uc3m | Universidad Carlos III de Madrid

Master Degree in Statistics for Data Science
2023-2024

Master Thesis

Comparison of quantum SVM and LAMDA in classification problems

Álvaro Fernández Blanco

Rosa Elvira Lillo Rodríguez
Jose Lisandro Aguilar Castro

Madrid, June 25, 2024

SUMMARY

Classification problems are pivotal in machine learning, with applications in fields such as medical diagnosis, image recognition, and fraud detection. Efficiently categorizing data into predefined classes enhances decision-making and operational processes. Support Vector Machines (SVM) and Learning Algorithm for Multivariate Data Analysis (LAMDA) are state-of-the-art techniques for these tasks. In addition, quantum computing has recently emerged as a transformative approach to enhance computational efficiency and security.

Within this general context, the present work studies the integration of SVM and LAMDA algorithms with quantum computing techniques to improve classification performance. Objectives include implementing and comparing quantum and classical versions of SVM and LAMDA, particularly for binary, multi-class, and multi-label classification (LAMDA only for binary classification). The structure of the thesis includes a review of quantum physics fundamentals, a detailed analysis of SVM and LAMDA classical and quantum algorithms, and a comparative study of their performance and efficiency.

Keywords: SVM, LAMDA, binary, multi-class, multi-label, quantum computing, qubit, efficiency, performance.

Contents

List of Figures

List of Tables

1 Introduction

Classification problems are a cornerstone in the field of machine learning, with applications spanning a wide range of domains such as medical diagnosis, image recognition, and anomaly or fraud detection [1]. These problems involve categorizing data into predefined classes, and their successful resolution can lead to optimized decision-making and enhanced operational efficiency across various sectors. Successful classification can streamline processes, improve accuracy in predictions, and facilitate better resource allocation, ultimately driving advancements and innovation in fields such as healthcare, finance, and technology. Among the numerous algorithms developed to tackle classification problems, Support Vector Machines (SVM) and Learning Algorithm for Multivariate Data Analysis (LAMDA) are state-of-the-art techniques [2] [3].

On the other hand, the concept of utilizing quantum mechanics to fundamentally enhance the computational efficiency and security of computers has been a pivotal idea at the intersection of physics, computer science, and mathematics for approximately four decades [4]. The versatility offered by replacing the classical bit by a qubit allows for performing an exponential number of operations with respect to the involved qubits, paving the way for a revolutionary approach to classification problem-solving.

Thus, combining classical machine learning techniques like SVM and LAMDA with the principles of quantum computing opens up promising horizons for enhancing the performance of classification algorithms. The construction of hybrid models has the potential to process vast amounts of data at unprecedented speeds, which is particularly beneficial for managing complex, high-dimensional datasets where classical algorithms could face limitations.

1.1 Work objectives

In order to tackle classification problems in a variety of scenarios through the aforementioned algorithms, the following objectives are established for this work.

- **General objective:** show the utility of SVM for binary, multi-class and multi-label classification problems, and of LAMDA for the resolution of binary classification problems, both from classical and quantum standpoints.

- **Specific objectives:**

  - Implement a quantum algorithm for binary, multi-class and multi-label classification SVM and compare it to its classical counterpart, in terms of computational efficiency and scores. Proceed analogously for binary LAMDA classification, in this case proposing a quantum alternative to the classical algorithm.

- Compare the performance and efficiency of SVM and LAMDA for binary classification problems, also contrasting the corresponding quantum implementations.

In accordance with these objectives, this work is structured as follows. Section 2 presents a revision of some fundamental concepts of quantum physics that contextualize the quantum algorithms presented in the following sections. Section 3 is subdivided into an introductory theory subsection, 3.1, and then presents the results obtained for the classification of two binary datasets, a multi-class and a multi-label one in 3.2. Section 4 focuses on classical and quantum approaches for the classification of the aforementioned binary datasets, described in Sections 4.1 and 4.2, respectively. Section 5 compares the performance and efficiency of SVM and LAMDA for binary classification. Finally, in Section 6, the main conclusions of this work are discussed.

2 Fundamentals of Quantum Physics

This chapter aims to present a somewhat concise description of the essential concepts that build the foundations of quantum mechanics and that are of use for contextualizing the ideas behind the quantum algorithms and implementations that will follow in forthcoming sections. Some of the concepts that are included in this summary are not intrinsically essential (this will be indicated in the respective subsections), but pave the way for a deeper understanding of the nature of quantum phenomena. This section is not conceived as a rigorous introduction to quantum mechanics, but as a slightly informal introduction orientated to the sections that this one precedes.

2.1 Preliminaries

Strictly speaking, the only system that exists is the Universe itself. Fortunately, it is almost always possible to work with an isolatable portion of the Universe that can be experimented upon, and whose interaction with the rest is small, controllable, or, in the best case, practically nonexistent. Classically, one can think of a state arbitrarily prepared and, through measurements, determine it with all the necessary precision.

Quantum mechanically, it is more delicate because, after a measurement, the system can change radically. This can be overcome by controlled preparation. If we have N identical systems prepared using the same experimental operations, we can assume that they are all in the same state. When we measure a physical quantity A on this state, we can obtain values A_1, A_2, ..., A_N. If N approaches infinity, we can even calculate a probability distribution (density) function.

This knowledge of the quantity A can be indefinitely improved by reducing the interval of the histogram (experimental resolution) and increasing N. We can consider other quantities B, C, ... and follow the same procedure. If we consider all

possible quantities and it turns out that differently prepared states lead to the same probability distributions, we will say that both states are equal.

2.2 Pure states and measurements

A pure state must be associated with a physical state for which the maximum information is available. In Classical Mechanics, this is straightforward: (r_1, r_2, ..., r_N, p_1, p_2, ..., p_N) defines a pure state with N point particles, where r_i, p_i, respectively, denote the position and momentum of particle i, $i \in \{1, \ldots, N\}$. In Quantum Mechanics, one faces two related problems:

- Certain quantities are not compatible; for example, we cannot know the position and momentum of a particle with unlimited precision, as $\Delta x \cdot \Delta p \geq \frac{\hbar}{2}$. This is known as Heisenberg's indeterminacy principle [5].

- After a measurement, the system is modified. It may even cease to exist. An example of this would be the absorption of a particle.

In this context, it is convenient to highlight two types of measurements.

1. Measurements of the second kind: after determining a quantity, it can change. Even the system as a whole may cease to exist.

2. Measurements of the first kind: allow the preparation of particles in a state with infinite precision.

2.3 Postulates of quantum mechanics

The mathematical formalism that permits a rigorous description of quantum mechanics is described through six postulates, which link quantum mechanics to the observable world [6].

Postulate I: Each quantum system has an associated (separable) Hilbert space, $\mathcal{H}$. At each given time t, a pure state of the quantum system is described by an element, denoted by $|\phi(t)\rangle$, of the Hilbert space $\mathcal{H}$. The norm of $|\phi(t)\rangle$ must be 1 $\forall t$.

Any vector $|\phi\rangle \in \mathcal{H}$ has a dual one $\langle\phi|$, which is a continuous linear functional that acts from $\mathcal{H}$ to $\mathbb{C}$, such that, for any vector $|\psi\rangle \in \mathcal{H}$, the action of $\langle\phi|$ on $|\psi\rangle$ is given by:

$$\langle\phi|\psi\rangle = (|\phi\rangle, |\psi\rangle),$$

where (,) denotes the inner product in Hilbert space $\mathcal{H}$. This is known as the bra-ket notation, and was introduced by Paul Dirac in 1939.

Postulate II: Every physical quantity, let us denote it as $\mathcal{A}$, is described by a linear operator, A, acting on $\mathcal{H}$. A must be self-adjoint and is called an observable. For instance, A could describe the position, momentum, or spin of a particle.

Postulate III: The outcome of the measurement of a physical quantity $\mathcal{A}$ can only be the eigenvalues of A, the self-adjoint operator describing $\mathcal{A}$.

Postulate IV:

a) Case of a non-degenerate eigenvalue. When the physical quantity $\mathcal{A}$ is measured on a quantum state described by ϕ (an element of $\mathcal{H}$), the probability that the outcome of that measurement be a_n, a non-degenerate eigenvalue of A, is given by:

$$|\langle a_n|\phi\rangle|^2,$$

where $|a_n\rangle$ is an eigenvector, with a norm equal to 1 associated to a_n: $A|a_n\rangle = a_n|a_n\rangle$.

b) Case of a degenerate eigenvalue. When the physical quantity $\mathcal{A}$ is measured on a quantum state described by ϕ (an element of $\mathcal{H}$), the probability that the outcome of that measurement be a_n, a degenerate eigenvalue of A, is given by:

$$\sum_{i=1}^{d_n} |\langle a_n, i|\phi\rangle|^2,$$

where $|a_n, i\rangle_{i=1,\ldots,d_n}$ is an orthonormal basis of the proper subspace of a_n, and d_n is the degree of degeneracy of a_n. Indeed,

$$A|a_n, i\rangle = a_n|a_n, i\rangle \quad \forall i \in \{1, \ldots, d_n\};\ \langle a_n, i|a_n, j\rangle = \delta_{ij}$$

and, if $|\psi\rangle$ is such that $A|\psi\rangle = a_n|\psi\rangle$, then $|\psi\rangle = \sum_{i=1}^{d_n} c_n|a_n, i\rangle$.

Postulate V: Reduction or collapse of the wave packet.

a) Case of a non-degenerate eigenvalue. If the outcome of a measurement of the physical quantity $\mathcal{A}$ on the quantum state described by $|\phi\rangle$ is the non-degenerate eigenvalue a_n, then the state of the quantum system immediately after the measurement is described by $|a_n\rangle$.

b) Case of a degenerate eigenvalue. If the outcome of a measurement of the physical quantity $\mathcal{A}$ on the quantum state described by $|\phi\rangle$ is the degenerate

eigenvalue a_n, then, the state of the quantum system immediately after the measurement is described by:

$$|\varphi\rangle = \frac{P(a_n)\,|\phi\rangle}{\sqrt{\langle\phi|P(a_n)|\phi\rangle}},$$

where $P(a_n)$ is the projector onto the proper subspace of a_n , that is,

$$P(a_n) = \sum_{i=1}^{d_n} |a_n, i\rangle\,\langle a_n, i|\,.$$

Postulate VI: If a quantum system is isolated, i.e., it does not interact with other systems, and its quantum state is described by $|\varphi\rangle$ at $t = t_0$, then the state of the system when $t > t_0$ is described by $|\phi(t)\rangle$, this ket being the unique solution to the Schrödinger equation, given by:

$$i\hbar\frac{\partial\,|\phi(t)\rangle}{\partial t} = H(t)\,|\phi(t)\rangle\,,$$

which satisfies the initial condition $|\phi(t = t_0)\rangle = |\varphi\rangle$. $H(t)$ is the quantum Hamiltonian, a self-adjoint operator which describes the energy of the system.

For further details on the mathematical grounds upon which these postulates are based (self-adjoint operators, spectrum, etc), see [6].

2.4 Entanglement and non-locality

This section briefly describes the non-local nature of the quantum phenomena, and introduces the notion of entanglement, which is a phenomenon that has no classical counterpart and that is of substantial importance in quantum computing. Finally, it addresses decoherence. This subsection is not essential to follow the forthcoming ones, although entanglement is a notion that appears assiduously, and decoherence is one of the major setbacks for constructing a quantum computer, which tackles the challenge of maintaining coherence.

2.4.1 EPR Paradox

The well-known EPR paradox is a thought experiment proposed by Einstein, Podolsky, and Rosen in 1935 [7], which questions whether quantum mechanics, through the wave function, provides a complete description of reality. The idea is as follows. Two particles, generically labeled as 1 and 2, with position and momentum operators X_1, P_1 for particle 1, and X_2, P_2 for particle 2, are considered. Due to the non-commutation of position and momentum operators for each particle, it is assumed that they are in an eigenstate of $X_1 - X_2$ and $P_1 + P_2$, which are

operators that do commute. Then, a measurement of the position of particle 1 is performed. The result should then determine the position of particle 2. However, if both particles are sufficiently separated, the principle of locality would imply that the measurement of the position of 1 should not affect particle 2. Nevertheless, this experiment shows that, regardless of the separation between 1 and 2, we could know with certainty the position of particle 2, even if X_2 does not commute with $P_1 + P_2$. Thus, it is inferred that quantum mechanics cannot be complete if only the information provided by the wave function is taken into account. The position of particle 2 must have been determined before the measurement on 1. This led to the conclusion that, beyond the wave function, there are other variables that would determine the outcome of a measurement, the so-called hidden variables.

In 1951, David Bohm proposed another thought experiment to emphasize this strange behavior, this time considering measurements on the spin of two particles instead of position and momentum [8, 9]. Again, the conclusion reached was the same: if locality is assumed, then physical states must have more information than described by quantum mechanics, and experimental results could be predicted if that hidden information were available. Alternatively, one could consider the possibility that quantum mechanics is non-local. This sparked a highly controversial debate over the next three decades, leading to the formulation of various hidden variable theories.

2.4.2 Bell Inequalities

A decade later, John Bell revisited Bohm's experiment, considering different spin directions for particles 1 and 2 [10]. In 1964, he concluded that any theory of hidden variables would lead to predictions different from those of quantum mechanics. To understand Bell's reasoning, it is illustrative to consider, for example, two spin-1/2 fermions described by the singlet state, $|\psi\rangle = (|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle)/\sqrt{2}$, allowing us to work with the spin operator, $\hat{\sigma} = (2/\hbar)\hat{\mathbf{S}}$. Two measurement directions are considered, defined by the unit vectors $\mathbf{a}$ and $\mathbf{b}$. The result of these measurements will be $\hat{\sigma}_a = \hat{\boldsymbol{\sigma}} \cdot \mathbf{a}$ and $\hat{\sigma}_b = \hat{\boldsymbol{\sigma}} \cdot \mathbf{b}$, and the correlation between these measurements is given by:

$$C(\mathbf{a}, \mathbf{b}) = \langle\psi| \, \hat{\sigma}_a \otimes \hat{\sigma}_b \, |\psi\rangle = -\mathbf{a} \cdot \mathbf{b}. \tag{1}$$

Now, consider a later version (1971) of Bell's calculation for a two-particle system where we measure observables $\hat{A}$ and $\hat{B}$ with values $A = \pm 1$, $B = \pm 1$. Let $\mathbf{a}$ and $\mathbf{b}$ be the parameters on which the first and second detector depend, respectively, and let λ be the hidden variable(s) determining the experimental outcome. It is assumed that λ is distributed according to a probability distribution characterized by a density $\hat{\rho}(\lambda)$ such that $\int d\hat{\rho}(\lambda) = 1$ and $\hat{\rho}(\lambda) \geq 0$.

The result of the measurement on the first particle depends only on $\mathbf{a}$ and λ, so $A \equiv A(\mathbf{a}, \lambda)$. Similarly, $B \equiv B(\mathbf{b}, \lambda)$. Thus, the correlation is given by:

$$C(\mathbf{a}, \mathbf{b}) = \int d\lambda \hat{\rho}(\lambda) A(\mathbf{a}, \lambda) B(\mathbf{b}, \lambda). \tag{2}$$

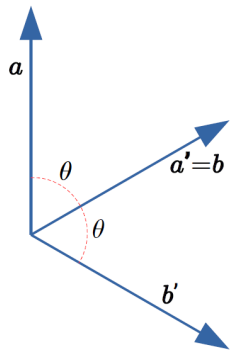


Fig. 1. Coplanar diagram of the unit vectors used to illustrate the violation of the Bell inequality in the case of spin singlet.

If we now consider two possible settings for each detector, $(\mathbf{a}, \mathbf{a}')$ and $(\mathbf{b}, \mathbf{b}')$, operating leads to the famous Bell inequality:

$$|C(\mathbf{a}, \mathbf{b}) - C(\mathbf{a}, \mathbf{b}')| + |C(\mathbf{a}', \mathbf{b}') - C(\mathbf{a}', \mathbf{b})| \leq 2. \tag{3}$$

If quantum mechanics had exclusively local nature, this inequality should be satisfied. However, it turns out that settings can be chosen such that the inequality is violated, confirming that quantum mechanics is a non-local theory. For example, in the case of the spin singlet, coplanar vectors $\mathbf{b} = \mathbf{a}'$ can be chosen, forming angles as shown in the schematic in Fig.1. Thus, by recovering the expression (3) for the quantum mechanics prediction, it follows that $C(\mathbf{a}, \mathbf{b}) = -\cos\theta$, $C(\mathbf{a}, \mathbf{b}') = -\cos 2\theta$, $C(\mathbf{a}', \mathbf{b}') = -\cos\theta$, and $C(\mathbf{a}', \mathbf{b}) = -1$. It can then be verified that the Bell inequality becomes:

$$h(\theta) \equiv |\cos\theta - \cos 2\theta| + |\cos\theta + 1| \leq 2. \tag{4}$$

The inequality is not satisfied for $\theta \in (0, \pi/2) \cup (3\pi/2, 2\pi)$, as illustrated in Fig.2.

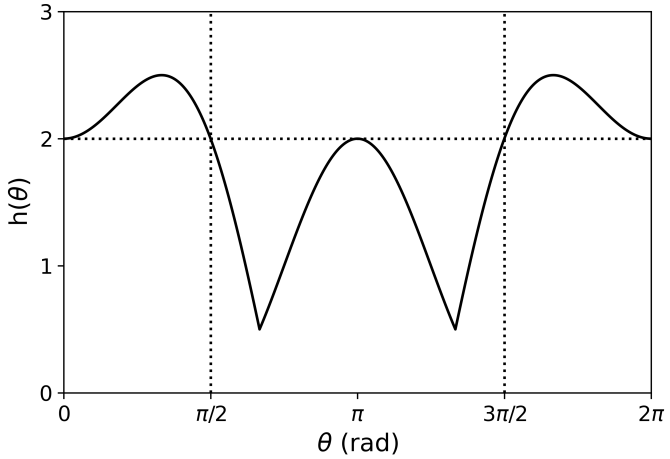


Fig. 2. Graphical representation corresponding to the violation of the Bell inequality (4).

After Bell's discovery, other inequalities were found, such as the CHSH [11], proposed by John Clauser, Michael Horne, Abner Shimony, and Richard Holt in the late 1960s. These are more favorable for experimental testing and overcome some probabilistic objections. In any case, these inequalities show that local hidden variable theories are ruled out, and quantum mechanics is consistent with all experimental results to date.

2.4.3 Entanglement

Entanglement is a phenomenon without a classical counterpart, which forces us to describe all components of a system using a single wave function, without the possibility of factorization into separate wave functions for each object. This is the key to the quantum correlation between particle states, regardless of how far apart they are (non-locality), and is at the heart of the EPR paradox discussed earlier. For example, consider:

$$|\psi\rangle = (\alpha_0 |\uparrow\rangle + \alpha_1 |\downarrow\rangle)(\beta_0 |\uparrow\rangle + \beta_1 |\downarrow\rangle) = \alpha_0\beta_0 |\uparrow\uparrow\rangle + \alpha_0\beta_1 |\uparrow\downarrow\rangle + \alpha_1\beta_0 |\downarrow\uparrow\rangle + \alpha_1\beta_1 |\downarrow\downarrow\rangle . \tag{5}$$

We have a state constructed using a product, which is therefore not entangled. On the other hand, if we consider a state of the form:

$$|\phi\rangle = \gamma_0 |\uparrow\uparrow\rangle + \gamma_1 |\downarrow\downarrow\rangle , \tag{6}$$

it would be an entangled state because if we assume that it comes from factorization and equate the coefficients with those of (1), then we arrive at an incompatible system.

A very relevant example of entangled states is the set of states known as the Bell Basis. This is a set of four normalized and entangled states that form a basis for a system with two two-level states (qubits). They are as follows:

$$\Psi_{12}^{(\pm)} = \sqrt{\frac{1}{2}}(|\uparrow\downarrow\rangle \pm |\downarrow\uparrow\rangle); \quad \Phi_{12}^{(\pm)} = \sqrt{\frac{1}{2}}(|\uparrow\uparrow\rangle \pm |\downarrow\downarrow\rangle). \tag{7}$$

These states are maximally entangled, as a measurement on the second system, in the basis $|\uparrow\rangle_2$, $|\downarrow\rangle_2$, completely determines the state of the first system. They are widely used in quantum information theory and quantum teleportation protocols, so they will be referred to frequently.

A tool that allows the identification of entangled states is the so-called *entanglement witnesses*. These are hermitian operators that define hyperplanes in the space of density operators, establishing a cut between some separable and all entangled states. These are observables $\hat{W}$ that satisfy $\mathrm{Tr}(\hat{W}\hat{\rho}_{sep}) \geq 0$ for separable states and $\mathrm{Tr}(\hat{W}\hat{\rho}) < 0$ for entangled ones. The closer the cut is to the set of separable states, the fewer entangled states share the hyperplane with separable ones, and therefore, the better the "witness" is.

Their usefulness lies in checking whether a given state operator represents an entangled state, which is guaranteed if $\mathrm{Tr}(\hat{W}\hat{\rho}) < 0$. Experimentally, measuring

the expected value of an observable becomes a practical approach, making them an effective tool.

2.4.4 Decoherence

Decoherence is a fundamental concept when discussing isolated systems and is the reason why quantum effects do not manifest at macroscopic scales in the classical world. While in a classical system, interaction with the environment is merely a disturbance to be mitigated in order to properly describe the physics, in a quantum system, coupling with the environment defines the observable physical properties of the system. There are two noteworthy consequences of the interaction between a quantum system and its environment [12].

- The irreversible disappearance of coherence, which is the source of phenomena like interference.

- The dynamic "definition" of the system's observables. This is often referred to as superselection induced by the environment, analogous to superselection rules that restrict allowed superpositions, thereby prohibiting the existence of certain observables. Thus, it explains that, ultimately, in the classical world, only certain more robust quantities like position or momentum can be observed.

A somewhat rudimentary but illustrative example is as follows: if one attempts to prepare a macroscopic object in a non-classical state, a coherent superposition of two clearly separated positions, incident light, thermal radiation, or even microwave background radiation would quickly lead to the decoherence of such a state.

Now, consider an entangled state of the form $|\Psi\rangle = \frac{1}{\sqrt{2}}(|\psi_1\rangle_1 |\phi_1\rangle_2 \pm |\psi_2\rangle_1 |\phi_2\rangle_2)$, where $|\psi_i\rangle_1$, $|\phi_i\rangle_2$, $i = 1, 2$, are not necessarily mutually orthogonal. This condition differs from the Bell states (7), and now entanglement is not necessarily maximal. The greater the overlap between $|\phi_1\rangle_2$ and $|\phi_2\rangle_2$, the more challenging it is to distinguish between these states through a projective measurement on system 2. This, in turn, complicates discerning whether system 1 is in the state $|\psi_1\rangle_1$ or $|\psi_2\rangle_1$. In the extreme case where $|\phi_1\rangle_2 = |\phi_2\rangle_2$, the state $|\Psi\rangle$ is factorizable, and the systems lack correlation.

If we now identify system 2 with the environment, it opens a path to describe decoherence. The information available about the system based on measurements of the environment increases with entanglement, and thus, with the distinguishability between the states of the environment. The more information there is, the less individuality the system retains. Thus, the coherence initially "localized" in the system becomes "shared" between the system and the environment, forming a composite system that can no longer be observed at the system level, leading to decoherence.

2.5 Density Operators

This subsection briefly presents the notion of density operators, which describe quantum systems in a complete way. It has been included for the sake of completeness but it is not necessary to follow the forthcoming sections.

The density operator or matrix provides a complete description of the knowledge about a quantum system. It incorporates classical uncertainty associated with each state $|\psi_i\rangle$ the system can be in, with probability p_i. It is defined as:

$$\hat{\rho} \equiv \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{8}$$

This expression results from expanding the one for the expected value of an operator $\hat{A}$, $\langle\hat{A}\rangle = \sum_i p_i \langle\psi_i| \hat{A} |\psi_i\rangle = \mathrm{Tr}(\hat{\rho}\hat{A})$. It is useful to recall that it is a self-adjoint and positive semidefinite operator, satisfying $\mathrm{Tr}(\hat{\rho}) = 1$ and also $\hat{\rho} \geq \hat{\rho}^2$ (i.e., $\langle\varphi| \hat{\rho} |\varphi\rangle \geq \langle\varphi| \hat{\rho}^2 |\varphi\rangle \quad \forall |\varphi\rangle$). This last property allows distinguishing pure states, where 8 reduces to $\hat{\rho} = |\psi\rangle \langle\psi|$, from mixed states. In the former case, the density operator satisfies the idempotence property $\hat{\rho} = \hat{\rho}^2$, which, in turn, implies $\mathrm{Tr}(\hat{\rho}^2) = 1$. However, for mixed states, $\hat{\rho} \neq \hat{\rho}^2$, leading to $\mathrm{Tr}(\hat{\rho}^2) < 1$.

Note that expression (8) provides information about the specific weight (p_i) of each state $|\psi_i\rangle$, but it cannot distinguish between different bases in which a state may be prepared. For example, if we consider the density operator:

$$\hat{\rho} = \frac{3}{4} |0\rangle \langle 0| + \frac{1}{4} |1\rangle \langle 1| , \tag{9}$$

we see that the system is in the state $|0\rangle$ with probability $3/4$ and in $|1\rangle$ with probability $1/4$. However, we can also consider that the system is prepared on the basis:

$$|a\rangle = \sqrt{\frac{3}{4}} |0\rangle + \sqrt{\frac{1}{4}} |1\rangle ; \quad |b\rangle = \sqrt{\frac{3}{4}} |0\rangle - \sqrt{\frac{1}{4}} |1\rangle , \tag{10}$$

with a $1/2$ probability in each of these states. Applying the definition (8), we find that the same matrix (9) is obtained. Through this simple example, we see that there is no unique set of privileged states.

Now, consider a composite system AB, described by $\hat{\rho}^{AB}$. Let $\hat{M}$ be an observable measured on A, and $\tilde{M}$ be the corresponding observable for the same measurement on the composite system. Then, $\tilde{M} = \hat{M} \otimes \mathbb{1}_B$, and averages over these observables must be consistent, implying that the density operators must satisfy:

$$\mathrm{Tr}(\hat{M}\hat{\rho}^A) = \mathrm{Tr}((\hat{M} \otimes \mathbb{1}_B)\hat{\rho}^{AB}). \tag{11}$$

Mathematically, there is a unique solution, given by $\hat{\rho}^A = Tr_B(\hat{\rho}^{AB})$, where $Tr_B(\hat{\rho}^{AB})$ is the partial trace, defined as:

$$Tr_B(|a_1\rangle \langle a_2| \otimes |b_1\rangle \langle b_2|) \equiv |a_1\rangle \langle a_2| \, \mathrm{Tr}(|b_1\rangle \langle b_2|), \tag{12}$$

where $|a_1\rangle$, $|a_2\rangle$ are arbitrary states of A and $|b_1\rangle$, $|b_2\rangle$ are states of B. Also, $\mathrm{Tr}(|b_1\rangle\langle b_2| = \langle b_2|b_1\rangle$, and the extension of (12) to an arbitrary operator of AB follows from imposing linearity on the arguments of Tr_B. Of course, the definition is entirely analogous for Tr_A. It is easy to verify that, for a product state $\hat{\rho}\otimes\hat{\sigma}$, the expected result is obtained: $\hat{\rho}^A = Tr_B(\hat{\rho}\otimes\hat{\sigma}) = \hat{\rho}\mathrm{Tr}(\hat{\sigma}) = \hat{\rho}$, and symmetrically $\hat{\rho}^B = \hat{\sigma}$. Of course, the interest in the above lies in the description of entangled states. Consider, for example, the Bell state $\Phi_{12}^{(+)} \equiv \sqrt{\frac{1}{2}}(|00\rangle + |11\rangle)$ defined in (7), its density matrix is:

$$\hat{\rho} = |\Psi_{12}^{(+)}\rangle\langle\Psi_{12}^{(+)}| = \frac{|00\rangle\langle 00| + |11\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 11|}{2}. \tag{13}$$

The reduced density operator for the first qubit is:

$$\hat{\rho}^1 = Tr_2(\hat{\rho}) = \frac{\mathrm{Tr}_2(|00\rangle\langle 00|) + Tr_2(|11\rangle\langle 00|) + Tr_2(|00\rangle\langle 11|) + Tr_2(|11\rangle\langle 11|)}{2} \tag{14}$$

$$= \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} = \frac{1}{2}. \tag{15}$$

2.6 Quantum Information Encoding

This is a relevant section, which introduces the quantum encoding protocols that will be implemented in the next sections.

2.6.1 Qubits and Bloch Representation

A qubit is simply a superposition of two generic quantum states in a two-level system, which can represent, for example, two spin states. We can write the states using the so-called Hopf coordinates,

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{16}$$

where $\varphi \in [0, 2\pi)$, $\theta \in [0, \pi]$, ultimately coinciding with a point on the unit sphere described in spherical coordinates $(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta) := \mathbf{a}$. This representation can be mapped onto a unit-radius sphere, the Bloch sphere (see Fig. 3). The relative phase φ encodes information that allows us to determine the results of interference measurements between states $|0\rangle$ and $|1\rangle$.

However, not only the surface has physical meaning. If we consider a point in the interior, it cannot represent a pure state, as it is not normalized, but it does represent mixed states. Indeed, first consider the density matrix of a pure state, in the $|0\rangle \to (1,0)^t, |1\rangle \to (0,1)^t$ representation. It results in:

$$\hat{\rho}_{|\psi\rangle} = |\psi\rangle\langle\psi| = \begin{pmatrix}\cos\frac{\theta}{2}\\ e^{i\varphi}\sin\frac{\theta}{2}\end{pmatrix}\cdot(\cos\frac{\theta}{2} \quad e^{-i\varphi}\sin\frac{\theta}{2}) = \frac{1}{2}\begin{pmatrix}1+\cos\theta & e^{-i\varphi}\sin\theta\\ e^{i\varphi}\sin\theta & 1-\cos\theta\end{pmatrix} = \frac{1}{2}(\mathbb{1}+\mathbf{a}\cdot\hat{\boldsymbol{\sigma}}) \tag{17}$$

Generalizing this for another vector $\mathbf{b} = (b_x, b_y, b_z)$, with $\|\mathbf{b}\| \leq 1$, the matrix is a density matrix that necessarily corresponds to a mixed state when $\|\mathbf{b}\| < 1$:

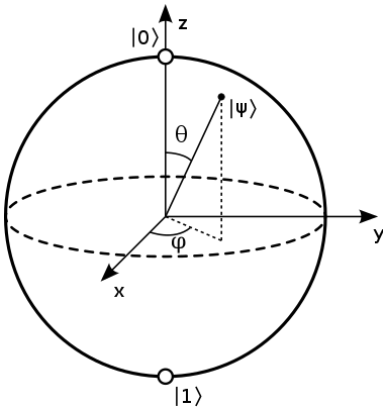


Fig. 3. Representation on the Bloch sphere of a generic qubit described by the expression (16), where the poles denote well-defined orthogonal states in the considered physical quantity (spin, polarization, etc.).

$$\hat{\rho}(\mathbf{b}) = \frac{1}{2}(\mathbb{1} + \mathbf{b} \cdot \hat{\boldsymbol{\sigma}}) \tag{18}$$

It is easy to verify that $\hat{\rho} = \hat{\rho}^{\dagger}$ and that $\mathrm{Tr}(\hat{\rho}) = 1$, and the condition $\hat{\rho} \geq 0$ follows from calculating the eigenvalues, which are $\lambda_{\pm} = 1 \pm \|\mathbf{b}\|/2 \geq 0$.

2.6.2 Qumodes

For the sake of completeness, another way of encoding information is worth mentioning: the so-called qumodes or quantum harmonic oscillators. In this case, the Hilbert space is of infinite dimension, and observables have a discrete, countable, or continuous spectrum. An example of the latter could be the amplitude or phase of the oscillator.

A pure state in position basis is written in the usual form $|\psi\rangle = \int dx\psi(x)\,|x\rangle$, and an arbitrary mixed state is given by:

$$\hat{\rho} = \int dsdtf(s,t)X(s)Z(t). \tag{19}$$

where $f(s,t)$ is a certain complex function, and $X(s) = e^{-2is\hat{p}}$, $Z(t) = e^{+2it\hat{x}}$ are the so-called Weyl-Heisenberg operators ($\hat{x}$ and $\hat{p}$ are the usual position and momentum operators of a harmonic oscillator).

Bell states for a system of two qumodes, which are maximally entangled, are given [13] by:

$$|\Psi(u,v)\rangle = \frac{1}{\sqrt{\pi}} \int dx e^{2ixv}\,|x\rangle\,|x-u\rangle\,. \tag{20}$$

2.6.3 Quantum Logic Gates

These are unitary operators that allow the manipulation of qubits in quantum circuits. The most used ones [14] are:

- **Pauli Gates (X, Y, Z).** These are Pauli matrices, and they apply a rotation of π radians around the corresponding axis on the Bloch sphere. The X gate (also known as NOT) exchanges $|0\rangle$ and $|1\rangle$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{21}$$

  The X gate (also known as NOT) exchanges $|0\rangle$ and $|1\rangle$. The generalized rotations of any angle θ around any axis (x, y, z respectively) of the Bloch sphere are given by:

$$R_X(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad R_Y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad R_Z = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \tag{22}$$

- **Hadamard Gate (H).** It maps eigenstates of S_z to those of S_x, $|0\rangle \to (|0\rangle + |1\rangle)/\sqrt{2}$, $|1\rangle \to (|0\rangle - |1\rangle)/\sqrt{2}$. Matrix form is:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{23}$$

  It is useful in quantum computers, which do not distinguish between S_x eigenstates, so this hidden information can be recovered thanks to this gate.

- **Control Gates.** They act on at least two qubits, one of which serves as control. If $U = \{u_{ij}\}_{i,j=0,1}$ is a gate on the second qubit, then it is called a CU gate, which can be CNOT, CY, or CZ, if U is one of the Pauli operators.

$$CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix} \longrightarrow \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{24}$$

  Note that the gate U acts on the second qubit only if the first (control) is $|1\rangle$.

- **SWAP Gate:** Swaps two qubits. If we swap $|10\rangle$ and $|01\rangle$ in the $|00\rangle, |10\rangle, |01\rangle, |11\rangle$ basis,

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{25}$$

Similarly, qumodes can be manipulated using their own gates. Among those for a single mode, we find [13]:

- **Weyl-Heisenberg (WH) Displacement Gate:**

$$Z(s)\,|p\rangle = |p+s\rangle\,, \quad Z(s)\,|x\rangle = e^{2isx}\,|x\rangle\,. \tag{26}$$

- **General Phase Gate:**

$$D = \exp[i\hat{f}(x)]. \tag{27}$$

- **Fourier Gate**, analogous to Hadamard for qumodes:

$$F\left|x\right\rangle_{pos} = \int d\gamma e^{2ix\gamma}\left|\gamma\right\rangle_{pos} = \left|x\right\rangle_{mom}. \tag{28}$$

Regarding two-mode gates, it is worth mentioning the C_Z gate $= \exp(2i\hat{x}\otimes\hat{x})$.

$$C_Z\left|x\right\rangle_{pos}\left|p\right\rangle_{mom} = \left|x\right\rangle_{pos}\left|p+x\right\rangle_{mom}. \tag{29}$$

2.7 Quantum Error Correction

This section has been included for the sake of completeness, since quantum error correction is a potential tool for tackling the loss of information during transmission.

The practical implementation of information transmission protocols must address the presence of noise. Quantum error correction techniques focus precisely on protecting and preserving this information to increase the fidelity of the transmission. The central idea is to encode the states in a way that is resistant to noise, and then decode them when it is appropriate to recover the original state. It is assumed that both encoding and decoding can be done without error, presenting a suitable scheme for transmitting quantum states through a noisy channel, but with nearly noise-free quantum computers at both ends of the channel.

In classical cases, a simple idea for encoding a bit is to replace it with several copies, for example, three copies of itself. Assuming that the probability p of noise swapping bit 0 and 1, and vice versa, is low (specifically if $p < 1/2$), a more reliable decoding would involve considering the bit that the majority of copies agree on, in this case, two or three. This is known as *majority decoding*. Another quite natural idea is to send the message repeatedly. In any case, the underlying notion is to add enough redundancy to the message to recover it after the effects of noise. A priori, translating this to the quantum context faces several challenges:

- The no-cloning theorem prevents duplicating quantum information, and even if it were possible, measuring and comparing the three states at the output of the channel would be unfeasible.

- Errors on a qubit are continuous, and determining exactly which errors have occurred for correction requires precision beyond reach.

- Measurements destroy information, making it impossible to observe the channel output similarly to how it was done in the classical context.

However, it is possible to overcome these difficulties. Consider a simplified model corresponding to a channel that leaves the qubits intact with probability $1-p$ and

flips them (i.e., applies a Pauli X gate) with probability p. This type of channel is known as a *bit flip channel*. Suppose we start with an initial state $a|0\rangle + b|1\rangle$ and encode it as:

$$|0\rangle \longrightarrow |000\rangle \qquad |1\rangle \longrightarrow |111\rangle \tag{30}$$

resulting in the encoded state $a|000\rangle + b|111\rangle$. This can be done using two consecutive CNOT gates, which will perform a flip on only one of the terms in the initial product state resulting from adding each qubit. Each of the three qubits is passed through an independent copy of the channel. Consider, analogous to the classical case, that one or none of the qubits are flipped, so the majority remain intact. Then, apply the following error correction procedure:

1. **Error Detection:** A measurement is made to determine if an error has occurred and, if so, which one. There are four possibilities or *syndromes*, given by the operators:

$$\begin{aligned} P_0 &\equiv |000\rangle\langle 000| + |111\rangle\langle 111| \\ P_1 &\equiv |100\rangle\langle 100| + |011\rangle\langle 011| \\ P_2 &\equiv |010\rangle\langle 010| + |101\rangle\langle 101| \\ P_3 &\equiv |001\rangle\langle 001| + |110\rangle\langle 110| \end{aligned}$$

   These correspond to the cases where no error occurs or the first, second, or third qubit is flipped, respectively. If, for example (the other cases are analogous), the second qubit was flipped, then the resulting state would be $|\psi\rangle \equiv a|010\rangle + b|101\rangle$, and it would be the case that $\langle\psi| P_i |\psi\rangle = \delta_{i2}$, with δ_{i2} being the Kronecker delta. The *syndrome* only provides information about which error occurred and leaves the state intact.

2. **State Recovery:** It is sufficient to apply the corresponding quantum gate to reverse the effect of noise. If the *syndrome* is 0, there is nothing to do, and if it is 1, 2, or 3, the respective qubit must be flipped, recovering the state with perfect precision.

It is worth noting the initial assumption of the maximum inversion of a qubit, which occurs with a probability of $(1-p)^3 + 3p(1-p)^2 = 1 - (3p^2 - 2p^3)$, obtained immediately using the binomial distribution. Like in the classical case, the probability of not correcting the error is $3p^2 - 2p^3$, so the encoding and decoding process again improves results when $p < 1/2$.

Despite being illustrative, the previous case is a simplification of a much more complex entity. Errors can be different and even have different consequences in different states. In the previous example, a state of the form $|\varphi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ would have been transmitted without any perception of error, being invariant under X.

A deeper analysis would quantify the fidelity of state transmission in each encoding protocol, which in simple models like the previous one would depend on p.

In this sense, a more sophisticated method, allowing the preservation of a qubit against errors generated by bit or phase flips, is the so-called *Shor Code* (see [15] for a detailed description).

2.8 Introduction to Qiskit

This section presents a simple chunk of code that implements a simple quantum circuit, employing library `Qiskit` from Python, which is the one that will be used for quantum simulations throughout this work. It is an accessory section that aims to present a simple example to contextualize the concepts discussed in the preceding subsections.

The code displayed at the end of this subsection implements a simple 2-qubit quantum circuit, divided in the following steps.

1. Create the two qubits, as well as the corresponding classical bits, each of which will collect the measurement from one of the qubits. Note that the qubit corresponds to a 2-dimensional system, but the measurement will collapse its state onto either $|0\rangle$ or $|1\rangle$, so the correspondence between qubits and bits is one-to-one.

2. Initialize the circuit. Originally, both qubits are in state $|0\rangle$. Then, a Hadamard gate is applied to the first qubit, followed by a CNOT gate that affects the whole system. Once both gates have been applied, Qiskit allows for returning the unitary operator, which describes the concatenation of the previous ones. Let us describe in detail how it would be computed manually and compare the theoretical and Qiskit results.

   The operator describing a Hadamard gate for the first qubit is represented by the tensor product of the Hadamard operator H in the first subspace times the identity operator $\mathbb{1}$ in the second one.

$$\begin{aligned} H \otimes \mathbb{1} &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & -1 \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \end{aligned} \tag{31}$$

   The CNOT gate is given by (24), and thus the unitary operator defining the circuit is given by:

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}. \tag{32}$$

The operator returned by Qiskit reads:

$$\begin{pmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0 & 0.7071 & 0.7071 \\ 0.7071 & -0.7071 & 0 & 0 \end{pmatrix}$$

.

The mismatch is fixed by swapping the second and third row and column. This is due to the fact that Qiskit orders qubits in the inverse way as it is usually done in theory. Thus, qubit q in a circuit with n qubits is accessed through the index $n-q-1$. In the previous example, this swaps the rows and columns where the first and second qubit do not coincide, that is, the second and third row (corresponding to $|01\rangle$ and $|10\rangle$ in the usual notation) and the second and third column ($\langle 01|$ and $\langle 10|$ in the usual notation).

3. Implement a measurement of each of the qubits separately. Given that a single measurement provides no useful information to perform statistics, a simulator is employed so that the previous measurements can be carried out a significant number of times, in this case, 100. Approximately one-half of the counts corresponds to $|00\rangle$ while the remaining half corresponds to $|11\rangle$. This was the result one could predict from theory, given that initially, the system was in state $|00\rangle$, whose associated vector is $(1,0,0,0)^t$. Thus, the resulting state, prior to the measurements, is given by the first column of the unitary operator, that is, $|\Psi_{final}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \Phi_{12}^{(+)}$. The final state is one of the Bell states. Once we have measured it 100 times, we obtain a count dictionary (which given the random nature of the quantum phenomena, may vary in each execution), which in this case reads $'00' : 49, '11' : 51$. As one could have expected from the theoretical considerations, approximately half of the observations correspond to $|00\rangle$, while the other (approximate) half correspond to $|11\rangle$. This accounts for the need to reproduce an experiment a certain amount of times in order to reach valid conclusions.

```
import qiskit as qk
# Creating Qubits
q = qk.QuantumRegister(2)
# Creating Classical Bits
c = qk.ClassicalRegister(2)
# Define and print an empty circuit.
# So far we only have an empty quantum circuit with 2 qubits (q0_0 and q0_1)
# and 2 classical registers (c0_0 and c0_1).
```

```
circuit = qk.QuantumCircuit(q, c)
print(circuit)
# Initialize empty circuit
circuit = qk.QuantumCircuit(q, c)
# Hadamard Gate on the first Qubit
circuit.h(q[0])
# CNOT Gate on the first and second Qubits
circuit.cx(q[0], q[1])
from qiskit.providers.aer import AerSimulator
# We copy the circuit because otherwise, the unitary operator does not allow
# for later measurement simulation.
circuit_copy = circuit.copy()
sim_u = AerSimulator(method = 'unitary')
circuit_copy.save_unitary()
result = qk.execute(circuit_copy, sim_u).result()
U = result.get_unitary(decimals = 4)
print(U)
# Measuring the Qubits.
circuit.measure(q, c)
print (circuit)
# We run the circuit on the quantum simulator.
# Using Qiskit Aer's Qasm Simulator: define where we want to execute the
# simulation.
simulator = qk.BasicAer.get_backend('qasm_simulator')
# Simulating the circuit using the simulator to obtain the result.
job = qk.execute(circuit, simulator, shots=100)
result = job.result()
# We obtain the aggregated binary results from the circuit.
counts = result.get_counts(circuit)
print (counts)
```

3 SVM

3.1 Theory revision

Support vector machines (SVMs) are supervised learning maximum-margin models with associated learning algorithms that analyze data for classification and regression analysis, which were developed in the last decade of the 20th century by Cortes and Vapnik [16], [17]. Let us revise the theoretical background behind SVM [18].

3.1.1 Linearly Separable Binary Classification

Suppose we have L training points, where each input $\mathbf{x}^i$ has D attributes and belongs to one of two classes $y_i = -1$ or $y_i = 1$, i.e the training data is of the form:

$$\{\mathbf{x}^i, y_i\} \quad \mathbf{x}^i \in \mathbb{R}^D, \quad y_i \in \{-1, 1\}, \quad \forall i = 1, \ldots, L.$$

We shall denote the components of $\mathbf{x}^i$ as $x_1^i, \ldots, x_D^i$, or simply $x_1, \ldots, x_D$ when the index i of the instance is irrelevant. Let us assume that data is linearly separable, so that a hyperplane in $\mathbb{R}^D$ can be drawn which separates the instances of the two classes. This hyperplane can be described by $\mathbf{w} \cdot x + b = 0 \;\; (\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R})$, where $\mathbf{w}$ is normal to the hyperplane and $|b|/||\mathbf{w}||$ is the distance from the hyperplane to the origin.

Support vectors are the samples closest to the separating hyperplane and the aim of Support Vector Machines (SVM) is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes. Then, the aim is to select the variables $\mathbf{w}$ and b such that:

$$\begin{array}{l} \mathbf{x}^i \cdot \mathbf{w} + b \geq +1 \text{ for } y_i = +1 \\ \mathbf{x}^i \cdot \mathbf{w} + b \leq -1 \text{ for } y_i = -1 \end{array} \quad \Longleftrightarrow \quad y_i(\mathbf{x}^i \cdot \mathbf{w} + b) - 1 \geq 0, \quad \forall i \in \{1, \ldots, L\}. \tag{33}$$

The support vectors lie then on the two planes $H_\pm \equiv \mathbf{x}^i \cdot \mathbf{w} + b = \pm 1$, the distance between them being $1/||\mathbf{w}||$. Since the aim is precisely to maximize this margin, one needs to minimize $||w||$, or equivalently minimize $\frac{1}{2}||\mathbf{w}||^2$. The Quadratic Programming (QP) optimization problem then reads:

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t} \quad y_i(\mathbf{x}^i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \in \{1, \ldots, L\}. \tag{34}$$

Problem (34) can be solved using Lagrange multipliers $\boldsymbol{\alpha}$, $\alpha_i \geq 0 \, \forall i \in \{1, \ldots, L\}$. One arrives then at the function:

$$L_P \equiv \frac{1}{2}||\mathbf{w}||^2 - \boldsymbol{\alpha}[y_i(\mathbf{x}^i \cdot \mathbf{w} + b) - 1] = \\ = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{L} \alpha_i y_i (\mathbf{x}^i \cdot \mathbf{w} + b) + \sum_{i=1}^{L} \alpha_i. \tag{35}$$

In order to find the $\mathbf{w}$ and b which minimize, and $\boldsymbol{\alpha}$ which maximizes (35) (whilst keeping $\alpha_i \geq 0 \ \forall i$), one differentiates L_P with respect to $\mathbf{w}$ and b and sets the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} = \sum_{i=1}^{L} \alpha_i y_i \mathbf{x}^i \tag{36}$$

$$\frac{\partial L_P}{\partial b} = 0 \longrightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{37}$$

Substituting (36) and (37) into (35) one ends up in what is called the dual form of L_P, which one needs to maximize:

$$\text{Maximize } L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \quad \text{s.t} \quad \alpha_i \geq 0, \quad \forall i \in \{1, \ldots, L\}, \ \sum_{i=1}^{L} \alpha_i y_i = 0, \tag{38}$$

where $H_{ij} = y_i y_j \mathbf{x}^i \cdot \mathbf{x}^j$. One can obtain $\boldsymbol{\alpha}$ for (38) with a QP solver, and then use (36) to retrieve $\mathbf{w}$. Regarding b, any data point satisfying (37) that is a support vector $\mathbf{x}^s$ will have the form $y_s(\mathbf{x}^s \cdot \mathbf{w} + b) = 1$. Substituting in (36) and multiplying both sides by y_s, recalling that $y_s^2 = 1$, it follows that:

$$b = y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}^m \cdot \mathbf{x}^s, \tag{39}$$

where S denotes the set of indices of the support vectors, i.e., it corresponds to the indices $i \in \{1, \ldots, L\}$ such that $\alpha_i > 0$. However, instead of taking an arbitrary support vector $\mathbf{x}^s$, it is more appropriate to take an average over all support vectors in S. Then,

$$b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}^m \cdot \mathbf{x}^s \right). \tag{40}$$

3.1.2 Binary Classification for Data that is not Fully Linearly Separable

In order to extend the SVM methodology to deal with data that is not fully linearly separable, we relax the constraints (33) slightly to allow for misclassified points. This is done by introducing a positive slack variable $\xi_i, i \in \{1, \ldots, L\}$.

$$\begin{array}{l} \mathbf{x}^i \cdot \mathbf{w} + b \geq +1 - \xi_i \text{ for } y_i = +1 \\ \mathbf{x}^i \cdot \mathbf{w} + b \leq -1 + \xi_i \text{ for } y_i = -1 \end{array} \quad \Longleftrightarrow \quad y_i(\mathbf{x}^i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0 \quad \forall i \in \{1, \ldots, L\}, \tag{41}$$

where $\xi_i \geq 0 \ \forall i \in \{1, \ldots, L\}$. Then, a suitable adaptation of (34) reads:

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{L}\xi_i \quad \text{s.t} \quad y_i(\mathbf{x}^i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0 \quad \forall i \in \{1, \ldots, L\}, \tag{42}$$

where the constant C controls the trade-off between the slack variable penalty and the size of the margin. The next step is then to reformulate (35), which can be done as follows:

$$L_P = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\alpha_i[y_i(\mathbf{x}^i \cdot \mathbf{w} + b) - 1 + \xi_i] - \sum_{i=1}^{L}\mu_i\xi_i, \tag{43}$$

where $\mu_i \geq 0 \ \forall i \in \{1, \ldots, L\}$ denotes the Lagrange multipliers associated to the restriction $\xi_i \geq 0 \ \forall i \in \{1, \ldots, L\}$. The aim is now to minimize L_P in (43) with respect to $\mathbf{w}$, b, and ξ_i, and maximize it with respect to $\boldsymbol{\alpha}$ (where $\alpha_i, \mu_i \geq 0 \ \forall i \in \{1, \ldots, L\}$). Setting the corresponding derivatives equal to zero renders (36), (37) and also:

$$\frac{\partial L_P}{\partial \xi_i} = 0 \longrightarrow C = \alpha_i + \mu_i \tag{44}$$

Substituting (36), (37) and (44) in (43), L_D has the same form as (38) before. However (44) together with $\mu_i \geq 0 \ \forall i$ 8i, implies that $\alpha \leq C$. We therefore need to find:

$$\max_{\alpha}\left[\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{H}\boldsymbol{\alpha}\right] \quad \text{s.t} \quad 0 \leq \alpha_i \leq C \quad \forall i \in \{1, \ldots, L\} \text{ and } \sum_{i=1}^{L}\alpha_i y_i = 0 \tag{45}$$

Parameter b is then computed analogously to (40), although in this case, the set of support vectors used to calculate it is determined by the indices i such that $0 < \alpha_i \leq C$.

3.1.3 Kernel functions

When applying SVM to linearly separable data, a matrix $\mathbf{H}$ was defined (see eq (38)) from the dot product of the input variables:

$$H_{ij} = y_i y_j \mathbf{x}^i \cdot \mathbf{x}^j \equiv y_i y_j k(\mathbf{x}^i, \mathbf{x}^j). \tag{46}$$

The function $k(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i)^T \cdot \mathbf{x}^j$ is an example of a family called kernel functions [19], which is known as linear kernel.

The set of kernel functions is composed of variants of the previous function in the sense that they are all based on calculating inner products of two vectors. This means that if the functions can be recast into a higher-dimensional space by some potentially non-linear feature mapping function $\phi(x)$, only inner products of the mapped inputs in the feature space need to be determined, without the explicit need to calculate function ϕ.

The reason why these kernel functions turn out to be useful is that there are certain classification problems that are not linearly separable in the space of the inputs $\mathbf{x}^i$, but which could be separable in a higher dimensionality feature space given a suitable mapping $x \to \phi(x)$. Some of the most recurring kernel functions are:

- Gaussian radial basis kernel function.

$$k(\mathbf{x}^i, \mathbf{x}^j) = e^{-\left(\frac{||\mathbf{x}^i - \mathbf{x}^j||^2}{2\sigma^2}\right)}. \tag{47}$$

- Polynomial kernel function.

$$k(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i \cdot \mathbf{x}^j + a)^b \tag{48}$$

  .

  If $a = 0, b = 1$ one retrieves the linear kernel.

- Sigmoidal kernel function.

$$k(\mathbf{x}^i, \mathbf{x}^j) = \tanh\left(a\mathbf{x}^i \cdot \mathbf{x}^j - b\right). \tag{49}$$

These are the kernel functions that will be used throughout this work. Of course,

kernel functions are an extensive family, whose characterization is given by Mercer's theorem, based on certain properties such as symmetry (see [20] for a detailed formulation).

3.1.4 Multi-class SVM

Sections 3.1.1, 3.1.2 and 3.1.3 provide the tools to implement binary SVM classification. The next step is to extend this to the classification scenario where instances are to be classified into more than two classes, i.e., the multi-class frame. Two of the earliest and most recurrent approaches are the so-called one-against-all and one-against-one [21].

1. **One-against-all approach:** it constructs K SVM models where K is the number of classes. The ith SVM is trained with all of the examples in the ith class with positive labels, while negative labels are assigned to all other examples. Then, K problems like (42) are solved, one for each class. Then, one ends up with K hyperplane configurations given by $\mathbf{w}_m$ and b_m, where $m \in \{1, \ldots, K\}$. One finally classifies each instance following the criterion:

$$\text{class of } \mathbf{x}^i = \arg \max_{m=1,\ldots,K} \mathbf{w}_m \cdot \mathbf{x}^i + b_m, \tag{50}$$

   or

$$\text{class of } \mathbf{x}^i = \arg \max_{m=1,\ldots,K} \mathbf{w}_m \cdot \phi\left(\mathbf{x}^i\right) + b_m \tag{51}$$

   if one has made use of a feature map in order to apply the so-called kernel trick (i.e., use kernel functions).

2. **One-against-one approach:** This method constructs $K \cdot (K-1)/2$ classifiers, where each one is trained on data from two classes. For training data from the nth and the mth classes, one solves the binary classification problem given by (42). There are different methods for doing future testing after all classifiers are constructed. One of the most efficient [21] consists of considering the sign of $\mathbf{w}_{nm} \cdot \mathbf{x}^i + b_{nm}$, where $n, m \in \{1, \ldots, K\}$, K denoting the number of classes. If this sign classifies the ith instance into the mth class, then the vote for the mth class is increased in one unit. Finally, the predicted class for $\mathbf{x}^i$ is the one with the most votes. In case two classes have identical votes, one simply selects the one with the smaller index, despite this not being a rigorous criterion.

There are more alternatives, although in this work we shall stick to the aforementioned ones, which are the ones applied in the `svm` and `svm.SVC` functionalities of the `sklearn` Python library. However, some of these alternatives, discussed in [21], succeed in implementing multi-class classification by solving only one single optimization problem, instead of K or $K \cdot (K-1)/2$ ones.

3.1.5 Multi-label SVM

Finally, one can also seek strategies to tackle classification problems where more than one label can be assigned to each instance, which are known as multi-label classification problems. Existing strategies to approach these problems can be roughly categorized into three families, based on the order of correlations between labels that the learning techniques have considered [22]:

1. **First-order strategy:** The task of multi-label learning is tackled in a label-by-label style, therefore ignoring the co-existence of the other labels, for instance by decomposing the multi-label learning problem into a number of independent binary classification problems (one per label). This can be done by binarizing the response variable so that, for each individual problem, which corresponds to one particular label, each instance is either assigned to this label or not. This methodology is known as binary relevance and counts on the advantage that it can be parallelized in the number of labels.

   The main asset of first-order strategies lies in its conceptual simplicity and high efficiency. On the other hand, the effectiveness of the resulting approaches might be suboptimal due to the ignorance of label correlations.

2. **Second-order strategy:** The task of multi-label learning is approached by considering pairwise relations between labels, such as the ranking between the relevant label and irrelevant label, or interaction between any pair of labels. As label correlations are exploited to some extent by second-order strategy, the resulting approaches can achieve a suitable generalization performance. However, there are certain real-world applications where label correlations go beyond the second-order assumption.

3. **High-order strategy:** one considers high-order relations among labels such as imposing all other labels' influences on each label, or addressing connections among random subsets of labels. Apparently high-order strategy has stronger correlation-modeling capabilities than first-order and second-order strategies, while on the other hand, it is computationally more demanding and less scalable.

   An example of these strategies is the random k-label sets algorithm, which transforms the multi-label learning problem into an ensemble of multi-class classification problems, where each component learner in the ensemble targets a random subset of labels of the label space upon which a multi-class classifier is induced. In the training phase, the algorithm converts the original multi-label training set into a multi-class training set by treating every distinct label set appearing in the training set as a new class. One of the main drawbacks of this method is incompleteness since it is confined to predicting label sets that appear in the training set. In addition, when there are many label combinations, complexity scales and there are fewer training examples per constructed

class, leading to inefficiency. However, it could be a suitable choice when there number of label combinations is reduced compared to the number of training instances.

3.1.6 Quantum SVM implementation

Quantum SVMs (QSVMs) are particular cases of SVMs that rely on the kernel trick. When moving data to a higher dimensional space, or feature space, in order to seek a linear separation, so far, we have only considered the Euclidean space. In the quantum frame, one uses as feature space a certain space of quantum states (Hilbert space). This idea inspired some works, such as [23], to try to use quantum circuits in order to compute kernels and, hopefully, obtain some advantage over classical computers by working in a sophisticated feature space.

Then, QSVMs follow the classical methodology described in the previous sections except for the computation of the kernel function, which requires a quantum computer to take two input vectors, map them onto a feature map, compute their inner product, and return it [24].

Given a feature map φ, for each input $\mathbf{x}$, we shall have a circuit $\Phi(\mathbf{x})$ such that the output of the feature map will be the quantum state $\varphi(\mathbf{x}) = \Phi(\mathbf{x})\,|0\rangle$ (quantum circuits are usually initialized on state $|0\rangle$). The kernel function will then be given by:

$$k(\mathbf{a}, \mathbf{b}) = |\langle\varphi(\mathbf{a})|\varphi(\mathbf{b})\rangle|^2 = \left|\langle 0|\,\Phi^\dagger(\mathbf{a})\Phi(\mathbf{b})\,|0\rangle\right|^2. \tag{52}$$

This computation is trivial for a quantum computer since it describes the probability of measuring all zeros after preparing the state $\Phi^\dagger(\mathbf{a})\Phi(\mathbf{b})\,|0\rangle$, given that the computational basis is orthonormal. It is also relevant to note that, since quantum circuits are always represented by unitary operators, then $\Phi^\dagger$ is simply the inverse of Φ. Furthermore, since Φ is given by a sequence of quantum gates, $\Phi^\dagger$ is retrieved by applying these gates from right to left and inverting them. Finally, these quantum kernels fulfill the conditions required to qualify as kernel functions [25].

Regarding feature maps, these are often defined by a parametrized circuit $\Phi(\mathbf{x})$ that depends on the original data and thus can be used to prepare a state that depends on it. Two of the main procedures to encode data are:

- **Angle encoding:** when used on an n-qubit circuit, this feature map can take up to n numerical inputs $x_1, \ldots, x_n$. The action of its circuit consists in the application of a rotation gate on each qubit j parametrized by the value x_j. In this case, we are using the x_j values as angles in the rotations, which accounts for the name of the encoding.

  It is important to recall that circuits usually take $|0\rangle$ as the initial state, hence applying an operator like a Pauli Z gate (see eq.22) directly has no effect. This is the reason why, when Z gates are used, it is customary to precede them by

Hadamard gates acting on each qubit.

Input variables for angle encoding should also be normalized between 0 and 4π (not 2π because in eq.16 angles are divided by 2), so as to cover the whole feature space, even if this identifies the two extrema of the data under the action of the feature map.

- **Amplitude encoding:** it can take 2^n inputs when implemented on an n-qubit circuit, which allows for handling datasets with a large number of variables. Given an input $(x_0, \ldots, x_{2^n-1})$, the (normalized) state is:

$$|\varphi(\mathbf{a})\rangle = \frac{1}{\sqrt{\sum_k x_k^2}} \sum_{k=0}^{2^n-1} x_k |k\rangle \tag{53}$$

  Note that input $|0\rangle$ is not supported by this encoding. This encoding is, however, complex to achieve in terms of elementary quantum gates. A detailed discussion of this matter can be found in [26].

A widespread example of feature map is `Qiskit`'s `ZZFeatureMap`, which can take n inputs on n qubits (resembling angle encoding) and builds a parametrized circuit as follows [25]:

1. Apply a Hadamard gate on each qu-bit.
2. Apply on each qubit j a rotation $R_Z(2x_j)$ (defined in Section 2.6.3).
3. For each pair of elements $\{j, k\}$, $j, k \in \{1, \ldots, m\}$ and $j < k$, do:
   (a) Apply a CNOT gate targetting qubit k controlled by qu-bit j.
   (b) Apply a rotation $R_Z\left(2(\pi - x_j)(\pi - x_k)\right)$.
   (c) Repeat step 3.a.

Figure 4 represents an example of a `ZZFeatureMap` feature map on three qubits.

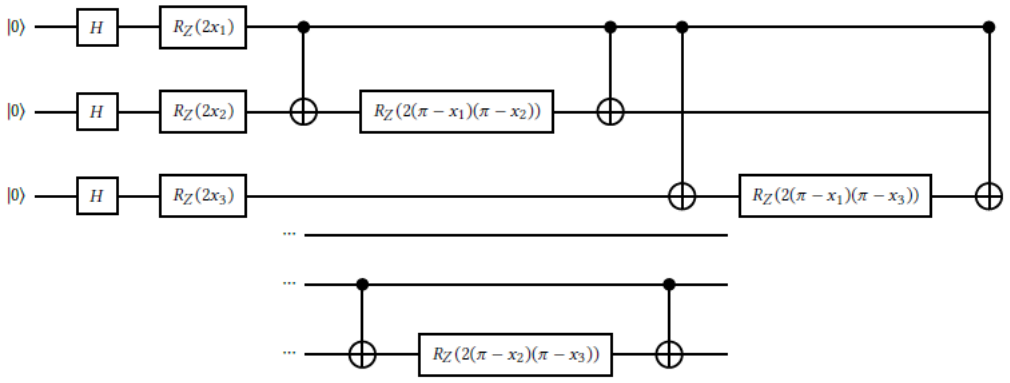


Fig. 4. Representation of a `ZZFeatureMap` feature map on three qubits with inputs x_1, x_2 and x_3.

Of course, there are other options, such as `Qiskit`'s `ZFeatureMap` or the more general `PauliFeatureMap`. In `Qiskit`, when calling the `QSVC` functionality, one needs to specify a kernel, which depends on a feature map. We shall consider two such kernels, `FidelityQuantumKernel` and `FidelityStatevectorKernel`, which compute the same inner product $|\langle\varphi(\mathbf{a})|\varphi(\mathbf{b})\rangle|$, but there are efficiency differences between them. It is also relevant to highlight the fact that all computational resources required to simulate quantum circuits, as well as computations on classical software, significantly increase computational time with respect to a quantum computer. In the latter, the number of operations is exponential with respect to the number of qubits, so memory limitations would appear on classical software for significantly smaller problems than in the quantum computing scenario. On the other hand, the construction of quantum computers has to face the major setback of noise and decoherence, which can be approached, among other techniques, through quantum error correction protocols (see Section 2.7).

3.2 Results

This section is devoted to contrasting the performance and computational efficiency of binary, multi-class and multi-label SVM algorithms that rely both on classical and quantum kernels. This comparison will be performed on two binary datasets, a multi-class and a multi-label one. More specifically, for the binary and multi-class scenarios, the comparison shall encompass linear SVM, gaussian kernel SVM and Quantum SVM. The latter is subdivided into the four combinations that span from considering the `Qiskit` feature maps `ZFeatureMap` and `ZZFeatureMap` together with kernels `FidelityStatevectorKernel` and `FidelityQuantumKernel`. Since the main goal is to contrast classical versus quantum algorithms, each of the aforementioned quantum configurations is individually compared to its classical counterparts in separate tables. Regarding multi-label SVM, we shall make use of the conclusions drawn from the previous scenarios to apply the most suitable quantum configuration in terms of time efficiency.

Each table summarizes the computational time and performance scores (in this

case, global accuracy and F1 score) of each algorithm for each combination of principal components, with training and testing observations. The aim is to create a significantly wide scenario variability so that conclusions on the suitability or optimality of a certain method for a given scenario can be inferred.

Regarding the F1 score, it is taken in the `weighted` mode, that is, the metrics for each label are computed, and then their average weighted by support (the number of true instances for each label) is calculated. This accounts for label imbalance, although in certain cases there is none, whenever both training and testing sets are taken with an equal cardinality, as happens in 3.2.2. This weighted score need not be between precision and recall.

For the sake of reproducibility, it is relevant to highlight that all the computations hereinafter (including Section 4) have been performed using a MSI laptop with 13th Gen Intel(R) Core(TM) i7-13700H 2.40 GHz processor and 32,0 GB RAM in 64 bits Windows 11 Pro.

3.2.1 SVM for binary classification

The dataset considered for this scenario is the Wisconsin breast cancer one [27] that belongs to the `Scikit-Learn` dataset library. It consists of 569 instances of 30 continuous variables, which include cell nuclei measures such as radius, perimeter, compactness, smoothness, concave points, texture, etc. The (binary) response variable indicates whether the tumor is benign (357 instances) or malignant (212) instances. The results drawn from applying the aforementioned SVM variants to the previously mentioned scenarios are listed in tables 1, 2, 3, and 4. Since there is no sheer imbalance in this dataset, both training and testing sets keep the global response variable proportion.

Principal components	**Train observations**	**Test observations**	**Linear SVM**			**Gaussian SVM**			**Quantum SVM**		
			Time	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**
2	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	86.7 %	0.870	0.1 s	80.0 %	0.805
	90	45	0.000 s	93.3 %	0.933	0.001 s	93.3 %	0.933	0.5 s	**95.6 %**	**0.955**
	180	90	0.002 s	92.2 %	0.922	0.001 s	**93.3 %**	**0.933**	1.0 s	**93.3 %**	**0.933**
5	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	**93.3 %**	**0.935**	0.3 s	86.7 %	0.870
	90	45	0.001 s	95.6 %	0.955	0.001 s	95.6 %	0.955	0.8 s	**97.8 %**	**0.978**
	180	90	0.003 s	93.3 %	0.933	0.001 s	**95.6 %**	**0.956**	1.9 s	94.4 %	0.945
10	30	15	0.003 s	93.3 %	0.935	0.001 s	**100 %**	**1**	0.6 s	80.0 %	0.803
	90	45	0.003 s	**97.8 %**	**0.978**	0.000 s	95.5 %	0.955	1.8 s	**97.8 %**	**0.978**
	180	90	0.003 s	95.6 %	0.956	0.002 s	**96.7 %**	**0.967**	11.1 s	93.3 %	0.933
20	30	15	0.004 s	**100 %**	**1**	0.001 s	93.3 %	0.931	226.3 s	86.7 %	0.867
	90	45	0.003 s	**97.8 %**	**0.978**	0.002 s	95.6 %	0.955	1609.5 s	95.6 %	0.955
	180	90	0.001 s	**94.4 %**	**0.945**	0.001 s	**94.4 %**	**0.945**	(3601.8 s)	(91.1 %)	(0.910)

Table 1: Linear SVM, Gaussian SVM and Quantum SVM results using `ZFeatureMap` feature map and `FidelityStatevectorKernel` kernel for the breast cancer dataset.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	86.7 %	0.870	0.3 s	46.7 %	0.467
	90	45	0.000 s	**93.3 %**	**0.933**	0.001 s	**93.3 %**	**0.933**	1.3 s	82.2 %	0.808
	180	90	0.002 s	92.2 %	0.922	0.001 s	**93.3 %**	**0.933**	3.1 s	87.8 %	0.879
5	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	**93.3 %**	**0.935**	3.0 s	66.7 %	0.800
	90	45	0.001 s	**95.6 %**	**0.955**	0.001 s	**95.6 %**	**0.955**	9.5 s	75.6 %	0.704
	180	90	0.003 s	93.3 %	0.933	0.001 s	**95.6 %**	**0.956**	17.7 s	80.0 %	0.775
10	30	15	0.003 s	93.3 %	0.935	0.001 s	**100 %**	**1**	13.8 s	66.7 %	0.800
	90	45	0.003 s	**97.8 %**	**0.978**	0.000 s	95.5 %	0.955	40.8 s	66.7 %	0.800
	180	90	0.003 s	95.6 %	0.956	0.002 s	**96.7 %**	**0.967**	181.4 s	66.7 %	0.800
20	30	15	0.004 s	**100 %**	**1**	0.001 s	93.3 %	0.931	2750.9 s	66.7 %	0.800
	90	45	0.003 s	**97.8 %**	**0.978**	0.002 s	95.6 %	0.955	(10885.9 s)	(66.7 %)	(0.800)
	180	90	0.001 s	**94.4 %**	**0.945**	0.001 s	**94.4 %**	**0.945**	(19902.2 s)	(66.7 %)	(0.800)

Table 2: Linear SVM, Gaussian SVM and Quantum SVM results using `ZZFeatureMap` feature map and `FidelityStatevectorKernel` kernel for the breast cancer dataset.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	86.7 %	0.870	2.3 s	80.0 %	0.805
	90	45	0.000 s	93.3 %	0.933	0.001 s	93.3 %	0.933	18.8 s	**95.6 %**	**0.955**
	180	90	0.002 s	92.2 %	0.922	0.001 s	**93.3 %**	**0.933**	86.9 s	**93.3 %**	**0.933**
5	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	**93.3 %**	**0.935**	4.7 s	86.7 %	0.870
	90	45	0.001 s	95.6 %	0.955	0.001 s	95.6 %	0.955	64.3 s	**97.8 %**	**0.978**
	180	90	0.003 s	93.3 %	0.933	0.001 s	**95.6 %**	**0.956**	232.9 s	94.4 %	0.945
10	30	15	0.003 s	93.3 %	0.935	0.001 s	**100 %**	**1**	23.3 s	80.0 %	0.804
	90	45	0.003 s	**97.8 %**	**0.978**	0.000 s	95.5 %	0.955	219.9 s	**97.8 %**	**0.978**
	180	90	0.003 s	95.6 %	0.956	0.002 s	**96.7 %**	**0.967**	1760.2 s	93.3 %	0.933
20	30	15	0.004 s	**100 %**	**1**	0.001 s	93.3 %	0.931	» 3600 s	-	-
	90	45	0.003 s	**97.8 %**	**0.978**	0.002 s	95.6 %	0.955	» 3600 s	-	-
	180	90	0.001 s	**94.4 %**	**0.945**	0.001 s	**94.4 %**	**0.945**	» 3600 s	-	-

Table 3: Linear SVM, Gaussian SVM and Quantum SVM results using `ZFeatureMap` feature map and `FidelityQuantumKernel` kernel for the breast cancer dataset.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	86.7 %	0.870	3.3 s	46.7 %	0.467
	90	45	0.000 s	**93.3 %**	**0.933**	0.001 s	**93.3 %**	**0.933**	45.6 s	82.2 %	0.808
	180	90	0.002 s	92.2 %	0.922	0.001 s	**93.3 %**	**0.933**	285.2 s	87.8 %	0.879
5	30	15	0.001 s	**93.3 %**	**0.935**	0.001 s	**93.3 %**	**0.935**	43.3 s	66.7 %	0.800
	90	45	0.001 s	**95.6 %**	**0.955**	0.001 s	**95.6 %**	**0.955**	479.1 s	75.6 %	0.704
	180	90	0.003 s	93.3 %	0.933	0.001 s	**95.6 %**	**0.956**	1861.2 s	80.0 %	0.775
10	30	15	0.003 s	93.3 %	0.935	0.001 s	**100 %**	**1**	211.8 s	66.7 %	0.800
	90	45	0.003 s	**97.8 %**	**0.978**	0.000 s	95.5 %	0.955	1868.8 s	66.7 %	0.800
	180	90	0.003 s	95.6 %	0.956	0.002 s	**96.7 %**	**0.967**	» 3600 s	-	-
20	30	15	0.004 s	**100 %**	**1**	0.001 s	93.3 %	0.931	» 3600 s	-	-
	90	45	0.003 s	**97.8 %**	**0.978**	0.002 s	95.6 %	0.955	» 3600 s	-	-
	180	90	0.001 s	**94.4 %**	**0.945**	0.001 s	**94.4 %**	**0.945**	» 3600 s	-	-

Table 4: Linear SVM, Gaussian SVM and Quantum SVM results using `ZZFeatureMap` feature map and `FidelityQuantumKernel` kernel for the breast cancer dataset.

Regarding computational efficiency, it is clear that in all cases classical SVM is about four orders of magnitude faster. Nonetheless, it is relevant to highlight that all the computations, including quantum ones, have been performed on a classical computer, not on a proper quantum computer, where there is no need for simulation, but only for letting the system evolve, which consumes significantly less time.

Comparing Tables 4 and 3 with Tables 1 and 2, one spots a difference of approximately one order of magnitude in computational time. In other words, `FidelityQuantumKernel` is about one order of magnitude slower than `FidelityStatevectorKernel`. On a finer scale, comparing Tables 4 and 2 with 3 and 2, one also notices that `ZFeatureMap` leads to faster computations than `ZZFeatureMap`. Given that classical computations tend to take thousandths of a second, for efficiency comparison purposes, the quantum simulations were stopped if they took longer than one hour (that is, a difference of six orders of magnitude). However, for the sake of completeness, in the cases when computation ended after several hours (no more than twelve), the results are included in brackets. This criterion shall also be held for the following sections.

As for performance, the only difference concerning quantum configurations falls on the feature maps. While `ZFeatureMap` has an associated accuracy of over 80% in all cases, while `ZZFeatureMap` seldom surpasses this threshold. F1 score is in general similar to accuracy, the main differences occurring the further these are to one. Therefore, the most suitable quantum configuration has proved to be `ZFeatureMap` and `FidelityStatevectorKernel` (Table 1). Let us now compare the results rendered by the latter configuration with those obtained classically.

- For $n = 2$, quantum methods outperform classical ones for 90 training observations, and are only outperformed when 180 observations are considered.

- For $n = 5$, the same applies, although this time, gaussian kernel SVM outperforms quantum SVM for 180 training instances.

- For $n = 10$, the highest accuracies are achieved by classical methods, although for 90 training instances quantum and linear SVM render the same accuracy.

- For $n = 20$, classical algorithms outperform quantum ones.

Taking everything into consideration, one observes that the potential scenarios where quantum methodology is prone to outperforming classical computations are the "intermediate" ones. In other words, for a (binary classification) dataset that consists of between 2 and 5 features, and such that about 100 observations are available for training, one could explore the quantum alternatives to slightly enhance classical performance without a significant time consumption (< 2 s).

Let us now consider a second dataset that consists of 64709 instances of COVID tests at the end of April 2020 in Israel. This dataset covers four categorical variables:

- `cough`: binary variable that indicates whether the patient coughed regularly.

- `fever`: binary variable that indicates whether the patient had fever.
- `gender`: gender of the patient, classified into `male`, `female` and `none`.
- `test_indication`: describes patient's recent activity, classified into `abroad`, `contact with positive` or `other`.

The (binary) response variable is labelled as `corona_result`. This dataset contrasts the previous one in the significantly lower number of variables, as well as the fact that it is made up of categorical variables, which only allow for $2 \cdot 2 \cdot 3 \cdot 3 = 36$ different types of patients. In addition, the response variable is substantially unbalanced, with only 1336 positive instances. Let us study the performance of both the classical and quantum SVM implementations in this type of scenario.

In both implementations, we shall train the model with a balanced (50/50) training set, and then, perform the testing on a set that keeps the original response variable proportions. In this case, given that there are only four columns in the dataset, no PCA will be performed. The results rendered both by classical and quantum algorithms are listed in Tables 5, 6, 7 and 8.

Train	**Test**	**Linear SVM**			**Gaussian SVM**			**Quantum SVM**		
observations	**observations**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**
200	100	0.002 s	0.920	0.926	0.001 s	0.940	0.940	0.4 s	**0.950**	**0.947**
1000	500	0.003 s	0.920	0.931	0.009 s	0.882	0.907	7.7 s	**0.932**	**0.938**
2000	1000	0.008 s	0.913	0.926	0.035 s	**0.924**	**0.936**	32.2 s	0.920	0.933

Table 5: Linear SVM, Gaussian SVM and Quantum SVM results using `ZFeatureMap` feature map and `FidelityStatevectorKernel` kernel for the COVID dataset.

Train	**Test**	**Linear SVM**			**Gaussian SVM**			**Quantum SVM**		
observations	**observations**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**
200	100	0.002 s	0.920	0.926	0.001 s	0.940	0.940	0.5 s	**0.950**	**0.947**
1000	500	0.003 s	**0.920**	**0.931**	0.009 s	0.882	0.907	8.1 s	0.908	0.923
2000	1000	0.008 s	0.913	0.926	0.035 s	**0.924**	**0.936**	32.6 s	0.891	0.914

Table 6: Linear SVM, Gaussian SVM and Quantum SVM results using `ZZFeatureMap` feature map and `FidelityStatevectorKernel` kernel for the COVID dataset.

Train	**Test**	**Linear SVM**			**Gaussian SVM**			**Quantum SVM**		
observations	**observations**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**	**Time**	**Accuracy**	**F1 score**
200	100	0.002 s	0.920	0.926	0.001 s	0.940	0.940	274.9 s	**0.950**	**0.947**
1000	500	0.003 s	**0.920**	**0.931**	0.009 s	0.882	0.907	>3600 s	-	-
2000	1000	0.008 s	0.913	0.926	0.035 s	**0.924**	**0.936**	>3600 s	-	-

Table 7: Linear SVM, Gaussian SVM and Quantum SVM results using `ZFeatureMap` feature map and `FidelityQuantumKernel` kernel for the COVID dataset.

Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
		Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
200	100	0.002 s	0.920	0.926	0.001 s	0.940	0.940	549.5 s	**0.950**	**0.947**
1000	500	0.003 s	**0.920**	**0.931**	0.009 s	0.882	0.907	>3600 s	-	-
2000	1000	0.008 s	0.913	0.926	0.035 s	**0.924**	**0.936**	>3600 s	-	-

Table 8: Linear SVM, Gaussian SVM and Quantum SVM results using `ZZFeatureMap` feature map and `FidelityQuantumKernel` kernel for the COVID dataset.

One observes that in the cases of 200 and 1000 training instances, the quantum configuration with `ZFeatureMap` feature map and `FidelityStatevectorKernel` kernel outperforms the linear and gaussian classical kernels, although the difference in scores is not substantial (usually 0.1-0.2). One also notes that, especially in the quantum scenario, neither accuracy nor F1 score increase so significantly when increasing the number of observations. The reason behind this, as discussed earlier, could lie in the fact that there is a maximum of 36 different observations in this dataset, so increasing the training set would not substantially increase the information of the model, but rather readjust the hyperplane depending on the weight or "influence" that each category has in each of the (maximum 36) types of instance. In fact, for 200 training instances, the highest overall accuracy, as well as the lowest false negative rate (2 %) was achieved by quantum configurations (see 6).

Regarding computational efficiency, as happened with the previous dataset, the quantum implementations are about three orders of magnitude slower (again, recalling the consideration that these simulations are performed on classical software). However, given the small number of variables considered, both methods render results in an acceptable time.

3.2.2 SVM for multi-class classification

The dataset considered for this section accounts for Dengue fever in Medellin. It consists of 32549 instances of 21 binary attributes related to symptoms like fever, hemorrhage, hypothermia, abdominal pain, etc. The response variable contains three classes of dengue (labelled 0, 1 and 2). Since there is a significantly higher number of instances than in the previous binary dataset, the number of training instances has been updated to cover a richer variety of scenarios. It is relevant to highlight that this is a balanced dataset, with 10210 observations for class 0 dengue, 11163 for class 1 and 11186 for class 2.

The results drawn from applying the aforementioned SVM variants to these scenarios are listed in Tables 9, 10, 11, and 12.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	66.7 %	0.634	0.002 s	**73.3 %**	**0.733**	0.1 s	46.7 %	0.478
	300	150	0.002 s	72.7 %	0.713	0.004 s	**82.7 %**	**0.819**	1.3 s	74.7 %	0.731
	1500	750	0.005 s	74.9 %	0.734	0.018 s	**82.5 %**	**0.823**	19.3 s	77.3 %	0.762
	3000	1500	0.010 s	78.7 %	0.776	0.065 s	**83.9 %**	**0.837**	72.3 s	79.7 %	0.787
5	30	15	0.001 s	**86.7 %**	**0.861**	0.001 s	80.0 %	0.798	0.2 s	66.7 %	0.674
	300	150	0.002 s	77.3 %	0.766	0.003 s	**85.3 %**	**0.849**	2.1 s	74.7 %	0.737
	1500	750	0.005 s	76.4 %	0.757	0.018 s	**85.6 %**	**0.854**	29.0 s	81.6 %	0.809
	3000	1500	0.008 s	76.1 %	0.754	0.065 s	**88.0 %**	**0.879**	76.6 s	84.1 %	0.839
10	30	15	0.001 s	**86.7 %**	**0.867**	0.001 s	**86.7 %**	**0.867**	0.4 s	66.7 %	0.677
	300	150	0.002 s	86.7 %	0.863	0.002 s	**92.0 %**	**0.919**	4.1 s	89.3 %	0.891
	1500	750	0.007 s	87.2 %	0.871	0.016 s	**94.5 %**	**0.945**	41.6 s	**94.5 %**	**0.945**
	3000	1500	0.018 s	88.7 %	0.886	0.084 s	95.7 %	0.956	115.7 s	**96.5 %**	**0.965**
20	30	15	0.001 s	**93.3 %**	**0.933**	0.001 s	86.7 %	0.867	111.4 s	86.7 %	0.861
	300	150	0.002 s	98.0 %	0.980	0.002 s	**98.7 %**	**0.987**	3341.5 s	**98.7 %**	**0.987**
	1500	750	0.011 s	**99.9 %**	**0.999**	0.024 s	**99.9 %**	**0.999**	» 3600 s	-	-
	3000	1500	0.016 s	99.7 %	0.997	0.039 s	**100 %**	**1.0**	» 3600 s	-	-

Table 9: Linear SVM, Gaussian SVM and Quantum SVM results using `ZFeatureMap` feature map and `FidelityStatevectorKernel` kernel for the Dengue dataset.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	66.7 %	0.634	0.002 s	**73.3 %**	**0.733**	0.4 s	60 %	0.606
	300	150	0.002 s	72.7 %	0.713	0.004 s	**82.7 %**	**0.819**	5.0 s	72.0 %	0.708
	1500	750	0.005 s	74.9 %	0.734	0.018 s	**82.5 %**	**0.823**	30.2 s	74.9 %	0.741
	3000	1500	0.010 s	78.7 %	0.776	0.065 s	**83.9 %**	**0.837**	256.5 s	77.0 %	0.763
5	30	15	0.001 s	**86.7 %**	**0.861**	0.001 s	80.0 %	0.798	4.7 s	26.7 %	0.274
	300	150	0.002 s	77.3 %	0.766	0.003 s	**85.3 %**	**0.849**	17.4 s	78.7 %	0.787
	1500	750	0.005 s	76.4 %	0.757	0.018 s	85.6 %	0.854	73.0 s	**95.7 %**	**0.957**
	3000	1500	0.008 s	76.1 %	0.754	0.065 s	88.0 %	0.879	355.8 s	**98.0 %**	**0.981**
10	30	15	0.001 s	**86.7 %**	**0.867**	0.001 s	**86.7 %**	**0.867**	20.9 s	53.3 %	0.511
	300	150	0.002 s	86.7 %	0.863	0.002 s	**92.0 %**	**0.919**	99.5 s	80.0 %	0.792
	1500	750	0.007 s	87.2 %	0.871	0.016 s	94.5 %	0.945	272.6 s	**99.2 %**	**0.992**
	3000	1500	0.018 s	88.7 %	0.886	0.084 s	95.7 %	0.956	741.4 s	**99.6 %**	**0.996**
20	30	15	0.001 s	**93.3 %**	**0.933**	0.001 s	86.7 %	0.867	2904.0 s	40.0 %	0.393
	300	150	0.002 s	98.0 %	0.980	0.002 s	**98.7 %**	**0.987**	» 3600 s	(80.0 %)	-
	1500	750	0.011 s	**99.9 %**	**0.999**	0.024 s	**99.9 %**	**0.999**	» 3600 s	-	-
	3000	1500	0.016 s	99.7 %	0.997	0.039 s	**100 %**	**1.0**	» 3600 s	-	-

Table 10: Linear SVM, Gaussian SVM and Quantum SVM results using `ZZFeatureMap` feature map and `FidelityStatevectorKernel` kernel for the Dengue dataset.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	66.7 %	0.634	0.002 s	**73.3 %**	**0.733**	2.1 s	46.7 %	0.478
	300	150	0.002 s	72.7 %	0.713	0.004 s	**82.7 %**	**0.819**	321.4 s	74.7 %	0.731
	1500	750	0.005 s	74.9 %	0.734	0.018 s	**82.5 %**	**0.823**	» 3600 s	-	-
	3000	1500	0.010 s	78.7 %	0.776	0.065 s	**83.9 %**	**0.837**	» 3600 s	-	-
5	30	15	0.001 s	**86.7 %**	**0.861**	0.001 s	80.0 %	0.798	4.7 s	66.7 %	0.674
	300	150	0.002 s	77.3 %	0.766	0.003 s	**85.3 %**	**0.849**	660.9 s	74.7 %	0.737
	1500	750	0.005 s	76.4 %	0.757	0.018 s	**85.6 %**	**0.854**	» 3600 s	-	-
	3000	1500	0.008 s	76.1 %	0.754	0.065 s	**88.0 %**	**0.879**	» 3600 s	-	-
10	30	15	0.001 s	**86.7 %**	**0.867**	0.001 s	**86.7 %**	**0.867**	21.1 s	66.7 %	0.677
	300	150	0.002 s	86.7 %	0.863	0.002 s	**92.0 %**	**0.919**	2734.8 s	89.3 %	0.892
	1500	750	0.007 s	87.2 %	0.871	0.016 s	**94.5 %**	**0.945**	» 3600 s	-	-
	3000	1500	0.018 s	88.7 %	0.886	0.084 s	**95.7 %**	**0.956**	» 3600 s	-	-
20	30	15	0.001 s	**93.3 %**	**0.933**	0.001 s	86.7 %	0.867	» 3600 s	-	-
	300	150	0.002 s	98.0 %	0.980	0.002 s	**98.7 %**	**0.987**	» 3600 s	-	-
	1500	750	0.011 s	**99.9 %**	**0.999**	0.024 s	**99.9 %**	**0.999**	» 3600 s	-	-
	3000	1500	0.016 s	99.7 %	0.997	0.039 s	**100 %**	**1.0**	» 3600 s	-	-

Table 11: Linear SVM, Gaussian SVM and Quantum SVM results using `ZFeatureMap` feature map and `FidelityQuantumKernel` kernel for the Dengue dataset.

Principal components	Train observations	Test observations	Linear SVM			Gaussian SVM			Quantum SVM		
			Time	Accuracy	F1 score	Time	Accuracy	F1 score	Time	Accuracy	F1 score
2	30	15	0.001 s	66.7 %	0.634	0.002 s	**73.3 %**	**0.733**	5.3 s	60.0 %	0.606
	300	150	0.002 s	72.7 %	0.713	0.004 s	**82.7 %**	**0.819**	774.9 s	72.0 %	0.708
	1500	750	0.005 s	74.9 %	0.734	0.018 s	**82.5 %**	**0.823**	» 3600 s	-	-
	3000	1500	0.010 s	78.7 %	0.776	0.065 s	**83.9 %**	**0.837**	» 3600 s	-	-
5	30	15	0.001 s	**86.7 %**	**0.861**	0.001 s	80.0 %	0.798	76.8 s	26.7 %	0.274
	300	150	0.002 s	77.3 %	0.766	0.003 s	**85.3 %**	**0.849**	» 3600 s	(78.7 %)	(0.787)
	1500	750	0.005 s	76.4 %	0.757	0.018 s	**85.6 %**	**0.854**	» 3600 s	-	-
	3000	1500	0.008 s	76.1 %	0.754	0.065 s	**88.0 %**	**0.879**	» 3600 s	-	-
10	30	15	0.001 s	**86.7 %**	**0.867**	0.001 s	**86.7 %**	**0.867**	426.7 s	53.3 %	0.511
	300	150	0.002 s	86.7 %	0.863	0.002 s	**92.0 %**	**0.919**	» 3600 s	-	-
	1500	750	0.007 s	87.2 %	0.871	0.016 s	**94.5 %**	**0.945**	» 3600 s	-	-
	3000	1500	0.018 s	88.7 %	0.886	0.084 s	**95.7 %**	**0.956**	» 3600 s	-	-
20	30	15	0.001 s	**93.3 %**	**0.933**	0.001 s	86.7 %	0.867	» 3600 s	-	-
	300	150	0.002 s	98.0 %	0.980	0.002 s	**98.7 %**	**0.987**	» 3600 s	-	-
	1500	750	0.011 s	**99.9 %**	**0.999**	0.024 s	**99.9 %**	**0.999**	» 3600 s	-	-
	3000	1500	0.016 s	99.7 %	0.997	0.039 s	**100 %**	**1.0**	» 3600 s	-	-

Table 12: Linear SVM, Gaussian SVM and Quantum SVM results using `ZZFeatureMap` feature map and `FidelityQuantumKernel` kernel for the Dengue dataset.

Regarding computational efficiency, the same considerations as for the binary case hold. Comparing Tables 12 and 11 with Tables 9 and 10, one spots a difference of at least one order of magnitude in computational time, so that `FidelityQuantumKernel` is at least one order of magnitude slower than `FidelityStatevectorKernel`, while `ZFeatureMap` leads to faster computations than `ZZFeatureMap`. A high number of training instances severely penalizes computational time. This also happens in the classical frame but on a significantly finer scale.

As for performance, the only difference concerning quantum configurations again falls on the feature maps. This time `ZZFeatureMap` reaches the highest levels of accuracy, although it is outperformed by `ZFeatureMap` in certain scenarios. F1

score is in general very similar to accuracy since this dataset has completely balanced categories, and so do the training and testing sets.

Then, the most suitable quantum configurations have proved to be `FidelityStatevectorKernel` together with `ZZFeatureMap` and also `ZFeatureMap` (Tables 10 and 9, respectively). Let us now compare the results rendered by these configurations with those obtained classically.

- For $n = 2$, quantum methods are outperformed by at least one of the classical ones in all scenarios.

- For $n = 5$, the same applies if we consider `ZFeatureMap`, but the `ZZFeatureMap` configuration outperforms classical algorithms for 1500 and 3000 training instances, by a margin ≥ 10 %.

- For $n = 10$, `ZFeatureMap` configuration achieves higher accuracy than classical algorithms for 3000 training instances, and is not outperformed classically for 1500 training instances. However, both the `ZFeatureMap` configuration and classical algorithms are outperformed by the `ZZFeatureMap` configuration, which surpasses 99 % accuracy in both of these scenarios.

- For $n = 20$ (almost equivalent to no PCA), classical algorithms in general outperform quantum ones (although for 300 instances `ZFeatureMap` provides the same accuracy, but requires significant computational time), which tend to be too computationally inefficient.

Taking everything into consideration, one again notices that the potential scenarios where quantum methodology is prone to outperforming classical computations are the "intermediate" ones. In other words, for a (multi-class) dataset that consists of between 5 and 10 features, and such that about 1500 - 3000 observations are available for training, one could explore the quantum alternatives to slightly or even moderately enhance classical performance. It is relevant to observe that the intervals of features and observations that enhance quantum performance have shifted towards higher values, as have the number of categories from binary to 3-classes. In addition, the required computational time has increased between one and two orders of magnitude, both in the classical and quantum cases, therefore the proportion remains similar.

3.2.3 SVM for multi-label classification

Let us now consider a dataset which contains 9.921 tweets labelled with the concerns towards vaccines. Between 1 and 3 labels, out of a set of 12, are assigned to each tweet. The dataset is split into 80 % training observations and 20 % testing ones since there is a relatively high proportion of labels with respect to the global number of instances.

Out of the approaches discussed in Section 3.1.5, the random k-labelsets algorithm proves inappropriate for this case, given that 288 training categories appear, many of which are made up of only one instance. This technique could have been useful, for instance, if no more than two labels were assigned to each instance. Then, there would be a maximum of $\binom{12}{2} + 12 = 78$ categories (counting both training and testing), so there would be more than 100 instances per class on average.

Let us then implement the (first-order) binary relevance strategy. After some cleaning tasks like removing emojis or mentions to other users, we perform the train-test split. This is the starting point both for the classical and quantum implementations.

Classical implementation

We binarize the training set, i.e., convert the multi-label response variable into a binary matrix with 12 columns, each representing a label, and as many rows as training instances. We then use a pipeline to remove both accents and frequent English words like "a", "the", etc. The pipeline also includes a linear kernel (which outperformed sigmoid, polynomial and Gaussian kernels in terms of scores) one-against-all, also known as one-versus-rest classifier. We finally fit the model and predict on the test set, obtaining the scores given by Table 13. The time required to fit the model is 177.0 s. Note that there are as many variables as different words among all tweets (in this case 16762), so there are more variables than instances.

Class	Precision	Recall	F1-score	Support
Conspiracy	0.84	0.18	0.29	91
Country	0.75	0.23	0.35	39
Ineffective	0.78	0.46	0.58	342
Ingredients	0.88	0.33	0.48	114
Mandatory	0.83	0.45	0.59	155
None	0.33	0.01	0.02	115
Pharma	0.71	0.37	0.48	258
Political	0.71	0.11	0.19	110
Religious	**1.00**	0.31	0.47	13
Rushed	0.83	0.56	0.67	298
Side-effect	0.86	**0.74**	**0.80**	764
Unnecessary	0.78	0.24	0.37	160
Micro avg	0.82	0.48	0.60	2459
Macro avg	0.78	0.33	0.44	2459
Weighted avg	0.79	0.48	0.57	2459
Samples avg	0.53	0.50	0.51	2459

Table 13: Multi-label classification report for the classical implementation. `Micro avg` counts the total true positives, false negatives and false positives in all classes and computes the metrics globally based on these sums. `Macro avg` averages the unweighted mean per label. `Weighted avg` averages the support-weighted mean per label. `Sample avg` calculates metrics for each instance, and finds their average (only meaningful for multi-label classification, where this differs from the functionality `accuracy_score`.

One observes that the precision scores are the highest ones, with a `Micro avg`, `Macro avg` and `Weighted avg` $\approx 80\%$, while recall average scores are between 30% and 50%. This means that, provided that a classifier has predicted a given label, it is correct in $\approx 80\%$ of cases. On the other hand, recall scores show that false negatives are in general more frequent than true positives, so the algorithm tends to assign less labels than it should.

If one checks the support column, one may find a plausible explanation for this. Variable `Side-effect` is by far the most widespread label (in the test set, but since train/test is random, percentages are similar in the train set), appearing with at least twice the frequency of the other labels. This is also the label with highest F1 score (harmonic mean of precision and recall). The rest of labels have significantly fewer observations and thus the classifier lacks the information to fit a more accurate model. For these labels, scores are rather variable. For instance, regarding variable `Religious`, with only 13 test instances, precision is 100%, and recall equals $31\% \approx 100 \cdot 4/13$, which means that only 4 of the 2459 test observations were classified into the this category, all of them correctly, but 9 more should have also been classified into this class.

In general, this classifier would display a better performance if more labels were

assigned to each instance, which would increase the number of training observations for each binary classification problem, enhancing the fit of the model.

Quantum implementation

In this case, given the computational times for quantum implementations in the previous sections, we shall reduce the binary problems to a relatively small number of variables. To that end, we first remove the words (variables) that among all tweets appear less than a certain threshold (in this case 20, since these words would be considered irrelevant) and then apply a PCA which reduces the number of variables to 10 principal components. The last step is then to solve each binarized problem as in Section 3.2.1. We use the optimal configuration in terms of computational efficiency, i.e., `ZFeatureMap` feature map and `FidelityStatevectorKernel` kernel. The time required for this computation, whose scores are shown in Table 14, was 10668 s.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Conspiracy	nan	0.00	nan	91
Country	nan	0.00	nan	39
Ineffective	nan	0.00	nan	342
Ingredients	nan	0.00	nan	114
Mandatory	nan	0.00	nan	155
None	nan	0.00	nan	115
Pharma	nan	0.00	nan	258
Political	nan	0.00	nan	110
Religious	nan	0.00	nan	13
Rushed	nan	0.00	nan	298
Side-effect	**0.52**	**0.35**	**0.42**	764
Unnecessary	nan	0.00	nan	160
Micro avg	0.52	0.11	0.18	2459
Macro avg	0.52	0.03	0.42	2459
Weighted avg	0.52	0.11	0.42	2459
Samples avg	0.52	0.12	0.95	2459

Table 14: Multi-label classification report for the quantum implementation.

The results are extremely unsatisfactory. First of all, the `nan` in precision come from zero divisions that correspond to all the binary problems such that no label was predicted. Therefore, neither true positives nor false positives were obtained, leading to a "0/0" indetermination. These `nan` values subsequently extended to the F1 score. Most predictions come as false negatives, so the denominator for recalls is non-zero, and the absence of true positives leads to zero recall. The only exception to this comes from variable `Side-effect`, which, as discussed earlier, counts on the most substantial support.

These results can be regarded as an extreme polarization of those obtained classically. The reason behind this could lie on the considerable loss of information that

the variable reduction through PCA implies. The prior step of removing words with less frequency than a certain threshold plays no role since it was removed and results did not change. In this case, the joint effect of scarce support and applying a PCA to drastically reduce (by a factor of 1600) the number of variables accounts for the triviality of the predictions.

In order to enhance the performance of this technique, one could consider more principal components but is limited by classical computation resources. In a proper quantum computer, using amplitude encoding, one could need 14 qubits to encode approximately all the variables ($2^{14} \approx 16000$).

Both in the classical and quantum scenarios, first-order algorithms were applied, which, in an informal way, are the natural generalizations of binary SVM to the multi-label frame. Other second or high-order algorithms, that also account for the correlation structure of interdependence of the labels, are bound to outperform the ones discussed in this section (although some like the k-labelsets algorithm may not be applicable in this context). Nonetheless, starting from the most simple or even naive approaches paves the way for a deeper understanding of the nature of the problem, as well as the need to implement more sophisticated alternatives.

4 LAMDA for binary classification

This section is devoted to the second classifier that this work aims to study: Learning Algorithm for Multivariate Data Analysis (LAMDA). Regarding this algorithm, binary classification will be explored, both classically, implementing techniques that are presented in the literature [3], and in the quantum scenario, where a quantum simulation-based alternative is proposed.

4.1 Classical implementation

LAMDA is an incremental conceptual clustering method based on fuzzy logic, which can be applied in the processes of classification and recognition of concepts (classes). LAMDA has the following features [3]:

- It does not require beforehand knowledge on the number of classes.
- Descriptors can be qualitative, quantitative or a combination of both.
- LAMDA can use a supervised learning stage followed by an unsupervised one.
- Classification and recognition of concepts are based on the maximum adequacy (MA) rule.
- This methodology has the possibility to control the selectivity of the classification (exigency level) through the parameter α.
- LAMDA models the concept of maximum entropy (homogeneity). This concept is represented by a class denominated Non-Informative Class (NIC). The NIC concept plays the role of a threshold of decision, in the classification process.

In this work, we shall consider a variant of LAMDA that sticks to the train and test methodology instead of continous learning. Given that this work will only apply LAMDA for binary classification, the number of classes will be known beforehand and NIC will be omitted.

In informal terms, LAMDA seeks to classify each observation into a class by defining certain scores that measure the degree to which, given this observation, each of its descriptor values corresponds to each class. These scores are called marginal adequacy degrees (MADs) and are computed through a fuzzy logic distribution. However, there are as many of these scores as descriptors for each class. In order to condense all this information into a single score, the global adequacy degree (GAD) is defined as a function of MADs. This function is usually a pondered combination of a T-norm and an S-conorm, which respectively generalize the conjunctive intersection ('AND') operator and the disjunctive ('OR') operator to multi-valued logic. This

weight is given by a coefficient called exigency level (α), which shall be taken as hyper-parameter. Once the GAD of an observation to each class is computed, the classification follows from taking the class whose GAD is the highest.

More in detail, the line of work that will be considered can be summarized in the following steps.

1. Let us first define the notation that we shall consider hereinafter.

   - $I = \{1, \ldots, n\}$. Set of instances.
   - $J = \{1, \ldots, p\}$. Set of descriptors.
   - $C = \{1, 2\}$. Classes.
   - x_i: Instance i (a row of the data frame).
   - x_i^j: value of descriptor $j(\in J)$ for instance $i(\in I)$.
   - $I_c = \{i \in I : x_i \in c, \quad c \in C\}$: subset of instances that belong to a given class.

2. Perform the train-test split.

3. For the training set:

   (a) Standardize the descriptors so that they belong to the interval $[0, 1]$.

   $$x_i^j \to \frac{x_i^j - \min\limits_{i \in I} x_i^j}{\max\limits_{i \in I} x_i^j - \min\limits_{i \in I} x_i^j}$$

   (b) Compute the average descriptor matrix, that is, a matrix M_{av} of size 2 x p such that each entry represents the average value of each descriptor corresponding to each class.

   $$M_{av}(c, j) = \frac{1}{|I_c|} \sum_{i \in I_c} x_i^j \tag{54}$$

   (c) Compute the MAD to each class for each descriptor of each instance. To that end, the fuzzy binomial distribution has been considered. Given an instance $i \in I$, a class $c \in C$, and denoting $\rho_c^j \equiv M_{av}(c, j)$

$$MAD(x_i^j, c) = (\rho_c^j)^{x_i^j}(1-\rho_c^j)^{(1-x_i^j)} \tag{55}$$

It may be informative on the behaviour of the MAD to note that, for a fixed ρ, the MAD is an increasing, decreasing or constant function of x for $\rho > 1/2$, $\rho < 1/2$, and $\rho = 1/2$ respectively.

(d) For each value of (the here hyper-parameter) exigency level α, compute the GAD of each instance for each class. Each observation is finally classified into the category with regard to which its GAD is the highest. Once this is done for all the observations, one can compare the accuracy and F1 score of the training predictions with the true value, and choose the α value that returns the highest score (in this case, the criterion is based on the F1 score since it takes into account both precision and recall). This parameter will be the one taken in the testing phase.

In order to compute the GAD of each instance $i \in I$ to a class $c \in C$, three (weighted) combinations of a T-norm and an S-conorm are proposed.

Firstly, the min-max approach:

$$GAD(i,c) = \alpha \min_{j\in J} MAD(x_i^j, c) + (1-\alpha) \max_{j\in J} MAD(x_i^j, c). \tag{56}$$

Secondly, the product approach:

$$GAD(i,c) = \alpha \prod_{j\in J} MAD(x_i^j, c) + (1-\alpha)\left(1 - \prod_{j\in J} MAD(x_i^j, c)\right). \tag{57}$$

Thirdly, the Lukasiewicz approach:

$$\begin{aligned} GAD(i,c) = \alpha \max\left(1 - n + \sum_{j\in J} MAD(x_i^j, c), 0\right) + \\ + (1-\alpha)\min\left(\sum_{j\in J} MAD(x_i^j, c), 1\right). \end{aligned} \tag{58}$$

The term weighted by α is the T-norm while the one weighted by $(1-\alpha)$ corresponds to the S-conorm. For the testing step, the combination (out of the three that have been presented) that provides the higher F1 score, together with the exigency level α resulting from the corresponding hyper-parameter tuning process will be considered.

4. For the test (validation) set: step 3.a is repeated, and then step 3.c is performed with the same matrix M_{av} computed for the training set. Finally, the GAD is computed for the T-norm, S-conorm and α obtained in step 3.d. Then, the classification of the test set is performed considering the highest GAD for each instance, and the score metrics are returned.

4.1.1 Binary dataset classification

Let us now apply the previous methodology to the binary dataset which was previously studied in Section 3.2.1, with a 0.75/0.25 train-test split, i.e., 426 training instances and 143 testing ones. Table 15 summarizes the training scores for each T-norm and S-conorm combination, obtained from the optimal exigency level α_{opt}.

Approach	$\boldsymbol{\alpha_{\text{opt}}}$	**Precision**	**F1 score**	**Fit time**
min-max	0	0.200	0.277	32.3 s
Product	0	**0.934**	**0.932**	20.9 s
Lukasiewicz	0	0.187	0.373	32.4 s

Table 15: LAMDA training scores for the implemented approaches and train test split 0.75/0.25 on the breast cancer dataset.

Clearly the product approach outperforms the others significantly. One can find an plausible explanation for this in the number of descriptors. For this dataset J is a set of 30 elements, so expression (56) is reducing the information provided by a vector of 30 MADs to a combination of its minimum and maximum, leading to a relevant loss of information. Similarly, expression (58) is considering a T-norm and S-conorm that for large enough n is bound to reduce to $\sum_{j\in J} MAD(x_i^j, c)$ since it would be unlikely that the MADs added up to more than $p-1$. However, a sum does not penalize low MADs as much as the product considered in (57), which stands in an intermediate position between the reduction of a vector to its extreme values as in the min-max approach and the over-conservative Lukasiewicz expression. For the considered number of descriptors, expression (57) provides a suitable balance that takes into consideration the information from each descriptor weighting it in an appropriately adjusted proportion. One could conjecture that for lower values of p, for instance $p \approx 5$, expressions (56) and (58) would lead to more accurate classifications.

It may be insightful to plot the both the true and predicted train target data on the axes of the first two principal components (see Figure 5).

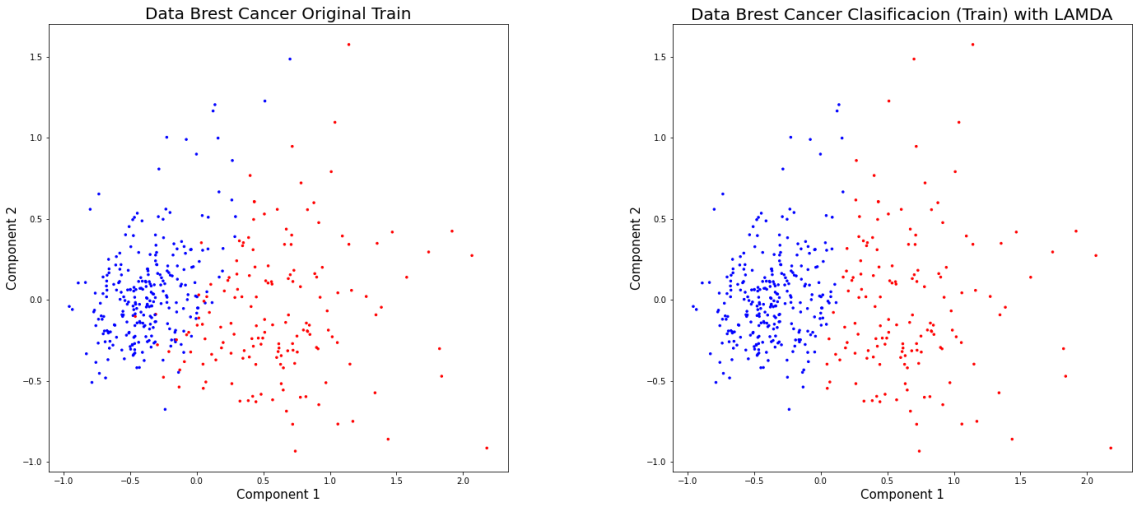


Fig. 5. Left: true train response variable (breast cancer dataset). Right: predicted train response variable. Blue instances stand for benign diagnosis while red ones correspond to malignant diagnosis. This color pattern will be kept hereinafter.

The misclassified observations lay on the border between the benign and malignant observations. LAMDA draws this border very rigidly, while in reality it is slightly dim.

In order to classify the test data, the product approach with $\alpha = 0$ was implemented, rendering the results listed in Table 16 and the plots in figure 6.

Approach	$\boldsymbol{\alpha}$	**Precision**	**F1 score**	**Test time**
Product	0	0.950	0.958	2.1 s

Table 16: (Classical) LAMDA testing scores for the breast cancer dataset.

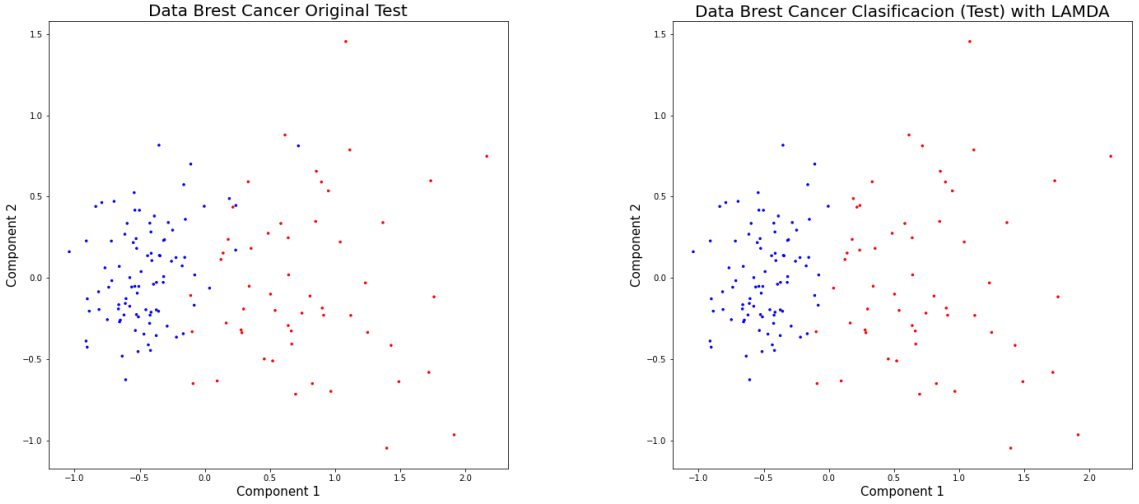


Fig. 6. Left: true test response variable. Right: predicted test response variable. Breast cancer dataset.

Both precision and F1 score are over 0.95, providing a satisfactory classification. As happened in the training phase, figure 6 shows that LAMDA again sets a rigid border between predicted categories.

Let us now repeat the previous implementation on the COVID dataset which was also introduced in Section 3.2.1. This time, given the dataset imbalance, the training phase has been performed on a 2400 instance training set, which is balanced regarding the response variable, while testing has been carried out on a 2000 instance set that keeps the original imbalance. The training scores are summarized in Table 17.

Approach	$\boldsymbol{\alpha_{\text{opt}}}$	**Precision**	**F1 score**	**Fit time**
min-max	0.15	0.314	0.473	52.4 s
Product	0	**0.789**	**0.775**	33.9 s
Lukasiewicz	0	0.250	0.500	54.1 s

Table 17: LAMDA training scores on the COVID dataset.

Compared to Table 15, one observes that scores have become more centralized (product approach scores have shrunk while the rest have increased), but the product approach still outperforms the other two alternatives. The reason for this score centralization lies in the discussion contemplated in Section 3.2.1: we are considering a balanced training set that consists of a maximum of 36 different observations, some of which share the two values of the response variable. Therefore, for each of this observations, the algorithm needs to "take a side" so that training scores are maximized. Even if some approaches are more efficient than others, the general tendency leads to rather centralized scores, given the balance of the training set.

In order to visualize this, one can again plot both the true and predicted train target data on the axes of the first two principal components, which is shown in Figure 7. However, this plot could be uninformative or even misleading, in the sense that, unlike 5, each dot does not represent a single observation, but a set of them, and the color corresponds to the category of the majority of observations corresponding to a given dot.

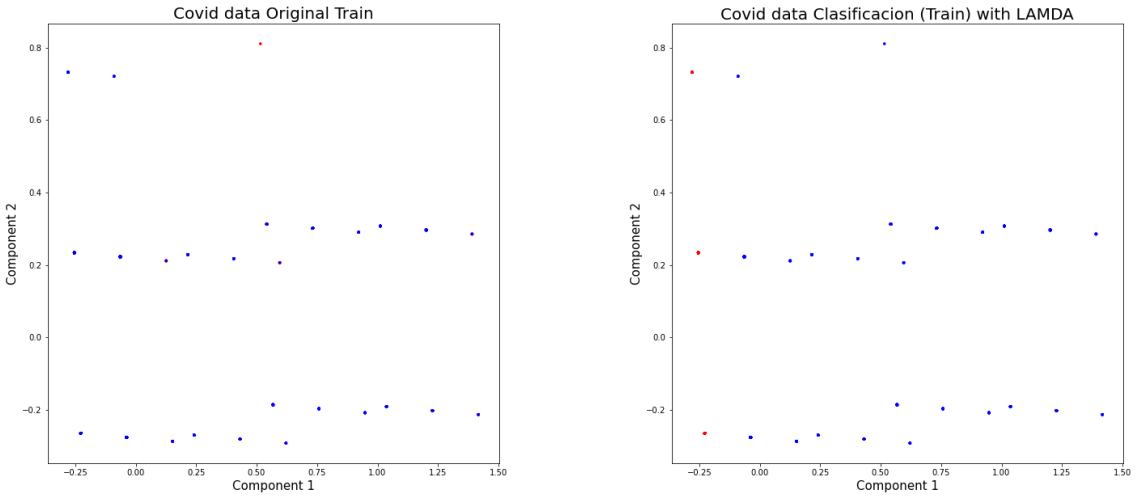


Fig. 7. Left: true train response variable (COVID dataset). Right: predicted train response variable. Blue instances stand for negative diagnosis while red ones correspond to positive diagnosis. This color pattern will be kept hereinafter.

Let us now implement the product approach on the testing set. Since in this case, it is unbalanced, both the `weighted` and `macro` precision and F1 scores will be taken into account (see caption of Figure 13 for the details on these scores). Results are described in Table 18 and the visualization is provided by Figure 8.

Approach	α	**Precision (weighted)**	**Precision (macro)**	**F1 (weighted) score**	**F1 (macro) score**	**Test time**
Product	0	0.942	0.602	0.898	0.631	3.7 s

Table 18: (Classical) LAMDA testing scores for the COVID dataset.

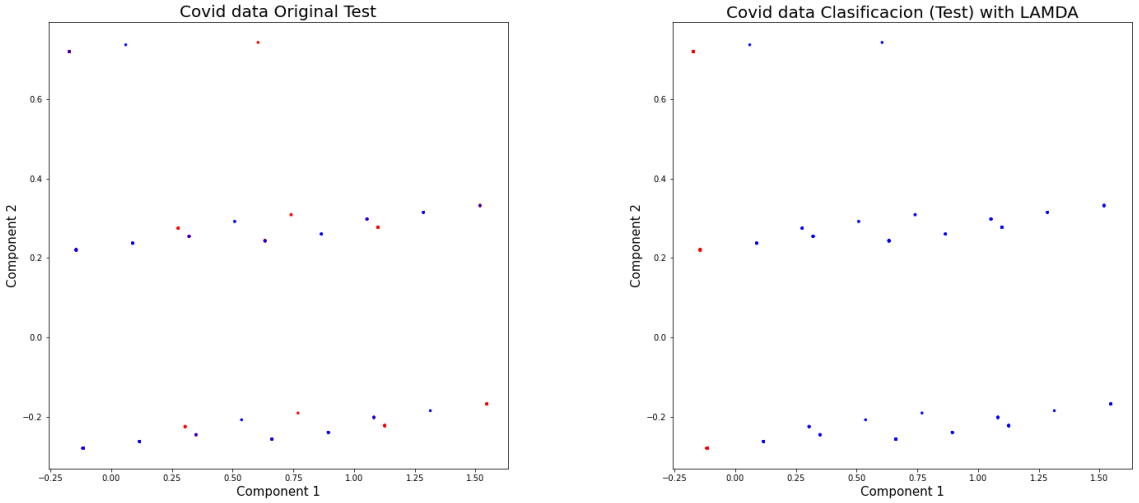


Fig. 8. Left: true test response variable. Right: predicted test response variable. Breast cancer dataset.

Despite the `weighted` scores being promising, `macro` ones are substantially lower, which is accounted for by the imbalance in the testing set, together with the tendency towards classifying instances to the dominant class, in this case, negative. As

happened with the breast cancer dataset, LAMDA draws a rigid border between the two classes in the PCA bidimensional plot.

In the next subsection, we shall propose a quantum approach in order to soften this border via the intrinsic randomness of a quantum algorithm.

4.2 Quantum implementation

The methodology for the proposed quantum implementation substitutes GAD computation for a quantum simulation-based alternative. More in detail, the procedure is identical to the one described in Section 4.1 up to step 3.d (included) for training and up to step 3.c (also included) for testing. In other words, the quantum simulation is performed after obtaining M_{av} for the training set and the MADs for the testing set. This would only cover up to step 3.b for the training set in 4.1, but the value of the optimum hyper-parameter α is highly relevant for the last quantum adjustments, which requires considering steps 3.c and 3.d for training.

Then, on the basis of the testing set MADs, we propose a quantum simulation as an alternative to GAD deterministic computation. In informal terms, one would expect an observation to fit into a given category when the majority of its descriptors show a higher degree of similarity with those assigned to the category. Since this study focuses on binary classification, and MADs are already normalized in the interval $[0, 1]$, one could seek to seize the synergy between binary classification and quantum qubit measurement, which would appear as a suitable generalization of fuzzy logic in the quantum frame.

As a first consideration, which is of the utmost importance, one needs to establish a link between MADs and classes. That is, when α is close to 0, then it is the S-conorm that defines the GAD. On the contrary, if α is close to 1, then it is the T-norm. In addition, T-norms ("intersections") are smaller than S-conorms ("unions"). Thus, when α is close to 0, smaller MADs lead to higher GADs and vice versa, so when comparing the MADs of an observation with regard to a given class, smaller MADs represent a higher degree of similarity to that class than higher ones, despite what one might have expected from the etymology of the acronym MAD (although this could be fixed by introducing a "dual" MAD, for instance $1 - MAD$). On the other hand, if α is close to 1, then higher MADs represent a higher degree of similarity to the given class. This highlights the importance of performing hyper-parameter tuning with α, which is bound to return an optimal value close to an extreme of the interval $[0, 1]$.

Let us suppose, without loss of generality, that the optimal exigency level is close to zero (although the formal condition is < 0.5), as happened in 4.1.1. Then, if for a given observation, we consider the two MAD vectors that result from joining the MADs of all descriptors for each class, then we could encode them in a quantum circuit and carry out a certain number M of measurements. Then, for each descriptor, one can compare the number of projections on $|0\rangle$ for each of the two classes. If both values coincide, then the descriptor is dismissed. Otherwise, the category

corresponding to the highest number of $|0\rangle$ projections is taken as a marginal class for that descriptor. Finally, one classifies the observation into the category corresponding to the class that has been pointed as the marginal class the majority of times.

More concisely, the algorithm steps are the following:

1. If $\alpha_{opt} < 0.5$, then perform the steps 2-6 as presented. Otherwise, perform them swapping $|0\rangle$ and $|1\rangle$.

2. For each instance (steps 2-6 are apply to each individual instance), concatenate the MADs of all the descriptors into a vector for each class, thus obtaining two vectors of length equal to the number of descriptors p.

3. Encode these two vectors into two separate quantum circuits of p qubits via angle encoding, in this case, through RY gates (rotation on the Y axis). The rotation matrix is given by:

$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

   on the $\{|0\rangle, |1\rangle\}$ basis (see Section 2.6.3). Since when initializing a quantum circuit in Qiskit, all qubits are started as $|0\rangle$, the encoding angle is given by $\theta = 2 \arcsin \sqrt{\mu}$, where μ is the corresponding entry of the MAD vector (recall that the measurement on $|1\rangle$ has probability $\sin^2\left(\frac{\theta}{2}\right)$, which accounts for the square root).

4. Simulate a certain number of measurements (in this case 120, although executions of 70-200 rendered similar or the same results) for both vectors.

5. Count the number of $|0\rangle$ projections for each vector and each descriptor, and create two logical vectors. The first will have a "True" entry in the position of those descriptors where the vector for the first class had strictly more $|0\rangle$ projections than the one for the second class. The second logical vector is obtained analogously swapping the classes.

6. Compare the number of "True" entries of both vectors. Classify the instance in the class whose logical vector has more "True" entries. In case of a tie, count the overall number of $|0\rangle$ projections of both MAD vectors and classify the instance in the class that has more (in the highly unlikely case of another tie, classify into any of the classes).

It is relevant to note that simulating a moderate number of times (100) allows for introducing a certain degree of randomness when classifying instances that lie on

the edge between the two classes, and may prevent a misclassification coming from an over-conservative strategy.

4.2.1 Results

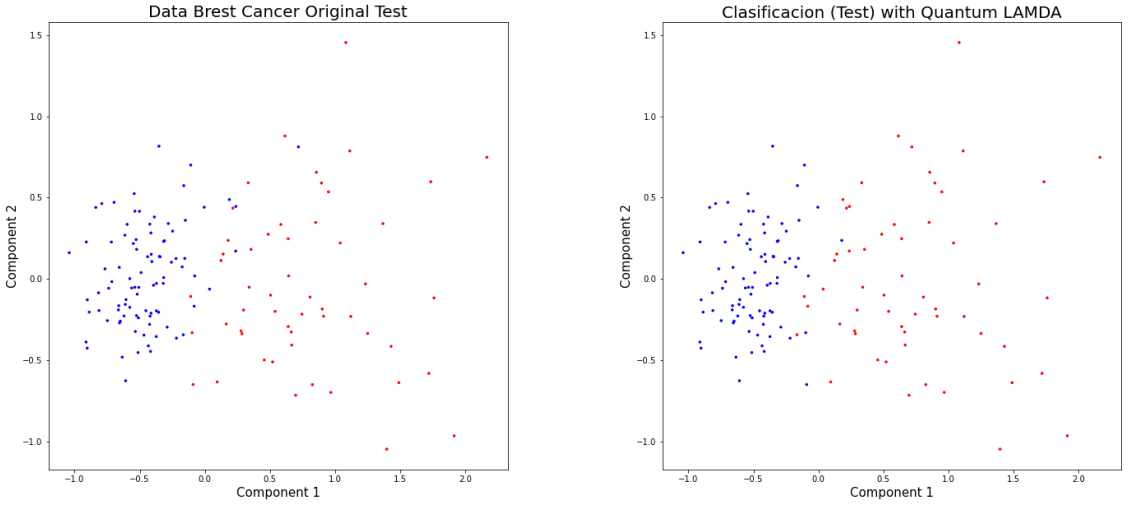
Let us present the results obtained for the quantum approach. Table 19 summarizes five executions of the program.

MAD approach	α	Precision	F1 score	Number of simulations	Test time
Product	0	0.913	0.924	200	2847 s
Product	0	**0.938**	**0.944**	130	4093 s
Product	0	**0.938**	**0.944**	120	3598 s
Product	0	0.921	0.930	100	2754 s
Product	0	0.921	0.930	70	3761 s

Table 19: Quantum LAMDA testing scores for the breast cancer dataset.

It is relevant to highlight that quantum simulations, unlike classical ones, are not reproducible given the proven intrinsic random nature of the quantum processes, this is why five simulations are presented. Several more were run, always displaying scores in the interval [0.92, 0.97] and execution times between 2700s and 4100s.

Graphically, Figure 9 allows for comparing the true and predicted test response variables with the quantum implementation for the extreme (70 and 200) and one of the intermediate (120) number of executions, which cover all the combinations of obtained scores.



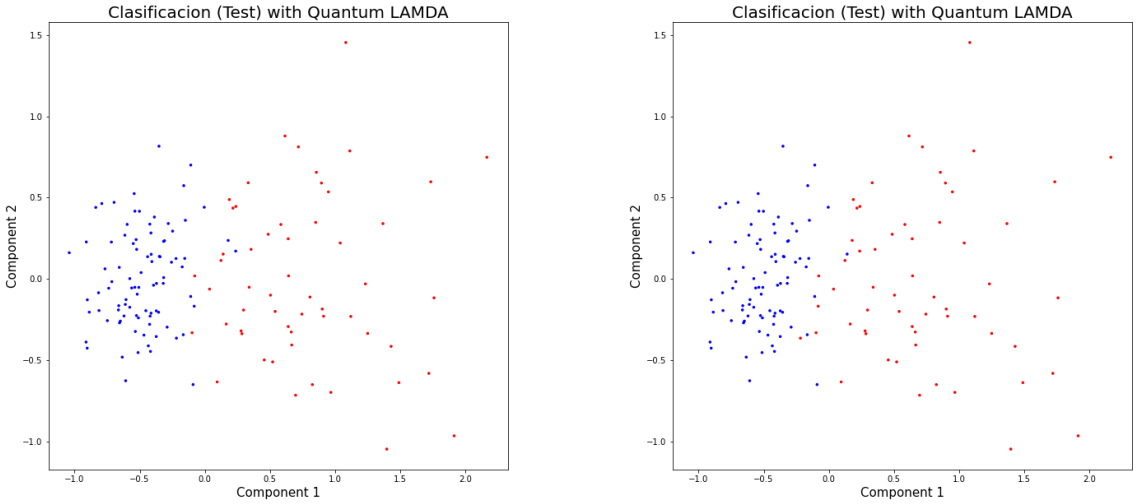


Fig. 9. (1,1) True test response variable. (1,2) Predicted test response variable with quantum implementation of LAMDA for 70 simulations per instance. (2,1) Predicted test response variable with quantum implementation of LAMDA for 120 simulations per instance. (2,2) Predicted test response variable with quantum implementation of LAMDA for 200 simulations per instance. Breast cancer dataset.

We observe how quantum simulations succeed in classifying instances that, informally, are slightly immersed in the dot cloud corresponding to the opposite class. Despite rendering slightly lower (1-2 %) scores, this quantum technique succeeds in dimming the boundary between two classes and could be of advantage when the plot of the true response variables is further from displaying a rigid borderline.

Let us now apply the quantum implementation to the COVID dataset. Results are summarized in Table 20 and plotted in Figure 10.

MAD approach	α	**Precision**		**F1 score**		**Number of simulations**	**Test time**
		Weighted	**Macro**	**Weighted**	**Macro**		
Product	0	0.926	0.590	0.916	0.606	200	47.1 s
Product	0	0.925	0.584	0.914	0.598	130	45.3 s
Product	0	0.920	0.565	0.911	0.576	120	45.6 s
Product	0	0.927	0.590	0.915	0.607	100	45.2 s
Product	0	**0.929**	**0.599**	**0.918**	**0.617**	70	44.5 s

Table 20: Quantum LAMDA testing scores for the COVID dataset.

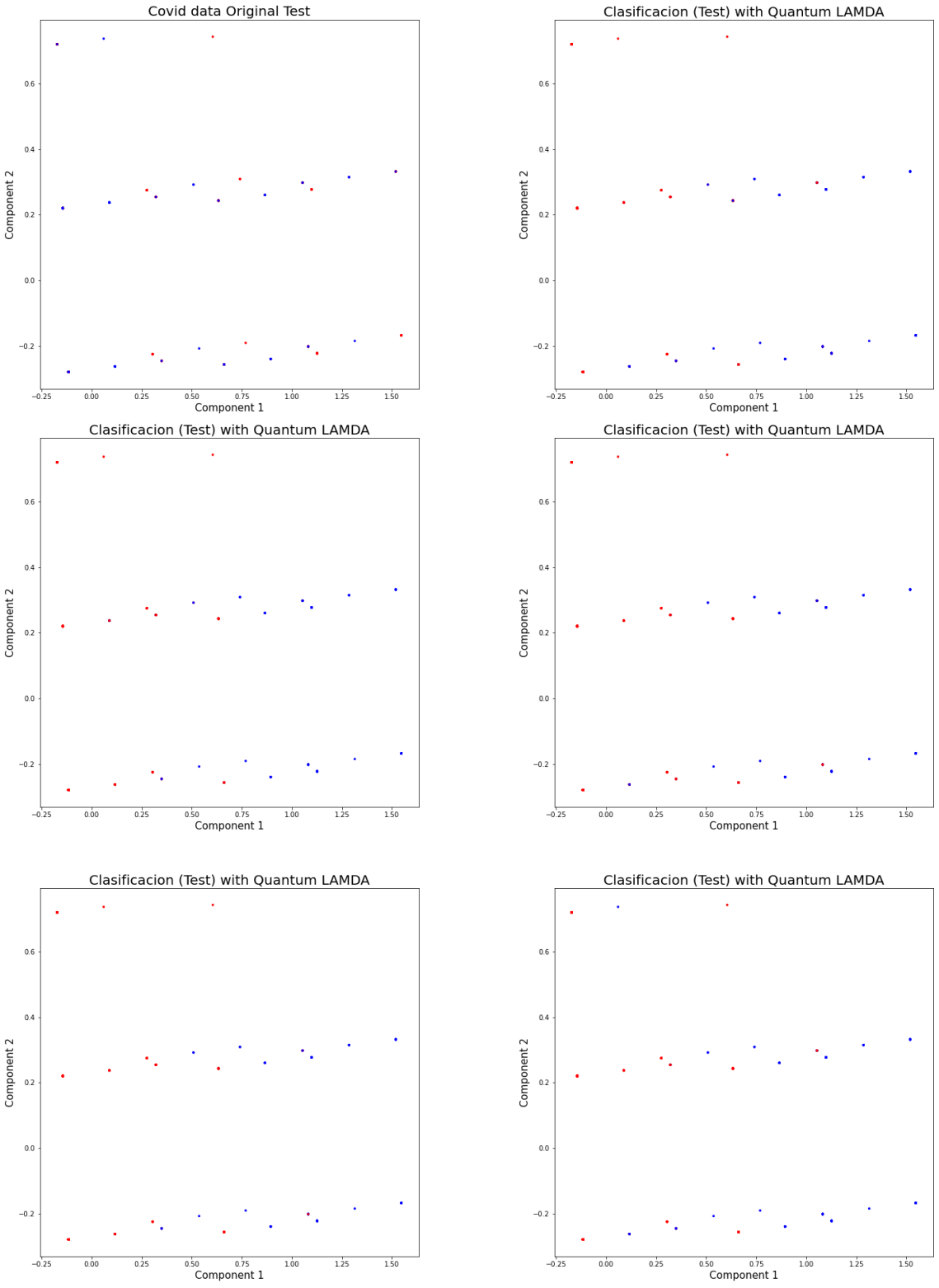


Fig. 10. (1,1) True test response variable. (1,2) Predicted test response variable with quantum implementation of LAMDA for 70 simulations per instance. (2,1) Predicted test response variable with quantum implementation of LAMDA for 100 simulations per instance. (2,2) Predicted test response variable with quantum implementation of LAMDA for 120 simulations per instance. (3,1) Predicted test response variable with quantum implementation of LAMDA for 130 simulations per instance. (3,2) Predicted test response variable with quantum implementation of LAMDA for 200 simulations per instance. COVID dataset.

One observes that the quantum approach renders similar scores for all the implemented number of simulations. Again, the same difference between `macro` and `weighted` scores becomes clear. Compared to the classical implementation (see 18), scores are similar, with the quantum method displaying a slightly higher F1 score but a slightly lower precision. Analogously to what happened with the breast cancer dataset, the quantum implementation dims the boundary between the two classes.

Regarding computational efficiency, it is true that this quantum implementation is about three orders of magnitude slower than the classical GAD one for the breast cancer dataset, but it is relevant to highlight again the computational hindrance of simulating quantum phenomena on classical software. On the other hand, this factor between computational times is reduced to one order of magnitude for the COVID dataset, since quantum circuits only need to encode vectors of four elements, instead of thirty, as happened with the breast cancer dataset.

5 SVM and LAMDA comparison on binary classification

Since the same datasets have been used both in Section 3.2.1 and Section 4, it is consistent to compare both SVM and LAMDA binary classifiers through the results discussed in these sections.

In the light of Tables 1-8 on the SVM side and Tables 16-20 on the side of LAMDA, one observes that SVM is about three orders of magnitude faster than LAMDA, both in the classical and quantum frames, respectively. The only exception to this lies in the similar computational time for quantum SVM and quantum LAMDA for the COVID dataset (see Tables 5-8 and Tables 17, 18, respectively). However, a significant part of the code used for SVM is internally programmed in Python libraries, which enhances the computational efficiency in contrast to the rather "manual" programming of LAMDA, even if computations are less sophisticated in the latter, in the sense that they imply performing elementary operations (sums, products, powers, min, max, etc).

For each training instance, LAMDA requires the computation of MAD for all descriptors and classes (in this case two), which scales the number of operations by a factor of twice the number of descriptors, plus two extra computations for computing the GAD for each class. Denoting the number of descriptors by d, this makes a total of $2(d+1)$ computations (which include two powers and a multiplication) per instance, and this is all scaled by the number of parameters tested in hyper-parameter tuning, in this case, 21 $(0, 0.05, \ldots, 1)$. Then, the previous total is updated to $42(d+1)$ computations per instance, in the training procedure implemented in this work, described in Section 4.1. On the other hand, SVM (see eq.42) implies solving a quadratic optimization problem with as many constraints as training instances, which is efficiently implemented internally, even if this QP solvers imply more sophisticated operations like computing gradients or Hessians. In any

case, these would concern polynomial functions, not sophisticated compositions of functions that require of subsequently demanding approximation algorithms.

Regarding the classical-quantum comparison, it is relevant again to outline the fact that all computational resources required to simulate quantum circuits, as well as computations on classical software, substantially increase computational time with respect to a quantum computer. In the latter, the number of operations is exponential with respect to the number of qubits, so memory limitations would appear on classical software for significantly smaller problems than in the quantum computing scenario.

As for performance, both classical and quantum SVM (see Tables 1 and 2) outperform LAMDA (see Tables 16 and 19) in terms of metrics for the breast cancer dataset. Some of the SVM configurations surpass 97.5% accuracy, while classical LAMDA, which displays the highest scores, closely followed by quantum LAMDA, reaches only 95%, although this is also a satisfactory result. In addition, while LAMDA uses all the variables (30), SVM predicts with higher accuracy with 20, 10 or even 5 principal components.

Regarding the COVID dataset, which is diametrically opposed to the breast cancer one in terms of number and type of variables, both SVM and LAMDA render lower scores, which can be accounted for by the bounded number (36) of different instances, as discussed in Section 3.2.1. In this case, the highest accuracy and `weighted` F1 scores come from quantum SVM (see Tables 5 and 6), reaching 95% accuracy and 0.947 F1 score. Despite the fact that the metrics displayed by LAMDA almost reach these scores as for precision, F1 scores slightly lag behind in both the classical and quantum implementations (see Tables 20 and 18), although quantum provides more satisfactory score. In general, for the COVID datasets, the differences in performance are slighter, given that the constrained number of different observations limited the performance of the algorithms.

In general, one could conclude that in terms of performance, both SVM and LAMDA render satisfactory results. However, SVM outperforms LAMDA in terms of computational efficiency and scores, even if LAMDA is a more accessible and intuitive alternative for manual programming.

6 Conclusions

The general objective of this Master Thesis consists of showing the usefulness of SVM and LAMDA for classification problems (in the case of LAMDA for binary ones) both from a classical and quantum standpoint. It can be stated that this purpose has been complied with, through the specific objectives presented in Section 1.1.

After the revision of quantum physics fundamentals in Section 2, the first of the specific objectives is approached in Sections 3 and 4. In Section 3.1, the theoretical basis behind both classical and quantum SVM is presented, and in Section 3.2, for

each type of classification problem (binary, multi-class and multi-label), classical and quantum results are compared in terms of performance and efficiency. Analogously, this analysis is repeated in Section 4 for binary classification problems with LAMDA. Concurrently, these sections pave the way for the explicit comparison between binary SVM and LAMDA, in both classical and quantum frames, performed in Section 5, with the subsequent attainment of the second specific objective.

Regarding the first of the specific objectives, quantum and classical algorithms have performed similarly both in SVM and LAMDA (except for multi-label SVM), with slight ($\approx 0.01 - 0.02$ differences) in accuracy or F1 scores. An exception to this tendency can be observed in multi-class SVM (see Section 3.2.2, where the quantum results for 5 principal components rendered 0.1 higher metrics. Regarding multi-label SVM, despite the intrinsic hindrance of most labels being scarcely represented, with subsequent setbacks for model fitting, the binary relevance approach managed to provide moderately acceptable results. However, the transition to the quantum frame took classification to the dominant category to the extreme in binarized problems, invalidating this quantum approach for a dataset with such characteristics. In terms of efficiency, as discussed in several occasions, quantum simulations usually required computational times that were about three orders of magnitude above classical ones. However, it is highly relevant to highlight the inefficiency derived from artificially simulating quantum circuits on classical software, and the fact that the intrinsic architecture of a quantum computer would lead to considerably faster computations.

As for the second specific objective, given the opposed nature of each binary dataset, one can conclude that SVM is more computationally efficient than LAMDA and a more suitable alternative in terms of scores for higher dimensional datasets, even if both methods render satisfactory predictions. For smaller datasets, this differences shrink, even if LAMDA is slightly outperformed by SVM. Nevertheless, as discussed in Section 5, SVM is more efficiently programmed internally, and LAMDA counts on the advantage of admitting straightforward parallelization, which could significantly enhance computational efficiency.

Finally, based on the conclusions of this Master Thesis, several potential avenues for future study emerge. Firstly, optimizing the computational efficiency of LAMDA through procedures such as parallel programming. Secondly, seeking an alternative to the multi-label quantum approach proposed in Section 3.2.3. Finally, extending LAMDA to multi-class and multi-label classification and comparing its performance to SVM's, in the line of this work, or including additional classifiers in this comparison.

References

[1] Pramila P. Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, 2018.

[2] Davide Anguita, Alessandro Ghio, Noemi Greco, Luca Oneto, and Sandro Ridella. Model selection for support vector machines: Advantages and disadvantages of the machine learning theory. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[3] Juan G., E. Guzman-Ramirez, and Oleksiy Pogrebnyak. Search Algorithm for Image Recognition Based on Learning Algorithm for Multivariate Data Analysis. In Taufik Abro, editor, *Search Algorithms for Engineering Optimization*. InTech, February 2013.

[4] Ronald de Wolf. The potential impact of quantum computers on society. *Ethics and Information Technology*, 19(4):271–276, December 2017.

[5] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Cambridge University Press, 3 edition, 2018.

[6] R. Shankar. *Principles of Quantum Mechanics*. Plenum Press, New York, 2nd edition, 1994.

[7] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, 1935.

[8] D. Bohm. *Quantum Theory*. Prentice-Hall, Englewood Cliffs, 1951.

[9] D. Bohm and Y. Aharonov. Discussion of Experimental Proof for the Paradox of Einstein, Rosen, and Podolsky. *Phys. Rev.*, 108(4):1070–1076, 1957.

[10] J. S. Bell. On the Einstein Podolsky Rosen paradox. *Phys. Phys. Fiz.*, 1(3):195–290, 1964.

[11] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880–884, 1969.

[12] M. Schlosshauer. *Decoherence and the Quantum to Classical Transition*. Springer, 2007.

[13] A. Furusawa and P. Van Loock. *Quantum Teleportation and Entanglement: A Hybrid Approach to Optical Quantum Information Processing*. Wiley, 1st edition, 2011.

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

[15] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[16] Vladimir N. Vapnik. The support vector method. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks — ICANN'97*, pages 261–271, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[17] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, September 1995.

[18] Nello Cristianini. An introduction to support vector machines : and other kernel-based learning methods.

[19] John Shawe-Taylor and Nello Cristianini. Kernel Methods for Pattern Analysis.

[20] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. Learning from data, e-chapter 8: Support vector machines. Online, 2012. Accessed: 2024-06-17.

[21] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[22] Min-Ling Zhang and Zhi-Hua Zhou. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, August 2014.

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

[24] Elías F. Combarro and Samuel González-Castillo. *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms*. Packt Publishing Limited, Birmingham, 2023.

[25] Maria Schuld. Supervised quantum machine learning models are kernel methods, 2021.

[26] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, Cham, 2021.

[27] Mangasarian-Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

Appendix A. Confusion matrices

In this section the confusion matrices corresponding to the binary and multi-class datasets considered for SVM throughout this work will be listed. Note that the following matrices are normalized by the length of the corresponding classifier vectors, and that rows indicate the true categories while columns denote the predicted ones.

The confusion matrices are presented for each number of principal components n, and each number of training observations (which has an associated number of testing observations corresponding to 1/2 of the training ones). For each given scenario (6 tables, presented as a 2 x 3 frame) , the matrices appear in the following order:

1. Linear SVM. Position (1,1).

2. Gaussian kernel SVM. Position (1,2).

3. Quantum SVM with `ZFeatureMap` feature map and `FidelityStatevectorKernel`. Position (1,3).

4. Quantum SVM with `ZZFeatureMap` feature map and `FidelityStatevectorKernel`. Position (2,1).

5. Quantum SVM with `ZFeatureMap` feature map and `FidelityQuantumKernel`. Position (2,2).

6. Quantum SVM with `ZZFeatureMap` feature map and `FidelityQuantumKernel`. Position (2,3).

A.1. Breast Cancer dataset

Note that in this case, given that the proportions between instances are approximately 2:1 for benign:malignant, the train and test sets are built to respect this proportionality (which still leads to a quite balanced distribution). The same would happen if the sample were large enough and completely random subsamples were taken. However, we lack this high number of observations, so in order to prevent sheer imbalance cases, these proportions have been fixed, so the columns of the following matrices add up to 2/3 and 1/3 respectively. To obtain the number of correctly predicted instances, one could also normalize each row by the column sums, that is, divide the first row by 2/3 and the second one by 1/3. Nonetheless, for the sake of consistency, the same normalization (the one described at the beginning of the section) has been implemented for all the datasets.

n = 2

- **30 training observations**

	B	M
B	0.600	0.067
M	0	0.333

	B	M
B	0.533	0.133
M	0	0.333

	B	M
B	0.467	0.200
M	0	0.333

	B	M
B	0.400	0.267
M	0.267	0.067

	B	M
B	0.467	0.200
M	0	0.333

	B	M
B	0.400	0.267
M	0.267	0.067

- **90 training observations**

	B	M
B	0.667	0
M	0.067	0.267

	B	M
B	0.644	0.022
M	0.044	0.289

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	0.644	0.022
M	0.155	0.178

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	0.644	0.022
M	0.155	0.178

- **180 training observations**

	B	M
B	0.633	0.033
M	0.044	0.289

	B	M
B	0.644	0.022
M	0.044	0.289

	B	M
B	0.644	0.022
M	0.044	0.289

	B	M
B	0.589	0.078
M	0.044	0.289

	B	M
B	0.644	0.022
M	0.044	0.289

	B	M
B	0.589	0.078
M	0.044	0.289

n = 5

- **30 training observations**

	B	M
B	0.600	0.067
M	0	0.333

	B	M
B	0.600	0.067
M	0	0.333

	B	M
B	0.533	0.133
M	0	0.333

	B	M
B	0.666	0
M	0.333	0

	B	M
B	0.533	0.133
M	0	0.333

	B	M
B	0.666	0
M	0.333	0

- **90 training observations**

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	0.667	0
M	0.022	0.311

	B	M
B	0.667	0
M	0.244	0.089

	B	M
B	0.667	0
M	0.022	0.311

	B	M
B	0.667	0
M	0.244	0.089

- **180 training observations**

	B	M
B	0.633	0.033
M	0.033	0.300

	B	M
B	0.633	0.033
M	0.011	0.322

	B	M
B	0.633	0.033
M	0.022	0.311

	B	M
B	0.656	0.011
M	0.189	0.144

	B	M
B	0.633	0.033
M	0.022	0.311

	B	M
B	0.656	0.011
M	0.189	0.144

n = 10

- **30 training observations**

	B	M
B	0.600	0.067
M	0	0.333

	B	M
B	0.667	0
M	0	0.333

	B	M
B	0.533	0.133
M	0.067	0.267

	B	M
B	0.667	0
M	0.333	0

	B	M
B	0.533	0.133
M	0.067	0.267

	B	M
B	0.667	0
M	0.333	0

- **90 training observations**

	B	M
B	0.667	0
M	0.022	0.311

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	0.667	0
M	0.022	0.311

	B	M
B	0.667	0
M	0.333	0

	B	M
B	0.667	0
M	0.022	0.311

	B	M
B	0.667	0
M	0.333	0

- **180 training observations**

	B	M
B	0.644	0.022
M	0.022	0.311

	B	M
B	0.633	0.033
M	0	0.333

	B	M
B	0.633	0.033
M	0.033	0.300

	B	M
B	0.667	0
M	0.333	0

	B	M
B	0.633	0.033
M	0.033	0.300

	B	M
B	-	-
M	-	-

n = 20

- **30 training observations**

	B	M
B	0.667	0
M	0	0.333

	B	M
B	0.667	0
M	0.067	0.267

	B	M
B	0.600	0.067
M	0.067	0.267

	B	M
B	0.667	0
M	0.333	0

	B	M
B	-	-
M	-	-

	B	M
B	-	-
M	-	-

- **90 training observations**

	B	M
B	0.667	0
M	0.022	0.311

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	0.667	0
M	0.044	0.289

	B	M
B	(0.667)	0
M	(0.333)	0

	B	M
B	-	-
M	-	-

	B	M
B	-	-
M	-	-

- **180 training observations**

	B	M
B	0.633	0.033
M	0.022	0.311

	B	M
B	0.633	0.033
M	0.022	0.311

	B	M
B	0.633	0.033
M	0.056	0.278

	B	M
B	-	-
M	-	-

	B	M
B	-	-
M	-	-

	B	M
B	-	-
M	-	-

A.2. Dengue dataset

n = 2

- **30 training observations**

	0	1	2
0	0.333	0	0
1	0	0.067	0.267
2	0	0.067	0.267

	0	1	2
0	0.333	0	0
1	0	0.200	0.133
2	0	0.133	0.200

	0	1	2
0	0.133	0.067	0.133
1	0	0.133	0.200
2	0	0.133	0.200

	0	1	2
0	0.200	0	0.133
1	0	0.267	0.067
2	0	0.200	0.133

	0	1	2
0	0.133	0.067	0.133
1	0	0.133	0.200
2	0	0.133	0.200

	0	1	2
0	0.200	0	0.133
1	0	0.267	0.067
2	0	0.200	0.133

- **300 training observations**

	0	1	2
0	0.333	0	0
1	0.033	0.160	0.140
2	0.060	0.040	0.233

	0	1	2
0	0.333	0	0
1	0.013	0.287	0.033
2	0.060	0.067	0.207

	0	1	2
0	0.333	0	0
1	0.033	0.253	0.047
2	0.087	0.087	0.160

	0	1	2
0	0.327	0	0.007
1	0.060	0.213	0.060
2	0.053	0.100	0.180

	0	1	2
0	0.333	0	0
1	0.033	0.253	0.047
2	0.087	0.087	0.160

	0	1	2
0	0.327	0	0.007
1	0.060	0.213	0.060
2	0.053	0.100	0.180

- **1500 training observations**

	0	1	2
0	0.333	0	0
1	0.015	0.145	0.173
2	0.024	0.039	0.271

	0	1	2
0	0.333	0	0
1	0.011	0.257	0.065
2	0.013	0.085	0.235

	0	1	2
0	0.333	0	0
1	0.013	0.260	0.060
2	0.059	0.095	0.180

	0	1	2
0	0.325	0	0.008
1	0.039	0.232	0.063
2	0.045	0.096	0.192

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **3000 training observations**

	0	1	2
0	0.333	0	0
1	0.014	0.167	0.152
2	0.017	0.029	0.287

	0	1	2
0	0.332	0	0.001
1	0.010	0.250	0.073
2	0.005	0.072	0.257

	0	1	2
0	0.333	0	0.001
1	0.013	0.274	0.046
2	0.048	0.095	0.190

	0	1	2
0	0.328	0	0.005
1	0.036	0.231	0.067
2	0.035	0.087	0.211

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

n = 5

- **30 training observations**

	0	1	2
0	0.333	0	0
1	0	0.200	0.133
2	0	0	0.333

	0	1	2
0	0.333	0	0
1	0	0.267	0.067
2	0	0.133	0.200

	0	1	2
0	0.200	0	0.133
1	0	0.267	0.067
2	0	0.133	0.200

	0	1	2
0	0.067	0.267	0
1	0	0.133	0.200
2	0	0.267	0.067

	0	1	2
0	0.200	0	0.133
1	0	0.267	0.067
2	0	0.133	0.200

	0	1	2
0	0.067	0.267	0
1	0	0.133	0.200
2	0	0.267	0.067

- **300 training observations**

	0	1	2
0	0.333	0	0
1	0.013	0.227	0.093
2	0.053	0.067	0.213

	0	1	2
0	0.333	0	0
1	0.013	0.287	0.033
2	0.047	0.053	0.233

	0	1	2
0	0.327	0	0.007
1	0.013	0.240	0.080
2	0.100	0.053	0.180

	0	1	2
0	0.313	0.013	0.007
1	0.007	0.267	0.006
2	0	0.127	0.207

	0	1	2
0	0.327	0	0.007
1	0.013	0.240	0.080
2	0.100	0.053	0.180

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **1500 training observations**

	0	1	2
0	0.333	0	0
1	0.012	0.180	0.141
2	0.020	0.063	0.251

	0	1	2
0	0.333	0	0
1	0.011	0.261	0.061
2	0.012	0.060	0.261

	0	1	2
0	0.333	0	0
1	0.015	0.273	0.045
2	0.052	0.072	0.209

	0	1	2
0	0.329	0	0.004
1	0.004	0.297	0.032
2	0	0.003	0.331

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **3000 training observations**

	0	1	2
0	0.333	0	0
1	0.014	0.179	0.141
2	0.015	0.069	0.249

	0	1	2
0	0.333	0	0
1	0.013	0.271	0.050
2	0.004	0.053	0.276

	0	1	2
0	0.325	0	0.008
1	0.012	0.261	0.061
2	0.025	0.054	0.255

	0	1	2
0	0.328	0.005	0
1	0.001	0.319	0.013
2	0	0	0.333

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

n = 10

- **30 training observations**

	0	1	2
0	0.333	0	0
1	0	0.267	0.067
2	0	0.067	0.267

	0	1	2
0	0.333	0	0
1	0	0.267	0.067
2	0	0.067	0.267

	0	1	2
0	0.200	0	0.133
1	0	0.200	0.133
2	0.067	0	0.267

	0	1	2
0	0.067	0.267	0
1	0	0.267	0.067
2	0	0.133	0.200

	0	1	2
0	0.200	0	0.133
1	0	0.200	0.133
2	0.067	0	0.267

	0	1	2
0	0.067	0.267	0
1	0	0.267	0.067
2	0	0.133	0.200

- **300 training observations**

	0	1	2
0	0.333	0	0
1	0.013	0.293	0.027
2	0.027	0.067	0.240

	0	1	2
0	0.333	0	0
1	0.013	0.313	0.007
2	0.013	0.047	0.273

	0	1	2
0	0.327	0	0.007
1	0.013	0.300	0.020
2	0.027	0.040	0.267

	0	1	2
0	0.307	0.027	0
1	0	0.333	0
2	0	0.173	0.160

	0	1	2
0	0.327	0	0.007
1	0.013	0.300	0.020
2	0.027	0.040	0.267

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **1500 training observations**

	0	1	2
0	0.333	0	0
1	0.008	0.263	0.063
2	0.008	0.049	0.276

	0	1	2
0	0.333	0	0
1	0.011	0.295	0.028
2	0.004	0.012	0.317

	0	1	2
0	0.333	0	0
1	0.011	0.295	0.028
2	0.004	0.012	0.317

	0	1	2
0	0.328	0.005	0
1	0	0.333	0
2	0	0.003	0.331

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **3000 training observations**

	0	1	2
0	0.333	0	0
1	0.007	0.272	0.054
2	0.004	0.047	0.282

	0	1	2
0	0.333	0	0
1	0.007	0.300	0.026
2	0.001	0.009	0.323

	0	1	2
0	0.333	0	0.001
1	0.001	0.306	0.021
2	0.001	0.005	0.327

	0	1	2
0	0.329	0.004	0
1	0	0.333	0
2	0	0	0.333

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

n = 20

- **30 training observations**

	0	1	2
0	0.333	0	0
1	0	0.267	0.067
2	0	0	0.333

	0	1	2
0	0.333	0	0
1	0	0.267	0.067
2	0	0.067	0.267

	0	1	2
0	0.333	0	0
1	0	0.200	0.133
2	0	0	0.333

	0	1	2
0	0.067	0.267	0
1	0	0.200	0.133
2	0	0.200	0.133

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **300 training observations**

	0	1	2
0	0.333	0	0
1	0.020	0.313	0
2	0	0	0.333

	0	1	2
0	0.333	0	0
1	0.013	0.320	0
2	0	0	0.333

	0	1	2
0	0.333	0	0
1	0.007	0.320	0.007
2	0	0	0.333

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **1500 training observations**

	0	1	2
0	0.333	0	0
1	0.001	0.332	0
2	0	0	0.333

	0	1	2
0	0.333	0	0
1	0	0.332	0.001
2	0	0	0.333

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

- **3000 training observations**

	0	1	2
0	0.333	0	0
1	0.003	0.331	0
2	0	0	0.333

	0	1	2
0	0.333	0	0
1	0	0.333	0
2	0	0	0.333

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

	0	1	2
0	-	-	-
1	-	-	-
2	-	-	-

A.3. COVID dataset

Here 0 means COVID positive and 1 COVID negative. Note that in this case, given that the proportions between instances are approximately 19:1 for negative:positive, the test set was built to respect this proportionality, because after taking most of the positive observations to create a balanced training set, the remaining ones were too scarce to maintain the original proportionality.

n = 4

- **200 training observations**

	0	1
0	0.90	0.05
1	0.03	0.02

	0	1
0	0.92	0.03
1	0.03	0.02

	0	1
0	0.93	0.02
1	0.03	0.02

	0	1
0	0.93	0.02
1	0.03	0.02

	0	1
0	0.93	0.02
1	0.03	0.02

	0	1
0	0.93	0.02
1	0.03	0.02

- **1000 training observations**

	0	1
0	0.888	0.062
1	0.018	0.032

	0	1
0	0.846	0.104
1	0.014	0.036

	0	1
0	0.904	0.046
1	0.022	0.028

	0	1
0	0.876	0.074
1	0.018	0.032

	0	1
0	-	-
1	-	-

	0	1
0	-	-
1	-	-

- **2000 training observations**

	0	1
0	0.881	0.069
1	0.018	0.032

	0	1
0	0.885	0.065
1	0.011	0.039

	0	1
0	0.882	0.068
1	0.012	0.038

	0	1
0	0.851	0.099
1	0.01	0.04

	0	1
0	-	-
1	-	-

	0	1
0	-	-
1	-	-

Appendix B. Codes

Codes for Sections 3 and 4 can be accessed through GitHub. The correspondence between codes and sections is the following.

- `SVM_B` and `SVM_B_2`: Section 3.2.1.
- `SVM_MC`: Section 3.2.2.
- `SVM_MC`: Section 3.2.3.
- `LAMDA_1` and `LAMDA_2`: Section 4.