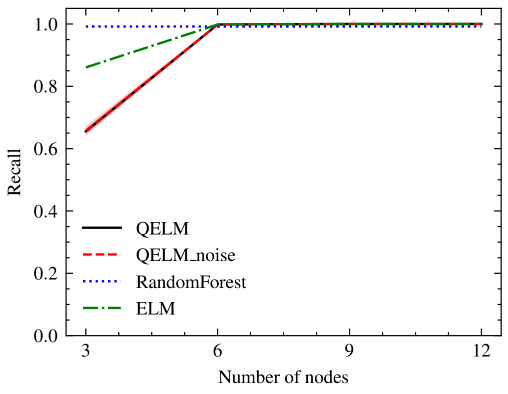
Figure 6.3: UCI Breastcancer results

6.2.3 UCI banknote authentication

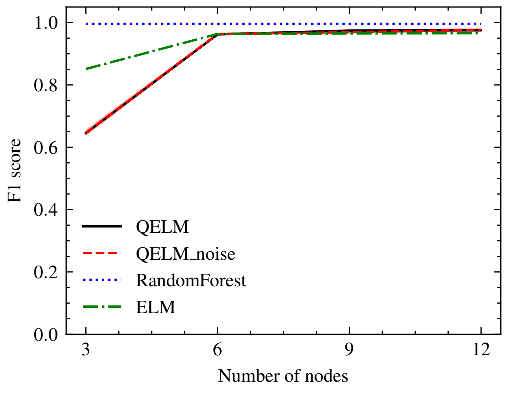
For this problem, as shown in Figure 6.4, our QELM model not only matches its classical counterpart but is also very close to the Random Forest Classifier, which is the standard benchmark for tabular data in the machine learning community. Here, we can see a clear performance boost when increasing the number of qubits of the quantum system, plateauing around six qubits. However, we get the same performance for both noisy and noiseless simulations, but highlight consistent results for both cases.



(a) Precision



(b) Recall

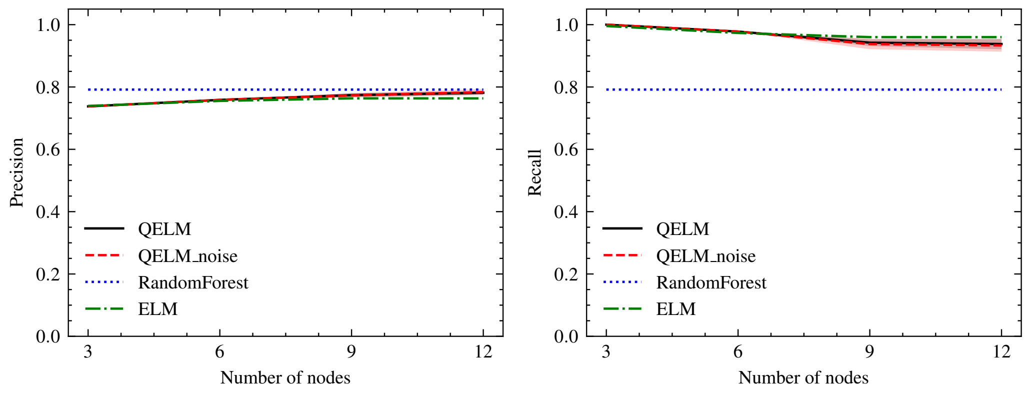


(c) F1 score

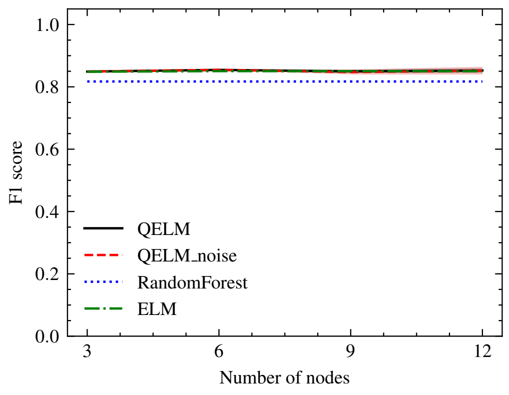
Figure 6.4: UCI Banknote authentication results

6.2.4 UCI haberman survival

As can be seen in Figure 6.5, both ELM and QELM have a similar performance, around 82% of F1 score, overcoming the Random Forest Classifier by a slight margin. Contrary to the previous problem, we do not see any significant improvement in the results when increasing the number of qubits in this problem, as well as we do not see any changes on the results when using noisy simulations. There have been other attempts to solve this problem using quantum methods like [38], where they achieve a performance similar to ours.



(a) Precision **(b)** Recall

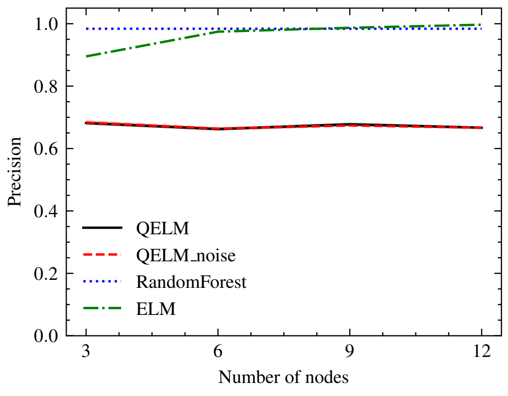


(c) F1 score

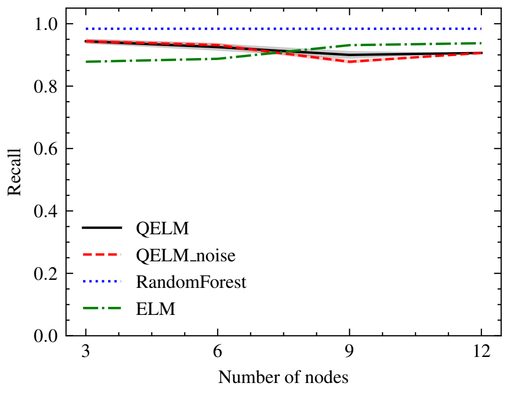
Figure 6.5: UCI haberman survival results

6.2.5 UCI early diabetes

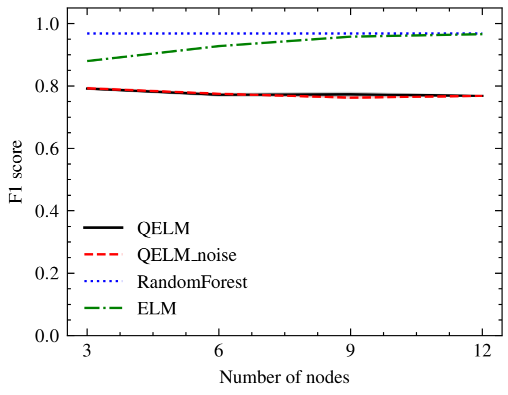
In the same way, as we did with the UCI breastcancer dataset in Section 6.2.2, we also reduce the dimensionality of this dataset to three features using PCA. As shown in Figure 6.6, this is the worst performance we get from our QELM model, with a gap between the classical models and the quantum model of roughly 20% in terms of F1 score. It is possible that by reducing the dimensionality, we also delete some vital information from the data, and our model is not capable of learning appropriately with a reduced number of relevant features. Additionally, as in the previous problem, we do not see any performance boost scaling the number of qubits or using noisy simulations.



(a) Precision



(b) Recall



(c) F1 score

Figure 6.6: UCI Early diabetes results

7. CHAPTER

Conclusions

Despite the recent efforts and improvements over the years in the field of quantum computing, the current NISQ quantum computers are far from ideal, suffering from unwanted noise. QELM tries to take advantage of this problem and presents itself as an alternative that can be implemented using the current real hardware.

We present and implement a circuit-based Quantum Extreme Learning Machine framework, successfully tackling supervised learning tasks in binary classification. However, our QELM could not outperform classical machine learning models, highlighting that it is important to maintain a grounded perspective when investigating novel, interesting quantum machine learning techniques.

Wanting to get closer to the results we would get with real hardware, we use noise models from real IBM quantum computers to simulate the dynamics of these systems. However, as these models are approximations of the error, and there is no classical computer that can faithfully simulate the complex dynamics of the quantum noise, we have not been able to find any performance boost when using them.

Quantum computers are growing fast in the number of qubits available and improving with new methodologies and topologies that reduce the upper bound of the noise present in the actual QPUs. All these advances will surely improve the current methods and may launch a new scenario where hybrid quantum-classical methods can provide a viable alternative to purely classical methods.

Finally, due to the versatility of the QELM and QRC frameworks, they can be implemented using a wide range of quantum platforms, opening a whole world of possibilities to quantum computing and machine learning researchers.

8. CHAPTER

Future work

The field of quantum based reservoir computing is still in its early stages, so there is much to be said for future directions. The first thing we could explore to improve this work could be to replicate this work but using a real quantum device, in order to determine if the results are improved by the richer dynamics of the quantum hardware, particularly with respect to noise variations and entanglement, but also to check if we could take advantage of the quantum speed-up [39]. As quantum computers become more accessible and their capabilities improve, there will be room for more information to be encoded into more qubits, as well as with more reliable results.

Other different encoding schemes could provide us with the opportunity to encode larger data into quantum computers. For example, in the idea of data reuploading [40], they discuss how to load a higher-dimensional data point by breaking it down into sets of sequentially parameterized gates using only one qubit.

Also, the use of other types of quantum substrates needs to be investigated. Nuclear magnetic resonance in molecules [17] and trapped ions are potential platforms for spin-based QELM systems [41], as well as spin systems in terms of quantum circuits as depicted in [42]. Neutral atom platforms are another option to consider, motivated by their long coherence times, quantum registers with a larger number of qubits, and higher connectivity [43]. Furthermore, other physical implementations are possible, for example, photonic experimental setups [44], arrays of quantum dots in semiconductor devices, and in coupled superconducting qubits [45].

Apart from the ideas shown above, we can consider other interesting lines of research, such as combining some quantum substrates as an ensemble of them. This has been explored in [46, 47], where they use several different substrates, running them in parallel and combining their outputs.

Finally, as we rely on quantum noise as a valuable resource and the current quantum computers are usually affected by these phenomena, studying what kind of noise in quantum hardware, and what structure of quantum processors is more suitable could be crucial [4, 48].

Bibliography

[1] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Commun. ACM*, vol. 57, p. 13–15, jun 2014.

[2] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," p. 13, 2006.

[3] P. Mujal, "Quantum Reservoir Computing for Speckle-Disorder Potentials," *arXiv:2201.11096*, Jan. 2022.

[4] Y. Suzuki, Q. Gao, K. C. Pradel, K. Yasuoka, and N. Yamamoto, "Natural quantum reservoir computing for temporal information processing," *Scientific Reports*, vol. 12, p. 1353, Jan. 2022.

[5] P. Dirac, "A new notation for quantum mechanics," 1939.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. USA: Cambridge University Press, 10th ed., 2011.

[7] K. Fujii and K. Nakajima, "Harnessing disordered-ensemble quantum dynamics for machine learning," *Phys. Rev. Applied*, vol. 8, p. 024030, Aug 2017.

[8] K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, and M. Kitagawa, "Boosting computational power through spatial multiplexing in quantum reservoir computing," *Phys. Rev. Applied*, vol. 11, p. 034021, Mar 2019.

[9] A. Kutvonen, T. Sagawa, and K. Fujii, "Optimizing a quantum reservoir computer for time series prediction," *arXiv:1807.03947*, June 2020.

[10] Q. H. Tran and K. Nakajima, "Higher-Order Quantum Reservoir Computing," Oct. 2020.

[11] J. Chen and H. I. Nurdin, “Learning Nonlinear Input-Output Maps with Dissipative Quantum Systems,” *Quantum Information Processing*, vol. 18, p. 198, July 2019.

[12] J. Chen, H. I. Nurdin, and N. Yamamoto, “Temporal information processing on noisy quantum computers,” *Phys. Rev. Applied*, vol. 14, p. 024065, Aug 2020.

[13] R. Martínez-Peña, J. Nokkala, G. L. Giorgi, R. Zambrini, and M. C. Soriano, “Information Processing Capacity of Spin-Based Quantum Reservoir Computing Systems,” *Cognitive Computation*, Oct. 2020.

[14] S. Dasgupta, K. E. Hamilton, and A. Banerjee, “Characterizing the memory capacity of transmon qubit reservoirs,” May 2022.

[15] L. C. G. Govia, G. J. Ribeill, G. E. Rowlands, H. K. Krovi, and T. A. Ohki, “Quantum reservoir computing with a single nonlinear oscillator,” *Phys. Rev. Research*, vol. 3, p. 013077, Jan 2021.

[16] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. H. Liew, “Quantum reservoir processing,” *npj Quantum Information*, vol. 5, p. 35, Dec. 2019.

[17] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and M. Kitagawa, “Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid,” June 2018.

[18] K. Fujii and K. Nakajima, “Quantum reservoir computing: a reservoir approach toward quantum machine learning on near-term quantum devices,” *arXiv:2011.04890*, Nov. 2020.

[19] L. Domingo, G. Carlo, and F. Borondo, “Optimal quantum reservoir computing for the NISQ era,” May 2022.

[20] R. O. Vallejos, F. de Melo, and G. G. Carlo, “The principle of majorization: application to random quantum circuits,” *Physical Review A*, vol. 104, p. 012602, July 2021. arXiv:2102.09999.

[21] F. Hu, S. Khan, G. Angelatos, and H. Tureci, “A Quantum Reservoir Computing Approach to Image Classification,” in *APS March Meeting Abstracts*, vol. 2021 of *APS Meeting Abstracts*, p. V32.001, Jan. 2021.

[22] M. V. Bastida, *Supervised Learning with Quantum Reservoir Computing*. Princeton University Senior Theses, Princeton University, 2021.

[23] H. Kawai and Y. O. Nakagawa, "Predicting excited states from ground state wavefunction by supervised quantum machine learning," *Machine Learning: Science and Technology*, vol. 1, p. 045027, Dec. 2020.

[24] A. Ben-Israel and T. Greville, "Generalized inverses: theory and applications," 1974.

[25] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406–413, July 1955.

[26] M. Schuld, R. Sweke, and J. J. Meyer, "The effect of data encoding on the expressive power of variational quantum machine learning models," *Physical Review A*, vol. 103, p. 032430, Mar. 2021.

[27] M. Weigold, J. Barzen, F. Leymann, and M. Salm, "Expanding Data Encoding Patterns For Quantum Algorithms," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pp. 95–101, Mar. 2021.

[28] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," Tech. Rep. arXiv:2001.03622, arXiv, Feb. 2020.

[29] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, p. 043001, Nov. 2019.

[30] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, p. 032420, Sept. 2020.

[31] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, "Hierarchical quantum classifiers," *npj Quantum Information*, vol. 4, pp. 1–8, Dec. 2018.

[32] E. Stoudenmire and D. J. Schwab, "Supervised Learning with Tensor Networks," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016.

[33] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, p. 032309, Sept. 2018.

[34] E. Altman, K. R. Brown, G. Carleo, L. D. Carr, E. Demler, C. Chin, B. DeMarco, S. E. Economou, M. A. Eriksson, K.-M. C. Fu, M. Greiner, K. R. A. Hazzard, R. G. Hulet, A. J. Kollar, B. L. Lev, M. D. Lukin, R. Ma, X. Mi, S. Misra, C. Monroe,

K. Murch, Z. Nazario, K.-K. Ni, A. C. Potter, P. Roushan, M. Saffman, M. Schleier-Smith, I. Siddiqi, R. Simmonds, M. Singh, I. B. Spielman, K. Temme, D. S. Weiss, J. Vuckovic, V. Vuletic, J. Ye, and M. Zwierlein, "Quantum Simulators: Architectures and Opportunities," *PRX Quantum*, vol. 2, p. 017003, Feb. 2021.

[35] C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, June 1998.

[36] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[37] S. Moradi, C. Brandner, C. Spielvogel, D. Krajnc, S. Hillmich, R. Wille, W. Drexler, and L. Papp, "Clinical data classification with noisy intermediate scale quantum computers," *Scientific Reports*, vol. 12, p. 1851, Feb. 2022.

[38] I. F. Araujo, D. K. Park, F. Petruccione, and A. J. da Silva, "A divide-and-conquer algorithm for quantum state preparation," *Scientific Reports*, vol. 11, mar 2021.

[39] Rønnow Troels F., Wang Zhihui, Job Joshua, Boixo Sergio, Isakov Sergei V., Wecker David, Martinis John M., Lidar Daniel A., and Troyer Matthias, "Defining and detecting quantum speedup," *Science*, vol. 345, pp. 420–424, July 2014.

[40] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, Feb. 2020.

[41] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, "Demonstration of the trapped-ion quantum CCD computer architecture," *Nature*, vol. 592, pp. 209–213, Apr. 2021.

[42] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, "Continuous-variable quantum neural networks," *Physical Review Research*, vol. 1, p. 033063, Oct. 2019.

[43] R. A. Bravo, K. Najafi, X. Gao, and S. F. Yelin, "Quantum reservoir computing using arrays of Rydberg atoms," *arXiv:2111.10956*, Nov. 2021.

[44] A. J. Kollár, M. Fitzpatrick, and A. A. Houck, "Hyperbolic lattices in circuit quantum electrodynamics," *Nature*, vol. 571, pp. 45–50, July 2019.

[45] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, pp. 209–212, Mar. 2019.

[46] G. Angelatos, H. Tureci, F. Hu, and S. Khan, "Quantum Annealing Systems as Reservoirs I: Design and Performance," in *APS March Meeting Abstracts*, vol. 2022 of *APS Meeting Abstracts*, p. W37.004, Jan. 2022.

[47] S. Vintskevich and D. Grigoriev, "Computing with two quantum reservoirs connected via optimized two-qubit nonselective measurements," May 2022.

[48] T. Kubota, Y. Suzuki, S. Kobayashi, Q. H. Tran, N. Yamamoto, and K. Nakajima, "Quantum Noise-Induced Reservoir Computing," July 2022.

[49] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, "Opportunities in Quantum Reservoir Computing and Extreme Learning Machines," *Advanced Quantum Technologies*, vol. 4, no. 8, p. 2100027, 2021.

[50] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*. Quantum Science and Technology, Springer International Publishing, 2018.