

UNIVERSITY OF VERONA
DEPARTMENT OF COMPUTER SCIENCE

GRADUATE SCHOOL OF NATURAL SCIENCES AND ENGINEERING
DOCTORAL PROGRAM IN COMPUTER SCIENCE
CYCLE 32

Quantum Approaches to Data Science and Data Analytics




S.S.D. INF/01

Coordinator: _____
Massimo Merro

Tutor: _____
Alessandra Di Pierro

Doctoral Student: _____
Riccardo Mengoni

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License, Italy. To read a copy of the licence, visit the web page:
<http://creativecommons.org/licenses/by-nc-nd/3.0/>

-  **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
-  **NonCommercial** — You may not use the material for commercial purposes.
-  **NoDerivatives** — If you remix, transform, or build upon the material, you may not distribute the modified material.

Algorithms and data structures for indexing, mining, and coding of sequential data — RICCARDO MENGONI
or
Combinatorics and Algorithmics of Sequential Data — Prefix normal words, colored strings, and space-aware encodings — RICCARDO MENGONI
PhD Thesis
Verona, May 14, 2020
ISBN <ISBN>

Contents

1	The Classical Setting	1
1.1	Classical Machine Learning	1
1.1.1	Supervised Learning	2
1.1.2	Unsupervised Learning	5
1.2	Topological Data Analysis	7
1.2.1	Algebraic Topology Background	7
1.2.2	Persistent Homology	10
1.2.3	TDA as a Resource for ML	12
2	Quantum Entanglement	13
2.1	Postulates of Quantum Mechanics	13
2.2	Entanglement for Pure States	14
2.2.1	Schmidt Decomposition	15
2.3	Entanglement for Mixed States	16
2.3.1	PPT Criterion for Separability	18
2.4	Entanglement Monotones	18
2.4.1	Measures of Entanglement	19
2.5	Entanglement Classification	19
3	Quantum Computing	21
3.1	Quantum Circuit Model	21
3.1.1	Quantum Gates	21
3.1.2	Quantum Parallelism	22
3.1.3	Quantum Algorithms	23
3.2	Adiabatic Quantum Computation and Quantum Annealing	26
3.2.1	Adiabatic Quantum Computation	26
3.2.2	Quantum Annealing	27
3.3	Topological Quantum Computation	29
3.3.1	Mathematical Background	29
3.3.2	Kauffman Bracket	30
3.3.3	Braids and Links	31
3.3.4	Computing with Anyons	31
3.3.5	Topological Quantum Computation of the Jones Polynomials	32
4	Quantum Machine Learning	35
4.1	QC: Machine Learning for Quantum Physics	36
4.2	CQ: Quantum Computing for Machine Learning	37
4.2.1	Essential Tools in QML	37

4.2.2	Quantum K-NN and K-Means	39
4.2.3	Quantum SVM	40
4.2.4	Quantum Computation of Hard Kernels	41
4.2.5	ML with a Quantum Annealer	44
5	Homological Analysis of Multi-qubit Entanglement	47
5.1	Creating the Qubit Data Cloud	47
5.1.1	Random State Generation	48
5.1.2	Entangled States Selection	48
5.1.3	Distances Calculation	48
5.2	Entanglement Classification	49
5.2.1	Classification of Three Qubits States	49
5.2.2	Classification of Four Qubits States	51
5.2.3	Classification of Five Qubits States	55
5.2.4	Classification of Six Qubits States	62
5.3	Class Frequencies	66
5.4	Comments	68
6	Quantum Kernel Methods	71
6.1	Quantum Error Correcting Output Codes	71
6.1.1	Error Correcting Output Codes (ECOC) in Classical Machine Learning	72
6.1.2	Strategy for Quantum Error Correcting Output Codes	73
6.1.3	Comments	76
6.2	Hamming Distance Kernelization via TQC	76
6.2.1	Topological Quantum Calculation of Hamming Distance Between Binary Strings	77
6.2.2	Hamming Distance Based Kernel	79
6.2.3	Comments	80
7	Machine Learning with the D-Wave	81
7.1	Quantum Annealing Approach to the Minimum Spanning Tree Problem with Δ -Degree Constraint	81
7.1.1	QUBO Formulation	81
7.1.2	Resources	84
7.1.3	Embedding	87
7.1.4	Experimental Results and Comments	89
7.2	Quantum Annealing Approach to Edit Distance Calculation	91
7.2.1	QUBO Formulation	91
7.2.2	Resources	92
7.2.3	Embedding	93
7.2.4	Experimental Results and Comments	94
8	Machine Learning with IBM Quantum Computer	97
8.1	Dataset and Preprocessing	97
8.2	Encoding Graphs into Quantum States	99
8.2.1	Example	100
8.3	Graphs Quantum Classifier	101
8.4	Experimental Results and Comments	103
	References	107

Introduction

Quantum mechanics is a branch of physics that started in 1900 with the pioneering work of Max Planck, who proposed a solution to the black-body radiation problem, and continued with Albert Einstein's 1905 paper explaining the photoelectric effect. In the 1920s, scientists like Schrödinger, Heisenberg, Born, Dirac and others further contributed to the development of the quantum theory and its formalism, which involves mathematical objects called *wave functions* for describing the state of a quantum system.

Today, quantum mechanics is considered to be the most complete physical theory of nature, which is able to explain phenomena that are characteristic of the atomic and subatomic world, succeeding where classical theories are inadequate.

One key feature of quantum mechanics is the so-called entanglement, i.e. a quantum correlation that cannot be explained by any local classical theory and that nowadays constitute a fundamental resource for many quantum-related tasks.

One of them is Quantum Computing, a well established research field where quantum phenomena like superposition and entanglement are employed to process information. The aim of quantum computing is to devise quantum algorithms which are able to manipulate quantum systems in order to obtain a quantum state representing the solution for a given problem. The power of superposition and entanglement can lead to a computational speed-up with respect to a classical computer. Quantum computers outputs are probabilistic, thus, contrary to classical deterministic algorithms, quantum algorithms may need to be repeated a number of times before getting a result.

There exist several equivalent models of quantum computation like quantum circuit model, adiabatic quantum computing (AQC) model and topological quantum computation (TQC) which work employing different quantum mechanical concepts. In the last decade, a large effort has been put forward by both universities and companies in the realization of a hardware able to perform quantum computation. The two main hardware available to the public are the IBM Q, a general purpose gate model quantum computer, and the D-Wave quantum annealer, i.e. a device specific for solving combinatorial optimization problems.

Simultaneously to the rise of quantum computing, also another research area has gained a lot of popularity in recent years, namely Machine Learning (ML).

We live undoubtedly in the era of big data, where information is collected by the most disparate devices. In this context, ML constitutes a set of techniques for identifying patterns among these huge data-sets and for inferring input-output relations from data in order to interpret previously unknown inputs. Some of the most common ML applications are spam mail filters, iris recognition for security systems, evaluation of consumer behaviour, assessing risks in the financial sector or developing strategies for computer games.

Machine learning tasks are typically classified into three broad categories depending on the nature of the learning process [3]: Supervised, Unsupervised and Reinforcement learning.

Within the supervised category, we mainly focus on classification algorithms like *K-nearest neighbours* and *kernel methods*; the latter are a set of algorithms, such as *support vector machines*, which

employ kernel functions to perform non-linear classifications. Unsupervised algorithms like *k-means* and *minimum spanning tree clustering algorithms* are also discussed in this thesis.

Moreover, a set of techniques known as Topological Data Analysis (TDA) that are able to improve many ML algorithms will be introduced. TDA provides a set of tools to recognize the global structure of multidimensional point clouds. The most powerful tool of TDA is perhaps Persistent Homology, i.e. an algebraic method for computing coarse topological features of a given data cloud that persist over many grouping scales [20, 21].

Recently a new interdisciplinary research topic going under the name of Quantum Machine Learning (QML) started to merge in different ways quantum computing and machine learning techniques in order to achieve improvements in both fields [15, 79, 80, 83, 84, 85, 86, 170]. It is possible to distinguish four approaches in QML, depending on the nature of the dataset under study and on the computation device being used [88]: in the first one (denoted as CC) the dataset represents some classical system and the algorithm runs on a classical computer but the machine learning algorithm is inspired by the formalism of quantum mechanics [89, 90, 91, 92, 93]; in the second approach (denoted as CQ) data is assumed to be classical but algorithms relies on the advantages of quantum computation in order to speed up classical methods of machine learning [94, 95, 96, 97, 98, 99, 100]; the third approach (denoted as QC) uses classical methods of machine learning to analyse quantum systems [101, 103, 105, 111, 112, 113, 114] and finally the last direction (denoted as QQ) imagines a general scenario where both the learning algorithm and the system under study are fully quantum.

In this thesis we propose and explore different research directions related to both the use of classical algorithm for the study of quantum systems (QC) and the employment of quantum computing to speed up hard ML computational tasks (CQ).

In the QC framework, we propose a TDA based on persistent homologies for the study of multipartite quantum entanglement, i.e. a complex form of entanglement that is shared between multiple quantum parties, which is in general very hard to characterise and classify.

In the CQ framework we present different strategies to perform ML tasks on quantum computational devices choosing the quantum computing model whose properties are more suitable for the ML algorithm being studied. In particular, from a purely theoretical viewpoint, we propose two different approaches based on kernel methods: one performs a multi-class classification known as error correcting output codes (ECOC) on a gate model quantum device while the other one computes a hamming distance based kernel via topological quantum computation.

On a more applicative side, we used two quantum devices currently available, i.e. the D-Wave quantum annealer and the IBM quantum computer for implementing two different tasks. A formulation of minimum spanning tree (MST) clustering and of an edit distance kernel are presented as quantum annealing problems that can be solved using the D-Wave device. A quantum classifier that performs facial expressions recognition is developed for the IBM quantum hardware.

The original work behind this doctoral thesis has already appeared in the papers:

1. *Homological analysis of multi-qubit entanglement*, A. Di Pierro, S. Mancini, L. Memarzadeh, R. Mengoni, EPL (Europhysics Letters) 123 (3), 30006
2. *Persistent homology analysis of multiqubit entanglement*, R. Mengoni, A. Di Pierro, L. Memarzadeh, S. Mancini, QIC Vol.20 No.5-6 (2020)
3. *Quantum error-correcting output codes*, D. Windridge, R. Mengoni, R. Nagarajan, International Journal of Quantum Information 16 (08), 1840003
4. *Hamming distance kernelisation via topological quantum computation*, A. Di Pierro, R. Mengoni, R. Nagarajan, D. Windridge, International Conference on Theory and Practice of Natural Computing, 269-280
5. *Kernel methods in Quantum Machine Learning*, R. Mengoni, A. Di Pierro, Quantum Mach. Intell. (2019) 1: 65

6. *Study network-related optimization problems using quantum alternating optimization ansatz*, Z. Wang, R. Mengoni et al. APS Meeting Abstracts
7. *Quantum Annealing approach to Edit Distance Calculation*, R. Mengoni, F. Trotti, A. Di Pierro, (work in progress)
8. *Facial expressions recognition with IBM quantum computer*, R. Mengoni, M. Incudini, A. Di Pierro, QTML2019 proceedings online, www.quantummachinelearning.org/qtml2019-program.html

The thesis is organized as follows. In Chapter 1 we give an overview of the classical computing techniques in Machine Learning and Topological Data Analysis. In Chapter 2 we present the quantum theory, with a focus on quantum entanglement. Then in Chapter 3 we introduce the topic of quantum computing in its many facets. In Chapter 4, quantum machine learning and its main algorithms are discussed. Finally in Chapters 5 up to 8 we explain the core work of this thesis: homological analysis of multiqubit entanglement (Ch.5); quantum error correcting output codes and Hamming distance kernelization via TQC (Ch.6); a quantum annealing implementation of the ML learning techniques for MST clustering and graph edit distance calculation (Ch.7); facial expressions recognition quantum algorithm on the IBM quantum computer (Ch.8).

The Classical Setting

In this chapter we give an overview of the classical computing techniques which are relevant for this thesis. In the first section the field of Machine Learning is introduced together with an explanation of its main algorithms. In the second part, we discuss the techniques known under the name of Topological Data Analysis.

1.1 Classical Machine Learning

The subset of Artificial Intelligence known as Machine Learning (ML) is recently showing an increasing popularity. Historically ML was addressed as the set of techniques able to identify patterns among huge datasets "without being explicitly programmed to perform that task" [76, 77]. In other words ML infers an input-output relation from data in order to interpret previously unknown inputs. In the last decade ML has undergone a great development because of its effectiveness in dealing with many IT problems like spam mail filters, iris recognition for security systems, evaluation of consumer behaviour, assessing risks in the financial sector or developing strategies for computer games.

Machine learning tasks are typically classified into three broad categories depending on the nature of the learning process [3]:

1) *Supervised learning*. The algorithm is provided with a set of correct input-output relations from which it has to infer a mapping that relates them. More in detail, it is given an ensemble of labelled data points usually called *training set*. This sample contains data in the form of multidimensional *feature vectors* which could for example represent the health condition of a patient and the labels indicating whether the subject is healthy or presents some diseases. The learning algorithm has the task to induce a classifier from these labelled samples. The classifier is a function that allocate labels to inputs, including those that are out of the training set which have never been previously analysed by the algorithm.

2) *Unsupervised learning*. The algorithm is provided with an unlabelled dataset with the goal of finding patterns and structures without any prior experience of the problem. A standard examples of unsupervised learning task is *clustering* where data-points in the form of multidimensional vector are assigned labels and grouped in such a way to minimize within-group distance while maximizing the margin between different classes. Other examples include *density estimation* which determines how data is distributed in space and *generative models* which are able to generate new data instances.

3) *Reinforcement learning*. In this setting, usually a computer program, called agent, interacts with a dynamic environment in which it must perform a certain goal. The agent could be rewarded or punished depending on which strategy it uses in order to reach its goal.

In the following sections we will discuss the machine learning methods that are relevant for this thesis, namely the K-nearest neighbours algorithm and kernel methods for what concerns supervised learning while different clustering algorithms are presented as examples of unsupervised learning.

1.1.1 Supervised Learning

K-Nearest Neighbours

K-Nearest Neighbours (k-NN) is a very popular and simple supervised learning algorithm used in pattern recognition where object classification is based on the characteristics of objects that are close to the one considered [4].

In general, consider a training set $\mathcal{T} = \{\vec{x}_m, l_m\}_{m=1}^M$ containing M feature vectors $\vec{x}_m \in \mathbb{R}^N$ with their respective class labels $l_m \in \{c_1, c_2, \dots, c_n\}$, where c_i identifies one of the possible n classes in which the training dataset is divided.

In the classification phase, an unclassified input \vec{x} (which represents an object) is assigned to a class c if such a class is the most frequent among the k training vectors closest to \vec{x} (see Fig.1.1). This is based on the idea that feature vectors which are close to each other encode similar properties. The proximity is measured based on some distance between points, usually Euclidean distance, but also others like Manhattan or Hamming distance are equally usable.

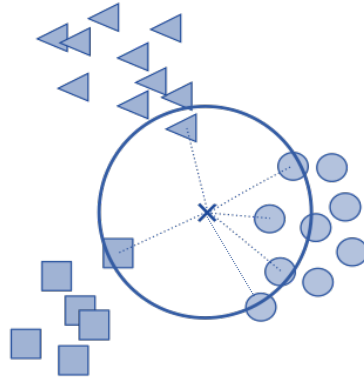


Fig. 1.1: The k-NN algorithm assigns a new input vector depicted as a cross to the most frequent class appearing among its closest $k = 6$ neighbours.

Since classification is based on the majority vote among k neighbours, k must be a positive, typically not very large, integer. In a binary classification context, in which there are only two classes, it is appropriate to choose odd k to avoid a situation of equality. However, the choice of k is not always easy and can influence the performance of the algorithm. Generally, increasing k leads to less noise-biased results but the criterion of class allocation for new input becomes more blurred [5]. From the complexity point of view, the k-NN algorithm takes $O(MN + kM)$, where, as said before, M is the cardinality of the training set and N the dimension of each sample.

A variation of the k-NN algorithm is the Nearest Centroid Classifier which, for each class, finds the centroid $\vec{\mu}_c$ i.e. the vector which is the mean between all those vectors with the same label

$$\vec{\mu}_c = \frac{1}{|N_c|} \sum_{l_m=c} \vec{x}_m \quad (1.1)$$

A new input is assigned to the class of the closest centroid; this is equal to a k -nearest neighbours where the training data has been pre-processed and $k = 1$.

Kernel Methods and Support Vector Machine

Kernel methods [115] are classification algorithms in ML that use a kernel function K in order to map data points, living in the input space V , to a higher dimensional feature space V' , where separability between classes of data becomes clearer. Kernel methods avoid the calculation of points in the new coordinates but rather perform the so called kernel trick that allow to work in the feature space V' simply computing the kernel of pairs of data points [115] (see. Fig.1.2).

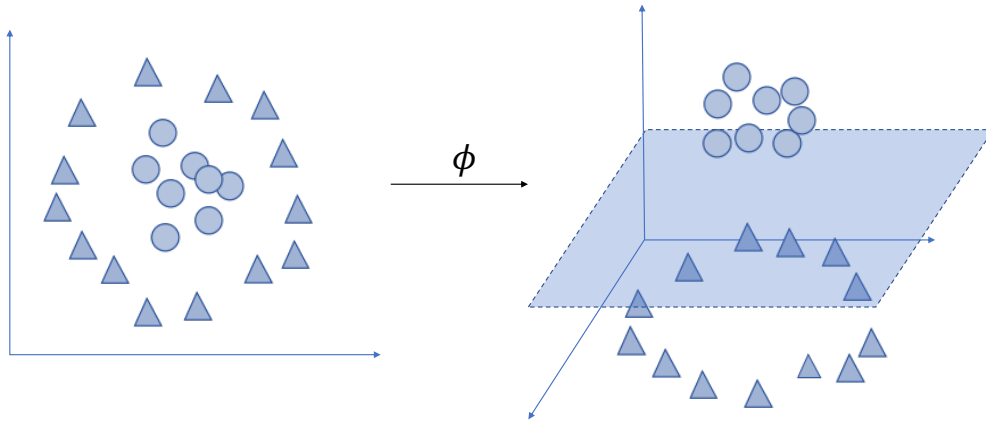


Fig. 1.2: Visual explanation of the kernel trick. After the application of the map ϕ , a previously non linearly separable dataset becomes such in a higher feature space.

If we consider a map ϕ such that $\phi : V \rightarrow V'$, the kernel $K : V \times V \rightarrow \mathbb{R}$ is the function representing the inner product in this high dimensional feature space V' :

$$K(\vec{x}_i, \vec{x}_j) \equiv \langle \phi(\vec{x}_i), \phi(\vec{x}_j) \rangle \quad (1.2)$$

(where $\langle \cdot, \cdot \rangle$ is the inner product in V') and it must satisfy the Mercer condition [11, 12] of positive semi-definiteness i.e. for all possible choices of M real numbers (c_1, \dots, c_M) , the following relation must hold

$$\sum_{i=1}^M \sum_{j=1}^M K(\vec{x}_i, \vec{x}_j) c_i c_j \geq 0. \quad (1.3)$$

Hence calculating the kernel $K(\vec{x}_i, \vec{x}_j)$ is computationally cheaper than computing each new coordinate $\phi(\vec{x})$; moreover it is worth noticing that at no stage it is required to compute $\phi(\vec{x}_i)$. In fact, the Mercer theorem guarantees the existence of a mapping ϕ whenever the kernel function $K(\vec{x}_i, \vec{x}_j)$ gives rise to a kernel matrix obeying the Mercer condition. Common examples of kernels defined on Euclidean space \mathbb{R}^d include:

- Linear kernel: $K(x, y) = x^T y$, $x, y \in \mathbb{R}^d$.
- Polynomial kernel: $K(x, y) = (x^T y + r)^n$, $x, y \in \mathbb{R}^d, r \geq 0$.
- Gaussian kernel: $K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$, $x, y \in \mathbb{R}^d, \sigma > 0$

Some kernels might be harder to compute than others. An example is the graph edit distance based kernels. The graph edit distance (GED) is a measure of similarity (or dissimilarity) between two graphs whose mathematical definition depends on the kind of graphs under study, i.e. whether graph are labelled, directed, planar etc. In general, given a set of graph edit operations, the GED between two graphs is defined as

$$GED(g_1, g_2) = \min_{(e_1, \dots, e_k) \in \mathcal{P}(g_1, g_2)} \sum_{i=1}^k c(e_i) \quad (1.4)$$

where $\mathcal{P}(g_1, g_2)$ denotes the set of edit paths transforming g_1 into (a graph isomorphic to) g_2 and $c(e) \geq 0$ is the cost of an edit operation e . The problem of computing GED is in general NP-complete, which means that also calculating the kernel which is based on the GED is a hard task.

Among the kernel methods, the best known example is Support Vector Machine (SVM), a supervised binary classifier that learns the optimal discriminative hyperplane when receiving in input a set of M labelled vectors $\{(\vec{x}, y) \mid \vec{x} \in \mathbb{R}^N, y \in \{-1, +1\}\}$. The SVM maximizes the distance, i.e. the margin, between the decision hyperplane and the closest points, called support vectors [70].

The SVM optimization problem with hard-margin can be formulated as follows:

$$\arg \min_{(\vec{w}, b)} \left\{ \frac{1}{2} \|\vec{w}\|^2 \right\} \quad (1.5)$$

subject to the constraint

$$\forall_i y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1$$

where (\vec{x}_i, y_i) , with $i = 1 \dots M$ and $y_i \in \{-1, +1\}$, is the couple of training vector and label; \vec{w} is the vector which is normal to the discriminative hyperplane, and b is the offset of the hyperplane.

An important extension of the SVM presented above is the so called soft margin SVM where the best hyperplane is the one that reaches the optimal trade-off between two factors: the minimization of the margin and the restrain of the point deviation from the margin, expressed using slack variables ξ_i tuned by the hyper-parameter C . The soft margin SVM optimization problem is of the form:

$$\arg \min_{(\vec{w}, b)} \left\{ \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^M \xi_i \right\} \quad (1.6)$$

subject to the constraint

$$\forall_i y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad (1.7)$$

Usually it is convenient to switch to the dual form where we introduce Lagrange multipliers α_i in order to include the constraint in the objective function:

$$\arg \max_{(\alpha_i)} \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\vec{x}_i^T \vec{x}_j) \quad (1.8)$$

subject to

$$\sum_i \alpha_i y_i = 0, \quad \forall_i \alpha_i \geq 0$$

where the relation $\vec{w} = \sum_i \alpha_i y_i \vec{x}_i$ has been used to obtain Eq. 1.8. It is worth noticing that only a sparse subset of the α_i s are non-zero and the corresponding \vec{x}_i are the support vectors which lie on the margin and determine the discriminant hyperplane.

In this context, a non-linear classification boundary for the SVM is obtained by replacing the term $(\vec{x}_i^T \vec{x}_j)$ in Eq. 1.8 with a kernel function $K(\vec{x}_i, \vec{x}_j) \equiv \vec{\phi}(\vec{x}_i)^T (\vec{\phi}(\vec{x}_j))$ satisfying the Mercer condition of positive semi-definiteness. The Lagrangian optimization problem for the soft margin SVM now becomes

$$\arg \max_{(\alpha_i)} \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) \quad (1.9)$$

subject to

$$\sum_i \alpha_i y_i = 0 \quad \text{with} \quad \forall_i \alpha_i \geq 0.$$

Note that the dual form of the SVM optimization problem is quadratic in the parameter α_i and it can be efficiently solved with quadratic programming algorithms.

A related method is the Least Squares Support Vector Machines (LS-SVM) [116] where the constraint defined in Eq. 1.7 is replaced with the equality constraint

$$\forall_i y_i(\vec{w} \cdot \vec{\phi}(\vec{x}_i) - b) = 1 - e_i, \quad (1.10)$$

where e_i are errors terms. In this way, optimal parameters $\vec{\alpha}$ and b that identify the decision hyperplane are found by solving a set of linear equations, instead of using quadratic programming. The LS-SVM problem can hence be formulated as follows

$$F \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} \doteq \begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & K + \gamma^{-1}I \end{pmatrix} \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix} \quad (1.11)$$

where F is a $(M + 1) \times (M + 1)$ matrix, $\vec{1}^T \equiv (1, 1, 1 \dots)^T$, K is the kernel matrix and γ^{-1} is the trade-off parameter that plays a similar role to C in soft margin SVM. Binary class labels are denoted by the vector $\vec{y} \in ([-1, 1]^M)^T$.

From a computational point of view [15], solving the quadratic programming problem or the least-squares SVM has $O(M^3)$ complexity. A bottleneck to the speed of computation is determined by the kernel: for a polynomial kernel $K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i^T \vec{x}_j + c)^d$ it takes $O(M^2d)$ but in other cases the complexity could be much higher.

1.1.2 Unsupervised Learning

K-Means Clustering

The k-means algorithm is an unsupervised ML method to cluster data, which allows us to subdivide objects into k partitions (or clusters), based on their attributes.

More in detail, consider an unlabelled dataset \mathcal{D} containing M feature vectors $\vec{x} \in \mathbb{R}^N$, the k-means algorithm aims to partition the dataset into $k \leq M$ sets $\mathbf{C} = \{C_1, C_2, \dots, C_k\}$ in order to minimize the within-cluster variance. The objective function is the following

$$\arg \min_{\mathbf{C}} \sum_{i=1}^k \sum_{\vec{x} \in C_i} \|\vec{x} - \vec{\mu}_i\|^2 \quad (1.12)$$

where $\vec{\mu}_i$ is the centroid or mean vector calculated among those points belonging to the same class C_i

$$\vec{\mu}_i = \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x}. \quad (1.13)$$

The algorithm at first randomly select k centroids and then follows an iterative procedure that alternates between two steps [6]. An assignment step where each vector \vec{x} is assigned to the cluster whose centroid $\vec{\mu}_i$ has the least squared Euclidean distance. An updating step where it is calculated the new centroids of the vectors in the new clusters.

The two steps are repeated until convergence, which is obtained when the points no longer change clusters. The algorithm does not guarantee a convergence to a global optimum but towards a local minimum [7]. Moreover the overall performance is largely affected by the initial choice of the centroids.

From the complexity point of view, finding the optimal solution for the k-means clustering problem is NP-hard in the Euclidean space [8], moreover the problem can be exactly solved in time $O(M^{Nk+1})$, where M is the number of elements in the dataset to be clustered and N is the dimension of the Euclidean space.

Minimum Spanning Tree Based Clustering

Before diving into this clustering method, let's briefly introduce the Minimum Spanning Tree (MST) problem.

Intro to the Minimum Spanning Tree problem

Consider a weighted graph $G = (V, E, w)$ composed of a set V of n vertices, a set E of edges and a function $w : E \rightarrow \mathbb{R}$ that associates a weight $w_{v,v'}$ to each edge $(v, v') \in E$.

The Minimum Spanning Tree (MST) problem aims at finding a subgraph T of G such that: it does not contain cycles (i.e. T is a tree), it touches all the n vertices of G and it minimizes the sum over all edge weights. Note that it is safe to consider only positive weights $w_{i,j}$ since the solution of MST problem does not change when we add a positive constant to the weights (see Fig.1.3).

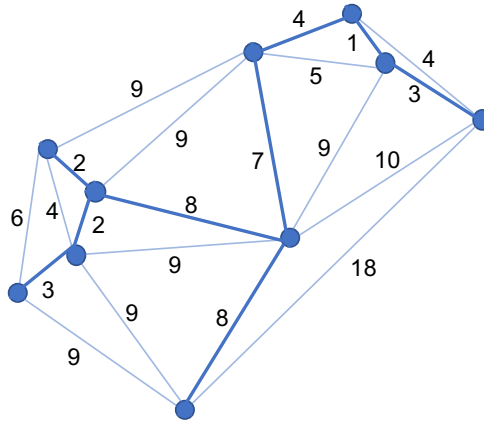


Fig. 1.3: Minimum spanning tree (MST) problem example. The edges that constitute the MST are highlighted. Notice the MST is a connected graph with n nodes and $n - 1$ edges.

In general, finding the MST of a given graph is not a hard task for a classical computer. In fact, given a graph where e is the number of edges and n is the number of vertices, the MST problem can be exactly solved in $O(e \log n)$ time. However, with an additional constraint called Δ -Degree constraint (i.e. each vertex in the MST must have degree less or equal than Δ), where $2 \leq \Delta < n - 1$, the problem becomes NP-hard and the corresponding decision problem NP-complete [9]. Finally, note that if the initial graph is complete, then a MST with Δ -Degree constraint is ensured to exist.

Clustering with the MST

Given an unlabelled dataset \mathcal{D} containing n feature vectors $\vec{x}_i \in \mathbb{R}^N$, the MST based clustering algorithm considers the n points \vec{x}_i as nodes of a complete weighted graph G , where weights $w_{i,j}$ represent the euclidean distance between any pair of points \vec{x}_i and \vec{x}_j .

$$w_{i,j} = d(\vec{x}_i, \vec{x}_j) \quad (1.14)$$

At this stage, the algorithm computes the MST (or the Degree constraint MST) of the weighted graph G . Once the MST is obtained, there are two different ways to produce a group of clusters[19].

If the number of clusters k is known in advance, the algorithm sorts the edges of the MST in descending order and remove the $(k - 1)$ edges with heaviest weights [16, 17], (see Fig.1.4).

Another approach to obtain clusters that does not require an a priori knowledge of k is the one where edges that satisfy an inconsistency measure are deleted from the tree. Such measure takes into account for those weights which are much larger than the average weight of the closest edges in the MST [18].

Finally, unlike traditional clustering algorithms, the MST based clustering does not assume any shape or structure of the underlying data and for this reason is well suited for detecting clusters with irregular boundaries [19].

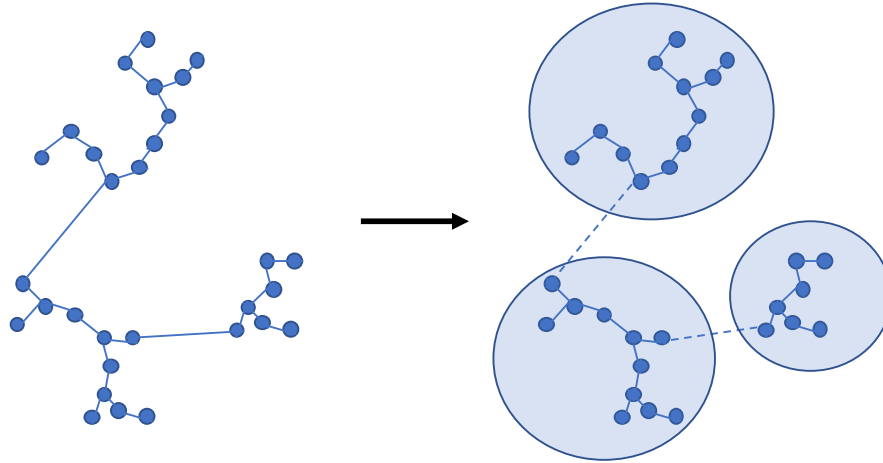


Fig. 1.4: Example of three-group MST-clustering after removal of two successive longest edges: (left) minimum spanning tree representation of the given points; (right) three-group clustering after the removal of two successive longest edges, dashed lines indicate the inconsistent edges.

1.2 Topological Data Analysis

A point cloud is a collection of points in some n -dimensional space \mathbb{S}_n . In many cases, analysing the global 'shape' of the point cloud gives essential insights about the problem it represents. Topological Data Analysis (TDA) provides a set of techniques to recognize the global structure of multidimensional point clouds. The most powerful tool of TDA is perhaps Persistent Homology, an algebraic method for computing coarse topological features of a given data cloud that persist over many grouping scales [20, 21].

1.2.1 Algebraic Topology Background

Algebraic topology is a branch of mathematics that employs abstract algebra to study topological spaces. The basic goal is to find invariants that classify topological spaces up to homotopy equivalence i.e. "continuous deformation" of one space into the other.

In this section we will review some key concepts of algebraic topology that are relevant for the technique known as Topological Data Analysis (TDA) [22, 23].

Convex set: A convex set is a region of a Euclidean space where every two points are connected by a straight line segment that is also within the region.

Convex Hull: The convex hull of a set X of points in an Euclidean space is the smallest convex set that contains X .

k -Simplex: A k -simplex $\sigma = (V_{j_0}, \dots, V_{j_k})$ is a k -dimensional object which is the convex hull of its $k+1$ points $\{V_{j_0}, \dots, V_{j_k}\}$. The dimension of a k -simplex is k and it can be understood as a k -dimensional

generalization of a triangle, for example, simplices of dimension 0, 1, 2 and 3 are respectively vertices, edges, triangles and tetrahedra as shown in Fig.1.5.

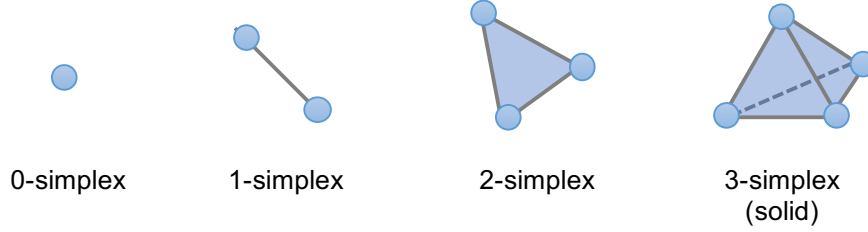


Fig. 1.5: Different k -simplices.

Face of a k -simplex: The convex hull of any nonempty subset of the $k + 1$ points of a k -simplex is called face of the simplex.

Simplicial complex: A simplicial complex \mathcal{K} is a finite set of simplices that satisfies the conditions:

- i) Any face of a simplex from \mathcal{K} is also in \mathcal{K} ,
- ii) The intersection of any two simplices $\sigma, \sigma' \in \mathcal{K}$ is either the empty set \emptyset or a face of both σ and σ' .

Loosely speaking, the simplicial complex is a collection of simplices and its dimension is equal to the largest dimension of its simplices, as shown in Fig.1.6.

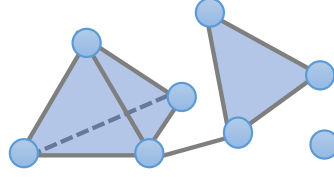


Fig. 1.6: An example of a simplicial complex.

Oriented simplex: An oriented k -simplex $\vec{\sigma}$ is a k -simplex with a fixed orientation, i.e. the order of the vertices is fixed. To denote an oriented simplex, brackets $[\cdot]$ instead of (\cdot) around the generating vertices are used. Oriented simplices have the following property when switching two vertices

$$[V_{j_0}, \dots, \hat{V}_{j_l}, \dots, \hat{V}_{j_m}, \dots, V_{j_k}] = -[V_{j_0}, \dots, \hat{V}_{j_m}, \dots, \hat{V}_{j_l}, \dots, V_{j_k}]$$

k -Chain group: A k -chain is a linear combination of oriented k -simplices with integer coefficients of the form

$$c = \sum_{i=1}^p \varepsilon_i \vec{\sigma}_i \quad (1.15)$$

where $\{\sigma_1, \dots, \sigma_p\}$ is the set containing all the k -simplices of the complex \mathcal{K} . The set of all k -chains of the complex \mathcal{K} is indicated with $C_k(\mathcal{K})$.

For example, the 1-chain of a path from some point v_1 to another point v_4 is $c = \vec{\sigma}_1 + \vec{\sigma}_2 + \vec{\sigma}_3$, where $\vec{\sigma}_1 = [v_1, v_2]$, $\vec{\sigma}_2 = [v_2, v_3]$ and $\vec{\sigma}_3 = [v_3, v_4]$ are the 1-simplices components of c .

k -Boundary operator: Given an oriented k -simplex $\vec{\sigma} = [V_{j_0}, \dots, V_{j_k}]$, the boundary operator is a map of the kind $\partial_k : C_k(\mathcal{K}) \rightarrow C_{k-1}(\mathcal{K})$ defined as

$$\partial_k(\vec{\sigma}) = \sum_{i=0}^k (-1)^i [V_{j_0}, \dots, \hat{V}_{j_i}, \dots, V_{j_k}]$$

where the hat accent on \hat{V}_{j_i} indicates that the point V_{j_i} is discarded and $[V_{j_0}, \dots, \hat{V}_{j_i}, \dots, V_{j_k}]$ is the oriented $(k-1)$ -simplex represented by all vertices in the set, except for V_{j_i} .

Roughly, the boundary map, applied to a k -chain outputs a $(k-1)$ -chain whose elements are the simplices that constitute the boundary of the k -chain. Considering the previous example, the boundary of the 1-chain $c = \vec{\sigma}_1 + \vec{\sigma}_2 + \vec{\sigma}_3$, of a path from point v_1 to v_4 is

$$\begin{aligned} \partial_1 c &= \partial_1(\vec{\sigma}_1 + \vec{\sigma}_2 + \vec{\sigma}_3) \\ &= \partial_1(\vec{\sigma}_1) + \partial_1(\vec{\sigma}_2) + \partial_1(\vec{\sigma}_3) \\ &= \partial_1([v_1, v_2]) + \partial_1([v_2, v_3]) + \partial_1([v_3, v_4]) \\ &= ([v_2] - [v_1]) + ([v_3] - [v_2]) + ([v_4] - [v_3]) \\ &= [v_4] - [v_1]. \end{aligned} \tag{1.16}$$

which in fact are the extremes (the boundary) of the path from point v_1 to v_4 .

k-Boundary group : The *k*-boundary group of a complex \mathcal{K} is defined as

$$B_k(\mathcal{K}) = \text{Im } \partial_{k+1} = \{c \in C_k(\mathcal{K}) \text{ such that } \exists c' \in C_{k+1}(\mathcal{K}) \text{ with } \partial_{k+1}(c') = c\} \tag{1.17}$$

This means that $B_k(\mathcal{K})$ of the complex \mathcal{K} contains all those k -chains that are boundaries of the $(k+1)$ -dimensional objects in \mathcal{K} ;

k-Cycle group : The *k*-cycle group of a complex \mathcal{K} is defined as

$$Z_k(\mathcal{K}) = \text{Ker } \partial_k = \{c \in C_k(\mathcal{K}) \text{ such that } \partial_k c = 0\} \tag{1.18}$$

which contains all those elements that have no boundary.

Homology group: Consider the simplicial complex \mathcal{K} . The k th homology group $H_k(\mathcal{K})$ associated to \mathcal{K} is defined as

$$H_k(\mathcal{K}) \equiv Z_k(\mathcal{K})/B_k(\mathcal{K}) \tag{1.19}$$

which includes those k -dimensional objects which have no boundary and are not boundary of anything that is $(k+1)$ -dimensional.

More in detail, for each simplicial complex \mathcal{K} there is a set of homological groups $\{H_0(\mathcal{K}), H_1(\mathcal{K}), H_2(\mathcal{K}), \dots\}$, where the k th homology group $H_k(\mathcal{K})$ is non-empty when the k -dimensional holes are in \mathcal{K} .

Betti numbers: The k -th Betti number β_k is defined as

$$\beta_k(\mathcal{K}) \equiv \dim H_k(\mathcal{K}). \tag{1.20}$$

Hence, the Betti numbers β_k , as well as the associated homology groups $H_k(\mathcal{K})$ of a simplicial complex \mathcal{K} , describes the connectivity existing in that complex. In other words, the k -th Betti number β_k counts the number of k -dimensional holes in a complex: β_0 is the number of connected components, β_1 is the number of 1-dimensional holes and β_2 is the number of two-dimensional holes i.e. voids.

Note that Betti numbers are topological invariants which means that if different spaces have the same Betti numbers then the spaces are homotopy equivalent.

1.2.2 Persistent Homology

Consider a dataset (or data cloud) represented by a set of points $\{x_\alpha\}$ in a Euclidean space. In order to capture the global topological features in the dataset, the corresponding space must first be represented as a simplicial complex as depicted in Fig. 1.7.

At first, points in the cloud are grouped together depending on their pairwise distances. More in detail, choosing a value of the grouping scale ϵ , it is possible to construct the graph whose vertices are the data points $\{x_\alpha\}$ and edges $e_{x_\alpha, x_{\alpha'}}$ are drawn when the $\frac{\epsilon}{2}$ -balls centred in the vertices x_α and $x_{\alpha'}$ intersect each other. Such graphs show connected components and hence clusters obtained at ϵ scale but do not provide information about higher-order features such as holes and voids.

In order to track high-dimensional features it is necessary to complete the corresponding graph to a simplicial complex by filling in the graph with simplices. At a given grouping scale ϵ , there are different methods to generate simplicial complexes \mathcal{K}_ϵ . One of this methods generates the so called Vietoris-Rips complex¹, \mathcal{R}_ϵ , where k -simplices correspond to $(k+1)$ points which are pairwise within distance ϵ . The Rips complex is closely related to another simplicial complex, called the Čech complex \mathcal{C}_ϵ , where s k -simplices are determined by $(k+1)$ points whose closed $\epsilon/2$ -ball neighborhood have a point of common intersection [25]. By the Čech theorem [24], the Čech complex has the same topological structure as the open sets ($\epsilon/2$ -balls) cover of the point cloud. This is not true for the Rips complex, which is more coarse than the Čech complex. Therefore, the latter is a more powerful tool for classification with respect to Rips which by the way is computationally easier to calculate.

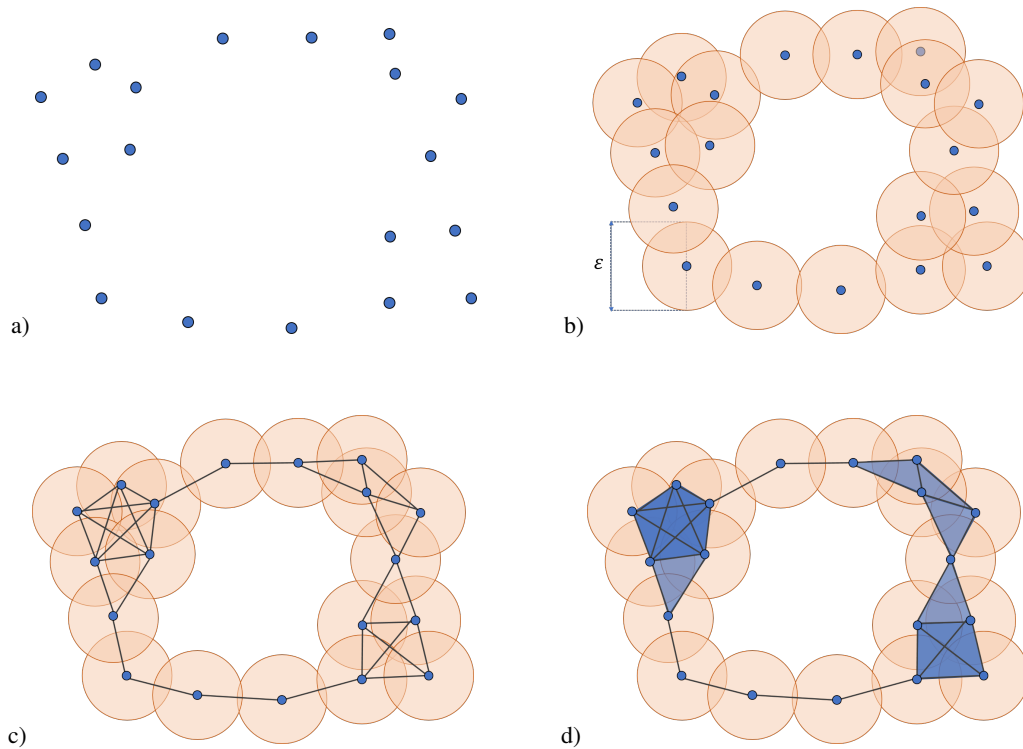


Fig. 1.7: Method to associate a simplicial complex to a data cloud: a) the point cloud; b) the choice of the grouping scale ϵ produces the graph shown in c); d) the graph is completed to a simplicial complex.

¹ For simplicity, it will be referred to as Rips complex in the following.

Topological features of the data cloud that appear at a chosen value of ϵ are obtained by considering the homology groups of the simplicial complex associated to the cloud. If ϵ is taken too small, then only multiple connected components are shown. On the other hand, when ϵ is large, any pairs of points get connected and a giant simplex with trivial homology is obtained.

However, it is preferable to make the whole process independent from the choice of ϵ . In order to obtain significant features it is necessary to consider all the range of ϵ . In the presence of a noisy dataset, those topological features which persist over a significant interval of the parameter ϵ are considered specific of that point cloud, while short-lived features are less important ones [25, 26].

Consider the filtration $K = \{\mathcal{K}_{\epsilon_i}\}_{i=1}^N$ i.e. a sequence of complexes associated to a given point cloud; instead of examining the homology of the individual terms $H_k(\mathcal{K}_{\epsilon_i})$, we look at the inclusion maps $I : H_k(\mathcal{K}_{\epsilon_i}) \rightarrow H_k(\mathcal{K}_{\epsilon_j})$ for all $i < j$. These maps are able to tell us which features persist since they reveal information that is not visible if we consider $H_k(\mathcal{K}_{\epsilon_i})$ and $H_k(\mathcal{K}_{\epsilon_j})$ separately.

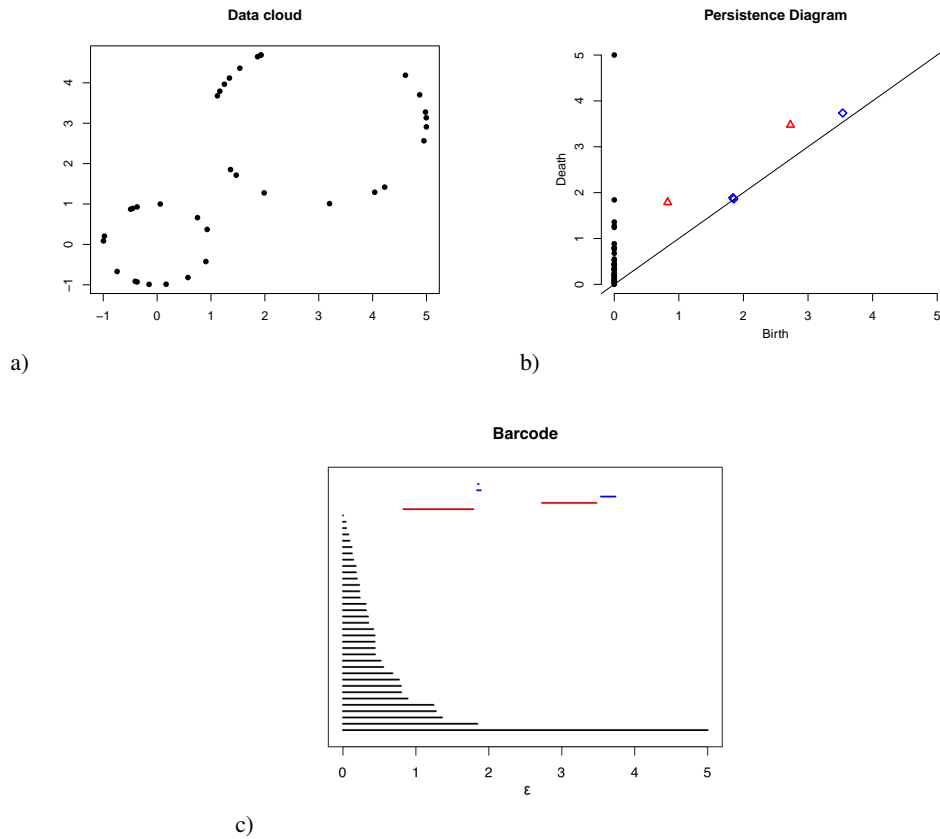


Fig. 1.8: a) the point cloud; b) and c) are respectively the persistent diagrams and the barcode associated to the point cloud: in the two diagrams, connected components are depicted in black, holes in red and voids in blue. When the grouping scale ϵ is small, many connected components are visible because points get connected only to the closest points. Increasing ϵ , higher order topological features like holes and voids appear. When the grouping scale is large enough, all points are connected to form a high dimensional simplex and hence a single connected component is left.

Barcodes. Given a filtration $K = \{\mathcal{K}_{\epsilon_i}\}_{i=1}^N$, a *barcode* is a graphical representation of $H_k(K)$ as a collection of horizontal line segments in a plane whose horizontal axis corresponds to the parameter ϵ and whose vertical axis represents homology groups, see c) in Fig.1.8. A barcode can be seen as a variation

over ϵ of the Betti numbers which count the number of n -dimensional holes on the simplicial complex \mathcal{K}_ϵ [25].

Persistence diagrams. Given the filtration $K = \{\mathcal{K}_{\epsilon_i}\}_{i=1}^N$ a persistence diagram is a multiset of points \vec{p} in \mathbb{R}^2 where each point $p = (b, d)$ tell us at which values of ϵ a k -dimensional hole appear (time b) and disappear (time d). Since not every k -dimensional hole has to disappear, those topological features that persist are mapped to points of the form (b, ∞) in the persistence diagram. Persistence diagrams are therefore concise graphical representations of $H_k(K)$ equivalent to barcodes, as shown in Fig.1.8.

1.2.3 TDA as a Resource for ML

ML is usually intended as a set of algorithms that can learn from data and make some kind of predictions about them. TDA is in general not considered part of the ML toolbox, even if the boundary between the two is blurred. However, rather than discussing about their differences, it is perhaps more useful to show how TDA and ML could play well together.

A solid approach in this direction was proposed in [27] where authors establish a connection between TDA and ML by designing a kernel defined on persistence diagrams that could improve kernel methods like SVM.

More in detail, given a set of persistence diagrams \mathcal{D} , the kernel is defined in terms of a feature map $\Phi_\sigma : \mathcal{D} \rightarrow L_2(\Omega)$, where $\Omega \subset \mathbb{R}^2$ indicates the closed half real plane above the diagonal, i.e. the portion of the plane where persistence diagrams live.

Since a persistence diagram \mathcal{D} is a multisets of points p in \mathbb{R}^2 which does not have the structure of a Hilbert space, it is possible to embed the diagram \mathcal{D} into a Hilbert space by representing it as a sum of Dirac delta distributions, one for each point p in \mathcal{D} .

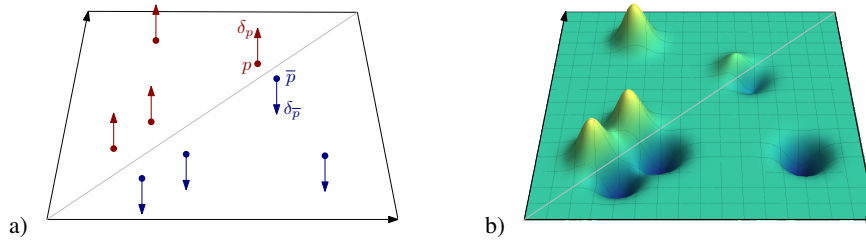


Fig. 1.9: (a) Given a persistence diagram \mathcal{D} , its domain is extended from Ω , i.e. the closed half plane above the diagonal, to \mathbb{R}^2 by adding points \bar{p} which are symmetric to p with respect to the diagonal. Each point $p \in \mathcal{D}$ is associated with a Dirac delta δ_p , while each \bar{p} with a $-\delta_{\bar{p}}$. (b) The modified persistent diagram on the left is used to define a heat diffusion problem with Dirac deltas as initial conditions (Figure from [27]).

Moreover, in order to take into account the distance between points p and the diagonal of the persistence diagram (see Fig.1.9a), the sum of Dirac deltas is used as an initial condition for a heat diffusion problem with a Dirichlet boundary condition on the diagonal.

The solution of this partial differential equation lead to the definition of the following kernel, tuned by the parameter σ

$$k_\sigma(F, G) = \frac{1}{8\pi\sigma} \sum_{\substack{p \in F \\ q \in G}} e^{-\frac{\|p-q\|^2}{8\sigma}} - e^{-\frac{\|p-\bar{q}\|^2}{8\sigma}}. \quad (1.21)$$

where $F, G \in \mathcal{D}$ are two persistence diagrams while p and q are respectively the persistence diagrams points of F and G and \bar{q} is the symmetric of q with respect to the diagonal.

Quantum Entanglement

Quantum entanglement is a purely quantum mechanical feature of global states of composite systems which cannot be written as a product of the states of individual subsystems (or a convex linear combination of them) [28].

Despite an initial scepticism in the early days of quantum theory [29], entanglement is nowadays considered a uniquely quantum mechanical resource and it plays a key role in many of the most interesting applications of quantum computation and quantum information [35].

In this chapter, after a brief overview of the postulates of quantum mechanics, the phenomenon of Quantum Entanglement will be introduced for both pure and mixed states. Finally, the problems of separability and characterization of entanglement will be extensively discussed.

2.1 Postulates of Quantum Mechanics

- I. *To a closed quantum system is associated a space of states \mathbb{H} which is a Hilbert space. The pure state of the system is then represented by a unit norm vector $|\Psi\rangle$ on such Hilbert space.*

The simplest quantum system we can consider is the qubit, to which is associated the Hilbert space \mathbb{C}^2 [35], a generic qubit state is described by a vector

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

where $\{|0\rangle, |1\rangle\}$ is the canonical basis and probability amplitudes i.e. coefficients $\alpha, \beta \in \mathbb{C}$ are such that $|\alpha|^2 + |\beta|^2 = 1$.

Quantum state $|\Psi\rangle$ is usually parametrized as

$$|\Psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle.$$

where $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$. This means that the set of states $\{|\Psi(\theta, \varphi)\rangle\}_{\theta, \varphi}$ corresponds to the surface of a 2-sphere $\mathbb{S}^2 \subset \mathbb{R}^3$ of radius one (known as the Bloch sphere).

- II. *The space of states of a composite physical system is the tensor product of the states spaces (Hilbert spaces) of its subsystems.*

- III. *The state change of a closed physical system with associated Hilbert space \mathbb{H} is described by a unitary operator $U : \mathbb{H} \rightarrow \mathbb{H}$*

$$|\Psi\rangle \rightarrow |\Psi'\rangle = U |\Psi\rangle$$

with $U^\dagger U = U U^\dagger = I$ where U^\dagger is the adjoint of U and I is the identity operator

The time evolution of the state of a closed quantum system is described by Schrödinger equation ¹

$$i \frac{d|\Psi\rangle}{dt} = H |\Psi\rangle \quad (2.1)$$

where H is known as the Hamiltonian, an Hermitian operator that describes the energy of the system. It is straightforward to see that the solution to Schrödinger equation is

$$|\Psi(t)\rangle = e^{-iHt} |\Psi(0)\rangle \quad (2.2)$$

Note that, since the operator H is Hermitian, the transformation $U = e^{-iHt}$ must be unitary, as claimed before.

IV. Given a closed quantum system, to any observable physical quantity A is associated a Hermitian operator in the space of states (Hilbert space \mathbb{H}), i.e. $A : \mathbb{H} \rightarrow \mathbb{H}$ such that $A = A^\dagger$. The possible measurement outcomes are the eigenvalues $\{a_j\}_j$ ($a_j \in \mathbb{R}$) of A . The probability that the outcome is a_j , given that the system was in the state $|\Psi\rangle$, is

$$p_\Psi(a_j) = \langle \Psi | P_j | \Psi \rangle,$$

with $P_j = |a_j\rangle \langle a_j|$ the projector onto the subspace of eigenvector $|a_j\rangle$ of A corresponding to the eigenvalue a_j .

As consequence of a measurement outcome a_j , the state of the system changes as follows

$$|\Psi\rangle \rightarrow |\Psi'\rangle = \frac{P_j |\Psi\rangle}{\sqrt{p_\Psi(a_j)}}. \quad (2.3)$$

2.2 Entanglement for Pure States

Consider a bipartite quantum systems composed of two subsystems A and B to which are associated the Hilbert spaces \mathbb{H}_A and \mathbb{H}_B with basis states respectively $\{|i_A\rangle\}_i$ and $\{|j_B\rangle\}_j$. According to the first and second postulate, the space of states of the bipartite composite system \mathbb{H}_{AB} is given by the tensor product of its two subsystems i.e. $\mathbb{H}_A \otimes \mathbb{H}_B$, having as basis states the set $\{|i_A j_B\rangle\}_{i,j}$. From the second postulate follows that any pure state of the composite system, $|\Psi_{AB}\rangle \in \mathbb{H}_A \otimes \mathbb{H}_B$, can be expressed as follows

$$|\Psi_{AB}\rangle = \sum_{i,j} c_{i,j} |i_A j_B\rangle \quad (2.4)$$

By looking at Eq. 2.4, it is possible to distinguish two kinds of states:

- *Pure states.* States $|\Psi_{AB}\rangle \in \mathbb{H}_{AB}$ that can be written as

$$|\Psi_{AB}\rangle = |\Psi_A\rangle \otimes |\Psi_B\rangle \quad (2.5)$$

with $|\Psi_A\rangle \in \mathbb{H}_A$ and $|\Psi_B\rangle \in \mathbb{H}_B$ are called factorizable or product states.

¹ Planck constant \hbar is set to one

- *Entangled states.* Those states $|\Psi_{AB}\rangle \in \mathbb{H}_{AB}$ that are not factorizable, i.e.

$$|\Psi_{AB}\rangle \neq |\Psi_A\rangle \otimes |\Psi_B\rangle \quad (2.6)$$

for any $|\Psi_A\rangle \in \mathbb{H}_A$ and $|\Psi_B\rangle \in \mathbb{H}_B$, are called entangled.

Note that factorizable states assign a precise state to each subsystem while for entangled states there is no way of characterizing one subsystem without referring to the other.

2.2.1 Schmidt Decomposition

The Schmidt decomposition is a useful mathematical description to characterize bipartite entanglement [28]. Consider a generic state $|\Psi_{AB}\rangle \in \mathbb{H}_{AB} = \mathbb{H}_A \otimes \mathbb{H}_B$ as in 2.4 where $\dim(\mathbb{H}_A) = d_A$ and $\dim(\mathbb{H}_B) = d_B$, written as

$$|\Psi_{AB}\rangle = \sum_{i=1}^{d_A} \sum_{j=1}^{d_B} c_{ij} |i_A j_B\rangle. \quad (2.7)$$

Coefficients $c_{ij} \in \mathbb{C}$ are such that $\sum_{i,j} |c_{ij}|^2 = 1$ and can be rearranged in a matrix form as follows

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1d_B} \\ c_{21} & c_{22} & \cdots & c_{2d_B} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d_A1} & c_{d_A2} & \cdots & c_{d_A d_B} \end{pmatrix}.$$

Applying Singular Values Decomposition [39] to matrix \mathbf{C} with complex entries, we obtain

$$\mathbf{C} = \mathbf{U} \mathbf{D} \mathbf{V} \quad (2.8)$$

where \mathbf{U} is a $d_A \times d_A$ unitary matrix, \mathbf{V} is a $d_B \times d_B$ unitary matrix and \mathbf{D} is a $d_A \times d_B$ diagonal matrix with non-negative real numbers on the principal diagonal. The diagonal entries, $\sqrt{\lambda_a}$, of \mathbf{D} are known as the singular values of \mathbf{C} .

Equation 2.7 can now be written as

$$|\Psi_{AB}\rangle = \sum_{i=1}^{d_A} \sum_{j=1}^{d_B} \sum_{a=1}^{\min\{d_A, d_B\}} U_{ia} \sqrt{\lambda_a} V_{aj} |i_A j_B\rangle.$$

where components c_{ij} are

$$c_{ij} = \sum_{a=1}^{\min\{d_A, d_B\}} U_{ia} \sqrt{\lambda_a} V_{aj} \quad (2.9)$$

Since vectors $\sum_{i=1}^{d_A} U_{ia} |i_A\rangle = |u_a\rangle_A$ and $\sum_{j=1}^{d_B} V_{aj} |j_B\rangle = |v_a\rangle_B$ form two new basis, respectively for \mathbb{H}_A and \mathbb{H}_B we can recast everything as

$$|\Psi_{AB}\rangle = \sum_{a=1}^{\min\{d_A, d_B\}} \sqrt{\lambda_a} \left(\sum_{i=1}^{d_A} U_{ia} |i_A\rangle \right) \left(\sum_{j=1}^{d_B} V_{aj} |j_B\rangle \right)$$

Finally we obtained the *Schmidt decomposition* of a pure bi-partite quantum state:

$$|\Psi_{AB}\rangle = \sum_{a=1}^{\min\{d_A, d_B\}} \sqrt{\lambda_a} |u_a\rangle_A |v_a\rangle_B \quad (2.10)$$

where coefficients $\sqrt{\lambda_a}$, known as *Schmidt coefficients*, are s.t. $\sum_a \lambda_a = 1$.

Schmidt Rank

The Schmidt rank is equal to the rank of the matrix \mathbf{C} i.e. to the number of non-zero Schmidt coefficients.

The Schmidt decomposition as it was defined for the state $|\Psi_{AB}\rangle$ of a bipartite system, give a mathematical characterization of bipartite entanglement:

- $|\Psi_{AB}\rangle$ is a *factorizable* state iff has Schmidt rank equal to one
- $|\Psi_{AB}\rangle$ is an *entangled* state iff it has Schmidt rank strictly greater than one

It is worth noting that the maximum Schmidt rank is $d = \min\{d_A, d_B\}$.

Maximally Entangled State

States $|\Psi_{AB}\rangle$ having maximum Schmidt rank and Schmidt coefficients all equal to $1/d$ are called *maximally entangled states*:

$$|\Psi_{AB}\rangle = \frac{1}{\sqrt{d}} \sum_{a=1}^d |u_a\rangle_A |v_a\rangle_B. \quad (2.11)$$

For bipartite systems, the Hilbert space $\mathbb{H} = \mathbb{C}^2 \otimes \mathbb{C}^2$ is spanned by the four Bell's maximally entangled basis states

$$|\psi^\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle|1\rangle \pm |1\rangle|0\rangle) \quad |\phi^\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle \pm |1\rangle|1\rangle). \quad (2.12)$$

These states have remarkable properties. A measurement of the observable Z on one of the two subsystems affects instantaneously the state of the other subsystem, that will collapse into state $|0\rangle$ or $|1\rangle$ depending on the outcome of the measurement. Thus, the result of the measurements for both subsystems are perfectly correlated: we know nothing at all about the subsystems, although we have maximal knowledge of the whole system.

2.3 Entanglement for Mixed States

Suppose to have a statistical mixture $\{p_i, |\psi_i\rangle\}$ on a Hilbert space \mathbb{H} i.e. an ensemble of quantum states $|\psi_i\rangle$ associated with some probability p_i , with $p_i \geq 0$ such that $\sum_i p_i = 1$.

A *mixed state* on $\{p_i, |\psi_i\rangle\}$ is defined as the *density operator*

$$\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|. \quad (2.13)$$

If we denote with $\mathcal{L}(\mathbb{H})$ the space of linear operators acting on the Hilbert space \mathbb{H} , then the class of density operators are characterized by the following properties.

An operator $\rho \in \mathcal{L}(\mathbb{H})$ is the density operator associated to some ensemble $\{p_i, |\psi_i\rangle\}$ if and only if it satisfies the conditions

- 1) $Tr(\rho) = 1$;
- 2) $\forall |\gamma\rangle \in \mathbb{H}, \langle \gamma|\rho|\gamma\rangle \geq 0$.

Given a bipartite system AB with associated Hilbert space $\mathbb{H}_{AB} = \mathbb{H}_A \otimes \mathbb{H}_B$ in the state ρ_{AB} , the state of the subsystem A is

$$\rho_A = Tr_B(\rho_{AB}).$$

where Tr_B stands for *partial trace* over B and it is defined as

$$Tr_B(\rho_{AB}) = \sum_{j=1}^{\dim \mathbb{H}_B} \langle j_B | \rho_{AB} | j_B \rangle$$

and $|j_B\rangle$ is an orthonormal basis of \mathbb{H}_B .

It is worth noticing that for a maximally entangled pure state

$$|\Psi_{AB}\rangle = \frac{1}{\sqrt{d}} \sum_{a=1}^d |u_a\rangle_A |v_a\rangle_B,$$

where $d = \min\{d_A, d_B\}$, the reduced density operator of subsystem A is

$$\rho_A = Tr_B(|\Psi_{AB}\rangle \langle \Psi_{AB}|) = I_A/d.$$

Hence, the reduced density operator of a maximally entangled state is called maximally mixed state i.e. a mixed state describing an ensemble of orthogonal states associated with a flat probability distribution.

Suppose that a mixed state of a bipartite composite system is described by the density operator ρ_{AB} acting on the Hilbert space $\mathbb{H}_{AB} = \mathbb{H}_A \otimes \mathbb{H}_B$. It is possible to distinguish the following states:

- *factorizable mixed states*. States ρ_{AB} that can be written as the tensor product of states ρ_A on \mathbb{H}_A and ρ_B on \mathbb{H}_B i.e

$$\rho_{AB} = \rho_A \otimes \rho_B \quad (2.14)$$

are called factorizable (or product) states.

- *Separable mixed states*. States ρ_{AB} that can be written as convex combination of states ρ_A^k on \mathbb{H}_A and ρ_B^k on \mathbb{H}_B i.e.

$$\rho_{AB} = \sum_k p_k \rho_A^k \otimes \rho_B^k \quad (2.15)$$

are separable states where coefficients p_k are s.t. $p_k \geq 0$ and $\sum_k p_k = 1$.

- *Entangled mixed states*. Those states that are not separable, i.e. can not be expressed as a convex combination of any state ρ_A^k on \mathbb{H}_A and ρ_B^k on \mathbb{H}_B

$$\rho_{AB} \neq \sum_k p_k \rho_A^k \otimes \rho_B^k \quad (2.16)$$

are called entangled states.

In practice it is hard to decide if a given states is separable or entangled simply looking at the definition. For this reason one of the fundamental problems concerning entanglement is the *separability problem* i.e. to establish if a given bipartite state is entangled or separable.

2.3.1 PPT Criterion for Separability

Being the separability problem for mixed states extremely complex, few criteria are known which work only in special cases. In the following we present a criterion for separability of a bipartite system known as Positive Partial Transpose (PPT) criterion. [37]

Let's first introduce the concept of *partial transposition*: consider a density operator ρ_{AB} acting on \mathbb{H}_{AB} . The partial transpose ρ^{TA} of ρ with respect to the subsystem A, is the Hermitian operator with unitary trace having elements

$$\langle i_A, j_B | \rho^{TA} | k_A, l_B \rangle \equiv \langle k_A, j_B | \rho | i_A, l_B \rangle \quad (2.17)$$

where $|i_A, j_B\rangle \in \mathbb{H}_A \otimes \mathbb{H}_B$ is a fixed orthonormal basis.

The *PPT criterion* states that: given a density operator ρ_{AB} on \mathbb{H}_{AB} , a necessary condition for its separability is that

$$\rho_{AB}^{TA} \geq 0.$$

It is possible to prove that the criterion of positive partial transpose is necessary and sufficient for any quantum system with a dimension of less than or equal to 6, i.e. when \mathbb{H}_{AB} is isomorphic to $\mathbb{C}^2 \otimes \mathbb{C}^2$ or to $\mathbb{C}^2 \otimes \mathbb{C}^3$ [28] [37]. In other words the PPT criterion states that if ρ_{AB}^{TA} has at least one negative eigenvalue, then the state ρ_{AB} will surely be entangled (note that it is independent of which is the transposed subsystem since $\rho_{AB}^{TA} = (\rho_{AB}^{TB})^T$).

2.4 Entanglement Monotones

Entanglement monotones are functions that measure the amount of entanglement in a quantum state. Such functions are nonnegative and their value does not increase under the action of Local Operations and Classical Communications (LOCC) [28].

Given a multipartite quantum system shared among different parties, LOCC transformations consist in a series of rounds of operations applied to the system. In each round, a given party operates locally on its subsystem and, in case a measurement occurred, a classical channel is used to communicate the result of the measurement to the other parties.

More in detail, an *entanglement monotone* is a function of a bipartite quantum state, $E(\rho_{AB})$, with the following properties:

- I) $E(\rho_{AB}) \geq 0$.
- II) $E(\rho_{AB}) = 0$ if ρ_{AB} is separable.
- III) $E(|\psi\rangle_{AB} \langle\psi|)$ is maximum when $|\psi\rangle_{AB}$ is a maximally entangled state.
- IV) E is constant under local unitaries, and cannot increase under LOCC.
- V) E is a convex function of the kind

$$E\left(\sum_i p_i \rho_i\right) \leq \sum_i p_i E(\rho_i) \quad (2.18)$$

whenever the ρ_i are Hermitian and $p_i \geq 0$ with $\sum_i p_i = 1$.

2.4.1 Measures of Entanglement

Concurrence

The concurrence is an entanglement monotone, ranging from zero to one, defined for a mixed state of two qubits $\rho = |\psi\rangle\langle\psi|$ where $|\psi\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ as [28]:

$$\mathcal{C}(\rho) \equiv \max(0, \lambda_1 - \lambda_2 - \lambda_3 - \lambda_4) \quad (2.19)$$

in which $\lambda_1, \dots, \lambda_4$ are the square roots of the eigenvalues (sorted in decreasing order) of the non-Hermitian matrix $\rho\tilde{\rho}$ with $\tilde{\rho} = (\sigma_y \otimes \sigma_y)\rho^*(\sigma_y \otimes \sigma_y)$ where ρ^* is the complex conjugate of the density matrix ρ in the computational basis and σ_y the Pauli y-matrix.

Note that the concurrence reaches its maximum when the two qubits are a Bell pair and it is zero when the state is separable. If the concurrence has a value strictly greater than zero and smaller than one, this means that the two qubits have some degree of entanglement.

Generalised Concurrence

Given an N -qubit quantum state $|\psi\rangle_N$, consider a bipartition of the N qubits into two subsystems A and B . The generalized concurrence \mathcal{C}_G measures the amount of entanglement between the two partitions A and B and it is defined as follows [142]:

$$\mathcal{C}_G(\rho_A) := \sqrt{2(1 - \text{Tr}(\rho_A^2))}. \quad (2.20)$$

2.5 Entanglement Classification

Since entanglement is a key resource in many quantum information technology tasks (see e.g. [41]), a careful characterization of entanglement is required. Usually, classes of entanglement are obtained by applying a set of operations to entangled states in order to convert them into other entangled states. If such operations allow the transformation, then the two states belong to the same entanglement class, otherwise they belong to classes of entanglement that are not equivalent.

In [42] LOCC operations have been used in order to check whether a pure state of a bipartite quantum system could be transformed into another pure state. In particular, it was shown the following. Consider two bipartite quantum states $|\psi\rangle_{AB}$ and $|\phi\rangle_{AB}$ on \mathbb{H}_{AB} of dimension d with Schmidt decomposition

$$\begin{aligned} |\psi\rangle_{AB} &= \sum_i \sqrt{\lambda_i} |i\rangle_A |i\rangle_B \\ |\phi\rangle_{AB} &= \sum_k \sqrt{\lambda'_k} |k\rangle_A |k\rangle_B \end{aligned} \quad (2.21)$$

If the Schmidt coefficients are arranged in decreasing order, i.e., $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ and $\lambda'_1 \geq \lambda'_2 \geq \dots \geq \lambda'_d$ then $|\psi\rangle$ can be transformed into $|\phi\rangle$ by LOCC if and only if

$$\sum_{i=1}^l \lambda_i \leq \sum_{i=1}^l \lambda'_i \quad (2.22)$$

for all $l = 1, 2, \dots, d$.

Moreover, pure states can be transformed into each other by means of LOCC if and only if they are related by local unitaries LU [43]. However, even simple bipartite systems are typically not related by LU which, in turn, means that infinitely many kinds of entanglement exist under LOCC.

While entanglement monotones have been proved to work mostly for bipartite states [28], for multipartite entanglement, approaches based on stochastic local operations and classical communication (SLOCC) seem more promising. In this scenario, two states are equivalent under SLOCC if there is a non-vanishing probability of success when trying to convert one state into the other.

In [44], equivalence classes between quantum states are constructed on the base of invariance under SLOCC. It was shown that each pure state of three entangled qubits can be converted into either the GHZ-state or the W-state, which leads to two inequivalent ways of entangling three qubits.

$$|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

$$|W\rangle = \frac{1}{2}(|001\rangle + |010\rangle + |100\rangle)$$

However, this leads to infinite (even uncountable) classes for more than three qubit systems. Hence this approach is not effective in the general case, although some ways out were devised for the case of four qubits. In [45] it is proposed a classification for pure entangled states of four qubit. This classification is obtained from the orbits generated by SLOCC operations and produces nine classes. Among these nine classes, one is called 'generic' and contains an uncountable number of SLOCC inequivalent classes of states, as shown in [46]. The other eight classes instead have W type entanglement.

Quantum Computing

Quantum Computing harnesses quantum phenomena like superposition and entanglement to process information.

Quantum algorithms manipulate a quantum system in order to obtain a quantum state whose components with high probability amplitude represent the solution for a given problem. The power of superposition and entanglement can lead to a computational speed-up with respect to a classical computer, since operations are performed on many states simultaneously. Quantum algorithms are repeated a number of times since the result is always probabilistic.

There exist different equivalent models of quantum computation: in this chapter, at first the quantum circuit model will be discussed, which is probably the most known, then adiabatic quantum computing will be introduced, with a particular emphasis to the technique of quantum annealing and finally topological quantum computation will be explained.

3.1 Quantum Circuit Model

In quantum circuit model, the quantum computation is performed by means of a sequence of quantum gates acting on registers of n qubits. Each quantum gate maps the input quantum state into other quantum state and therefore it must be described by a unitary operator. The computation usually ends with a measurement that collapse the system into a basis state with a given probability.

3.1.1 Quantum Gates

For what concerns single qubits gates it is useful to introduce the Pauli operators defined as

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (3.1)$$

Notice that $\{|0\rangle, |1\rangle\}$ are Z eigenvectors with corresponding eigenvalues $\{+1, -1\}$. Analogously

$$\{|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}\} \quad (3.2)$$

are X eigenvectors with corresponding eigenvalues $\{+1, -1\}$. Another widely used single qubit gate is the Hadamard gate (graphical representation on the right)

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{---} \boxed{H} \text{---} \quad (3.3)$$

which produces superposition of states as follows:

$$\begin{aligned} H|0\rangle &= |+\rangle \\ H|1\rangle &= |-\rangle. \end{aligned} \tag{3.4}$$

For what concerns two qubit gates, the *SWAP* gate is represented by

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{array}{c} \text{---} \\ \text{---} \end{array} \boxed{\text{SWAP}} \begin{array}{c} \text{---} \\ \text{---} \end{array} \tag{3.5}$$

and it has the effect of swapping the states of two qubits $|\psi\rangle$ and $|\phi\rangle$ as

$$SWAP|\psi\rangle|\phi\rangle = |\phi\rangle|\psi\rangle \tag{3.6}$$

Another important two qubit gate is the two qubit CNOT gate,

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{array}{c} \bullet \\ \text{---} \\ \oplus \\ \text{---} \end{array} \tag{3.7}$$

which flips the state of the second qubit when the first qubit is in $|1\rangle$

$$CNOT \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle \right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{3.8}$$

3.1.2 Quantum Parallelism

The speed-up of quantum algorithm is achieved through quantum parallelism i.e. the ability of the quantum computer to exist in a superposition of qubits states.

Consider the function $f : \{0, 1\} \rightarrow \{0, 1\}$ and a two qubits register initially both in the state $|0\rangle$. With an appropriate sequence of gates it is possible to construct the unitary operator U_f which acts as follows

$$U_f |x, 0\rangle = |x, f(x)\rangle \tag{3.9}$$

Suppose now to combine the unitary U_f with a Hadamard gate as in Fig.3.1.

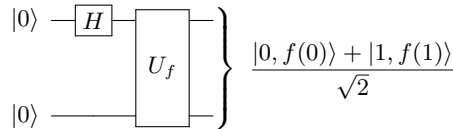


Fig. 3.1: Circuit showing quantum parallelism.

With a single operation it is possible to obtain a quantum superposition that involves the function f on both inputs 1 and 0. The drawback is that a final measurement of the qubits state yields only one evaluated value. However, the procedure of quantum parallelism has a key role as an intermediate step in quantum algorithm.

3.1.3 Quantum Algorithms

Quantum Fourier Transform

The quantum Fourier transform (QFT) is the quantum analogue of the classical discrete Fourier transform (DFT) which maps a vector $x = (x_0, x_1, \dots, x_{N-1}) \in \mathbb{C}^N$ to a new vector $y = (y_0, y_1, \dots, y_{N-1}) \in \mathbb{C}^N$, where $N = 2^n$, as follows

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{kj}{N}} \quad (3.10)$$

The QFT acts on the 2^n amplitudes of the quantum state

$$|\psi\rangle = \sum_{j=0}^{N-1} a_j |j\rangle \quad (3.11)$$

and maps them to the new amplitudes of the quantum state $|\phi\rangle$

$$|\phi\rangle = \sum_{k=0}^{N-1} b_k |k\rangle \quad \text{where} \quad b_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi i \frac{kj}{N}} \quad (3.12)$$

The quantum gate associated to the *QFT* has the following matrix representation, where $\omega = e^{\frac{2\pi i}{N}}$

$$\mathcal{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix} \quad (3.13)$$

The discrete Fourier transform on 2^n amplitudes can be implemented as a quantum circuit of only $O(n^2)$ gates while the classical discrete Fourier transform takes $O(n2^n)$ on a classical computer. The exponential speed-up of the quantum Fourier makes it one of the most widely used subroutines of many quantum algorithms.

Quantum Phase Estimation

Given a unitary matrix U and a quantum state $|\psi\rangle$ such that

$$U|\psi\rangle = e^{2\pi i\theta} |\psi\rangle \quad (3.14)$$

the quantum phase estimation algorithm finds the value of the eigenvalue i.e. of θ with high probability within an error ε , employing $O(1/\varepsilon)$ controlled unitary operations. The quantum circuit associate to the algorithm is constructed as in Fig3.2. Consider a register of n qubits in the state $|0\rangle$ and another register of m qubits initially in the state $|\psi\rangle$ for a global input state of $|0\rangle^{\otimes n} |\psi\rangle$.

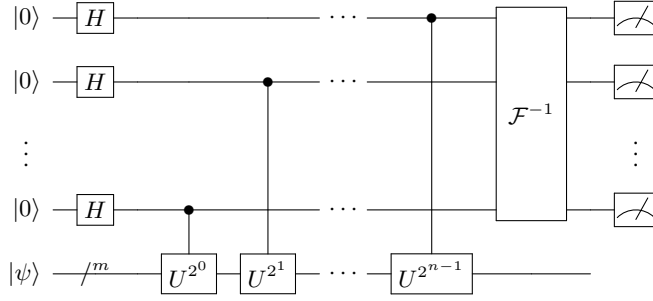


Fig. 3.2: Circuit realizing the quantum phase estimation.

An n -qubit Hadamard gate $H^{\otimes n}$ is applied to the first n qubits obtaining the state

$$\frac{1}{\sqrt{2^n}} (|0\rangle + |1\rangle)^{\otimes n} |\psi\rangle \quad (3.15)$$

and subsequently n control unitary gates $Control - U^{2^j}$ with $0 \leq j \leq n - 1$ are applied to $|\psi\rangle$. The global state at this stage can be written as

$$\frac{1}{\sqrt{2^n}} \underbrace{\left(|0\rangle + e^{2\pi i 2^{n-1} \theta} |1\rangle \right)}_{\text{qubit 1}} \otimes \underbrace{\left(|0\rangle + e^{2\pi i 2^1 \theta} |1\rangle \right)}_{\text{qubit 2}} \otimes \cdots \otimes \underbrace{\left(|0\rangle + e^{2\pi i 2^0 \theta} |1\rangle \right)}_{\text{qubit n}} \otimes |\psi\rangle \quad (3.16)$$

which could be rewritten as

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta} |k\rangle |\psi\rangle \quad (3.17)$$

The last gate is an inverse QFT, \mathcal{F}^{-1} , on the first n qubits which leads to the state

$$\frac{1}{2^n} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta} e^{-2\pi i \frac{kj}{2^n}} |j\rangle |\psi\rangle \quad (3.18)$$

At this point the value of $\theta \in [0, 1]$ is approximated as

$$2^n \theta = a + 2^n \delta, \quad (3.19)$$

where a is the nearest integer to $2^n \theta$ and $2^n \delta$ satisfies $0 \leq |2^n \delta| \leq \frac{1}{2}$. In case error $\delta = 0$, measuring the first n qubits always leaves them in the state $|a\rangle = |2^n \theta\rangle$ from which it is possible to read off the phase θ and hence obtain an eigenvalue. If the error $\delta \neq 0$ instead the algorithm yields the correct phase θ with probability $Pr(|a\rangle) \geq \frac{4}{\pi^2} \approx 0.405$ [47].

Grover's Algorithm

Grover's algorithm is a quantum search algorithm usually described in the context of searching an element in an unstructured database. More formally, it is considered an unknown or extremely complex function $f(x) : \{0, 1\}^n \rightarrow \{0, 1\}$ that returns 1 only for a particular unknown instance $\tilde{x} \in \{0, 1\}^n$. Grover's search algorithm uses internal calls to a quantum oracle O_f , which is a black-box quantum gate (or a set of quantum gates) acting on n input qubits that evaluates the function f at a given x . In order to find \tilde{x} , Grover's search only uses $\sqrt{2^n}$ calls to the oracle while a classical machine need in the worst case 2^n function evaluations. Grover's algorithm hence provides a quadratic speed-up with respect to the classical case and it is constructed as follows.

Consider at first the function f , it can be written as

$$f(x) = \begin{cases} 1 & x = \tilde{x} \\ 0 & \text{otherwise} \end{cases} \quad (3.20)$$

The oracle O_f evaluating this function is a quantum gate that flips the amplitude sign of the state to which it is applied in case $f(x) = 1$

$$O|x\rangle = (-1)^{f(x)}|x\rangle. \quad (3.21)$$

The algorithm can be represented by the quantum circuit shown in Fig. (3.3).

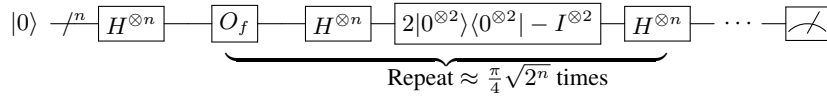


Fig. 3.3: Circuit representation of the Grover's search algorithm.

The n qubits in the state $|0\rangle^{\otimes n}$ are put in a superposition thanks to the n -qubit Hadamard gate $H^{\otimes n}$ resulting in the state

$$\frac{1}{\sqrt{2^n}}(|00\dots 0\rangle + |10\dots 0\rangle + \dots + |11\dots 1\rangle) \quad (3.22)$$

Subsequently the following two steps are applied number of times $\approx \frac{\pi}{4} \sqrt{2^n}$.

Step I. Application of the quantum oracle O_f gate

$$|\psi_2^{(i)}\rangle = O_f|\psi_1\rangle. \quad (3.23)$$

Step II. Application of the gate

$$H^{\otimes n} (2|0^{\otimes n}\rangle\langle 0^{\otimes n}| - I^{\otimes n}) H^{\otimes n} \quad (3.24)$$

Finally a measurement is performed and the solution states $|\tilde{x}\rangle$ is found with a high probability.

3.2 Adiabatic Quantum Computation and Quantum Annealing

Adiabatic quantum computation (AQC) is a model of quantum computation which relies on the adiabatic theorem in order to perform calculations. It is closely related to Quantum Annealing (QA), a meta-heuristic technique used to find the global minimum of an objective function which exploits the power of quantum fluctuations to explore the space of possible candidate solutions [48]. Like in simulated annealing, whose temperature parameter determines the probability of jumping between local minima, in QA the tunnelling field strength is responsible for the transition to better minima.

In this chapter we present an overview of both AQC and QA as well as a discussion of the properties of the current available QA hardware provided by the Canadian company D-Wave.

3.2.1 Adiabatic Quantum Computation

Adiabatic quantum computation (AQC) is a scheme of quantum computation where a quantum system is initially prepared in the ground state, i.e. the state with lowest energy of a simple initial Hamiltonian which is gradually deformed to reach a desired complicated Hamiltonian. The evolution must be very slow in order to assure that the system remains in its ground state throughout the whole process. AQC is in fact based on the adiabatic theorem introduced by Born and Fock in which states the following: *“A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalues and the rest of the Hamiltonian spectrum”*[49].

More in detail, consider an initial Hamiltonian $H(0) = H_0$ whose lowest energy eigenstate $|G_0\rangle$ is easy to obtain and a problem Hamiltonian H_P which might be easy to construct but whose ground state $|G_P\rangle$ is difficult to compute. Usually $|G_P\rangle$ is also the solution of an optimization problem which is exponentially hard to find with classical algorithms. The time dependent Hamiltonian $H_{AQC}(t)$ of the whole AQC process could be expressed as follows

$$H_{AQC}(t) = \left(1 - \frac{t}{\tau}\right) H_0 + \frac{t}{\tau} H_P \quad (3.25)$$

where time t goes from $0 \rightarrow \tau$ with τ being the adiabatic time scale.

Assume that the system ground state $|G(t)\rangle$ calculated at generic instant t is non-degenerate (with associated ground energy $E_0(t)$) and that the first excited state at given instant t is $|1(t)\rangle$ with associated energy $E_1(t)$. If we denote the generic state of the system as $|\psi(t)\rangle$, then the quantum adiabatic theorem states that the condition for $|\psi(t)\rangle$ to be arbitrarily close to the instantaneous ground state $|G(t)\rangle$, i.e.

$$|\langle G(t)|\psi(t)\rangle|^2 = 1 - \epsilon^2 \quad (3.26)$$

for an arbitrary small $\epsilon \ll 1$, is given by the following relation

$$\frac{\max_{0 \leq t \leq \tau} \left| \langle 1(t) | \frac{dH(t)}{dt} | G(t) \rangle \right|}{\min_{0 \leq t \leq \tau} \Delta(t)^2} = \epsilon. \quad (3.27)$$

The term $\Delta(t) = E_1(t) - E_0(t)$ denotes the energy gap between the instantaneous first excited and ground states and the min and max are calculated among the whole time interval $t \in [0, \tau]$. In other words, since the numerator of Eq. 3.27 is proportional to $1/\tau$, the upper bound on the rate at which the Hamiltonian can be evolved maintaining the adiabatic condition is

$$\tau \propto \frac{1}{\epsilon \times \min_{0 \leq t \leq \tau} \Delta(t)^2}. \quad (3.28)$$

Note that a bottleneck to AQC is due to the fact that if the energy gap $\Delta(t) \rightarrow 0$ as the system size increases then the runtime for the entire algorithm τ increases, according to Eq. 3.28, quadratically with the size of the gap [50].

3.2.2 Quantum Annealing

Different algorithms can be considered for finding a global minimum of a given objective function when we are in the presence of several local minima.

Simulated Annealing (SA), for example, is a very popular algorithm used to solve this kind of optimization problem that imitate the thermalization of a physical system releasing energy to a cooling reservoir. In SA, the initial high temperature induces thermal excitations which can allow the system to escape local minima, then the cooling pushes the system toward nearby low energy states. However, the thermal transition probability depends only on the height h of the potential wall to overcome $e^{-\frac{h}{k_B T}}$, which means that SA in general fails when it has to deal with very high barriers.

A Quantum Annealing approach was proposed to find the global minimum of a given objective function: a heuristic similar to SA but that exploits quantum tunnelling in order to escape local minima. The advantage is that the tunnelling probability depends both on the height h and the width w of the potential barrier $e^{-\frac{w\sqrt{h}}{\Gamma}}$, where Γ is the transverse field strength. This gives to QA the ability to move in an energy landscape where local minima are separated by tall barriers, provided that they are narrow enough.

The time-dependent Hamiltonian used in the process of QA is similar to the one of AQC

$$H(t) = -A(t)H_0 + B(t)H_P, \quad (3.29)$$

where H_0 is the initial Hamiltonian, H_P is the problem Hamiltonian and the functions $A(t)$ and $B(t)$ define the annealing schedule. The evolution is parametrized in time $t \in [0, T_{QA}]$, where T_{QA} is the total annealing time. The functions $A(t)$ and $B(t)$ are such that, at the beginning of the annealing, the transverse field strength $A(t)$ is large i.e. $A(0) \gg B(0)$ and the dynamics is dominated by the tunnelling Hamiltonian H_0 ; at the end of the annealing the functions satisfy the relation $A(T_{QA}) \ll B(T_{QA})$ and the dynamic is entirely governed by the problem Hamiltonian H_P .

The main difference between AQC and QA is that AQC require a unitary and adiabatic (slow) evolution of a closed quantum system, while QA allows for a non-adiabatic (fast) evolution of an open quantum system which is usually few millikelvin above absolute zero. This means that, since the adiabatic theorem does not hold for QA, we are usually not guaranteed to end up in the ground state of the system. For this reason, the QA process is usually repeated a number of times to increase the probability of reaching the ground state. Moreover, it often happens that the configurations that are returned are very close to the lowest-energy state and hence are still very interesting.

D-Wave Quantum Annealer

In 2011, the Canadian company *D-Wave Systems* announced the first commercial quantum annealer on the market by the name *D-Wave One* which used a 128 qubit processor chipset. Nowadays the company provides cloud access to the current architecture with a 2048-qubit QPU known as *D-Wave 2000Q*.

The device solves combinatorial optimization problems expressed as Quadratic Unconstrained Binary Optimization (QUBO) problems of the kind

$$O(x) = \sum_i h_i x_i + \sum_{i>j} J_{i,j} x_i x_j \quad (3.30)$$

where $x_i \in \{0, 1\}$ are binary variables and parameters h_i and J_{ij} encode the optimization problem.

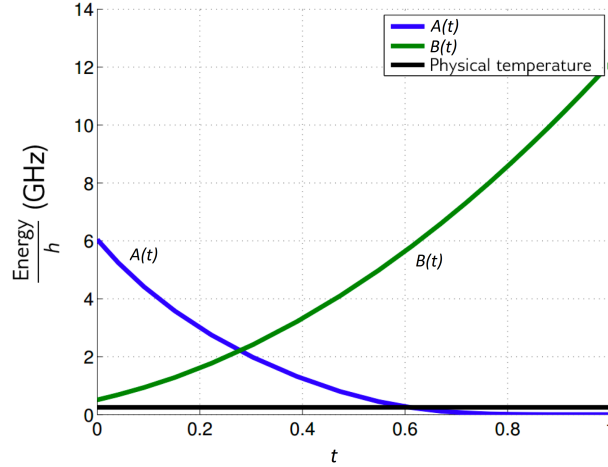


Fig. 3.4: D-Wave annealing parameters.

Hence the D-Wave device aims at solving such problem via a QA process that is described by the following Hamiltonian

$$\mathcal{H}_{QA} = A(s) \sum_i \hat{\sigma}_x^{(i)} + B(s) H_P \quad (3.31)$$

where annealing parameters $A(s)$ and $B(s)$ are smooth functions as those represented in Fig.3.4.

The term H_P is the Ising Hamiltonian associated to the objective function $O(x)$

$$H_P = \sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \quad (3.32)$$

where $\hat{\sigma}_x^{(i)}$ and $\hat{\sigma}_z^{(i)}$ are respectively the x and z Pauli operators. Coefficients h_i are realized by applying an external magnetic field to site i while J_{ij} are coupling interaction between the spin in sites i and the one in site j which can be either ferromagnetic ($J_{ij} < 0$, that tends to align spins) or anti-ferromagnetic ($J_{ij} > 0$, that tends to misalign spins). Despite $\sigma^{(i)} \in \{+1, -1\}$ are spin variables, it is possible to show that the Ising energy minimization is equivalent to the QUBO problem, this means that solving the latter corresponds to finding the ground state energy of the associated Ising model.

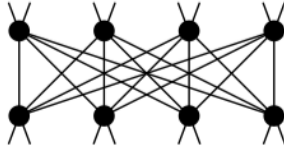


Fig. 3.5: D-Wave unit cell.

Moreover, in order to solve an instance of a QUBO problem with a D-Wave machine we need to embed the logical formulation of a given problem (i.e. the logical Ising problem) to the physical fixed architecture of the quantum processor (i.e. the physical Ising problem). This architecture is composed by a matrix of unit cells (Figure 3.5) that is a set of 8 qubits arranged in a bipartite graph. These unit cells are connected in a structure called *chimera graph*. Such embedding is found using a heuristic technique described in [51] and available in the D-Wave python libraries as a function that goes under the name of *find_embedding*.

3.3 Topological Quantum Computation

Topological Quantum Computation (TQC) [63, 64, 65] is based on the existence of some special particles, called *anyons*, whose statistics substantially differ from the more common physical particles observed in nature. Anyons can exist in two dimensional quantum systems and can not be identified neither with bosons nor with fermions. Their behaviour could be described by the statistics generated exchanging one particle with another since this exchange rotates the quantum state of the system producing arbitrary phases [66] (for Abelian anyons) or even unitary operations (for non-Abelian anyons). By exchanging non-Abelian anyons, it is possible to obtain significant changes in the state of the system, which can be used to perform quantum computation.

Quantum computers can benefit from the use of topological properties as far as they can guarantee a form of robustness [63]. This is possible because in a *topological quantum computer* information is encoded in the collective states of anyons, which are naturally protected from decoherence by their braiding behaviour. In the following we give an explanation of the basic features of the TQC computational paradigm.

3.3.1 Mathematical Background

In this section we briefly review the main concepts in Topology that are relevant for the work presented in this thesis, namely those of knots/links, braiding and related results.

Knot theory [53, 54] studies the topological properties of mathematical knots and links. A knot is an embedding of a circle in the 3-dimensional Euclidean space \mathbb{R}^3 , up to continuous deformations, and a link is a collection a knots that may be linked or knotted together.

A fundamental question in knot theory is whether two knot diagrams, i.e. projections of knots on the plane, represent the same knot or rather they are distinct.

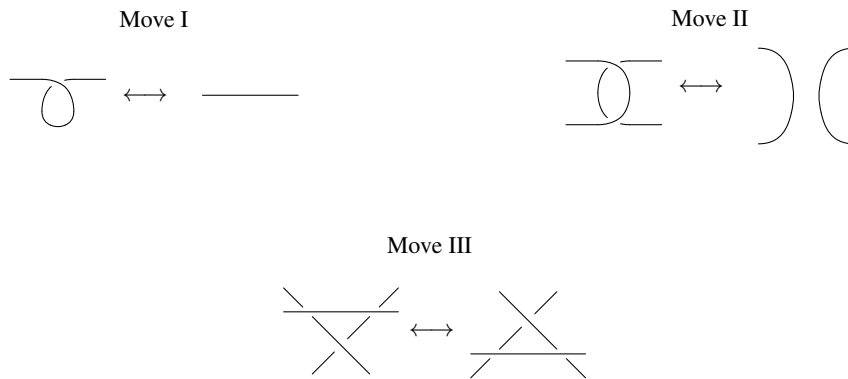


Fig. 3.6: The Reidemeister moves

The Reidemeister theorem [55] says that two links can be continuously deformed into each other if and only if any diagram of one can be transformed into a diagram of the other by a sequence of moves called Reidemeister moves [56]. If there exists such a transformation the two links are said to be isotopic. The Reidemeister moves can be of three types, as depicted in Figure 3.6. Move I undoes a twist of a single strand, move II separates two unbraided strands and finally move III slides a strand under a crossing.

A powerful knot invariant is the Jones polynomial $V_L(A)$ [57] which is a Laurent polynomial in the variable A with integer coefficients. Given two links L_1 and L_2 and their respective Jones polynomials

$V_{L_1}(A)$ and $V_{L_2}(A)$, the following relation holds true:

$$L_1 = L_2 \Rightarrow V_{L_1}(A) = V_{L_2}(A) \text{ or, equivalently, } V_{L_1}(A) \neq V_{L_2}(A) \Rightarrow L_1 \neq L_2.$$

A useful formulation of this polynomial due to Kauffman [58, 59] is given in terms of the so-called bracket polynomial or Kauffman bracket, defined in the following section.

3.3.2 Kauffman Bracket

The Kauffman bracket of any (unoriented) link diagram D , denoted $\langle L \rangle$, is a Laurent polynomial in the variable A , characterized by the three rules:

1. $\langle \bigcirc \rangle = 1$, where \bigcirc is the standard diagram of the loop
2. $\langle D \sqcup \bigcirc \rangle = (-A^2 - A^{-2}) \langle D \rangle = d \langle D \rangle$, where \sqcup denotes the distant union¹ and $(-A^2 - A^{-2}) = d$.
3. $\langle \begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array} \rangle = A \langle \begin{array}{c} \diagdown \diagup \\ \diagdown \diagup \end{array} \rangle + A^{-1} \langle \begin{array}{c} \diagup \diagdown \\ \diagup \diagdown \end{array} \rangle$

where $\begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array}$ and $\begin{array}{c} \diagdown \diagup \\ \diagdown \diagup \end{array}$ represent some regions of link diagrams where they differ.

Rule 3 expresses the *skein relation*: it takes in input a crossing r_i and dissolves it generating two new links that are equal to the original link except for r_i , and therefore with a smaller number of crossings. By applying it recursively to a link we obtain at the end a number of links with no crossings but only simple loops, though this number is exponential in the number of crossings. Rule 1 and Rule 2 show how to calculate the polynomial after the decomposition achieved by applying Rule 3. Note that the Kauffman bracket of a link diagram is invariant under Reidemeister moves II and III but it is not invariant under move I. For every two links L and M , the distant union $L \sqcup M$ has the property:

$$\langle L \sqcup M \rangle = (-A^2 - A^{-2}) \langle L \rangle \langle M \rangle = d \langle L \rangle \langle M \rangle$$

The Kauffman Bracket of the Hopf Link

We show here the calculation of the Kauffman bracket for the simplest non-trivial link with more than one component, i.e. the Hopf link depicted below [60].



By applying Rule 3 to the upper crossing we get

$$\langle \text{Hopf Link} \rangle = A \langle \text{Link with one crossing} \rangle + A^{-1} \langle \text{Link with one crossing} \rangle$$

Now we use also Rules 1 and 2 to compute the new two brackets separately:

$$\langle \text{Link with one crossing} \rangle = A \langle \text{Link with two crossings} \rangle + A^{-1} \langle \text{Link with one crossing} \rangle = Ad + A^{-1} = (-A)^3$$

$$\langle \text{Link with one crossing} \rangle = A \langle \text{Link with one crossing} \rangle + A^{-1} \langle \text{Link with two crossings} \rangle = A + dA^{-1} = (-A)^{-3}$$

¹ The distant union of two arbitrary links L and M , denoted by $L \sqcup M$ is obtained by first moving L and M so that they are separated by a plane, and then taking the union.

Finally we get

$$\langle \text{Hopf link} \rangle = A \langle \text{unknot} \rangle + A^{-1} \langle \text{unknot} \rangle = -A^4 - A^{-4}$$

It is worth noting that the Hopf link calculated here and the one obtained by reversing all the crossings have the same Kauffman brackets, i.e.

$$\langle \text{Hopf link} \rangle = \langle \text{Hopf link} \rangle$$

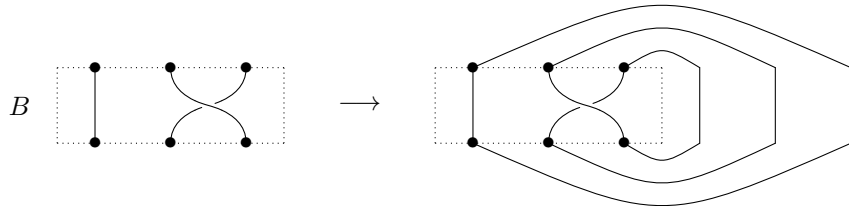
3.3.3 Braids and Links

A braid can be visualised as an intertwining of some number of strands, i.e. strings attached to top and bottom bars such that each string never turns back.

Given n strands, the operator σ_i performs a crossing between the i^{th} strand and the $(i + 1)^{th}$, keeping the former above the latter. In a similar way, the operator σ_i^{-1} denotes a crossing of the i^{th} strand below the $(i + 1)^{th}$. A generic braid B on n strings is obtained by iteratively applying the σ_i and σ_i^{-1} operators in order to form a braid-word, e.g. $\sigma_1\sigma_2\sigma_1^{-1}\sigma_4$. It is well-know that the operators σ_i and σ_i^{-1} on n strands define a group B_n called *braid group* [52].

Markov trace

Given a braid B , its Markov trace is the closure obtained connecting opposite endpoints of B together, as shown below.



The relation between links and open ended strands is defined by two important theorems [61, 62].

Alexander’s theorem

Every link (or knot) can be obtained as the closure of a braid.

The result of the Markov closure of a braid B is a link that we will denote by $L = (B)^{Markov}$. The result of the Markov closure of a braid B is a link that we denote by $L = (B)^{Markov}$.

Markov’s theorem

The closure of two braids B_1 and B_2 gives the same link (or knot) if and only if it is possible to transform one braid into the other by successive applications of the Markov moves:

- 1) conjugation: $B = \sigma_i B \sigma_i^{-1} = \sigma_i^{-1} B \sigma_i$, where $B \in B_n$
- 2) stabilization: $B = B \sigma_n^{-1} = B \sigma_n$, where $\sigma_n, B \sigma_n$ and $B \sigma_n^{-1} \in B_{n+1}$.

3.3.4 Computing with Anyons

In order to perform a topological quantum computation we need to fix an *anyon system*, i.e. a system with a fixed number anyons for which we specify: (1) the type, i.e. the anyon physical charge, (2) the

fusion rules N_{ab}^c (i.e. the laws of interaction), (3) the F -matrices, and (4) the R -matrices. The role of these latter will be made clear in the following.

The *fusion rules*, give the charge of a composite particle in terms of its constituents. The fusion rule $a \otimes b = N_{ab}^c c$ indicates the different ways of fusing a and b into c ; these are exactly N_{ab}^c . Dually, we can look at these rules as *splitting rules* giving the constituent charges of a composite particle.

An anyon type a for which $\sum_c N_{ab}^c > 1$ is called *non-Abelian*. In other words, a non-Abelian anyon is one for which the fusion with another anyon may result in anyons of more than one type. This property is essential for computation because it implies the possibility of constructing non trivial computational spaces, i.e. spaces of dimension $n \geq 1$ of ground states where to store and elaborate information. Such spaces correspond to so-called *fusion spaces*. The fusion space, V_{ab}^c , of a particle c , or dually its splitting space V_c^{ab} , is the Hilbert space spanned by all the different (orthogonal) ground states of charge c obtained by the different fusion channels. The dimension of such a space is called the *quantum dimension* of c ; clearly this is 1 for Abelian anyons.

Considering the dual splitting process, a non-Abelian anyon can therefore have more than one splitting rule that applies to it, e.g. $a \otimes b = c$ and $e \otimes b = c$. Given an anyon of type c we can split it into two new anyons a, b and obtain a tree with root c and a, b as leaves. By applying another rule to a , say $a = c \otimes d$, we will obtain a tree with leaf anyons c, d, b and root c . The same result can also be obtained by splitting the original anyon c into e, b and, supposing that there exists a fusion rule of the form $c \otimes d = e$, we can again split e into the leaves c and d . The two resulting, which have leaf anyons and root anyon of same type and differ only for the internal anyons a, e , represent two orthogonal vectors of the Hilbert space V_c^{cab} . Applying the fusion rules in different order generates other (non orthogonal) trees which have different shapes but contain the same information. This is because the total charge is conserved by locally exchanging two anyons, a property that deserves the ‘topological’ attribute to anyon systems and that determines the fault-tolerance of the quantum computational paradigm based on them.

The idea behind the use of anyons for performing computation is to exploit the properties of their statistical behavior; this essentially means to look at the exchanges of the anyons of the system as a process evolving in time, i.e., looking at an anyon system as a 2+1 dimensional space. This corresponds to *braiding* the threads (a.k.a. world-lines) starting from each anyon of the system. Particle trajectories are braided according to rules specifying how pairs (or bipartite subsystems) behave under exchange. The braiding process causes non-trivial unitary rotations of the fusion space resulting in a *computation*. Equivalently, a topological quantum computation can be seen as a splitting process (creating the initial configuration) followed by a braiding process (the unitary transformation) followed by a fusion process (measuring the final state). The latter essentially consists in checking whether the initial anyons fuse back to the vacuum from which they were created by splitting.

Finally, it is worth underlining that for certain models of anyons, such as the Fibonacci model, it is possible to reproduce any unitary operation to arbitrary accuracy (up to a global phase factor) by braiding anyons, making them universal for quantum computation. Moreover, Non-Abelian anyons are thought to be capable of universal quantum computation by braiding if the square of their quantum dimension is not an integer. Fibonacci anyons have in fact a quantum dimension of $\frac{1+\sqrt{5}}{2}$.

3.3.5 Topological Quantum Computation of the Jones Polynomials

Consider n anyons created in pairs from the vacuum. Each anyonic pair is in the vacuum fusion channel with initial state denoted by $|\psi\rangle$. The final state $\langle\psi|$ corresponds to a fusion of these anyons back into the vacuum [60].

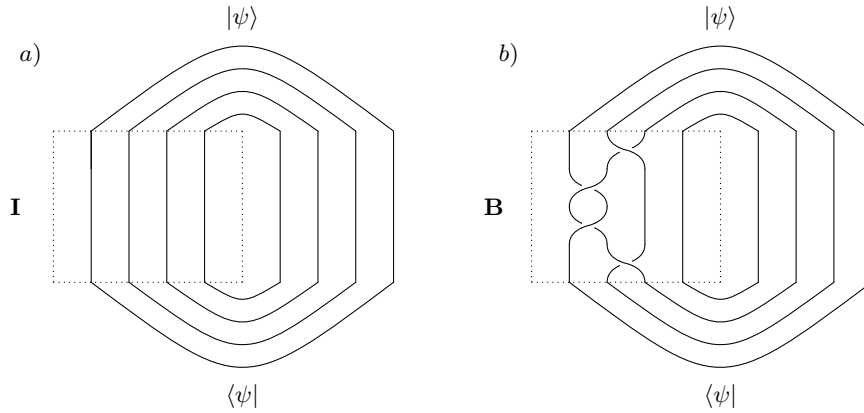


Fig. 3.7: Two anyonic quantum evolutions. In both cases pairs of anyons are created from the vacuum and then fused back into it. In a) no braiding, i.e the identity operator, is performed, in b) some braiding operator is applied.

As shown in Figure 3.7 part *a*, if no braiding is performed on the anyons (**I** stands for the identity), then the probability that they fuse back to the vacuum in the same pairwise order is trivially given by

$$\langle \psi | \mathbf{I} | \psi \rangle = \langle \psi | \psi \rangle = 1.$$

Consider instead the situation represented in Figure 3.7 part *b*, where, after creating $n = 8$ anyons in pairs from the vacuum, we braid half of them with each other to produce the anyonic unitary evolution represented by the operator **B**. In this case, the probability amplitude of fusing the anyons in the same pairwise order to obtain the vacuum state is given by

$$\langle \psi | \mathbf{B} | \psi \rangle = \frac{\langle (B)^{Markov} \rangle}{d^{n-1}}, \quad \text{where } d = (-A^2 - A^{-2}). \quad (3.33)$$

This equation expresses the relation between the probability amplitude of obtaining the vacuum state after the braiding given by the operator **B** and the Kauffman bracket of the link obtained from the Markov trace of braid B , i.e. $(B)^{Markov}$. The Jones polynomial is defined in terms of the Kauffman bracket, which is responsible for the complexity of computing the Jones polynomial. The problem of approximating the Jones polynomials at any fixed root of unity is a BQP problem. In fact it can be solved in polynomial time only by a quantum computer.

Quantum Machine Learning

In the last few years, a new interdisciplinary research topic going under the name of Quantum machine learning (QML) has emerged [15, 79, 80, 83, 84, 85, 86, 170]. Quantum Machine Learning has established itself as one of the most promising applications for quantum computers and Noisy Intermediate Scale Quantum (NISQ) devices. The aim of QML is to merge in different ways quantum computing and data mining techniques in order to achieve improvements in both fields. As shown in Fig. 4.1, it is possible to distinguish four approaches in QML, depending on the nature of the dataset under study and on the computation device being used [88].

		Type of Algorithm	
		classical	quantum
Type of Data	classical	CC	CQ
	quantum	QC	QQ

Fig. 4.1: The first letter refers to whether the system under study is classical or quantum, while the second letter defines whether a classical or quantum information processing device is used.

The Classical-Classical (CC) class refers to ordinary machine learning or machine learning algorithm that are inspired by the formalism of quantum mechanics. Here the dataset represent some classical system and the algorithm used can run on a classical computer [89, 90, 91, 92, 93]. In Classical-Quantum (CQ), algorithms rely on the advantages of quantum computation in order to speed up classical methods of machine learning or improve their performances. Again in this class data is assumed to be classical [94, 95, 96, 97, 98, 99, 100]. On the other hand, Quantum-Classical (QC) refers to the use of classical methods of machine learning to analyse quantum systems [101, 103, 105, 111, 112, 113, 114]. Finally, in Quantum-Quantum (QQ) both the learning algorithm and the system under study are fully quantum.

The aim of this chapter is to review the main approaches, focusing in particular on the more advanced branches of QML which are the QC and CQ sections.

4.1 QC: Machine Learning for Quantum Physics

It is well known that randomness and counter intuitive behaviour are the characteristic traits of quantum mechanics. In this scenario, techniques of ML could be of great help, unveiling patterns in quantum physics problems that are hard to grasp with the human intellect alone. Recently, many ML concepts and tools have been applied to the study of quantum systems. In the following, we discuss some of the most interesting approaches.

One of the main direction of research in this topic involves the use of ML to help understand many-body quantum systems and, in particular, quantum phase transitions. Many-body systems are usually described via phase diagrams showing qualitative changes in the system when its local parameters are varied. In some cases, phase transitions are identified by looking for discontinuities or singularities in the local parameters or its derivatives. However, in general, classification and detection of quantum phase transitions are hard issues in the theory of many-body physics because they often appear at considerably large system sizes and hence the exponential growth of the Hilbert space makes them hard to characterize. In order to avoid expensive computations, it has been proposed ML as a novel approach to provide good solutions in a reasonable amount of time.

An interesting work in this direction [102] shows how to use state-of-the-art supervised learning technique (domain adversarial neural networks) to successfully predict phase transitions in close agreement with standard methods for different many-body systems: the Ising model, the Bose-Hubbard model and the Su-Schrieffer-Heeger model with disorder.

Later the same authors [103] have shown how an unsupervised ML approach based on neural networks can be used to identify new parameters that efficiently capture the physics of phase transitions in systems of interacting quantum particles subjected to a static disordered background potential.

Finally, a recent paper [104] has shown how to identify quantum phase transitions of the axial next-nearest-neighbour Ising (ANNNI) model employing both unsupervised and supervised machine learning techniques like K-NN, Random Forest, Multilayer Perceptron and comparing them to different analytical solutions.

Besides detecting phase transitions, ML has also been applied to several quantum information tasks. A basic example is quantum state tomography (QST). The aim of QST is to carefully measure a quantum system in order to reconstruct its unknown density matrix. QST is a core problem in quantum information which needs an number of resources scaling exponentially with the size of the system in order to perform full tomography.

An interesting approach in this direction [105] employs ML in order to retrieve information about the amount of entanglement in an unknown quantum state. In general, the only way of measuring entanglement is through full quantum tomography. Authors instead propose a scheme where a neural network only requires a polynomial number of measurements to obtain a good approximation of the negativity, which is an entanglement measure.

Finally, in [106] authors extended the use of ML to the problem of quantum process tomography, where the ability of neural networks to model non-Markovian evolution of open quantum systems is investigated.

Another intriguing field of application for ML is the one concerning the discovery of new quantum algorithms, experiments and physical concepts. In 2014, it was proposed a method [107] for quantum algorithm design assisted by machine learning where a learning system, assisted by classical feedback computer, evolves into a quantum algorithm. The authors proved that their system is able to learn how to solve an oracle decision problem, called the Deutsch-Jozsa problem.

Two years later, Krenn et al. [108] found new experimental implementations for the creation and manipulation of complex entangled quantum states using a ML algorithm that exploits asymmetric techniques which are challenging to understand intuitively.

More recently [113] it was proposed a ML algorithm based on genetic algorithm and neural networks for the design of quantum optics experiments. The task in this case is to produce specific quantum states

such as Schrödinger cat states and cubic phase states which the algorithm successfully found with fidelity of over 96%.

Finally, the use of ML has also been proposed for the discovery of new unbiased physical concepts from experimental data. In their work [114], the authors used a neural network to compress data into a simple representation, which is used by the network itself to gain information about the quantum system. More specifically, given a dataset containing outcomes of quantum measurement, the network is able to recognize how many degrees of freedom are needed to describe the quantum state.

Other remarkable applications of ML to quantum information and computation concern the validation of quantum devices [101], the reduction of measurement errors for trapped-ion qubits [109] and the improvement of quantum error correction schemes [110].

4.2 CQ: Quantum Computing for Machine Learning

Inside the CQ section, the main idea is to use quantum mechanics in order to obtain a computational advantage for a specific class of ML techniques. In this section we introduce at first the tools, i.e. the quantum subroutines that are commonly used in this quantum machine learning algorithms. Hence the focus will be on the explanation of the main algorithm belonging to this class.

4.2.1 Essential Tools in QML

Amplitude Encoding

Many quantum machine learning algorithms are based on the idea of amplitude encoding, i.e. an encoding of the inputs of computations in the amplitudes of a quantum state of n qubits. Since a state of n qubits is described by $N = 2^n$ complex amplitudes, such an encoding automatically produces an exponential compression of the data.

More in details, a classical vectors $\vec{x} \in \mathbb{R}_N$ defined as

$$\vec{x} = (x_1, x_2, x_3, \dots, x_N) \quad (4.1)$$

could be represented by means of a quantum state $|\vec{x}\rangle$ on n qubits as

$$|\vec{x}\rangle = \frac{1}{|\vec{x}|} \sum_{k=1}^N x_k |k\rangle \quad (4.2)$$

where components of the vectors \vec{x} become amplitudes of the quantum state.

However, a crucial bottleneck of many algorithms that use amplitudes encoding is the preparation of states like $|\vec{x}\rangle$, which often requires the initialization of a quantum state whose amplitudes reflect the features of an entire dataset. While for specific cases efficient methods for state preparation are known, usually this step hides the complexity of the quantum algorithm.

A Quantum Random Access Memory (QRAM) is a device which is able of answering queries in quantum superposition. Specifically, the QRAM uses $O(N)$ memory cells to store the x_i values with $i = 1, 2, \dots, N$ and achieve the following transformation with a runtime of $O(\log^2 N)$ [86, 87]

$$\sum_{i=1}^N \alpha_i |i\rangle |0\rangle \rightarrow \sum_{i=1}^N \alpha_i |i\rangle |x_i\rangle \quad (4.3)$$

where $|i\rangle$ is the index register. Moreover a QRAM can produce in $O(\log N)$ the state $|\vec{x}\rangle$ of Eq. 4.2 under the condition that the state to prepare is sufficiently uniform [80]. However, despite there are many ideas of possible QRAM architectures [80], the question of whether a hardware realising a QRAM can be built is still open.

Swap Test

The Swap test refers to a quantum circuit shown in Fig.4.2 that is often used as a subroutine of many quantum machine learning algorithms. The circuit takes in input the quantum states $|\psi\rangle$ and $|\phi\rangle$ as well as an ancillary qubit, initially in the state $|0\rangle_a$

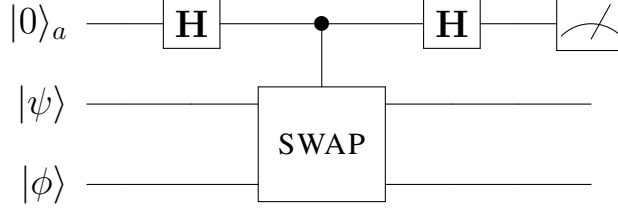


Fig. 4.2: Swap test circuit.

Then a Hadamard gate is applied to the ancillary qubit $|0\rangle_a$ followed by a controlled swap. After the final Hadamard gate on the ancilla, the global state of the system can be written as

$$\frac{1}{2} [|0\rangle_a (|\psi\rangle|\phi\rangle + |\phi\rangle|\psi\rangle) + |1\rangle_a (|\psi\rangle|\phi\rangle - |\phi\rangle|\psi\rangle)] \quad (4.4)$$

The probability of measuring the ancilla in the ground state is given by

$$P(|0\rangle_a) = \frac{1}{2} + \frac{1}{2} |\langle\psi|\phi\rangle|^2. \quad (4.5)$$

Hence it is possible to estimate the probability $P(|0\rangle_a)$ by repeated measurement of the ancilla and obtain the value of the fidelity, i.e. the scalar product between the two quantum states

$$|\langle\psi|\phi\rangle| = \sqrt{2P(|0\rangle_a) - 1} \quad (4.6)$$

Moreover, the time complexity of the swap test, given by the number of elementary gates such as controlled-swap gates, increases linearly with the number of qubits that constitute the quantum states $|\psi\rangle$ and $|\phi\rangle$.

Quantum Algorithm for Linear Systems of Equations

The quantum algorithm for linear systems of equations [121], usually abbreviated as HHL algorithm is a quantum algorithm for solving linear systems which provides a speedup over its classical counterpart. The problem the algorithm solves is the following, given a $N \times N$ Hermitian matrix A and a vector \vec{b} , we want to find the solution \vec{x} satisfying the equation

$$A\vec{x} = \vec{b} \quad (4.7)$$

The quantum algorithm aims at finding the solution $|x\rangle$ of the equation

$$\hat{A}|x\rangle = |b\rangle \quad (4.8)$$

starting from the application of the unitary operator $e^{-i\hat{A}t}$, obtained via Hamiltonian simulation performed in $O(\log N)$ steps [122], to the quantum state

$$|b\rangle = \sum_{i=1}^N b_i |i\rangle. \quad (4.9)$$

Then, an ancillary qubit in the state $|0\rangle$ is added to the system and the state $|b\rangle$ is decomposed in the eigenbasis $|e_i\rangle$ of \hat{A} using the quantum phase estimation algorithm [123] which retrieves the eigenvalues λ_i . The state after the decomposition is

$$\sum_{i=1}^N \beta_i |e_i\rangle |\lambda_i\rangle \quad (4.10)$$

where $|b\rangle = \sum_{i=1}^N \beta_i |e_i\rangle$. Now applying an inversion of the eigenvalue with a controlled rotation and uncomputing the eigenvalue qubit we obtain the desired state $|x\rangle$

$$\sum_{i=1}^N \frac{\beta_i}{\lambda_i} |e_i\rangle = \hat{A}^{-1} |b\rangle = |x\rangle. \quad (4.11)$$

Note that, provided that the matrix A is sufficiently sparse with low condition number

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, \quad (4.12)$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ are maximal and minimal singular values of A , then the algorithm has a runtime of $O(\log(N)d^2\kappa^2)$, where N is the number of variables in the linear system and d is the sparsity. The fastest classical algorithm for computing vector \vec{x} takes $O(N\kappa d)$. The apparent exponential speed-up over the best classical algorithm does not actually yield any computational improvement since the quantum algorithm outputs the solution in the form of a quantum state $|x\rangle$. Hence, in order to reconstruct the classical vector \vec{x} , a sampling procedure that would neutralize any computational advantage is required.

4.2.2 Quantum K-NN and K-Means

In both the quantum versions of k-NN and k-means [81] a quantum advantage is obtained thanks to the more efficient calculation of euclidean distances between vectors on a quantum computer. In particular, given two vectors \vec{a} and \vec{b} in \mathbb{R}_N , while a standard classical algorithm takes $O(N)$ to calculate $|\vec{a} - \vec{b}|$, the quantum algorithm based on the swap test runs in $O(\log(N))$.

The algorithm starts with the representation of the vectors in terms of quantum states

$$|a\rangle = \frac{1}{|\vec{a}|} \sum_{i=1}^N a_i |i\rangle. \quad (4.13)$$

$$|b\rangle = \frac{1}{|\vec{b}|} \sum_{j=1}^N b_j |j\rangle. \quad (4.14)$$

Then, using a QRAM-like device the two following states are initialized.

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0, a\rangle + |1, b\rangle) \quad (4.15)$$

$$|\phi\rangle = \frac{1}{\sqrt{Z}} (|\vec{a}| |0\rangle - |\vec{b}| |1\rangle) \quad (4.16)$$

where $Z = |\vec{a}|^2 + |\vec{b}|^2$. Evaluating $|\langle\psi|\phi\rangle|^2$ via swap test leads to an estimation of the euclidean distance

$$\left| \vec{a} - \vec{b} \right|^2 = 2Z |\langle \psi | \phi \rangle|^2 \quad (4.17)$$

In a recent work [82], it was proposed a classical quantum-inspired algorithm for clustering that uses sampling strategy to obtain $\left| \vec{a} - \vec{b} \right|$. The runtime of the algorithm is polylogarithmic in input size, matching the runtime of the quantum algorithms with only quadratic slowdown.

4.2.3 Quantum SVM

The first quantum approach to SVM is due to Anguita et al. [117]. In their work, they consider a discretized version of the SVM that also takes into account the generalization error of the classifier. This setting inhibits the use of well-known quadratic programming algorithms and optimization can turn into a problem with NP complexity.

Authors propose to represent different configurations of the α_i as quantum states $|\alpha_0 \alpha_1 \dots \alpha_M\rangle$ and then use Grover quantum algorithm in order to perform an exhaustive search over the configuration space and find the maximum of the cost function Eq. 1.8 in $O(\sqrt{2^M})$ instead of $O(2^M)$.

A different approach was proposed by Rebentrost, Mohseni and Lloyd [118], which presented a completely new quantum algorithm that realizes SVM on a circuit based quantum computer. This formulation became very popular in the last few years and it is often addressed as the Quantum SVM (QSVM) algorithm. The starting point to understand QSVM is the amplitude encoding of classical input training vectors \vec{x} into quantum states $|\vec{x}\rangle$ as

$$|\vec{x}\rangle = \frac{1}{|\vec{x}|} \sum_{k=1}^N x_k |k\rangle \quad (4.18)$$

Authors claim this whole set of M states could in principle be constructed querying a Quantum Random Access Memory (QRAM), which uses $O(MN)$ hardware resources and $O(\log MN)$ operations to obtain them [119].

In the preliminary step of the QSVM algorithm, it is exploited the fact that a dot product can be estimated faster using QRAM and repeating the SWAP test algorithm on a quantum computer [120]. More precisely, if the desired accuracy of the estimation is ϵ , then the overall complexity of evaluating a single dot product $\vec{x}_i^T \vec{x}_j$ is $O(\epsilon^{-1} \log N)$. Calculating the kernel matrix takes therefore $O(M^2 \epsilon^{-1} \log N)$ instead of $O(M^2 N \log(1/\epsilon))$ required in the classical case.

However, the main idea of QSVM algorithm is to use the LS-SVM of Eq. 1.11 and rewrite it in terms of quantum states as

$$\hat{F} |b, \vec{\alpha}\rangle = |\vec{y}\rangle \quad (4.19)$$

where $\hat{F} = F/\text{tr}(F)$ with $\|F\| \leq 1$, parameters $\vec{\alpha}$ and b identify the decision hyperplane and \vec{y} denotes the binary class labels. At this point it is applied the efficient quantum matrix inversion [121] in order to generate the quantum state $|b, \vec{\alpha}\rangle$. This algorithm at first requires the simulation of matrix exponential $e^{-i\hat{F}\Delta t}$, which can be performed in $O(\log N)$ steps [122].

Secondly, we can add an ancillary qubit initially in the state $|0\rangle$ and use the quantum phase estimation algorithm [123] to express the state $|\vec{y}\rangle$ in the eigenbasis $|e_i\rangle$ of \hat{F} and store an approximation of the eigenvalues λ_i of \hat{F} in the ancilla

$$|\vec{y}\rangle |0\rangle \rightarrow \sum_{i=1}^{M+1} \langle e_i | \vec{y} \rangle |e_i\rangle |\lambda_i\rangle \quad (4.20)$$

Now apply an inversion of the eigenvalue with a controlled rotation and un-compute the eigenvalue qubit to obtain

$$\sum_{i=1}^{M+1} \frac{\langle e_i | \vec{y} \rangle}{\lambda_i} |e_i\rangle = \hat{F}^{-1} | \vec{y} \rangle = |b, \vec{\alpha}\rangle. \quad (4.21)$$

In the training set basis, the solution state for the LS-SVM is

$$|b, \vec{\alpha}\rangle = \frac{1}{b^2 + \sum_{k=1}^M \alpha_k^2} \left(b |0\rangle + \sum_{k=1}^M \alpha_k |k\rangle \right). \quad (4.22)$$

The process of classifying new data $|\vec{x}\rangle$ requires the implementation of the state $|\tilde{u}\rangle$ obtained by calling the quantum data oracle using $|b, \vec{\alpha}\rangle$

$$|\tilde{u}\rangle = \frac{1}{\left(b^2 + \sum_{k=1}^M \alpha_k^2 |\vec{x}_k|^2\right)^{\frac{1}{2}}} \left(b |0\rangle |0\rangle + \sum_{k=1}^M |\vec{x}_k| \alpha_k |k\rangle |\vec{x}_k\rangle \right) \quad (4.23)$$

and also the query state

$$|\tilde{x}\rangle = \frac{1}{M|\vec{x}|^2 + 1} \left(|0\rangle |0\rangle + \sum_{k=1}^M |\vec{x}| |k\rangle |\vec{x}\rangle \right). \quad (4.24)$$

The classification is obtained calculating the inner product $\langle \tilde{x} | \tilde{u} \rangle$ via a swap test [120]. With the help of an ancillary qubit, the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a |\tilde{u}\rangle + |1\rangle_a |\tilde{x}\rangle)$ is constructed and then measured in the state $|\phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_a - |1\rangle_a)$ with a success probability given by $P = |\langle \psi | \phi \rangle|^2 = \frac{1}{2} (1 - \langle \tilde{x} | \tilde{u} \rangle)$.

Probability P can be estimated to accuracy ϵ in $O\left(\frac{P(1-P)}{\epsilon^2}\right)$ times. Class label is decided depending on the value of P : if it is greater than $\frac{1}{2}$ then $|\vec{x}\rangle$ is labelled -1 ; if it is less than $\frac{1}{2}$, in this case the label of $|\vec{x}\rangle$ is 1 .

The overall time complexity for both training and classification of the LS-SVM is of $O(\log(NM))$.

In the QSVM algorithm, kernelization can be achieved acting on the training vector basis i.e. mapping each $|\vec{x}_i\rangle$ to a d -times tensor product

$$|\phi(\vec{x}_i)\rangle = |\vec{x}_i\rangle_1 \otimes |\vec{x}_i\rangle_2 \otimes \dots \otimes |\vec{x}_i\rangle_d \quad (4.25)$$

in order to obtain polynomial kernels like the following in $O(d\epsilon^{-1}\log N)$

$$K(\langle \vec{x}_i | \vec{x}_j \rangle) \equiv \langle \phi(\vec{x}_i) | \phi(\vec{x}_j) \rangle = \langle \vec{x}_i | \vec{x}_j \rangle^d \quad (4.26)$$

Note that in the QSVM, kernel evaluation is directly performed in the high dimensional feature quantum space, while in classical SVM the kernel trick avoids such expensive calculation. However this apparent issue is solved considering the exponential quantum speed-up obtained in the evaluation of inner product.

Experimental implementations of the QSVM have been shown here [124, 125].

4.2.4 Quantum Computation of Hard Kernels

In this section we review the main proposals having as core idea the computation of classically hard kernel via a quantum device. In this context we can recognize two common threads.

On one side, a hybrid classical-quantum learning model takes classical input and evaluates a kernel function on a quantum devices, while classification is performed in the standard classical manner (e.g employing a SVM algorithm).

In the second approach instead, a kernel based variational quantum circuit is trained to classify input data. More in detail, a variational quantum circuit [127] is a hybrid quantum-classical algorithm employing a quantum circuit $U(\theta)$ that depends on a set of parameters θ which are varied in order to minimize

a given objective function (see Fig.4.3). Such objective function is evaluated through the output measurements of the variational circuit. Hence the training is performed by a classical iterative optimization algorithm that in every step finds the better candidates θ starting from random (or pre-trained) initial values.

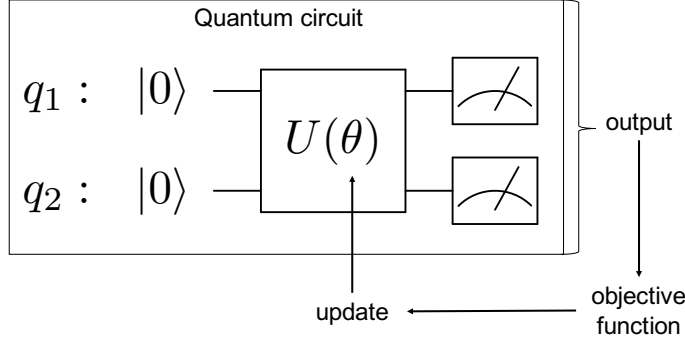


Fig. 4.3: Schematisation of a variational quantum circuit.

Optimization algorithms use the quantum circuit as a black-box and follow under two main categories:

- Derivative-free training algorithm [128] that do not involve the computation of the gradients to determine how to update the parameter θ . Such methods are easy to use but not very stable when they have to deal with a large sets of parameters.
- Gradient-based training methods compute derivatives of the objective function f and thereby of the variational circuit outputs giving information on the steepest descent. Usually the approximation of the gradient is performed using finite-differences method [129] or simultaneous perturbation stochastic approximation (SPSA) [130], which work as follows for parameter $\theta_i \in \theta$ and $\Delta\theta_i$ small enough

$$\delta_{\theta_i} f(\theta_i) \approx \frac{f(\theta_i - \frac{\Delta\theta_i}{2}) - f(\theta_i + \frac{\Delta\theta_i}{2})}{\Delta\theta_i} \quad (4.27)$$

Schuld and Killoran recently explored this concepts [131] underlining the relation between feature maps, kernel methods and quantum computing. The authors explain that the key element in both quantum computing and kernel methods is to perform computations in a high dimensional (possibly infinite) Hilbert space via an efficient manipulation of inputs.

In fact it is possible to interpret the encoding of classical inputs \vec{x}_i into a quantum state $|\phi(\vec{x})\rangle$ as a feature map ϕ which maps classical vectors to the Hilbert space associated with a system of qubits. As said before, two ways of exploiting this parallelism are described.

In the first approach, called by the authors *implicit*, a quantum device takes classical input and evaluates a kernel function as part of a hybrid classification model. This requires the use of a quantum circuit $U_\phi(x)$ realizing the mapping

$$\phi : \vec{x} \rightarrow |\phi(\vec{x})\rangle = U_\phi(x)|000\dots0\rangle \quad (4.28)$$

and which is able to produce a kernel

$$K(\vec{x}_i, \vec{x}_j) = \langle 000\dots0 | U_\phi^\dagger(x_i) U_\phi(x_j) | 000\dots0 \rangle \quad (4.29)$$

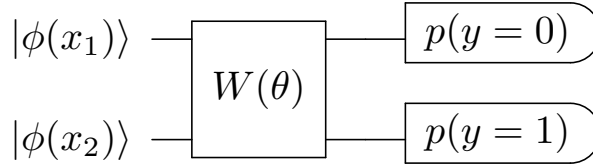
In order for quantum computing to be helpful, such kernel shouldn't be efficiently simulated by a classical computer. It is therefore posed the question of what type of feature map circuits U_ϕ lead to powerful kernels for classical learning models like SVM but at the same time are classically intractable. Authors suggest that a way to achieve such goal is to employ non-Gaussian elements (e.g. cubic phase

gate or photon number measurements) as part of the quantum circuit $U_\phi(x)$ realizing the mapping to the feature space.

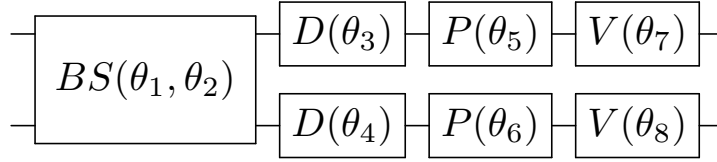
The second approach, addressed in the paper as *explicit*, uses a variational quantum circuit to directly learn a decision boundary in the quantum Hilbert space. In their example, they first translate classical input to a quantum squeezed state

$$\vec{x} \rightarrow |\phi(\vec{x})\rangle = \frac{1}{\sqrt{\cosh(c)}} \sum_{n=0}^{\infty} \frac{\sqrt{(2n)!}}{2^n n!} (-\exp^{i\vec{x}} \tanh(c))^n |2n\rangle, \quad (4.30)$$

then apply to $|\phi(\vec{x})\rangle$ a parametrised continuous-variable circuit



$W(\theta)$ is a repetition of the following gates



where

$$BS(\theta_1, \theta_2) = e^{\theta_1 (e^{i\theta_2} \hat{a}_1^\dagger \hat{a}_2 - e^{-i\theta_2} \hat{a}_1 \hat{a}_2^\dagger)},$$

with $\theta_1, \theta_2 \in \mathbb{R}$ and \hat{a}, \hat{a}^\dagger annihilation and creation operators defined as

$$\begin{aligned} \hat{a}|n\rangle &= \sqrt{n}|n-1\rangle; \\ \hat{a}^\dagger|n\rangle &= \sqrt{n+1}|n+1\rangle; \end{aligned} \quad (4.31)$$

The displacement

$$D(z) = e^{\sqrt{2}i(\text{Im}(z)\hat{x} - \text{Re}(z)\hat{p})},$$

with complex displacement z and finally the quadratic and cubic phase gates

$$P(u) = e^{i\frac{u}{2}\hat{x}^2}$$

$$V(u) = e^{i\frac{u}{3}\hat{x}^3}.$$

The probability of measuring the state $|n_1, n_2\rangle$ in the state $|2, 0\rangle$ or $|0, 2\rangle$ is interpreted as the probability that the classifier predicts class $y = 0$ or $y = 1$

$$p(|2, 0\rangle) = p(y = 0) \quad \text{and} \quad p(|0, 2\rangle) = p(y = 1)$$

The authors trained such model on the 'moons' dataset using stochastic gradient descent and showed that training loss converges to zero after about 200 iterations. The authors claim that the advantage of the explicit approach could be that the classifier is now defined directly on the quantum Hilbert space.

Hence leaving the restriction of accessing the quantum Hilbert space only by means of the kernel, as in the implicit approach.

Along the same path, simultaneously to [131], Havlicek et al. [132] propose two classifiers which map classical data into a quantum feature Hilbert space. Again one SVM classifier is based on a variational circuit that generates a separating hyperplane in the quantum feature space while the other classifier only estimates the kernel function on the quantum computer.

The two methods are tested on an artificial dataset $\vec{x} \in T \cup S \equiv \Omega \subset (0, 2\pi]^2$ where T and S are respectively the training and test sets. This classical input is previously encoded as $\phi_S(\vec{x}) \in \mathbb{R}$ where $\phi_S(\vec{x}) = (\pi - x_1)(\pi - x_2)$.

Stressing on the fact that to obtain an advantage over classical approaches, feature map needs to be based on a circuit that is hard to classically simulate, the authors propose a feature map on n -qubits generated by the unitary

$$\mathcal{U}_{\Phi}(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n} \quad (4.32)$$

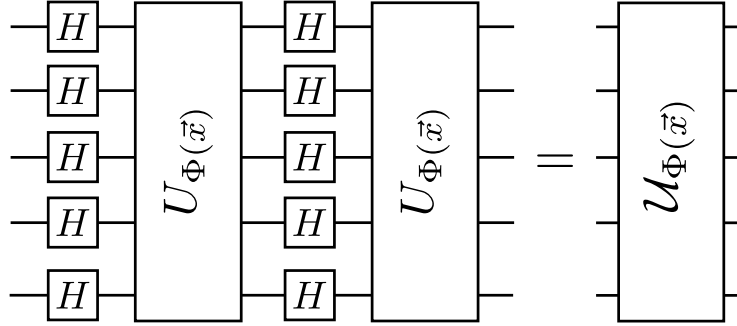


Fig. 4.4: Circuit representation of the feature map family considered. A series of Hadamard gates are applied before a diagonal phase gate. Then such layer is applied a second time.

where H is the Hadamard gate and

$$U_{\Phi(\vec{x})} = \exp \left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{k \in S} Z_k \right) \quad (4.33)$$

with Z_k being the Pauli Z operator acting on the k^{th} qubit and S the test set. Such circuit acts on $|0\rangle^n$ as initial state and uses classical data previously encoded $\phi_S(\vec{x})$.

The authors conjecture that the kernel obtained using a circuit $\mathcal{U}_{\Phi}(\vec{x})$ is hard to estimate up to an additive polynomially small error by classical means. The intuition for this conjecture stems from the fact that the feature map $\mathcal{U}_{\Phi}(\vec{x})$ belongs to a particular quantum circuit family used for estimating the hidden shift problem for boolean functions [132, 134, 135].

4.2.5 ML with a Quantum Annealer

We have seen in Sec. 3.2.2 that quantum annealing is usually employed to deal with combinatorial optimization problems expressed as a QUBO (or Ising problem, equivalently) of the form

$$O(x) = \sum_i h_i x_i + \sum_{i>j} J_{i,j} x_i x_j \quad (4.34)$$

where $x_i \in \{0, 1\}$ are binary variables and parameters h_i and J_{ij} define the optimization problem.

It is possible to perform ML experiments on the D-Wave as long as a computational aspect of the ML problem is expressed as a QUBO problem. An example could be found in the work [136] where authors propose a simple formulation of a clustering technique.

The idea is the following: consider a dataset \mathcal{D} containing M feature vectors $\vec{x}_i \in \mathbb{R}^N$ that we want to partition into $k \leq M$ sets $\mathbf{C} = \{C_1, C_2, \dots, C_K\}$ in order to minimize the within-cluster variance. We can start defining a binary variable $z_i^{(k)}$ that is 1 only when the vector \vec{x}_i belongs to class C_i , in other words

$$z_i^{(k)} = \begin{cases} 1 & \text{iff } \vec{x}_i \in C_i \\ 0 & \text{iff } \vec{x}_i \notin C_i \end{cases}$$

Since every vector \vec{x}_i must be associated to a single class, this means that $\forall \vec{x}_i$ the relation $\sum_{k=1}^K z_i^{(k)} = 1$ should hold. This condition can be rephrased as an objective function as follows

$$f_1(z) = \sum_{i=1}^M \left(\sum_{k=1}^K z_i^{(k)} - 1 \right)^2. \quad (4.35)$$

Moreover, since we want the distance between vectors $D_{i,j} = |\vec{x}_i - \vec{x}_j|$ to define the clusters, i.e we want to group together vectors that are closer, it is necessary to add the penalty term

$$f_2(z) = \sum_{k=1}^K \sum_{i \neq j} D_{i,j} z_i^{(k)} z_j^{(k)} \quad (4.36)$$

The QUBO associated to the clustering problem is therefore

$$O(z) = \lambda \sum_{i=1}^M \left(\sum_{k=1}^K z_i^{(k)} - 1 \right)^2 + \sum_{k=1}^K \sum_{i \neq j} D_{i,j} z_i^{(k)} z_j^{(k)} \quad (4.37)$$

where parameter λ was inserted to modulate the strength of the first term, which represent the hard constraint of the problem. Other version of clustering algorithms, like the bi-clustering problem (where rows and columns of a matrix are simultaneously clustered) appearing in [100] have been successfully translated in QUBO form and analysed using the D-Wave quantum annealer.

Homological Analysis of Multi-qubit Entanglement

As discussed in Chapter 2, in the last few decades entanglement has been recognized as a key resource for obtaining a quantum boost in many information technology tasks [41] and hence the interest moved from a purely foundational aspects to a more practical one. In this context, it is of uppermost importance to have a careful characterization of entanglement, although it is a challenging task when multipartite entangled systems are involved.

Initial work on the classification of entangled states was focused on the quantification through entanglement monotones (see Sec.2.4), which have been proved to work well mostly for bipartite states [28]. For multipartite entanglement, the main approach constructs equivalence classes on the base of invariance under stochastic local operations and classical communication (SLOCC) [44]. However, this leads to infinite (even uncountable) classes for more than three qubit systems. Hence this technique is not effective in the general case, although some ways out were devised for the case of four qubits [45].

Recently it was proposed an alternative route to entanglement classification represented by the analysis of topological features of multipartite quantum states [137]. Topological data analysis (see Sec.1.2) has recently gained a lot of attention in the classical framework thanks to its suitability for the analysis of huge datasets represented in the form of point clouds. Among these techniques, Persistent Homologies (PH) played a pivotal role [21, 139]. It is a particular sampling-based technique from algebraic topology aiming at extracting topological information from high-dimensional datasets.

Interestingly, a quantum approach to computing persistent homology has been devised in [122] in order to achieve more efficient algorithms for classical data analysis. Here instead we focus on the usage of persistent homologies in the study of pure quantum states by discussing how [137] and [138] employed this technique for characterizing multipartite entanglement.

In general the PH technique analyse the entanglement of multiqubit state vectors which are represented as a dataset by introducing a metric-like measure of pairwise entanglement between qubits. More specifically, given a vector state of N qubits, we consider N points each one representing a qubit and then arrange them in such a way that their pairwise distance is inversely proportional to the two-points correlation. Then, a barcode is constructed for this dataset. Two states are in the same class if they have the same barcodes. Actually, we do not care about the position of a bar, but only to its presence or absence. An interesting new classification of ‘genuinely’ entanglement is provided for states containing up to six qubits. Finally, we compute the relative occurrence frequency of the various classes of entangled states, obtained by means of a random generation of state.

5.1 Creating the Qubit Data Cloud

In this section, the methodology used for creating the data cloud which is the basis for classifying entangled states is discussed. We restrict our attention to N qubit states showing "genuine" entanglement, i.e. that are N -partite entangled or “fully inseparable”.

The approach starts with a random generation of pure states among which we select, using generalised concurrence measure, those showing genuine entanglement. At this stage, a data cloud is associated to a state in such a way that each qubit is identified with a single point in the cloud, while a distance between pairs of points is defined using a semi-metric that takes into account the pairwise entanglement shared by the two qubits that the points represent. Semi-distances between qubits are stored in a matrix D which will be the input of the persistent homology algorithm.

Note that in the definitions given in Section 1.2 we refer for simplicity to Euclidean spaces. Since here we are dealing instead with a semi-metric space, it is worth stressing that computing persistent homology is still possible in our case. In fact, a distance between pairs of points which does not satisfy triangular inequality is still sufficient for constructing Rips complexes.

Moreover, it is worth underlining that multipartite entanglement is not uniquely determined by means of bipartite entanglement between pairs of qubits. However, the idea here is to study the information we get from looking at multipartite entanglement through the lenses of TDA.

5.1.1 Random State Generation

In order to randomly generate a pure state of N qubits, we employ the following parametrization [140, 141]

$$|\psi\rangle = \sum_{n=0}^{2^N-1} \nu_n |n\rangle, \quad (5.1)$$

with

$$\nu_0 = \cos \theta_{2^N-1}, \quad (5.2)$$

$$\nu_{n>0} = e^{i\phi_n} \cos \theta_{2^N-1-n} \prod_{l=2^N-n}^{2^N-1} \sin \theta_l. \quad (5.3)$$

and

$$\theta_n := \arcsin \left(\xi_n^{\frac{1}{2^n}} \right). \quad (5.4)$$

The independent random variables $\phi_{n \geq 1}$ and $\xi_{n \geq 0}$ are uniformly distributed in the intervals:

$$\phi_{n \geq 1} \in [0, 2\pi), \quad \xi_n \in [0, 1].$$

Note that the state parametrization here presented is equivalent to that of random Haar states. In fact, following [140], the Hurwitz parametrization for generating pure states as vectors of a random unitary matrix distributed according to the Haar measure was used.

5.1.2 Entangled States Selection

After generating a random N -qubit state $|\psi\rangle$ we check that it is actually N -partite entangled. This happens iff for every bipartition A/\hat{A} (where \hat{A} denotes the complement set of A) of the N -qubit, $\mathcal{C}_G(\rho_A) \neq 0$, where \mathcal{C}_G is the generalized concurrence defined in Sec.2.4 as

$$\mathcal{C}_G(\rho_A) := \sqrt{2(1 - \text{Tr}(\rho_A^2))}. \quad (5.5)$$

5.1.3 Distances Calculation

It is possible to generate barcodes for simplicial complexes corresponding to a points (i.e.qubits) cloud by giving in input to the persistent homology algorithm the matrix D of all pairwise distances between points.

In [137], a semi-distance $1/E_{i,j}$, was proposed, where $E_{i,j}$ is an entanglement monotone calculated between qubit i and qubit j . This semi-distance goes from 1 (when the two qubit are maximally entangled) to $+\infty$ (when they are separable). Here we use the following semi-metric:

$$D_{i,j} = 1 - \exp \left\{ 1 - \frac{1}{E_{i,j}} \right\}, \quad (5.6)$$

where $E_{i,j}$, is chosen to be the concurrence $C_{i,j}$ between qubit i and qubit j . The semi-distance $D_{i,j}$ goes from 0 to 1 as the entanglement decreases, and remains finite for separable states.

Recall that, given a state ρ on N qubits, the concurrence between two qubits i and j is obtained by first tracing out all other $N - 2$ qubits. This gives the reduced density matrix ρ_{ij} . Then

$$C_{i,j} := \max\{0, \lambda_1 - \lambda_2 - \lambda_3 - \lambda_4\}, \quad (5.7)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, are the square root of the eigenvalues (in decreasing order) of the matrix $\rho_{ij}(\sigma_y \otimes \sigma_y)\rho_{ij}^*(\sigma_y \otimes \sigma_y)$ [143], with σ_y the well known Pauli matrix and ρ_{ij}^* the complex conjugate of $\rho_{i,j}$ in the computational basis.

5.2 Entanglement Classification

We have used the TDA package for computing persistent homology and barcodes developed for the *R software*. The classification is obtained grouping together those states with the same barcode.

In the following barcodes, black lines represent connected components (i.e. homology group H_0), red lines represent holes (i.e. homology group H_1) and blue lines represent voids (i.e. homology group H_2). All barcodes are generated using the Rips complex.

5.2.1 Classification of Three Qubits States

In the case of genuine three-partite entangled states of three qubits it is possible to recognize the following three classes. Barcodes are shown starting from the most frequent to the least frequent one.

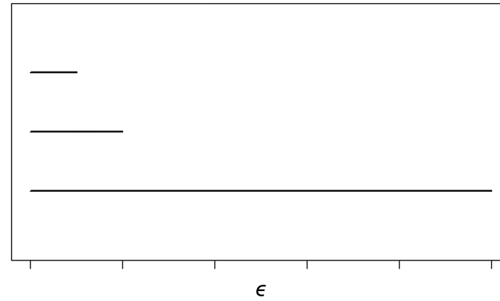


Fig. 5.1: Barcode for the class labelled as **3B1**. Black lines in the barcode represent connected components, i.e. homology group H_0 .

This class contains those states that have one of the two following properties:

- States with $C_{i,j}, C_{j,k} > 0$ and $C_{i,k} = 0$; their associated point cloud is made of three points where the first and the second are at distance $D(i,j)$, the second and the third are at distance $D(j,k)$, but the first and the third points are at maximum distance. This can be seen in the barcode of Figure 5.1 where two of the three lines vanishes and only one persists.

- States where, at a finite value ϵ^* of $D(i, j)$, the three points get connected one with each other to form a triangular graph that is immediately filled with a 2-simplex. This means that for $\epsilon \geq \epsilon^*$ we are left with only one connected component (the 2-simplex), again shown by the barcode of Figure 5.1.

A representatives for this class, could be

$$|3, B1\rangle = |W\rangle = \frac{1}{\sqrt{3}} (|100\rangle + |010\rangle + |001\rangle) \quad (5.8)$$

States of this class constitute a subset of the W-class found using the SLOCC classification.

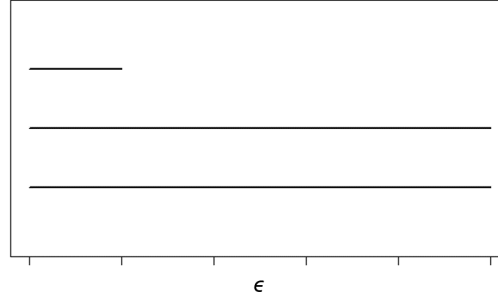


Fig. 5.2: Barcode for the class labelled as **3B2**. Black lines in the barcode represent connected components, i.e. homology group H_0 .

Another class is defined by states where $C_{i,j} > 0$, $C_{j,k} = 0$ and $C_{i,k} = 0$; the associated point cloud is made of three points where the first and the second are at a distance $D < 1$, while the third point is at distance $D = 1$ from the other two. The barcode shows three different connected components in the interval $[0, D(i, j)]$, while for values greater than $D(i, j)$ only two of them persist as shown in Figure 5.2. The state

$$|3, B2\rangle = |\psi_b\rangle_3 = \frac{1}{\sqrt{3}} (|000\rangle + |011\rangle + |111\rangle) \quad (5.9)$$

can be chosen to be a representative for this class. Note that this class constitutes a subset of the W-class found under SLOCC [44].

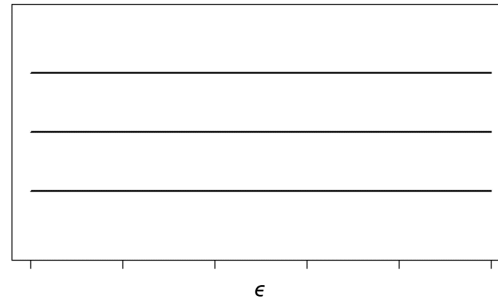


Fig. 5.3: Barcode for the class labelled as **3B3**. Black lines in the barcode represent connected components, i.e. homology group H_0 .

This class collects all those states where entanglement between any possible pair of the three qubits gives $C_{i,j} = 0$, for all i, j . This implies that each point of our three points cloud is maximally far away from the others. Hence they generate a barcode that only displays the 0-order homology group H_0 , i.e. the connected components as shown in Figure 5.3.

The state

$$|3, B3\rangle = |GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \quad (5.10)$$

can be chosen to be a representative for this class, which exactly corresponds to the GHZ-class of [44].

In summary, we have shown that a classification performed using barcodes obtained with distance $D(i, j)$ is able to distinguish 3 different classes of true entangled states of three qubits, hence going beyond the known SLOCC classification.

5.2.2 Classification of Four Qubits States

Barcodes generated by 4-partite entangled states of 4 qubits are shown starting from the most frequent to the least frequent one. Recall that black lines in the barcodes represent connected components (i.e. homology group H_0) and red lines represent holes (i.e. homology group H_1). All barcodes are generated using the Rips complex.

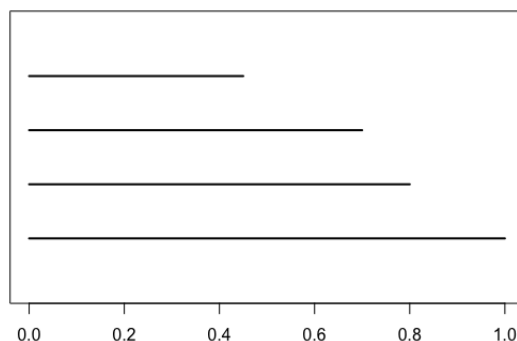


Fig. 5.4: Barcode of the class labelled as **4B1**.

Genuine entangled states with the barcode shown in Figure 5.4 have a total of four connected components: three of them end at value of $\epsilon < 1$, while only one component persists over all the range of ϵ . The fact that only one connected component persists means that the state form a single cluster of qubits grouped by pairwise entanglement without showing higher homological features. A representative of such class is the $|W\rangle$ state.

$$|4, B1\rangle = |W\rangle = \frac{1}{2}(|0001\rangle + |0010\rangle + |0100\rangle + |1000\rangle) \quad (5.11)$$

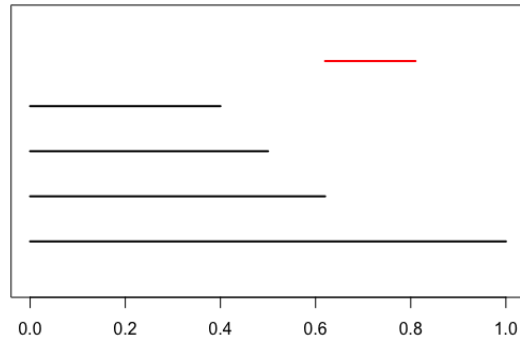


Fig. 5.5: Barcode of the class labelled as **4B2**.

States belonging to class of Figure 5.5 form again a single persistent component of pairwise entanglement between qubits. However in this case a hole, denoted by the red bar H_1 , appears when only one connected component is left. Such a hole has limited life-span since disappears when ϵ is sufficiently large. A state showing such a behaviour is the following:

$$|4, B2\rangle = \frac{1}{2\sqrt{2}} \left(\sqrt{2}|0000\rangle + |0011\rangle + |0110\rangle + |1001\rangle + |1100\rangle + \sqrt{2}|1111\rangle \right) \quad (5.12)$$

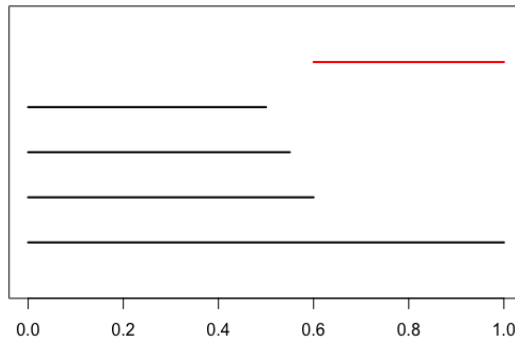


Fig. 5.6: Barcode of the class labelled as **4B3**.

In the case shown in Figure 5.6, a single connected component is left and a persistent hole is present. States with this barcode have the characteristic that each qubit is pairwise entangled to other two qubits and completely un-entangled with a third qubit. A state showing such properties is

$$|4, B3\rangle = \frac{1}{2} (|0000\rangle + |0011\rangle + |1010\rangle + |1111\rangle)$$

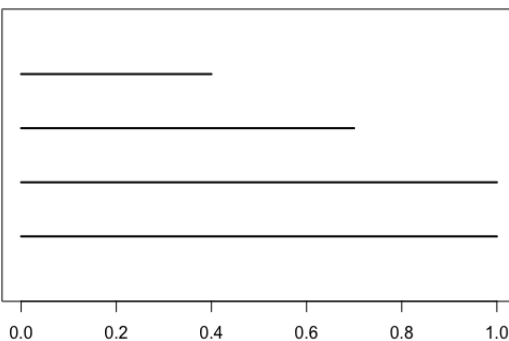


Fig. 5.7: Barcode of the class labelled as **4B4**

In the class represented by the barcode in Figure 5.7 we find genuinely entangled states with no higher homological feature than H_0 which have two different connected component that persist over the range of ϵ . This means that such states have two sets of qubits which are internally connected by pairwise entanglement to form a component, but no connection is present among qubits of different sets. Yet a single qubit in a set could be entangled to the other set as a whole. An example for this class is the state:

$$|4, B4\rangle = \frac{1}{2} (|0011\rangle + |1011\rangle + |1101\rangle + |1110\rangle) \quad (5.13)$$

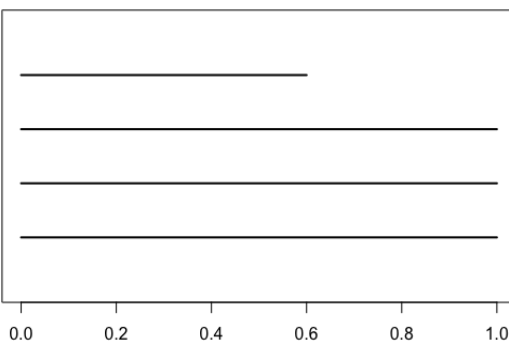


Fig. 5.8: Barcode of the class labelled as **4B5**

Like the previous case, states with the barcode of Figure 5.8 only show four connected components, three of which persist while one has limited lifetime. The characteristic of these states is that there are always 2 qubits which do not share any pairwise entanglement with another qubit, while the other two do. A representative state for this class is

$$|4, B5\rangle = \frac{1}{\sqrt{3}} (|0000\rangle + |0111\rangle + |1101\rangle) \quad (5.14)$$

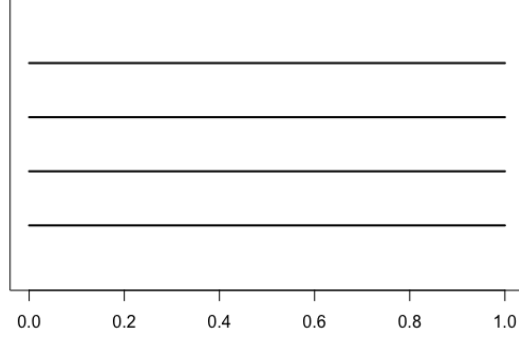


Fig. 5.9: Barcode of the class labelled as **4B6**

States of the kind shown in Figure 5.9 do not have any pairwise entanglement among qubits. For this reason no qubit get connected to another and we see four distinct components that persist. A representative of this class is the $|\text{GHZ}\rangle$ state.

$$|4, B6\rangle = |\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \quad (5.15)$$

As we can observe in Fig.5.22, there exist six different classes of four qubit genuine entangled states, based on the persistent homology classification.

Comparison with SLOCC and Generalisation

In [45] it is proposed a classification for pure entangled states of four qubit. This classification is obtained from the orbits generated by SLOCC operations and produces nine classes. Among these nine classes, one is called 'generic' and contains an uncountable number of SLOCC inequivalent classes of states, as shown in [46]. A comparison between the classification based on persistent homology and the one presented in [44] is possible in the three qubit case, as discussed before. However, in the four qubit case, a comparison between our approach and the one in [45] does not allow us to clearly establish a correspondence between classes as in the three qubit case. In order to better understand this, it is useful to consider few specific examples. Consider the class

$$L_{ab3} = a(|0000\rangle + |1111\rangle) + \frac{a+b}{2}(|0101\rangle + |1010\rangle) + \frac{a-b}{2}(|0110\rangle + |1001\rangle) + \frac{i}{\sqrt{2}}(|0001\rangle + |0010\rangle + |0111\rangle + |1011\rangle) \quad (5.16)$$

defined in [45]. It is easy to check that for the different values of the parameters a and b we obtain the following states:

- for $a = b = 0$, a W-like state with associated barcode **4B1**;
- for $a = 0$ and $b = 1$, a state with barcode **4B3**;
- for $a = b = 1$, a state with barcode **4B4**.

Moreover states belonging to the class $L_{0_{\tau \oplus 1}}$, which are obtained from the state $|L_{0_{\tau \oplus 1}}\rangle$ below, up to permutations of the qubits,

$$|L_{0_{7\oplus 1}}\rangle = |0000\rangle + |1011\rangle + |1101\rangle + |1110\rangle \tag{5.17}$$

have the same barcode of $|GHZ\rangle$ which instead belongs to the generic class G_{abcd} for $a = d$ and $b = c = 0$. This makes it impossible to fully characterize the SLOCC classification proposed in [45] in terms of barcodes and vice versa. The SLOCC approach is indeed based on a classification criteria (equivalence with respect to local operations) that are not comparable to ours, which are of different nature (the change in the topology of the point cloud formed by the qubit, depending on the pairwise entanglement strength). Therefore the classifications obtained are intrinsically different besides leading to a different number of classes: nine with SLOCC and six with persistent homology.

We will nevertheless argue in the following that our approach is more robust in terms of increasing the number of qubits. In fact, it was shown in [44] that for systems of size $N \geq 4$ there exist infinitely many inequivalent kinds of entanglement under SLOCC and no finite classification is known for states of those sizes. If we restrict to the case of genuine multipartite entangled state of N qubit, it is easy to see that our persistent homology approach always provides a finite number of different classes for any value of N . This is due to the fact that the total number of possible homology groups that can be obtained considering a dataset of points is always finite. The number of possible barcodes B_N obtained with a dataset of N points can in fact be bounded from the above as follows

$$B_N < \left(\sum_{d=0}^{\frac{N(N-1)}{2}} G_N(d) d! \right) \tag{5.18}$$

where $G_N(d)$ is the total number of possible graphs with N vertices and d edges, up to permutations of the vertices, and the factorial $d!$ takes into account all the possible ways of constructing $G_N(d)$.

In light of this concept, we extend our classification of genuine entanglement to the cases of five and six qubits as shown in the following sections.

5.2.3 Classification of Five Qubits States

Let’s hence consider randomly generated 5-partite entangled states of 5 qubits. Barcodes are shown below starting from the barcode more likely to appear to the least frequent one. Recall that black lines in the barcodes represent connected components (i.e. homology group H_0) and red lines represent holes (i.e. homology group H_1). All barcodes are generated using the Rips complex.

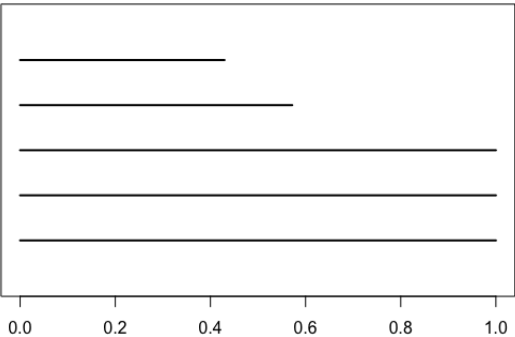


Fig. 5.10: Barcode of the class labelled as **5B1**

In the five qubit case, the most frequent class shows a barcode like the one in Figure 5.10 with three connected components that persist in the range of D . States of this kind have at least one qubit (up to

two qubits) that does not share pairwise entanglement with any other qubit. This configuration does not generate higher homology groups than H_0 . An example of state in this class is

$$|5, B1\rangle = \frac{1}{\sqrt{5}} (|00001\rangle + |00011\rangle + |00110\rangle + |01000\rangle + |11011\rangle) \quad (5.19)$$

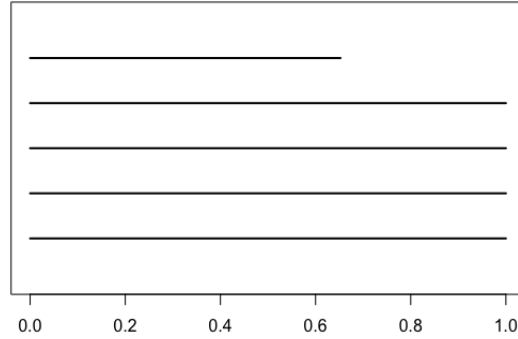


Fig. 5.11: Barcode of the class labelled as **5B2**

As we can see, the barcode of Figure 5.11 shows four persistent connected components i.e. only two qubits among five share pairwise entanglement while all remaining qubits act as independent connected component. A representative state for this class is

$$|5, B2\rangle = \frac{1}{\sqrt{5}} (|00001\rangle + |00011\rangle + |00100\rangle + |01100\rangle + |11010\rangle) \quad (5.20)$$

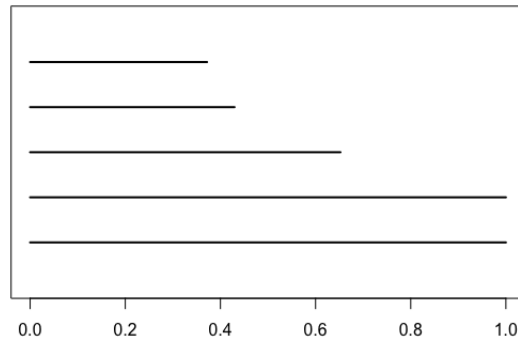


Fig. 5.12: Barcode of the class labelled as **5B3**

In this class identified by the barcode of Figure 5.12, two connected components persist while the other three have limited lifetime. Two clusters of qubits connected by pairwise entanglement are hence created and no holes or higher topological features appear. An example of state in this class is the following

$$|5, B3\rangle = \frac{1}{\sqrt{5}} (|00001\rangle + |00010\rangle + |00100\rangle + |01000\rangle + |10111\rangle) \quad (5.21)$$

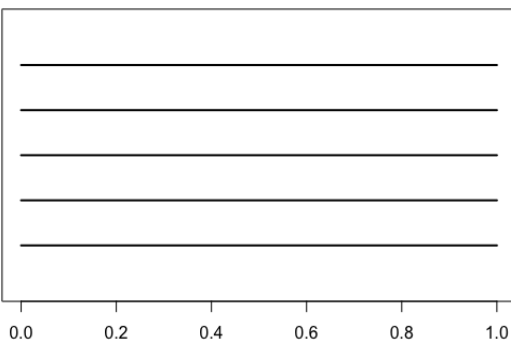


Fig. 5.13: Barcode of the class labelled as **5B4**

Figure 5.13 show the barcode of the class where we find GHZ like states, i.e. those states where there are no entangled pair of qubits and hence show five persistent connected components in the barcode.

$$|5, B4\rangle = \frac{1}{\sqrt{2}} (|00000\rangle + |11111\rangle) \quad (5.22)$$

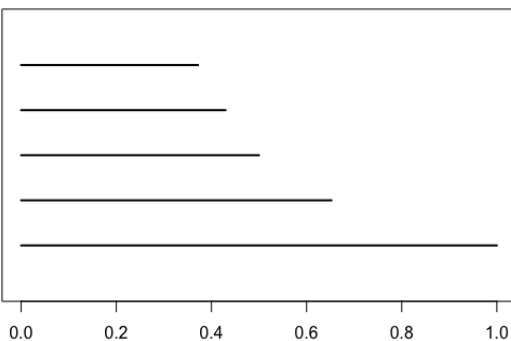


Fig. 5.14: Barcode of the class labelled as **5B5**

After we find the barcode shown in Figure 5.14 and relative to those states like $|W\rangle$ which have only one connected component and hence pairwise entanglement creates a single cluster of qubits.

$$|5, B5\rangle = \frac{1}{\sqrt{5}} (|00001\rangle + |00010\rangle + |00100\rangle + |01000\rangle + |10000\rangle) \quad (5.23)$$

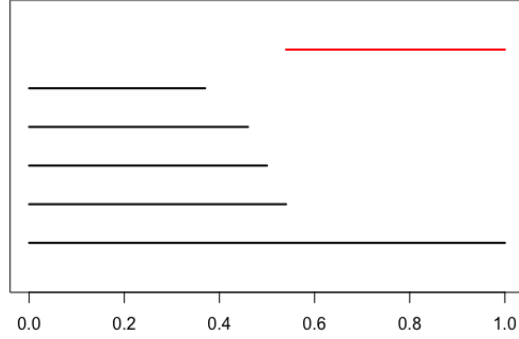


Fig. 5.15: Barcode of the class labelled as **5B6**

Figure 5.15 shows the barcode of the first class of 5 qubits genuinely entangled states that present a first order homology group H_1 , i.e. a hole, in the barcode. States in this class have their qubits connected to form a single persistent component when $\epsilon \approx 1$. Note also that some subsets of qubits do not share any pairwise entanglement and hence are responsible for the persistent hole. An example of state in this class is

$$|5, B6\rangle = \frac{1}{\sqrt{6}} (|00000\rangle + |11000\rangle + |01100\rangle + |00110\rangle + |00011\rangle + |10001\rangle) \quad (5.24)$$

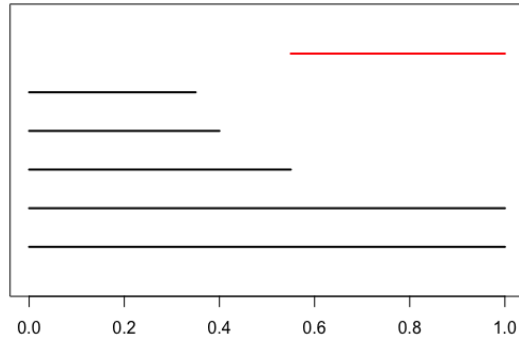


Fig. 5.16: Barcode of the class labelled as **5B7**

As we can see in Figure 5.16, like in the previous class, a hole is created at some value of ϵ and persists up to the upper limit of the semi-metric D . However here while four qubits are responsible for one connected component and for the H_1 homology, the remaining fifth qubit does not share any pairwise entanglement with the others and creates a persistent component on its own. A representative state of this kind is

$$|5, B7\rangle = \frac{1}{\sqrt{5}} (|00010\rangle + |00011\rangle + |00101\rangle + |10111\rangle + |11011\rangle) \quad (5.25)$$

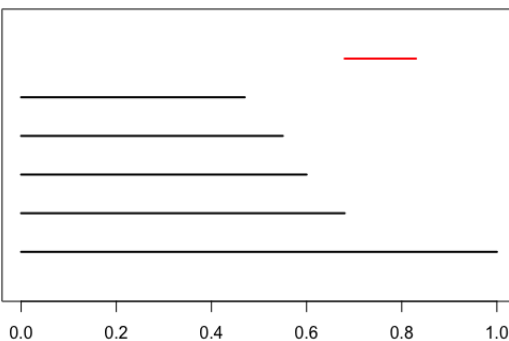


Fig. 5.17: Barcode of the class labelled as **5B8**

States in the class of Fig.5.17 have similar properties to those in the class of Fig.5.15, however in this case the H_1 homology does not persist since connections among qubits creating the hole appear at some value of ϵ . An example state with such barcode could be

$$|5, B8\rangle = \frac{1}{\sqrt{10}} \left(\sqrt{5}|00000\rangle + |11000\rangle + |01100\rangle + |00110\rangle + |00011\rangle + |10001\rangle \right). \quad (5.26)$$

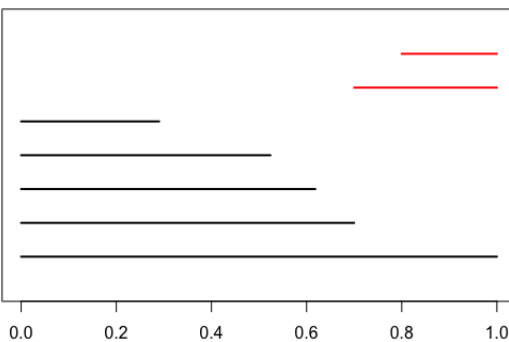


Fig. 5.18: Barcode of the class labelled as **5B9**

The class characterized by the barcode depicted in Figure 5.18 shows a single persistent connected component of qubits grouped by pairwise entanglement but also two holes which appear at some ϵ and persist for higher values. A representative for this class is the following

$$|5, B9\rangle = \sqrt{\frac{2}{5}} (|00000\rangle + |01010\rangle) + \frac{1}{5} (|00011\rangle + |00101\rangle + |01100\rangle + |11000\rangle + |10001\rangle) \quad (5.27)$$

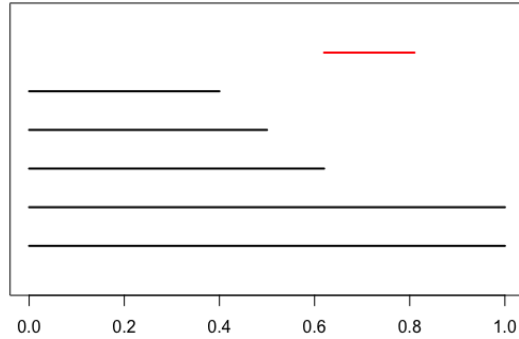


Fig. 5.19: Barcode of the class labelled as **5B10**

States belonging to the class of Figure 5.19 have similar properties to those in class with barcode in Figure 5.16, i.e. two persistent connected components, one of which is made up of a single qubit which does not share pairwise entanglement with the other four. In the other instead, the remaining four qubits get connected to form a non-persistent hole. An example state with such barcode could be

$$|5, B10\rangle = \frac{1}{\sqrt{6}} (|01000\rangle + |01010\rangle + |10000\rangle + |10001\rangle + |10110\rangle + |11010\rangle) \quad (5.28)$$

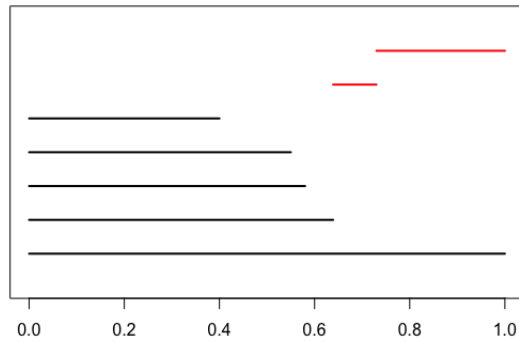


Fig. 5.20: Barcode of the class labelled as **5B11**

A single persistent connected component and two holes characterize the barcode of this class, as shown in Figure 5.20. Note that one of the two homology group generators H_1 appears only in a limited interval while the other one persists over ϵ . An example state with such barcode:

$$|5, B11\rangle = \frac{1}{\sqrt{11}} \left(\sqrt{5}|00010\rangle + \sqrt{2}|00100\rangle + \sqrt{2}|10000\rangle + |10101\rangle + |11100\rangle \right) \quad (5.29)$$

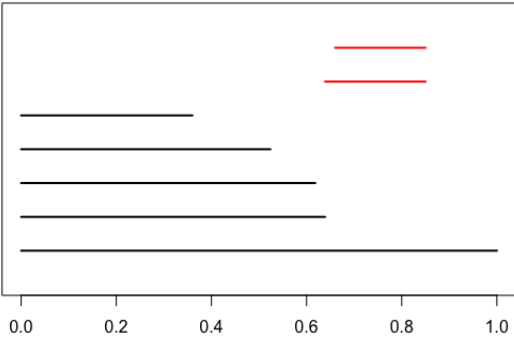


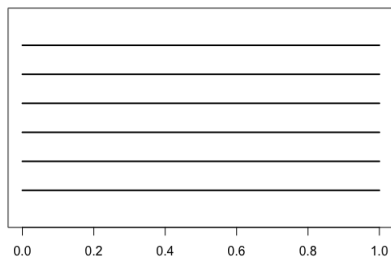
Fig. 5.21: Barcode of the class labelled as **5B12**

The least frequent class, barcode in Figure 5.21, is the one composed of those states where qubits get connected to form a single connected component allowing the presence of two hole that however do not persist. An example state with such barcode is the following

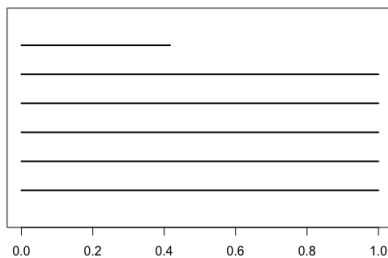
$$|5, B12\rangle = \frac{1}{\sqrt{2}} |00000\rangle + \frac{1}{\sqrt{10}} (|11000\rangle + |01100\rangle + |01010\rangle + |00101\rangle + |10001\rangle) \quad (5.30)$$

5.2.4 Classification of Six Qubits States

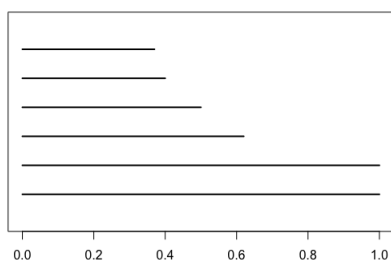
By randomly generating six-partite genuine entangled states of six qubits, we obtained 33 different classes. Here we report the 33 associated barcodes starting from the most frequent to the least frequent one. Recall that black lines in the barcodes represent connected components (i.e. homology group H_0), red lines represent holes (i.e. homology group H_1) and blue lines represent voids (i.e. homology group H_2). All barcodes are generated using the Rips complex.



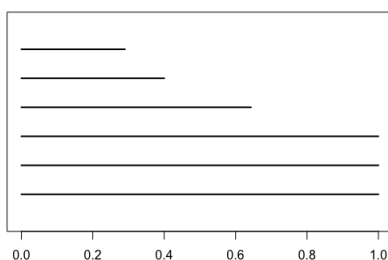
6B1



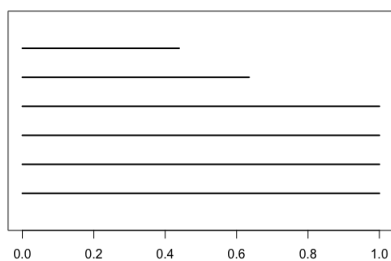
6B2



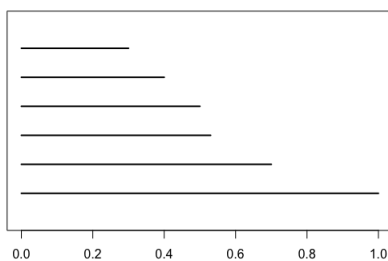
6B3



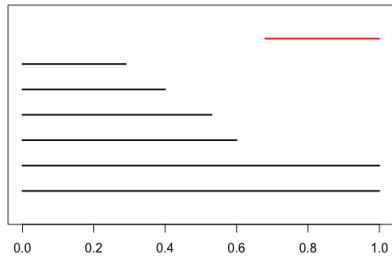
6B4



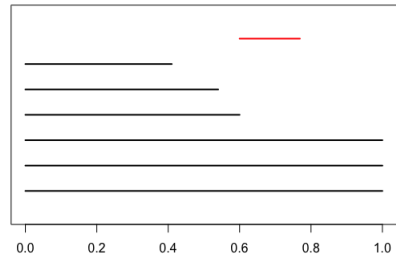
6B5



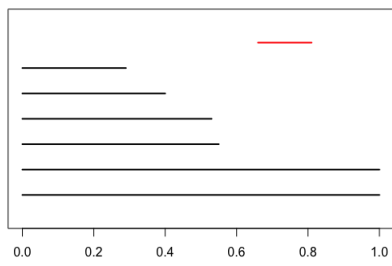
6B6



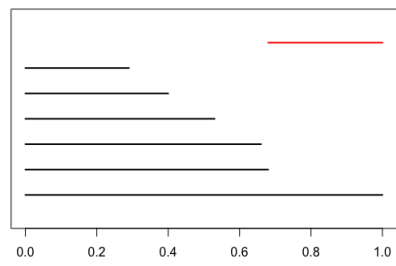
6B7



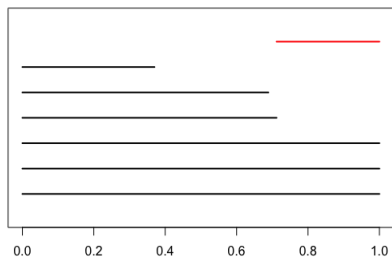
6B8



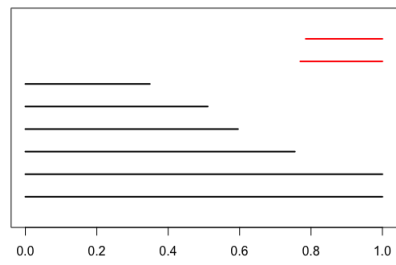
6B9



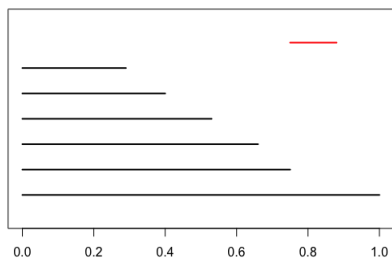
6B10



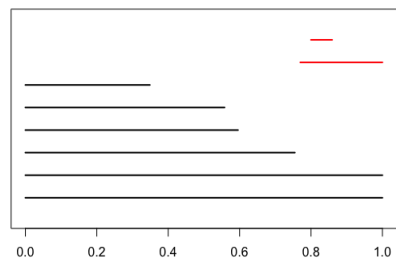
6B11



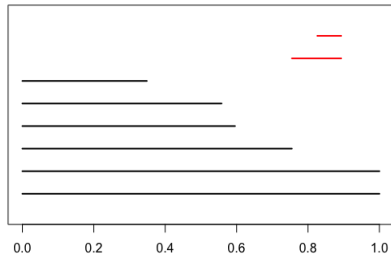
6B12



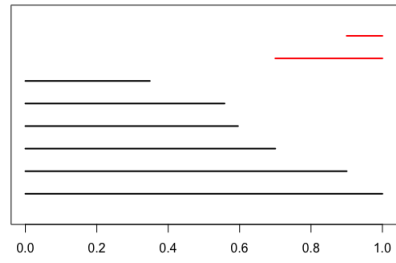
6B13



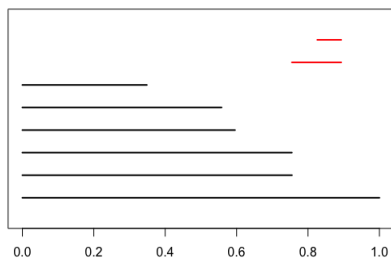
6B14



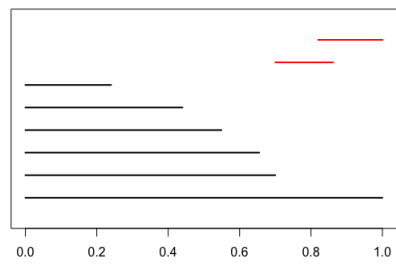
6B15



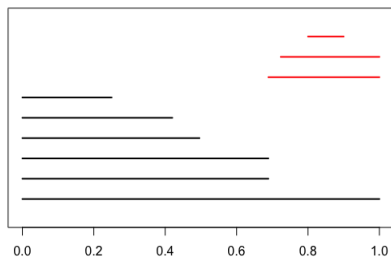
6B16



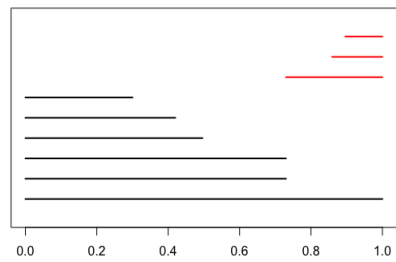
6B17



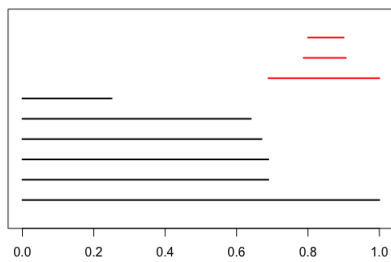
6B18



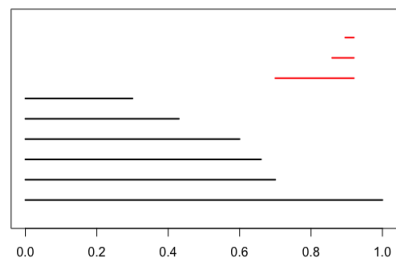
6B19



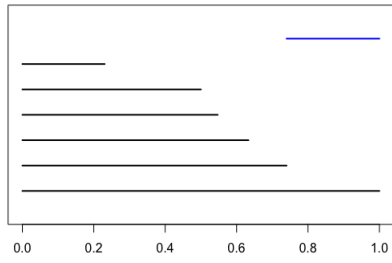
6B20



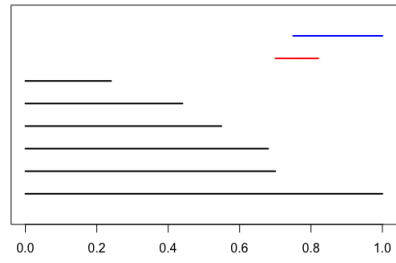
6B21



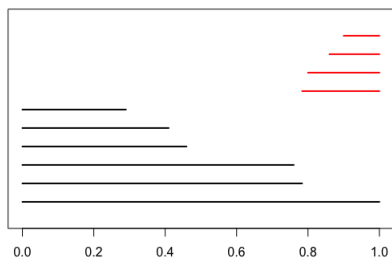
6B22



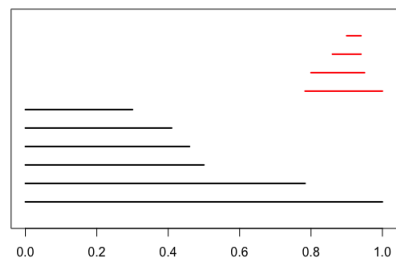
6B23



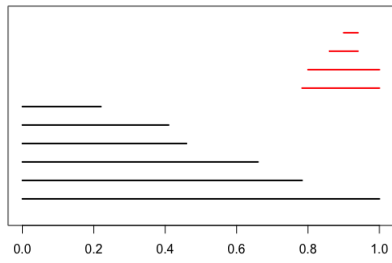
6B24



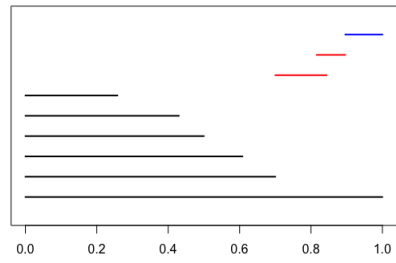
6B25



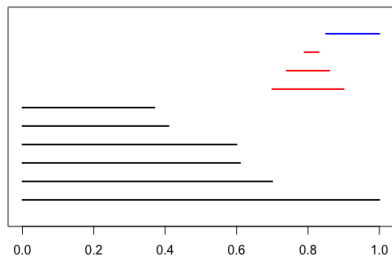
6B26



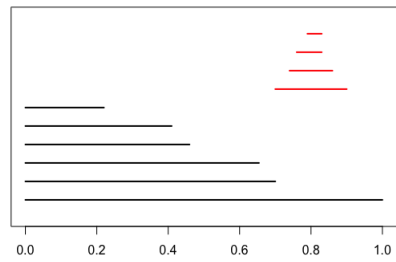
6B27



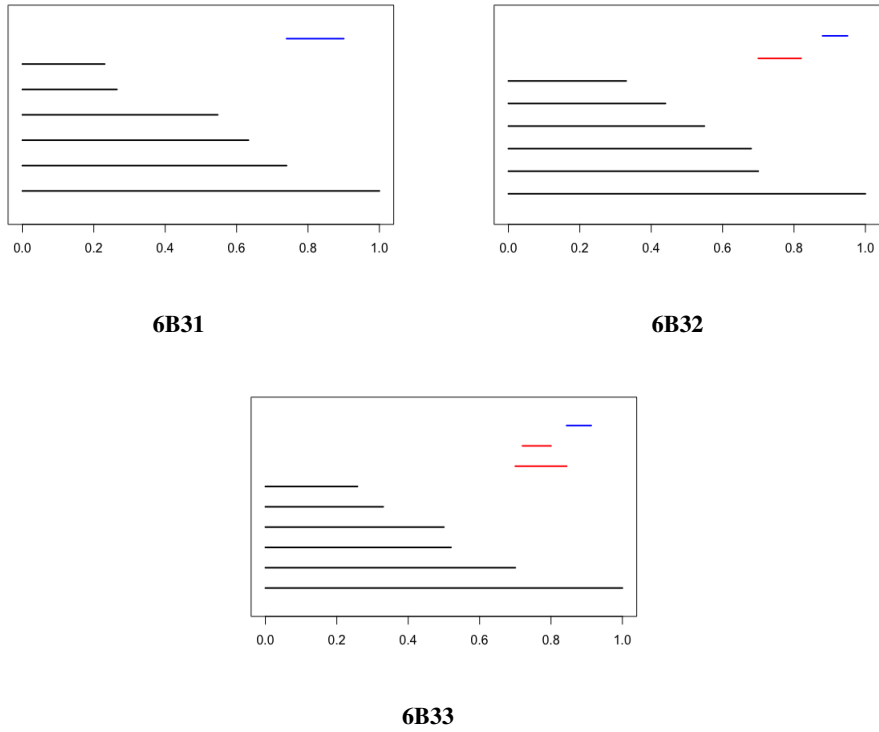
6B28



6B29



6B30



5.3 Class Frequencies

As we can observe in Fig.5.22, there exist six different classes of four qubit genuine entangled states, based on the persistent homology classification. The most frequent class (61.70%) is the one where only one component persists (**4B1**). States like *W* belong to this class. With frequencies of 17.31% and 10.60% we find states with barcodes **4B2** and **4B3** showing one persistent connected component and a hole (red line) that in the case of **4B3** is also persistent. The last three barcodes, in order **4B4**, **4B5** and **4B6** show an increasing number of disconnected components. States that are GHZ-like are hence the least frequent (0.52%).

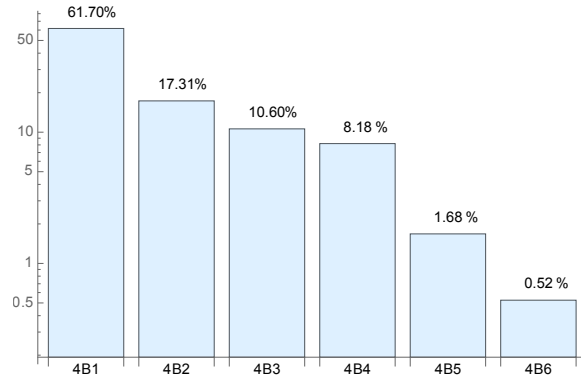


Fig. 5.22: Barcode frequencies (in Log scale) for four qubits genuine entangled states

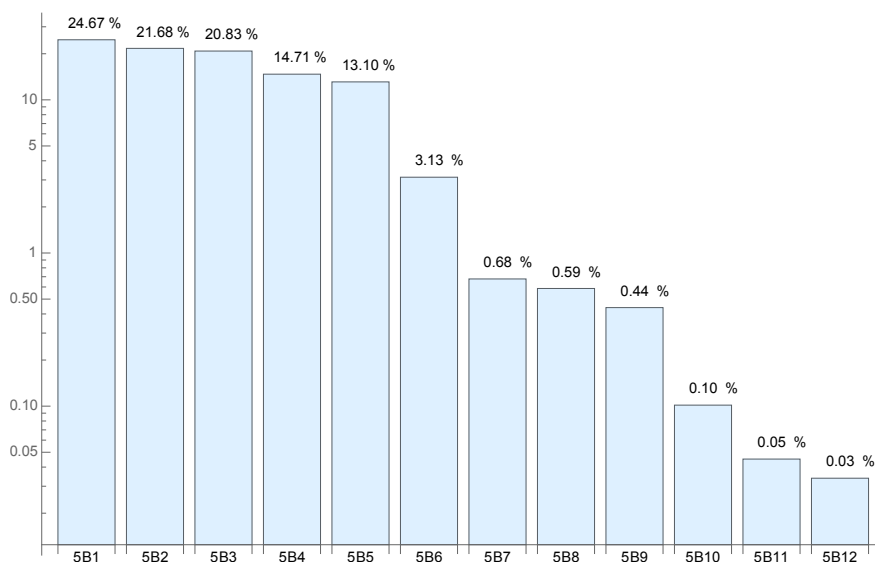


Fig. 5.23: Barcode frequencies (in Log scale) for five qubits genuine entangled states

For what concerns the five qubits case, we obtained twelve classes. By looking at the chart in Figure 5.23 it can be noticed that the most frequent barcode (**5B1**) belongs to states with three connected components, followed by states with barcodes showing four, two, five and one persistent components (respectively **5B2**, **5B3**, **5B4**, **5B5**). The 95% of all randomly generated states fall inside one of these first five classes. After them, barcodes with higher dimensional homology features start to appear: at first those with one hole and then those with two. The only exception is given by barcode **5B10** (showing one short-lived hole but two connected components) since it is less frequent than **5B9** (one connected component and two persistent holes).

By randomly generating six-partite genuine entangled states of six qubits, we obtained 33 different classes. Their frequencies are shown in Figure 5.24. With a frequency of 68%, the class defined by barcode **6B1** is by far the most frequent. Such a class consists of GHZ-like states that present six different connected components, i.e. the single qubits with no pairwise entanglement. As we have seen, for the five qubit case, the first classes in frequency, from **6B1** to **6B6**, are those containing states showing only connected components (H_0 homology group i.e. black bars). Then states with one hole start to appear, and later those showing two holes, with the exception of barcodes **6B12** and **6B13**. Barcodes showing multiple holes and voids, from **6B19** to **6B33** are also possible but they do not appear in the histogram since their frequency is very low ($\ll 0.004\%$). Finally, note that these frequencies are obtained using the parametrization of Eq. 5.1 which produces states according to the Haar measure, a uniform probability distribution over all quantum states.

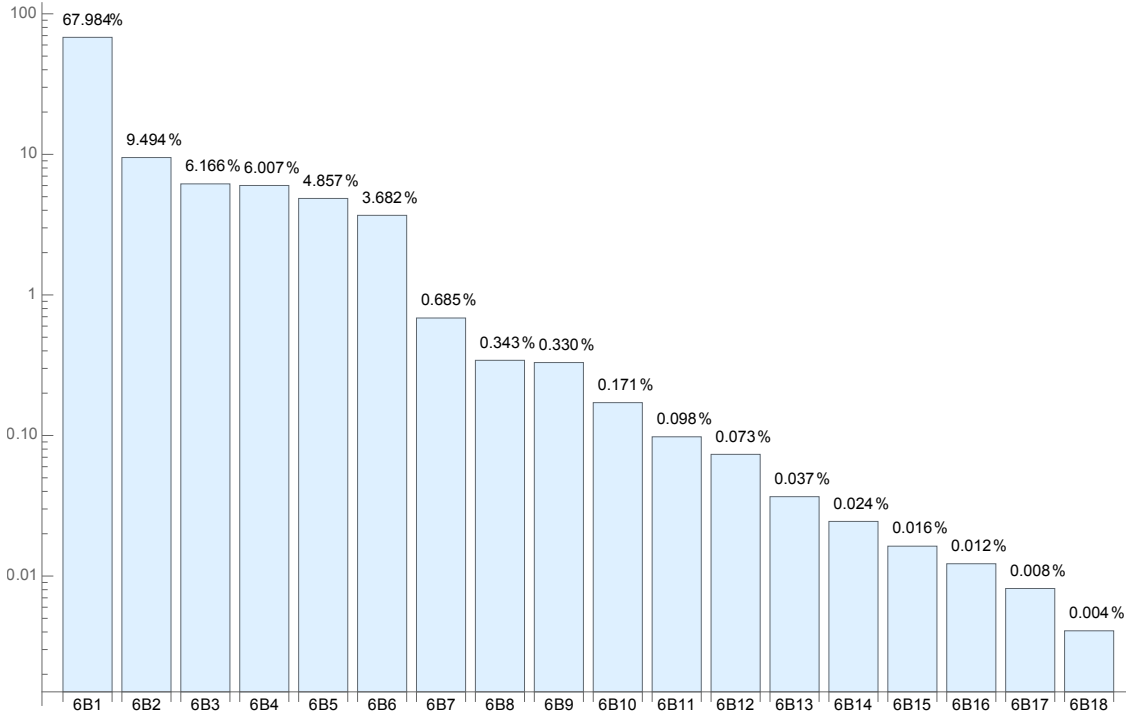


Fig. 5.24: Barcode frequencies (in Log Scale) for six qubits genuine entangled states

5.4 Comments

The classification that we have carried out for three, four, five and six qubits entangled states shows that it is possible to distinguish respectively three, six, twelve and thirty-three different classes by persistent homological barcodes. In general, given a N qubit genuinely entangled state, it is always possible to come up with a finite classification where the total number of possible barcodes B_N is bounded.

Furthermore, patterns seem to emerge in our classifications by looking at the frequencies of barcodes. First of all, those states which are characterized by the only H_0 homology group become more likely as the number of qubits N increases. This is followed by the group of those states showing also H_1 which again are followed by those with a much richer topology. While this fact could be explained from a topological point of view by claiming that, with a limited number of points complex homological patterns in the barcode are harder to obtain, it is still interesting to notice that the same reasoning also holds true for quantum state barcodes.

Among those states with only the H_0 homology group, it is worth noticing that the W-like class, with only one persistent connected component, decreases its frequency with the increase of N . In fact, except for $N = 4$ where we find this class in the first place, in the $N = 5$ and $N = 6$ cases, it falls to the last position. Conversely, the class of states which have N persistent components, like GHZ, gradually increase their frequency, starting from the bottom at $N = 4$ and becoming the most popular at $N = 6$.

In general we can say that increasing the number of qubits makes the randomly generated states easily fall inside classes with more persistent connected components. This seems to indicate that, increasing the number of parties, qubits in a genuine entangled state tend to dislike pairwise entanglement and rather share it with the whole set of other qubits.

The proposed method could be particularly useful to classify eigenstates of solvable Hamiltonians such as [144]. For non-integrable systems, the method can still be applied to approximate eigenstate derived by various techniques such as perturbation theory or numerical analysis. The procedure is to first

verify whether or not the eigenstate has genuine multipartite entanglement (as explained in section III.B) and if it is so it is possible to verify the different classes they belong to.

Another consideration is related to the subject of quantum algorithms and their computational complexity classes. As quantum speedup is based on the entanglement employed in the algorithm, it would be interesting to investigate possible links between quantum computational complexity classes and entanglement classification (see [145] along this line), here provided by persistent homology.

Last, the proposed approach could be extended to the case of mixed states and then find applications in the context of open systems dynamics [146].

Quantum Kernel Methods

Kernel functions are similarity functions that constitute a very useful tool in ML. In fact kernel functions can be expressed as inner product in another (usually larger) space without explicit feature mapping, enabling linear learning algorithms to learn highly non-linear decision boundaries. In many applications where data classification is based on dissimilarity measures (e.g. string matching for pattern recognition), kernels provide a valid alternative method for classification.

In Section 4.2 we have seen that quite a lot of effort have been put into the development of quantum algorithms based on kernel methods. The main approaches to quantize them either rely on the formulation of a quantum algorithm that can realize the respective of SVM but running on quantum computer (see Sec.4.2.3) or exploit the power of quantum computing to deal with classically intractable kernels (see Sec.4.2.4).

In this chapter, two approaches are proposed. The first one deals with Error Correcting Output Codes (ECOC), which are a standard technique in Machine Learning for efficiently rendering the collective outputs of a binary classifier, such as the Support Vector Machine, as a *multi-class* decision procedure. Appropriate choice of error correcting codes further enables incorrect individual classification decisions to be effectively corrected in the composite output. In the next section, we present a quantum algorithm based on the quantum Support Vector Machine which is able to reproduce the ECOC process with a speedup associated with efficient QRAM calls.

The second approach proposes to use the framework of TQC (see Sec.3.3) to estimate kernels that could be employed in supervised learning tasks. As an example, we show how a Hamming distance based kernel between binary strings can be obtained by an encoding of those strings as some special braiding in TQC. Hamming distance is derived as the Jones polynomial of a particular link. Exploit the computational features of TQC it is possible to compare two strings and compute a Hamming distance kernel function between them.

6.1 Quantum Error Correcting Output Codes

A well studied aspect of Quantum Computing is that of error correction [35], which is crucial in order to protect quantum algorithms from errors induced by environmental de-coherence [152]. Within the emerging subtopic of QML, however, other forms of error become apparent; in particular, *classification error*.

It will be the endeavour of this section to demonstrate that decision errors with respect to the output of *quantum classifier ensembles* are also amenable to error correction. In particular, we demonstrate that the existing advantages of quantizing machine learning algorithms demonstrated in [71, 72, 74, 75, 118, 122] can be further applied to the problem of multi-class ensemble decision-error correction. This will lead to a cumulative performance boost i.e. with respect to both the collaborative decision process and the underlying classifiers in the ensemble.

At first we will look at the individual classifiers of the ensemble in the classical setting; in particular we will focus on the Support Vector Machine (SVM) as this exhibits the dual characteristics of being both a binary and discriminative (as opposed to generative) classifier. Hence the standard classical setting for Error Correcting Output Codes (ECOC) is presented, followed by a discussion of our proposal for a quantum version of ECOC for multi-class classification problems.

6.1.1 Error Correcting Output Codes (ECOC) in Classical Machine Learning

Real world data typically exhibits multiple classes - for example photographs of street scenes may exhibit buildings of various kinds, pedestrians, vehicle, distinct species of animal and plant life etc. Machine learning is therefore commonly tasked with identifying from amongst the different classes when presented with novel data. A powerful way to approach these problems is to break the multi-class problem down into a series of smaller binary classification tasks.

Such two-class problems can then be treated by appropriate binary classifiers (e.g. SVMs) whose decision outputs are combined to provide the sought multi-class classification. Perhaps the simplest such approach is ‘one versus one’, in which classifiers are trained for all possible bi-partitions of classes and a majority vote of their decisions is applied. A more efficient alternative with respect to the number of classifiers is the ‘one versus all’ approach in which a binary classifier is built to distinguish each class from all the others. Again, a decision is made by majority vote to obtain a final class decision allocation. Thus, both methods suffice to convert binary classifiers into multi-class classifiers.

A key difference between ‘one versus one’ and ‘one versus all’ is that both methods have a diverse degree of resilience to *classification error*. The committee decision making process (e.g. majority vote) of the ensemble of classifiers potentially allows for some individually-incorrect decisions, whilst still arriving at a correct collective decision. They are thus, to an extent, *error-correcting*. However, it is demonstrable that neither of these approaches is optimal in this respect nor are they optimal in terms of the training requirements of the classifiers. For this, we need to consider Error Correcting Output Codes (ECOC) [153, 154].

Suppose, again, that (\vec{x}_i, y_i) with $i = 1 \dots M$ are the training vectors/labels, where $\vec{x}_i \in \mathbb{R}^N$ and the label set now extends to $y_i \in \{\omega_1, \omega_2, \dots, \omega_c\}$. There are in general L binary classifiers in the ensemble which are denoted as $h \in \{h_1, h_2, \dots, h_L\}$.

In the ‘one versus all’ method there is one binary classifiers for each class, hence $L = c$ in this case. The classifier h_i labels data from class i as positive and all data from the other classes as negative. A new instance is classified in the class whose corresponding classifier output has the largest value. The committee decision is thus defined as [158]

$$y = \arg \max_{i \in \{1, \dots, c\}} h_i(\vec{x}) \quad (6.1)$$

The ECOC method utilises a *codeword* for each class ω_i . There hence exists a $c \times L$ code matrix \mathcal{M} with values \mathcal{M}_{ij} , with each of the \mathcal{M}_{ij} values drawn from the set $\{1, -1\}$ ¹. The code matrix \mathcal{M} represents L distinct binary classification problems, where each of the individual codes divides the set of classes into two meta-classes (in the literature it *dichotomises* them). It is important to note that the division of the set of class labels into two meta-classes for each of dichotomisers is carried over to the training vectors themselves, i.e. each of the binary dichotomisers are trained on all of the training vectors to maximize their generalising capacity. Both ‘one versus one’ and ‘one versus all’ can thus be phrased in ECOC terms.

The matrix formulation adopted illustrates an important duality: while matrix columns represent the meta-structure of the dichotomisers, matrix rows define uniquely-identifying codewords for each of the

¹ This is the simplest form of ECOC; three valued \mathcal{M}_{ij} are possible i.e. $\mathcal{M}_{ij} \in \{-1, 0, 1\}$, where the zero value represents omission of a class from the meta-class allocation.

underlying classes class ω_i . There are hence two stages to the ECOC process: an encoding and a decoding stage. The coding stage is the constitution of an appropriate code matrix \mathcal{M} ; the decoding process is the derivation of a collective decision from the set of dichotomisers. To see how this works, consider an unlabelled test vector \vec{x} . Each of the meta-class dichotomisers predicts a value in the set $\{1, -1\}$ such that the test vector generates a codeword of length L . This codeword is then compared against the set of codewords constituting the row-wise entries of \mathcal{M} ie $\mathcal{M}_{(i,\cdot)}$. The class i with the closest code value is then allocated as the final ensemble decision:

$$y = \arg \min_{i \in 1, \dots, c} \{ \sum_j |h_j - \mathcal{M}_{ij}|^2 \} \quad (6.2)$$

The error correcting capacity of the ECOC matrix is thus determined by the minimal Hamming distance between codes.

Because the mapping of arbitrary codewords in the test vectors space onto the codeword contained in \mathcal{M} is many to one, the ECOC coding/decoding process has an intrinsic error correction property. A subset of the L dichotomisers can reach incorrect classification decisions with regard to the test vector while retaining a correct ensemble decision. Their errors have, in effect, cancelled themselves out (it may be shown that the ECOC ensemble reduces both classifier bias and variance errors [156]). This property is invaluable in any non-trivial, non-linearly-separable classification problem where there is an intrinsic, inalienable likelihood of error lower-bounded by the Bayes error for each dichotomiser.

6.1.2 Strategy for Quantum Error Correcting Output Codes

Our approach to quantize the ECOC is based on the implementation of the QSVM, discussed in Section 4.2.3, of which we present the multi-class version. Since we have to deal with L binary classifiers, h_j , with $j = 1, 2, \dots, L$, we assume that the QRAM contains the binary vector labels $\vec{y}_j \in \{+1, -1\}^M$ for each of the dichotomisers and that these too can be queried in superposition

$$\sum_{j=1}^L |\vec{y}_j\rangle |j\rangle \quad (6.3)$$

Moreover we assume that the code-word matrix \mathcal{M} is given from the beginning and that the QRAM contains also the binary vectors associated to each row i of the code-word matrix.

Projecting $|y\rangle$ into the eigenbasis $|e_i\rangle$ of the SVM matrix \hat{F} of Eq. 1.11, with eigenvalues λ_i , we obtain what follows

$$\sum_{j=1}^L |\vec{y}_j\rangle |j\rangle |0\rangle \rightarrow \sum_{j=1}^L \sum_{i=1}^{M+1} \langle e_i | \vec{y}_j \rangle |e_i\rangle |j\rangle |\lambda_i\rangle \rightarrow \sum_{j=1}^L \sum_{i=1}^{M+1} \frac{\langle e_i | \vec{y}_j \rangle}{\lambda_i} |j\rangle |e_i\rangle \quad (6.4)$$

A quantum phase estimation algorithm (first arrow of Eq. 6.4) with a successive employment of an eigenvalues inversion are performed. The eigenvalue qubit is hence uncomputed. In the training set basis this gives the solution state for the SVM parameters $\vec{\alpha}_j$ and b_j associated to the j^{th} dichotomiser.

$$\frac{1}{\sqrt{L}} \sum_{j=1}^L |\vec{\alpha}_j, b_j\rangle |j\rangle = \frac{1}{\sqrt{L}} \sum_{j=1}^L \left(\frac{b_j}{N_j} |0\rangle |j\rangle + \sum_{k=1}^M \frac{\alpha_{j,k}}{N_j} |k\rangle |j\rangle \right) \quad (6.5)$$

where $N_j = \left(b_j^2 + \sum_{k=1}^M \alpha_{j,k}^2 \right)^{\frac{1}{2}}$ is the normalization factor. In order to classify a new input, it is necessary to construct the state $|u\rangle$ which is obtained by calling the QRAM using the state of Eq. 6.5.

$$|u\rangle = \frac{1}{\sqrt{L}} \sum_{j=1}^L \left(\frac{b_j}{Z_j} |0\rangle |0\rangle |j\rangle + \sum_{k=1}^M \frac{\alpha_{j,k}}{Z_j} |\vec{x}_k\rangle |k\rangle |\vec{x}_k\rangle |j\rangle \right) \quad (6.6)$$

where $Z_j = \left(b_j^2 + \sum_{k=1}^M \alpha_{j,k}^2 |\vec{x}_k|^2\right)^{\frac{1}{2}}$ is the normalization constant. In addition, the query state associated to new unlabelled vector \vec{x} is constructed as follows

$$|\tilde{x}\rangle = \frac{1}{\sqrt{L}} \sum_{j=1}^L \frac{1}{M|\vec{x}|^2 + 1} \left(|0\rangle |0\rangle |j\rangle + \sum_{k=1}^M |\vec{x}| |k\rangle |\vec{x}\rangle |j\rangle \right). \quad (6.7)$$

We can thus rewrite Eq.s 6.6-6.7 respectively as

$$|u\rangle = \frac{1}{\sqrt{L}} \sum_{j=1}^L |u_j\rangle |j\rangle \quad (6.8)$$

$$|\tilde{x}\rangle = |\tilde{x}_j\rangle \left(\frac{1}{\sqrt{L}} \sum_{j=1}^L |j\rangle \right) \quad (6.9)$$

where

$$|\tilde{x}_j\rangle = \frac{1}{M|\vec{x}|^2 + 1} \left(|0\rangle |0\rangle + \sum_{k=1}^M |\vec{x}| |k\rangle |\vec{x}\rangle \right) \quad \text{and} \quad |u_j\rangle = \frac{b_j}{Z_j} |0\rangle |0\rangle + \sum_{k=1}^M \frac{\alpha_{j,k}}{Z_j} |\vec{x}_k| |k\rangle |\vec{x}_k\rangle$$

In order to obtain the required *multi-class* decision and implement the ECOC scheme, we need an additional stage. As stated in the beginning of this section, we assume that row vectors of the code-matrix \mathcal{M} are contained in the QRAM. Hence, instead of the state $|u\rangle$ alone, consider the state $|u^i\rangle$ associated to the i^{th} row of the code-matrix constructed as follows. Given QRAM access that performs

$$\frac{1}{\sqrt{L}} \sum_{j=1}^L |u_j\rangle |j\rangle \rightarrow \frac{1}{\sqrt{L}} \sum_{j=1}^L |u_j\rangle |j\rangle |\mathcal{M}_{(i,j)}\rangle \quad (6.10)$$

where $|\mathcal{M}_{(i,j)}\rangle$ is the qubit register which encode the values of the i^{th} row of $\mathcal{M}_{(i,j)}$ in binary, i.e. 0 for 1 and 1 for -1 ; applying a Z gate to the last register $|\mathcal{M}_{(i,j)}\rangle$ and uncomputing it we obtain

$$|u^i\rangle = \frac{1}{\sqrt{L}} \sum_{j=1}^L \mathcal{M}_{(i,j)} |u_j\rangle |j\rangle \quad (6.11)$$

Since the QRAM enables efficient storage of matrix values with access in quantum parallel, state preparation for storing each row of the binary-valued matrix \mathcal{M} thus takes place in $O(\log L)$ steps. This is finally followed by a swap test that reveals the value of $\langle \tilde{x} | u^i \rangle$: after the construction of state $\frac{1}{\sqrt{2}}(|0\rangle |u^i\rangle + |1\rangle |\tilde{x}\rangle)$, the probability of measuring the first qubit in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ is given by

$$P_i = \frac{1}{2} (1 - \langle \tilde{x} | u^i \rangle) \quad (6.12)$$

which is obtained with accuracy ϵ in $O\left(\frac{P_i(1-P_i)}{\epsilon^2}\right)$. Note that the inner product $\langle \tilde{x} | u^i \rangle$ can be written as

$$\begin{aligned} \langle \tilde{x} | u^i \rangle &= \frac{1}{L} \left(\langle \tilde{x}_{j'} | \sum_{j'=1}^L |j'\rangle \right) \sum_{j=1}^L \mathcal{M}_{(i,j)} |u_j\rangle |j\rangle = \\ &= \frac{1}{L} \sum_{j=1}^L \mathcal{M}_{(i,j)} \langle \tilde{x}_j | u_j \rangle \end{aligned} \quad (6.13)$$

where the term $\langle \tilde{x}_j | u_j \rangle$ is the one responsible for the classification of the unlabelled state $|\vec{x}\rangle$ with respect to the j^{th} dichotomiser. In fact, if the value of $\langle \tilde{x}_j | u_j \rangle > 0$, then the label associated to $|\vec{x}\rangle$ is $+1$. Conversely, for $\langle \tilde{x}_j | u_j \rangle < 0$, the classification gives a -1 .

The role of $\mathcal{M}_{(i,j)}$ inside the summation appearing in Eq. 6.13 can be better understood with a simple example. Suppose of having a set of three binary classifiers $\{h_1, h_2, h_3\}$ and three classes $\{w_1, w_2, w_3\}$ with ECOC code-words $c_1 = \{1, 1, 1\}$, $c_2 = \{-1, -1, -1\}$ and $c_3 = \{1, -1, -1\}$ respectively. Matrix \mathcal{M} hence has the following form

$$\mathcal{M} = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ 1 & -1 & -1 \end{pmatrix} \quad (6.14)$$

For the first class w_1 , Eq. 6.13 becomes

$$\begin{aligned} \langle \tilde{x}|u^1 \rangle &= \frac{1}{3} (\mathcal{M}_{(1,1)} \langle \tilde{x}_1|u_1 \rangle + \mathcal{M}_{(1,2)} \langle \tilde{x}_2|u_2 \rangle + \mathcal{M}_{(1,3)} \langle \tilde{x}_3|u_3 \rangle) = \\ &= \frac{1}{3} (\langle \tilde{x}_1|u_1 \rangle + \langle \tilde{x}_2|u_2 \rangle + \langle \tilde{x}_3|u_3 \rangle), \end{aligned} \quad (6.15)$$

for the second class w_2 instead we get

$$\begin{aligned} \langle \tilde{x}|u^2 \rangle &= \frac{1}{3} (\mathcal{M}_{(2,1)} \langle \tilde{x}_1|u_1 \rangle + \mathcal{M}_{(2,2)} \langle \tilde{x}_2|u_2 \rangle + \mathcal{M}_{(2,3)} \langle \tilde{x}_3|u_3 \rangle) = \\ &= \frac{1}{3} (-\langle \tilde{x}_1|u_1 \rangle - \langle \tilde{x}_2|u_2 \rangle - \langle \tilde{x}_3|u_3 \rangle) \end{aligned} \quad (6.16)$$

and for the third class w_3 we obtain

$$\begin{aligned} \langle \tilde{x}|u^3 \rangle &= \frac{1}{3} (\mathcal{M}_{(3,1)} \langle \tilde{x}_1|u_1 \rangle + \mathcal{M}_{(3,2)} \langle \tilde{x}_2|u_2 \rangle + \mathcal{M}_{(3,3)} \langle \tilde{x}_3|u_3 \rangle) = \\ &= \frac{1}{3} (\langle \tilde{x}_1|u_1 \rangle - \langle \tilde{x}_2|u_2 \rangle - \langle \tilde{x}_3|u_3 \rangle). \end{aligned} \quad (6.17)$$

Suppose now that the single dichotomisers classify the unlabelled state $|\tilde{x}\rangle$ as shown below

$$\begin{aligned} \langle \tilde{x}_1|u_1 \rangle > 0 &\rightarrow |\tilde{x}\rangle \text{ labelled } +1 \text{ by } h_1, \\ \langle \tilde{x}_2|u_2 \rangle > 0 &\rightarrow |\tilde{x}\rangle \text{ labelled } +1 \text{ by } h_2, \\ \langle \tilde{x}_3|u_3 \rangle < 0 &\rightarrow |\tilde{x}\rangle \text{ labelled } -1 \text{ by } h_3. \end{aligned} \quad (6.18)$$

To the state $|\tilde{x}\rangle$, it is hence associated the codeword $c_x = \{+1, +1, -1\}$. We can now rewrite Eq.s (6.15-6.16-6.17) respectively as follows

$$\langle \tilde{x}|u^1 \rangle = \frac{1}{3} (|\langle \tilde{x}_1|u_1 \rangle| + |\langle \tilde{x}_2|u_2 \rangle| - |\langle \tilde{x}_3|u_3 \rangle|) \quad (6.19)$$

$$\langle \tilde{x}|u^2 \rangle = \frac{1}{3} (-|\langle \tilde{x}_1|u_1 \rangle| - |\langle \tilde{x}_2|u_2 \rangle| + |\langle \tilde{x}_3|u_3 \rangle|) \quad (6.20)$$

$$\langle \tilde{x}|u^3 \rangle = \frac{1}{3} (|\langle \tilde{x}_1|u_1 \rangle| - |\langle \tilde{x}_2|u_2 \rangle| + |\langle \tilde{x}_3|u_3 \rangle|) \quad (6.21)$$

As we can see from Eq.s (6.19),(6.20) and (6.21), when the sign of $\langle \tilde{x}_j|u_j \rangle$ is in accordance with the sign of the correspondent element $\mathcal{M}_{(i,j)}$, the summation in (6.13) obtains a positive contribution. Conversely, when the sign of $\langle \tilde{x}_j|u_j \rangle$ is opposite to the one of $\mathcal{M}_{(i,j)}$, the summation in (6.13) gets a negative contribution. The effect of this procedure is to increase the value of $\langle \tilde{x}|u^i \rangle$ whose related class i has the closest code-word with respect to c_x . The unlabelled vector $|\tilde{x}\rangle$ is therefore assigned to the class with the highest $\langle \tilde{x}|u^i \rangle$.

Each P_i is an estimate of the distance from the correct code-word for the class and the classification that the dichotomisers actually generated. The estimation of probability P_i of equation (6.12) is used to decide the final class allocation via an *argmax* process. If all dichotomisers are ideal, the classes fully linearly separable and the query vector within the hamming bound for the ECOC codes, then the probability mass is entirely centred on the relevant class.

Note that we have an implicit logarithmic compression of the ECOC scheme since training and label vectors are \log_2 compressed and accessed simultaneously via the QRAM. Moreover only $\log_2 L + 1$ additional qubit to the Q-SVM are required to index all the L dichotomisers in the state $|j\rangle$. ECOC gains thus exist in addition to the QSVM speedup.

6.1.3 Comments

The emergent field of Quantum Machine Learning proposes to leverage the capabilities of quantum computation in order to achieve greater machine learning algorithms performance than would be possible classically. One of the principal quantum machine learning algorithms, the quantum support vector machine of Rebentrost, Mohseni & Lloyd [118] is able to obtain a significant computational speed increase in the case of a binary SVM classifier.

The work proposed here is an extension of the contribution [118] to the multi-class scenario. This is possible due to the quantization of the ECOC scheme, a method that combines many binary classifiers, called dichotomisers, to solve a multi-class problem. Our quantum implementation of ECOC is able to estimate the distance from the correct code-word for each class and the classification that the dichotomisers actually generated. It does so with an additional speedup associated with efficient QRAM calls. The class with the closest code value is then allocated. This work constitutes a fruitful expansion of the current range of quantum algorithms that can be applied to recognize patterns in data, specifically in the context of multi-class classification.

6.2 Hamming Distance Kernelization via TQC

In Sec. 4.2.4 the relation between feature maps, kernel methods and quantum computation was discussed. In particular, it is interesting the idea of interpreting the encoding of classical inputs \vec{x}_i into a quantum state $|\phi(\vec{x})\rangle$ as a feature map ϕ which maps classical vectors to the Hilbert space associated with a system of qubits. In the standard circuit based quantum computing framework, this requires a circuit $U_\phi(x)$ realizing the mapping

$$\phi : \vec{x} \rightarrow |\phi(\vec{x})\rangle = U_\phi(x)|000..0\rangle \quad (6.22)$$

obtaining a kernel as inner product between states

$$K(\vec{x}_i, \vec{x}_j) = \langle 000..0 | U_\phi^\dagger(x_i) U_\phi(x_j) | 000..0 \rangle \quad (6.23)$$

In order for quantum computing to be helpful, such kernel shouldn't be efficiently simulated by a classical computer. It is therefore posed the question of what type of feature map circuits U_ϕ lead to powerful kernels for classical learning models like SVM but that, at the same time, are classically intractable.

In Sec.3.3 we saw that TQC is equivalent in computational power to the other models of quantum computation such as the quantum circuit model and the quantum Turing machine model. However, certain algorithms are more naturally implementable on a topological quantum computer. A well-known example of such algorithms is the one for evaluating invariant for knots and links called the Jones polynomial [69]. The problem of approximating the Jones polynomials at any fixed root of unity is a BQP problem, it is in fact classically hard and requires all the power that quantum computing can offer. In particular, Jones polynomial naturally arise from the probability outcomes

$$\langle \psi | \mathbf{B} | \psi \rangle = \frac{\langle (B)^{Markov} \rangle}{d^{n-1}}, \quad \text{with } d = (-A^2 - A^{-2}) \quad (6.24)$$

where $|\psi\rangle$ is the quantum state of the vacuum, \mathbf{B} is the unitary quantum evolution to which is associated a braiding B , the knot (or link) $(B)^{Markov}$ is obtained from the closure of the braid B and $\langle \cdot \rangle$ is the Kauffman polynomial which is proportional to the Jones polynomial.

The idea again is to interpret the quantum evolution of TQC, expressed in terms of braiding, as a feature map that is able to map some classical input \vec{x} into the quantum Hilbert space of TQC

$$\phi : \vec{x} \rightarrow |\phi(\vec{x})\rangle = \mathbf{B}_{\vec{x}} |\psi\rangle \quad (6.25)$$

Hence obtaining a kernel whose evaluation depends on the Jones polynomial of the knot (or link) generated from the closure of the composed braid $B_x^\dagger B_y$

$$K(\vec{x}, \vec{y}) \equiv \langle \psi | \mathbf{B}_x^\dagger \mathbf{B}_y | \psi \rangle = \frac{\langle (B_x^\dagger B_y)^{Markov} \rangle}{d^{n-1}}$$

In order to have an advantage in using TQC, the above kernel should be classically hard to approximate. The key point is to associate a braiding to each classical vector in a meaningful way. In addition, the braiding must be complicated enough to produce a knot (or link) whose Jones polynomial is hard to calculate. However, designing such an encoding of classical data into braids could be a hard task. In the following we provide an example of how it is possible to construct a simple Hamming distance based kernel from a braiding encoding of binary strings.

The Hamming distance of two strings is defined as the number of positions in which the strings are different. As well as its use throughout computer science, the Hamming distance is interesting from the perspectives of statistical data analysis and machine learning in that it is used in many *kernel functions*. We show the relationship between Hamming distance and Topology and we use it to define a quantum algorithm computing a Hamming distance based kernel.

The quantum algorithm we present is essentially the application of the Jones polynomial algorithm after an appropriate problem reduction. This is obtained by an encoding of binary strings as some special braiding in TQC and deriving their Hamming distance as the Jones polynomial of the link resulting from the closure of the obtained braids. We can then exploit the computational features of TQC for comparing two strings and obtain an estimation of the Hamming distance between them.

With the present work we aim at studying the suitability of TQC for kernel methods.

6.2.1 Topological Quantum Calculation of Hamming Distance Between Binary Strings

In this section we define a topological quantum algorithm for the approximation of the Hamming distance between two binary strings. This will be the base for the definition of a distance based kernel.

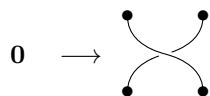
Hamming distance

Given two binary strings u and v of length n , the Hamming distance $d_H(u, v)$ is the number of components (bits) by which the strings u and v differ from each other.

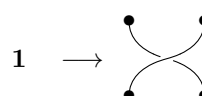
Encoding Binary Strings in TQC

Given a binary string u , we associate to each 0 and 1 in u a particular braiding between two strands as follows:

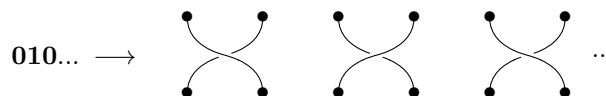
▷ 0 is identified with the crossing σ_i



▷ 1 is identified with the crossing σ_i^\dagger



Note that, using this encoding, a given binary string of length n is uniquely represented by a pairwise braiding of $2n$ strands i.e. by a braid $B \in B_{2n}$ as shown below.



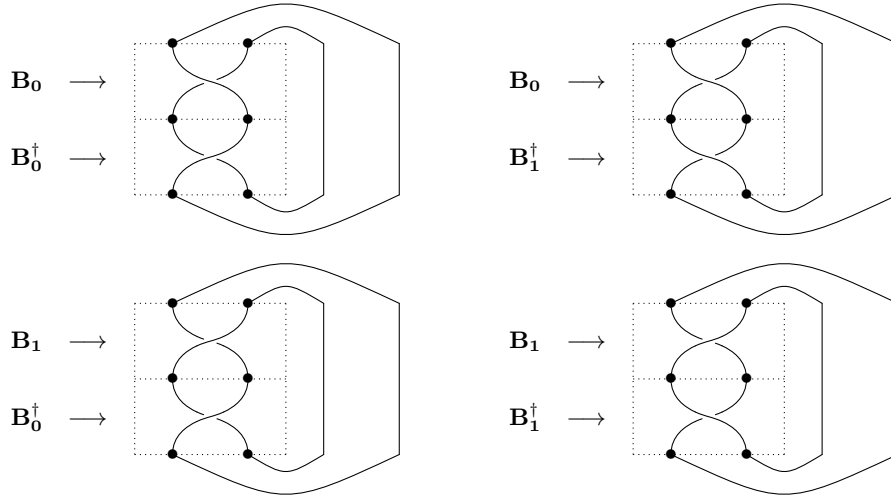


Fig. 6.1: Links associated to the Hamming distance between two single-digit binary strings.

Hamming Distance Calculation: Simple Case

Given two binary strings of length one ($n = 1$), u and v , we consider the braiding operators, \mathbf{B}_u and \mathbf{B}_v , associated to u and v , respectively. Then we construct the composite braiding operator $\mathbf{B}_u^\dagger \mathbf{B}_v$ and apply the Markov trace, obtaining a link. Our aim is to calculate the Hamming distance $d_H(u, v)$ by exploiting the properties of the Kauffman brackets associated to these links. All the possible cases are shown below.

As we can see from Figure 6.1,

- $d_H(0, 0)$ and $d_H(1, 1)$ can be continuously transformed in two loops (using the Reidemeister move) with Kauffman brackets (using rules in Section 3.3.3)

$$\langle \bigcirc \sqcup \bigcirc \rangle = (-A^2 - A^{-2}) \langle \bigcirc \rangle = d \langle \bigcirc \rangle = d$$

- $d_H(1, 0)$ and $d_H(0, 1)$ are both represented by the Hopf link with Kauffman brackets (calculated as in Section 3.3.2)

$$\langle \mathbf{Hopf} \rangle = (-A^4 - A^{-4})$$

If we could perform the calculation of such Kauffman brackets using anyons, as discussed in Section 3.2, we would get:

- for $d_H(0, 0)$ and $d_H(1, 1)$

$$\langle \psi | \mathbf{B}_u^\dagger \mathbf{B}_v | \psi \rangle = \frac{\langle (\mathbf{B}_u^\dagger \mathbf{B}_v)^{Markov} \rangle}{d^{2n-1}} = \frac{\langle \bigcirc \sqcup \bigcirc \rangle}{d^{2-1}} = \frac{d}{d} = 1$$

- for $d_H(1, 0)$ and $d_H(0, 1)$

$$\langle \psi | \mathbf{B}_u^\dagger \mathbf{B}_v | \psi \rangle = \frac{\langle (\mathbf{B}_u^\dagger \mathbf{B}_v)^{Markov} \rangle}{d^{2n-1}} = \frac{\langle \mathbf{Hopf} \rangle}{d}$$

This means that, when the Hamming distance is zero (i.e. in the cases $d_H(0, 0)$ and $d_H(1, 1)$), the probability of the anyons fusing back into the vacuum is 1. When the hamming distance is 1 instead (i.e. in both cases $d_H(0, 1)$ and $d_H(1, 0)$), this probability reduces to $\left| \frac{\langle \mathbf{Hopf} \rangle}{d} \right|^2$.

Hamming Distance Calculation: General case

What was shown in the previous paragraph can be easily generalised. Consider two binary strings u and v of length $n > 0$ such that $d_H(u, v) = k$.

This means that Markov trace of the $2n$ strand used in the encoding will give a number $2(n - k)$ of loops and k Hopf links. Hence, the Kauffman bracket is calculated considering the distant union \sqcup between all these $k + 2(n - k) = 2n - k$ links. What we get from anyon braiding is the following:

$$\begin{aligned} \langle \psi | \mathbf{B}_u^\dagger \mathbf{B}_v | \psi \rangle &= \frac{\langle (\mathbf{B}_u^\dagger \mathbf{B}_v)^{Markov} \rangle}{d^{2n-1}} = \frac{\langle \left(\bigsqcup_{i=1}^{2(n-k)} \bigcirc \right) \sqcup \left(\bigsqcup_{j=1}^k \mathbf{Hopf} \right) \rangle}{d^{2n-1}} = \\ &= d^{2(n-k)} \frac{\langle \left(\bigsqcup_{j=1}^k \mathbf{Hopf} \right) \rangle}{d^{2n-1}} = d^{2(n-k)} d^{k-1} \frac{\langle \mathbf{Hopf} \rangle^k}{d^{2n-1}} = \frac{\langle \mathbf{Hopf} \rangle^k}{d^k} \end{aligned}$$

where Property 1.1.1 and the rules of the Kauffman brackets have been used. Finally we can write

$$\langle \psi | \mathbf{B}_u^\dagger \mathbf{B}_v | \psi \rangle = \left(\frac{\langle \mathbf{Hopf} \rangle}{d} \right)^{d_H(u, v)} \quad (6.26)$$

which means that, given two arbitrary binary string u and v , of length n , their associated braiding \mathbf{B}_u and \mathbf{B}_v are such that the probability amplitude of $2n$ anyons fusing back into the vacuum after a braid $\mathbf{B}_u^\dagger \mathbf{B}_v$ is given by a constant $\frac{\langle \mathbf{Hopf} \rangle}{d}$ multiplied by itself a number of times equal to the Hamming distance between the two strings $d_H(u, v)$.

6.2.2 Hamming Distance Based Kernel

Kernel functions are generalised inner products that profoundly extend the capabilities of any mathematical optimisation that can be written in terms of a *Gram matrix* of discrete vectors (for example, a Gram matrix of vectors over training examples in machine-learning or samples requiring interpolation in regression). In particular, the Gram matrix $(\vec{x}_i^T \vec{x}_j)$ may be freely replaced by any kernel function $K(\vec{x}_i, \vec{x}_j)$ that satisfies the Mercer condition, i.e. a condition guaranteeing positive semi-definiteness. Many optimisation problems fall into this category (e.g. the dual form of the support vector machine training problem [70]).

The Mercer space is given in terms of the input space x via $\vec{\phi}(\vec{x})$, where $K(\vec{x}_i, \vec{x}_j) \equiv \vec{\phi}(\vec{x}_i)^T (\vec{\phi}(\vec{x}_j))$. The Mercer condition guarantees the existence of $\vec{\phi}$, but the kernel itself may be defined based on any similarity function that gives rise to a legitimate kernel matrix. A kernel enforces a *feature mapping* of the input objects into a Hilbert space. However, the feature mapping does not need at any stage to be directly computed in itself, the kernel matrix alone is sufficient.

Here, we show how a kernel can be naturally defined using TQC. To this purpose we use the Hamming distance as a demonstrative example of an approach to the definition of kernel methods that may involve more complex distance notions (note that the Hamming distance is essentially the simplest case of an edit distance, which excludes edit operations such as insertion, deletion and substitution; these clearly provide a more general and accurate measure of sequence dissimilarities).

The topological quantum computation of the Hamming distance shown in Section 6.2.1 can be used to define a kernel function. In fact, the encoding of binary strings as vectors $\mathbf{B} | \psi \rangle$ in the anyonic space allows us to define an embedding ϕ into the Hilbert space \mathcal{H} defined by the fusion space of the anyonic configurations, i.e. for each string u , the mapping $\phi(u)$ is such that $\phi(u) = \mathbf{B}_u | \psi \rangle \in \mathcal{H}$. With this, using Equation 6.26 we can define a string kernel by

$$K(u, v) \equiv \langle \psi | \mathbf{B}_u \mathbf{B}_v^\dagger | \psi \rangle = \left(\frac{\langle \mathbf{Hopf} \rangle}{d} \right)^{d_H(u, v)} = \left(\frac{A^4 + A^{-4}}{A^2 + A^{-2}} \right)^{d_H(u, v)}$$

If we work with so-called Fibonacci anyons, we have that $A = e^{\pi i/10}$ and the resulting kernel matrix is positive semi-definite. Thus it satisfies the Mercer condition for a valid kernel. Moreover, we can show

that the Euclidean distance in the Mercer space, i.e. the fusion space \mathcal{H} , can be defined in terms of $\frac{\langle \text{Hopf} \rangle}{d}$. In fact, we have, using the fact that vectors in \mathcal{H} are normalized to unity,

$$\|\phi(u) - \phi(v)\|_{\mathcal{H}}^2 = \|\phi(u)\|_{\mathcal{H}}^2 + \|\phi(v)\|_{\mathcal{H}}^2 - 2\phi(u)^T \phi(v) = 2 - 2K(u, v)$$

6.2.3 Comments

By suitably encoding the Hamming distance calculation problem into a link invariant problem, we have shown how to solve it by means of a topological quantum computer. We have also shown that the anyonic encoding of the string data and their braiding evolution naturally define a kernel function. The choice of a simple distance such as the Hamming distance allowed us to focus on the description of the approach rather than on the technicalities of the encodings of more complex distance notions.

We are not aware of other approaches that associate topological properties to a given problem with no intrinsic topology, in order to exploit TQC. Our aim is to further investigate the potential offered by topological quantum algorithmic techniques for Machine Learning. It will be the subject of future work to extend the range of applicability of TQC to the computation of hard kernels. An example could be the edit distance based kernel between graphs. However an encoding of problems of this kind into a braid/link is a totally unexplored territory, yet worth investigating.

Machine Learning with the D-Wave

In Section 3.2.2 it was explained how quantum annealing is currently used to approach combinatorial optimization problems. In 4.2.5 it was shown how to express a computationally intensive aspect of a ML algorithm in the form of QUBO (or Ising problem) which is the standard way of formulating problems that can be approached by the D-Wave quantum annealer.

Here we consider two different ML problems, namely the minimum spanning tree based clustering and the edit distance based kernel calculation and we show how to make them amenable for a QA treatment. The first section of this chapter is therefore dedicated to the formulation of the QUBO associated to the MST problem with Δ -Degree Constraint, which is the hard computation at the core of the MST based clustering, as explained in Sec. 1.1.2.

The second part of this chapter instead is devoted to the QA based calculation of the graph edit distances, which can be used to speed up the training of a graph classifier employing the edit distance based kernel. Both approaches are tested on the most advanced QA hardware available to the public, namely the D-Wave 2000Q introduced in 3.2.2.

7.1 Quantum Annealing Approach to the Minimum Spanning Tree Problem with Δ -Degree Constraint

Consider a weighted graph $G = (V, E, w)$ composed of a set V of n vertices, a set E of edges and a function $w : E \rightarrow \mathbb{R}$ that associates a weight $w_{v,v'} > 0$ to each edge $(v, v') \in E$.

The Minimum Spanning Tree (MST) problem introduced in Sec.1.1.2 aims at finding the tree T , subgraph of G , that minimizes the sum over all edge weights, touching all the vertices of G . The Δ -Degree constraint (i.e. each vertex in the MST must have degree less or equal than Δ), with $2 \leq \Delta < n - 1$, is introduced to make the problem NP-hard [9].

The aim of the following sections would be at first to reformulate the aforementioned problem in terms that could be understood by the D-wave quantum annealer. This means constructing a QUBO model that realizes the MST with Δ -Degree constraint problem. Then, after a discussion about the properties of the model, the performances of the quantum device for solving this problem are analysed. Note that the focus will be on complete graphs, for which a MST with Δ -Degree constraint surely exists.

7.1.1 QUBO Formulation

Given a complete weighted graph $G = (V, E, w)$ with n vertices and weights $w_{v,v'}$, consider a total ordering of the vertices, i.e. a bijective map that associates to each vertex only one index $i \in \{1, 2, \dots, n\}$ and vice versa. Since our task is to obtain a tree that covers all the nodes of G , we can arbitrary nominate one of the vertices to be the root. In other words we randomly pick one vertex and upgrade it to *root*, denoted by r and associated to the index $i = 1$.

We can now represent the ordering of vertices with the binary variable $x_{v,i}$, defined for each $v \in V \setminus \{r\}$ and $i \geq 2$ as follows

$$x_{v,i} = \begin{cases} 1 & \text{iff } v \text{ is in position } i \\ 0 & \text{otherwise} \end{cases}$$

The relation between a vertex p which is parent of the i^{th} vertex is expressed by the binary variable $y_{p,i}$ as follows

$$y_{p,i} = \begin{cases} 1 & \text{iff } p \text{ is parent of vertex in position } i \\ 0 & \text{otherwise} \end{cases}$$

Because the root has no parent, $y_{p,1}$ must be zero for every vertex. Moreover, since the root has been set from the beginning, it is already known that $y_{p,2} = 1$ only when p is the root. Hence, we can safely restrict the variable $y_{p,i}$ to the cases where $p \in V$ and $i \geq 3$.

Let's now introduce the hard constraints these two variables have to satisfy in order to realize the MST. Since we considered a total ordering of the vertices, the following two QUBO terms should be considered:

$$\sum_{v \in V \setminus \{r\}} \left(\sum_{i=2}^n x_{v,i} - 1 \right)^2 + \sum_{i=2}^n \left(\sum_{v \in V \setminus \{r\}} x_{v,i} - 1 \right)^2 \quad (7.1)$$

Where the first term ensures that only one i is associated to every v while the second term guarantees the converse.

Moreover, a necessary property to obtain a tree requires that each vertex has only one parent. Such property is expressed with the QUBO constraint

$$\sum_{i=3}^n \left(\sum_{p \in V} y_{p,i} - 1 \right)^2 \quad (7.2)$$

Now we need to secure that, if p is in the parent of the vertex in position i , then p shouldn't be at greater or equal position. In QUBO form, such condition becomes

$$\sum_{p \in V \setminus \{r\}} \sum_{i=3}^n \sum_{j \geq i} y_{p,i} x_{p,j} \quad (7.3)$$

Finally, in order to make sure that parents and children are indeed neighbours, the following term that penalizes non-neighbours should be taken into account

$$\sum_{p \in V} \sum_{v \notin N(p), v \neq r} \sum_{i=3}^n x_{v,i} y_{p,i} \quad (7.4)$$

Note that this term does not give any contribution when the graph is complete, since all vertices are neighbours. Anyway we decided to insert this term from completeness, since it becomes relevant when the input graph is not complete.

QUBO terms introduced so far guarantee that the sub-graph of G we are looking for is actually a tree and hence we obtained all the hard constraints necessary in the MST problem.

In order to realize the Δ -Degree constraint, a number $\Delta - 1$ of binary slack variables $z_{p,j}$ are introduced for each vertex. The QUBO term for this constrain is

$$\sum_{p \in V} \left(\sum_{i=3}^n y_{p,i} + \sum_{j=1}^{\Delta-1} z_{p,j} - \Delta + 1 \right)^2 \quad (7.5)$$

where the first term counts the number of children of p having $i \geq 3$, while the plus one at the end takes into account:

- when $p = r$, the edge between the root and the vertex labelled with $i = 2$;
- when $p \neq r$, the edge between p and its parent.

Since $(\sum_{j=1}^{\Delta-1} z_{p,j}) \in \{0, 1, \dots, \Delta - 1\}$ and $(\sum_{j=1}^{\Delta-1} z_{p,j} - \Delta + 1) \in \{0, 1, \dots, 1 - \Delta\}$, Eq. 7.5 can be simplified as follows

$$\sum_{p \in V} \left(\sum_{i=3}^n y_{p,i} - \sum_{j=1}^{\Delta-1} z_{p,j} \right)^2 \quad (7.6)$$

The final QUBO form for the MST problem with Δ -degree constrain is

$$\begin{aligned} & A \sum_{v \in V \setminus \{r\}} \left(\sum_{i=2}^n x_{v,i} - 1 \right)^2 + A \sum_{i=2}^n \left(\sum_{v \in V \setminus \{r\}} x_{v,i} - 1 \right)^2 + \\ & + A \sum_{i=3}^n \left(\sum_{p \in V} y_{p,i} - 1 \right)^2 + A \sum_{p \in V \setminus \{r\}} \sum_{i=3}^n \sum_{j \geq i} y_{p,i} x_{p,j} + \\ & + A \left(\sum_{p \in V} \sum_{v \notin N(p), v \neq r} \sum_{i=3}^n x_{v,i} y_{p,i} \right) + A \sum_{p \in V} \left(\sum_{i=3}^n y_{p,i} - \sum_{j=1}^{\Delta-1} z_{p,j} \right)^2 + \\ & + B \left(\sum_{v \in N(r)} x_{v,2} w_{r,v} + \sum_{v \in V \setminus \{r\}} \sum_{p \in V} \sum_{i=3}^n y_{p,i} x_{v,i} w_{p,v} \right) \end{aligned} \quad (7.7)$$

where the last term minimizes the sum of edge weights w .

As we can see from Eq. 7.7, parameter A have been introduced to define the strength of the hard constraints with respect to the cost term, multiplied instead by parameter B . In general, $A \gg B$ to ensure that the problem is well defined. In our case, the minimum value of A for which the problem is well defined is given by the relation

$$\frac{A}{B} > \max_{v, v' \in V} \{w_{v,v'}\} \quad (7.8)$$

This is due to the fact that valid solutions (i.e.those that satisfy all the hard constraints) and the closest non valid solutions (i.e.those that do not satisfy at least one hard constraint) are one bit-flip far from each other. Hence we must avoid the situation where, with a single bit-flip form a valid solution, we gain more from the B term than what we lose dissatisfying one of the hard constraints. Selecting A and B according to inequality of Eq. 7.8 is sufficient to satisfy such requirement.

7.1.2 Resources

Number of Logical Qubits

A number $(n - 1)^2$ of qubits are needed to represent binary variables $x_{v,i}$, other $n(n - 2)$ qubits are used for the variables $y_{p,i}$ and finally $n(\Delta - 1)$ qubits are associated to $z_{p,j}$. The total number of logical qubits is therefore

$$n_L = (n - 1)^2 + n(n - 2) + n(\Delta - 1) \quad (7.9)$$

In the next histogram, total number of logical qubits n_L is shown varying Δ and n (number of vertices in the input graph G). As we can see n_L scales as n^2 when Δ is fixed while it grows as Δ for chosen n .

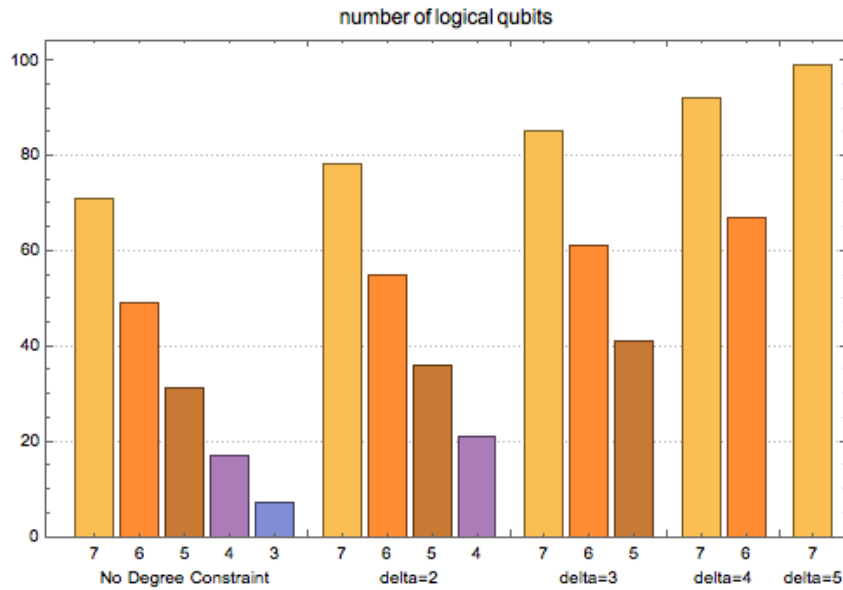


Fig. 7.1: number of logical qubits varying (n, Δ) . Bars in the same group have the same Δ , bars of same colour have the same n .

Connectivity

Each QUBO term present in Eq. 7.7 is responsible for the creation of different coupling (i.e. connections) between logical qubits. The first two constraints expressed by Eq. 7.1 connects all $x_{v,i}$ with the same i and all those with the same v creating the cliques shown in the example below.

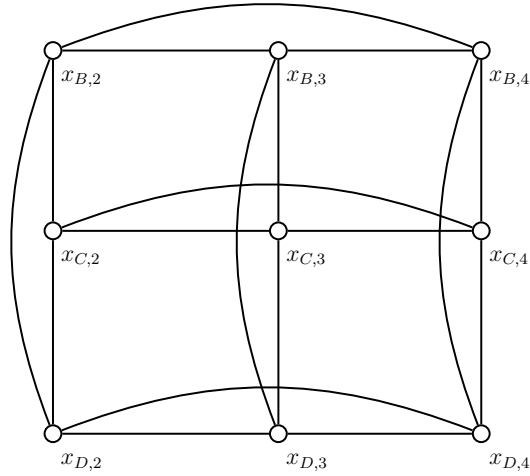


Fig. 7.2: Example of the connectivity for the $x_{v,i}$ elements, due to the first two terms. Consider $G = \{V, E, w\}$ with $V = \{A, B, C, D\}$ and vertex $A = root$. Each row creates a clique. Same happens for columns.

The constraint given by Eq. 7.2 connects all $y_{p,i}$ i with the same index i which hence create a clique.

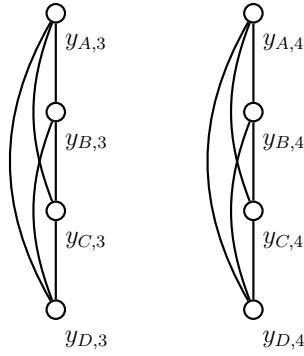


Fig. 7.3: Example of the connectivity for the $y_{p,i}$ elements, due to the third term. Consider $G = \{V, E\}$ with $V = \{A, B, C, D\}$ and vertex $A = root$. Each column form a clique.

The fourth term (the one of Eq. 7.3) connects each $y_{p,i}$ with all the $x_{p,j}$ with same p and $j \geq i$. The $x_{p,2}$ are not connected to any of the $y_{p,i}$. This term alone does not create any new clique but highly connects x and y variables.

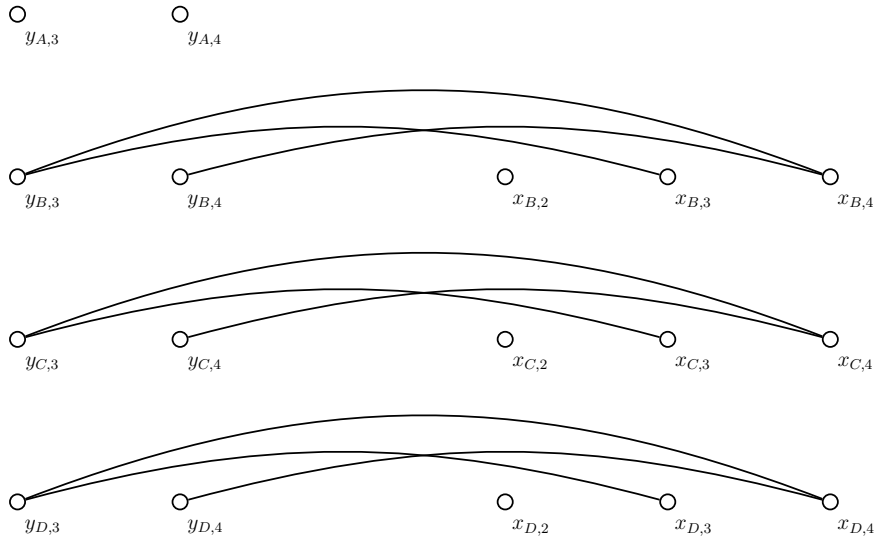


Fig. 7.4: Example of the connectivity between $y_{p,i}$ and $x_{p,j}$ elements, due to the fourth term. Consider $G = \{V, E\}$ with $V = \{A, B, C, D\}$ and vertex $A = root$.

As written before, the term of Eq. 7.4 does not give any contribution in the case of complete graphs. For sparse graphs, it becomes highly important.

The Δ -Degree constraint expressed by Eq. 7.5 connects each $y_{p,i}$ to all the $y_{p,j}$ with different i and same p but also to all the $z_{p,i}$ with same p . This creates a cliques like the one shown below

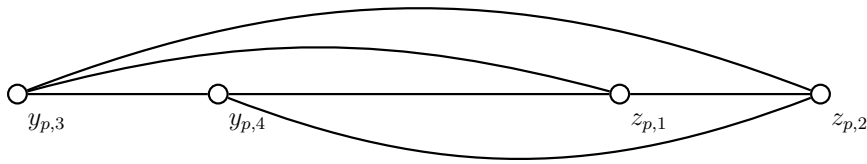


Fig. 7.5: Assume $\Delta = 3$. Example of the connectivity between $y_{p,i}$ and $z_{p,i}$ elements (for a generic p), due to the sixth term. Consider $G = \{V, E\}$ with $V = \{A, B, C, D\}$ and vertex $A = root$.

Finally the cost term that minimizes the total edge weights generates connections between each $x_{v,i}$ and all the $y_{p,i}$ with same i and different p for which it exists the edge (p, v) in G .

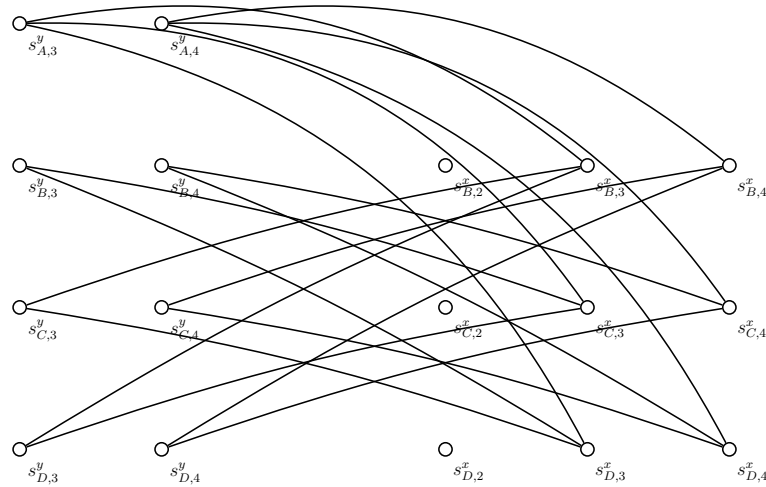


Fig. 7.6: Example of the connectivity between $y_{p,i}$ and $x_{v,i}$ elements, due to the last term of Eq. 7.7. Consider a complete graph $G = \{V, E\}$ with $V = \{A, B, C, D\}$ and vertex $A = root$.

In general, the maximum qubit degree appearing in the QUBO is $4n - 7$ while the total number of edges is given by $\frac{1}{2}(-10 + (29 - 5\Delta + \Delta^2)n + 2(-13 + \Delta)n^2 + 7n^3)$. The table shown below report those connectivity properties.

n	Δ	logical qubits n_L	connections	density δ	max degree
4	2	21	89	0.42381	9
5	2	36	215	0.34127	13
5	3	41	240	0.292683	13
6	2	55	424	0.285522	17
6	3	61	460	0.251366	17
6	4	67	502	0.227047	17
7	2	78	737	0.245421	21
7	3	85	786	0.220168	21
7	4	92	842	0.201147	21
7	5	99	905	0.186559	21

Table 7.1: Comparison between different instances for complete graphs. Here δ indicates the density of connections in the QUBO.

7.1.3 Embedding

In order to use the D-Wave quantum annealer to solve our optimization problem, it is necessary to embed the QUBO structure into the D-Wave architecture.

As we have seen in the previous section, our QUBO for the MST problem with Δ -Degree constraint contains clusters of cliques which are highly connected to each other. This make it impossible to directly embed our problem on the D-Wave Chimera architecture. An appropriate embedding technique is therefore needed. Two different embeddings were taken into consideration: the *fully_connected* deterministic embedding [159] where each logical qubit is mapped to a fixed number of physical qubits and the *find_embedding* heuristic algorithm [51] which tries to find the best representation of the original problem minimizing the number of physical qubits.

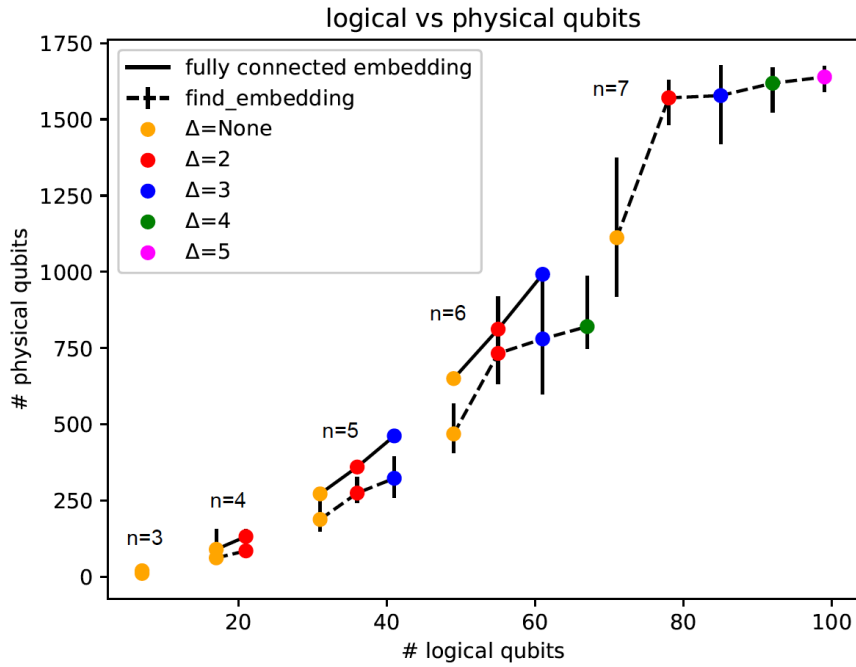
Results obtained using this two embeddings strategies are shown in the following table where: n_p^{fully} is

the number of physical qubits required by the *fully_connected* embedding; n_p^{find} is the median number of physical qubits found by *find_embedding* algorithm; $min(n_p^{find})$ and $max(n_p^{find})$ are respectively the best and worst case scenario obtained with *find_embedding* and finally n_{chain} tells us the length of the largest chain in the best embedding.

n	Δ	n_L	n_p^{fully}	n_p^{find}	# trials	$min(n_p^{find})$	$max(n_p^{find})$	n_{chain}
3	/	7	20	11	10	11	13	2
4	/	17	90	62	100	51	155	4
4	2	21	132	85	100	69	156	5
5	/	31	272	188	100	148	264	8
5	2	36	360	274.5	50	242	327	10
5	3	41	462	323	50	260	394	11
6	/	49	650	468.5	50	407	567	15
6	2	55	812	732.5	20	633	918	20
6	3	61	992	780	20	598	1002	19
6	4	67	/	820.5	20	748	987 A	22
7	/	71	/	1112	20	919	1373	26
7	2	78	/	1570.5	10	1481	1629	33
7	3	85	/	1578	10	1420	1678	36
7	4	92	/	1618.5	10	1522	1671	36
7	5	99	/	1639	10	1591	1675	36

Table 7.2: Fully connected embedding and find_embedding comparison for different instances for complete graphs.

Comparison between *fully_connected* and *find_embedding* becomes clearer in the following plot that show the behaviour of n_p^{fully} and n_p^{find} with respect to the number of logical qubits.



As we can see, the median obtained with *find_embedding* is always lower than *fully_connected*, moreover for $n_L > 61$, it is impossible to find a *fully_connected* embedding because the number of qubits needed would exceed the physical qubits available in the current version of the D-Wave.

For this reason we decided to run our instances only using the *find_embedding* heuristic algorithm, whose median number of physical qubits are displayed in the histogram below, varying (n, Δ) . As we can see, fixing Δ , the n_p^{find} medians scale approximately as n^4 , where n is the number of vertices of the input graph G .

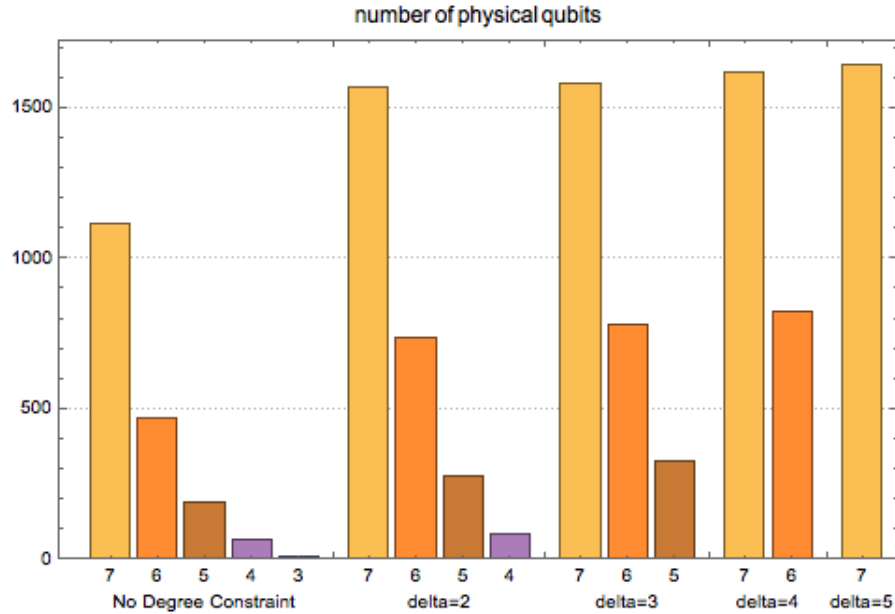


Fig. 7.7: number of physical qubits varying (n, Δ) . Bars in the same group have the same Δ , bars of same colour have the same n .

7.1.4 Experimental Results and Comments

We used the D-Wave 2000Q quantum annealer to solve the Δ -Degree MST for complete graphs of the following sizes:

- $n = 4$ with $\Delta = 2$
- $n = 5$ with $\Delta = 2, 3$
- $n = 6$ with $\Delta = 2, 3$

where different values of the QUBO parameters $A = 2, 3, 4, 5$ and $B = 1$ have been used for each (n, Δ) in order to study which one leads to better performances.

We generated 200 random instances with edge weights randomly chosen among the set $[0.5, 0.7, 0.9, 0.11, 0.13, 0.15]$ and used the best embedding (i.e. with the minimum number of physical qubits) found by *find_embedding* after 20 trials.

Moreover, the new D-Wave feature *extended_j_range* was employed in the embedding. Such tool enables chains of physical qubits representing a single logical qubit to have a ferromagnetic coupling $J_{chain} = -2$, while all the problem coupling strengths are scaled in the range $[-1, 1]$.

When running the D-Wave there should be a trade-off between finding the solution with a high probability in a single long run and running the algorithm multiple times with a shorter runtime and a

smaller single-run success probability. This trade-off is reflected in the *time-to solution (TTS)* metric, which measures the time required to find the ground state at least once with some desired probability (here taken to be 0.99)

$$T_{99} = \frac{\ln(1 - 0.99)}{\ln(1 - p)} T_{Anneal} \tag{7.10}$$

where p is the probability of finding the optimal solution, checked via an MST exact classical solver.

The following results show the median of the TTS (with probability T_{99}) with error bars that indicate the 35 and 65 percentiles. We used 100000 annealing runs for each instance and an annealing time of $T_{Anneal} = 1\mu s$ per run. The

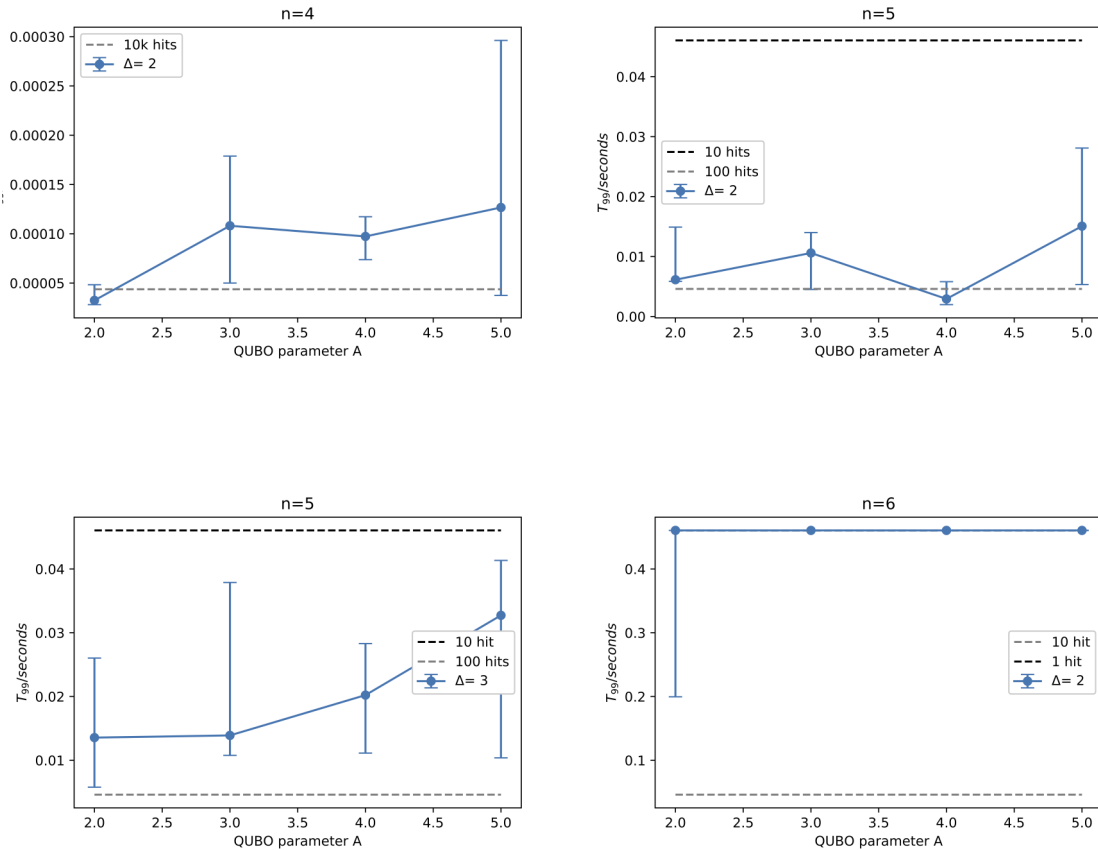


Fig. 7.8: TTS as function of parameter A. Other annealing parameters: *auto_scale*: False (Required when using extended j range. Manual rescaling needed); *flux_drift_compensation*: True (Required when using extended j range, it automatically compensate for biases introduced by strong negative couplings); *gauges*: None (gauges are incompatible with extended j range)

The improvement obtained with *extended_j_range* is remarkable in the cases where $n = 4, 5$. For $n = 6$ the improvement is small because the typical length of the chains (around 20) is way above the optimal range (5-7 qubits) where *extended_j_range* does its best.

From the graphs shown before it is possible to see that the D-wave is able to solve problems of small size involving up to 5 nodes and a $\Delta = 3$ degree constraint. Solutions for $n=6$ and $\Delta = 2$ is also found, but with a very low probability. This lead us to the two following conclusions: first, despite the D-Wave

has more the 2000 working qubits, it starts to fail when the Δ degree MST problem involves around 700 physical qubits; secondly the low density of connections in the physical hardware produces a large increase in the number of binary variables (and therefore qubits) being used.

Performances of the quantum annealer could be improved applying reverse annealing techniques for local refinement of the solution [160] or adding a pause in the annealing schedule for each run of D-Wave [161]. Moreover it could be interesting to study a hybrid approach [162] where a classical solver decompose a QUBO problem by splitting it into sub-problems. Each one of them is solved using the quantum annealer. Such approach is able to deal with larger problems which are inaccessible by the D-Wave alone.

7.2 Quantum Annealing Approach to Edit Distance Calculation

The purpose of this project is to analyse, study and implement the graph edit distance problem using a Quantum Annealer, specifically the D-Wave. A QUBO formulation of the problem was created and then implemented; this formulation was tested on graphs with different number of nodes and edges. By the results achieved, it has been noted that the correct solution is obtained with a fairly high probability for small graphs; instead, increasing in the number of nodes, the probability of obtaining correct solutions decreases until it reaches zero. This happens because the number of physical qubits required for the problem increases with the number of nodes, and consequently, grows the difficulty of mapping the logical problem into the physical and hence the creation of a correct embedding becomes a hard task.

7.2.1 QUBO Formulation

Consider two undirected simple graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, where V_i is the number of nodes and E_i is the number of edges of graph G_i . The graph edit distance (GED) allows us to understand how different is the graph G_1 from G_2 , i.e. the number of operations of node insertion, node elimination or edge insertion and elimination that are necessary to make one graph isomorphic to the other. The question of whether two graphs are isomorphic is believed to be hard, but its classification into a complexity class is still not clear, in fact it is an NP-intermediate problem [163]. Calculating the GED is based on the concept of graph isomorphism and therefore it is at least as difficult as the isomorphism problem.

The QUBO formulation that will be given for this problem does not solve the general GED, but it implements a more restrictive form where only edges are manipulated. Thus, by construction, the number of nodes of G_1 and G_2 must be the same.

Given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ we define a binary variable $x_{v,i}$ that realize a bijection between V_1 and V_2

$$x_{v,i} = \begin{cases} 1 & \text{if vertex } v \in V_2 \text{ gets mapped to vertex } i \in V_1 \\ 0 & \text{otherwise} \end{cases}$$

We are now introducing the initial Hamiltonian, i.e. the hard constraints, which must necessarily be satisfied in order to obtain a solution. To guarantee such bijective mapping it is necessary to formulate \mathcal{H}_A as follows:

$$\mathcal{H}_A = A \sum_v (1 - \sum_i x_{v,i})^2 + A \sum_i (1 - \sum_v x_{v,i})^2 \quad (7.11)$$

As said before the number of nodes of the two graphs are the same, in this case \mathcal{H}_A must be zero in the optimal case. At this stage we also need a second term \mathcal{H}_B which penalize a bad mapping i.e. when two vertices i and j in G_1 that are connected by an edge, $(i, j) \in E_1$, are mapped into vertices u and v in G_2 which are not connected, $(u, v) \notin E_2$ and vice versa.

$$\mathcal{H}_B = B \sum_{i,j \notin E_1} \sum_{u,v \in E_2} x_{u,i} x_{v,j} + B \sum_{i,j \in E_1} \sum_{u,v \notin E_2} x_{u,i} x_{v,j} \quad (7.12)$$

Summarizing, \mathcal{H}_A represent the hard constraints which must be inevitably satisfied i.e. $\mathcal{H}_A = 0$. This is due to the fact that having a bijection between V_1 and V_2 is a necessary requirement for the calculation of the GED. The \mathcal{H}_B term represent the quantity that we want to optimize and whose minimum value corresponds to the GED.

The complete QUBO formulation is:

$$\begin{aligned} \mathcal{H}_{Qubo} = & A \sum_v (1 - \sum_i x_{v,i})^2 + A \sum_i (1 - \sum_v x_{v,i})^2 + \\ & + B \sum_{i,j \notin E_1} \sum_{u,v \in E_2} x_{u,i} x_{v,j} + B \sum_{i,j \in E_1} \sum_{u,v \notin E_2} x_{u,i} x_{v,j} \end{aligned} \quad (7.13)$$

So if the result of Eq. 7.13 is equal to 0, then the two graphs are isomorphic and their GED is 0. If the result is greater than 0, the minimum of \mathcal{H}_{Qubo} will be also the minimum distance between the two graphs, i.e. \mathcal{H}_{Qubo} will return the GED.

Since \mathcal{H}_A is a hard constraint, parameter A should be much higher than B if we want \mathcal{H}_A to be surely satisfied. However, since in the D-Wave quantum annealer all the QUBO coefficients (both linear, h_i , and quadratic, J_{ij}) get normalized in the range $[-1, 1]$, it is important to chose A and B in such a way that the normalized coefficients do not become too small. For this reason, a good choice for parameters A and B is the one where

$$\min_{i,j,k} (h_i^A, J_{jk}^A) \geq \max_{i,j,k} (h_i^B, J_{jk}^B) \quad (7.14)$$

where h_i^A and J_{jk}^A are respectively the linear and quadratic coefficients appearing in \mathcal{H}_A while h_i^B and J_{jk}^B are linear and quadratic coefficients of \mathcal{H}_B .

7.2.2 Resources

Number of Logical Qubit

From the binary variable $x_{v,i}$ result n^2 logical qubits, where n is the number of vertices of bot G_1 and G_2 . In the histogram of Fig.7.9 it is shown how the number of logical qubits varies as the nodes of the graphs n increase.

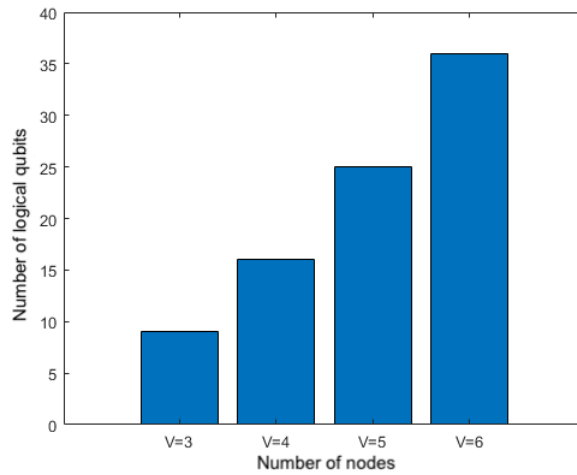


Fig. 7.9: Number of logical qubits used when the number of nodes varies

Connectivity

Each QUBO term in Eq. 7.13 is responsible for the creation of different couplings (i.e. connections) between logical qubits. As an example, in the case of graphs with 3 nodes, terms \mathcal{H}_A and \mathcal{H}_B give the following qubit connectivity:

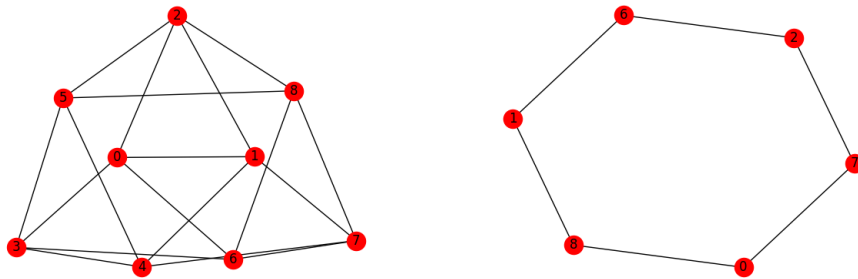


Fig. 7.10: Connectivity between logical qubits generated by \mathcal{H}_A (on the left) and by \mathcal{H}_B (on the right).

Therefore the QUBO, which is the sum of \mathcal{H}_A and of \mathcal{H}_B , is represented by the following graph:

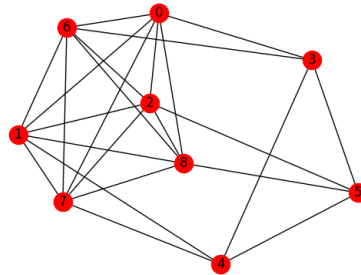


Fig. 7.11: Graph structure associated to the QUBO in the case of problems involving three-nodes graphs.

7.2.3 Embedding

In order to use the D-Wave for solving the problem, it is necessary to match the structure created by the QUBO formulation (Fig.7.11) to the physical structure of the quantum annealer, i.e. find the correct embedding. In order to do so, we used the heuristic algorithm *find_embedding* which tries to find the best representation of the original problem by minimizing the number of physical qubits [51]. As it is possible to see in the histogram of Fig.7.12, the number of physical qubits grows rapidly with the number of nodes.

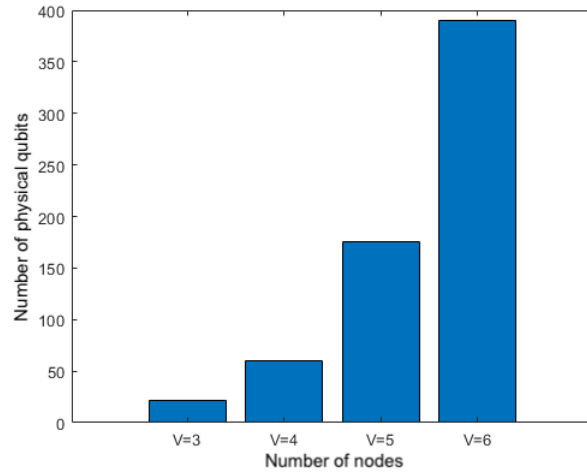


Fig. 7.12: Number of physical qubits used varying the number of nodes

Comparing the two histograms as shown in Fig. 7.13, the increase in the gap between logical and physical qubits becomes evident. This is mainly due to the fact that, since the topology of the QUBO problem is highly connected, long chains of physical qubits are required to represent a single logical qubit.

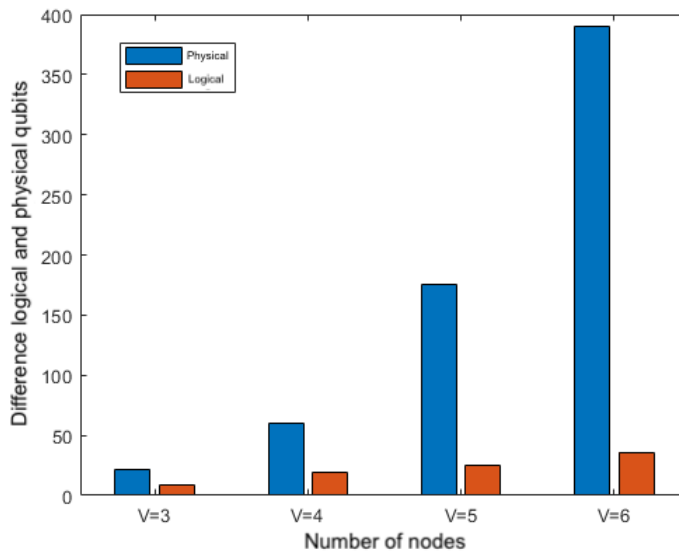


Fig. 7.13: Difference between logical and physical qubits

7.2.4 Experimental Results and Comments

The D-Wave 2000Q™ System has been employed in order to calculate the GED between graphs of the following dimensions:

- Number of edges equal to 3; QUBO parameters: $A = 1, 1.5, 2$ and $B = 1$

- Number of edges equal to 4; QUBO parameters: $A = 1, 1.5, 2$ and $B = 1$
- Number of edges equal to 5; QUBO parameters: $A = 1, 1.5, 2$ and $B = 1$
- Number of edges equal to 6; QUBO parameters: $A = 1, 1.5, 2$ and $B = 1$

Different values for the QUBO parameters have been set. The A parameter is modified in the various cases in order to find the best configuration that leads to the optimum, while the parameter B is left unchanged. Moreover, the D-Wave schedule has been set so that the annealing time is constant and equal to $1\mu sec$ per anneal. Since the D-Wave returns probabilistic results, it is necessary to repeat the run a high number of times (one thousand in this case) in order to obtain a sample of correct solutions from which it is possible to calculate the probability of success. The D-Wave solutions were checked via an exact GED classical solver.

The graphs selected for testing the QUBO on the D-Wave quantum annealer are the following:

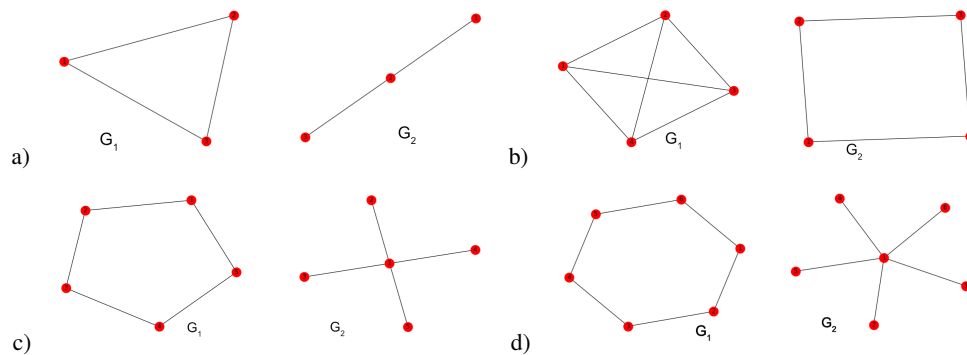


Fig. 7.14: Graphs selected in the case of : a) three nodes, b) four nodes, c) five nodes, d) six nodes.

The values of the GED for the selected graphs, varying the number of nodes n , are the following

$$GED(G_1, G_2) = \begin{cases} 1 & \text{for } n = 3 \\ 2 & \text{for } n = 4 \\ 5 & \text{for } n = 5 \\ 7 & \text{for } n = 6 \end{cases}$$

The histograms below show the number of times the optimal solution was found when the parameter A was changed. In the case of graphs with six nodes, the correct solution has never been found.

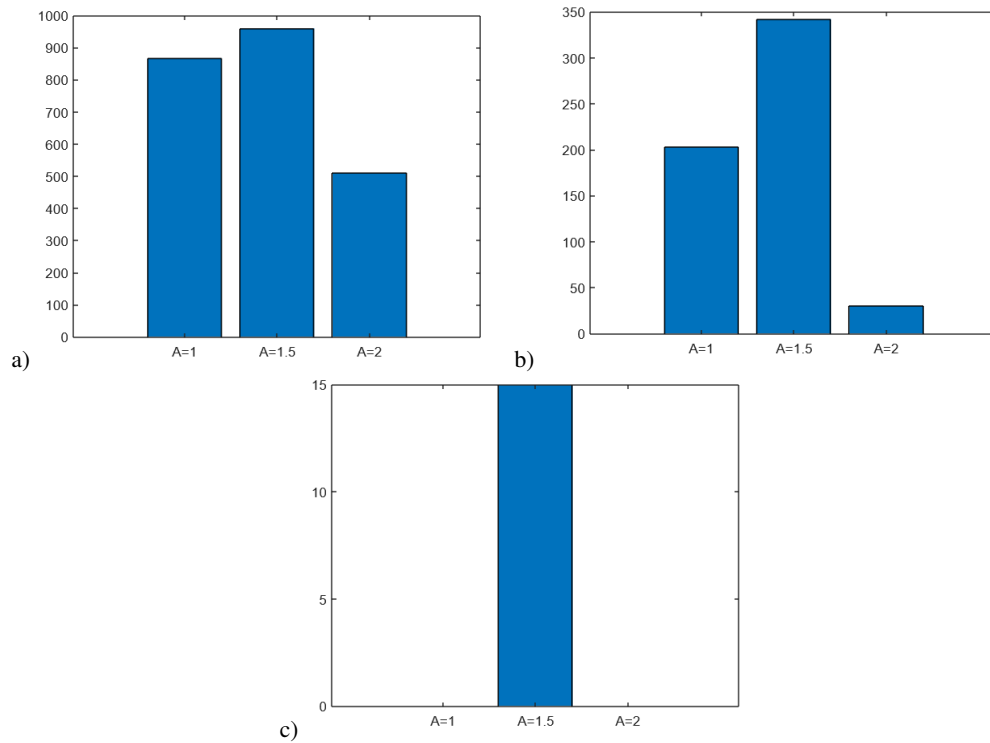


Fig. 7.15: Number of occurrences of the solution in the cases of: a) three nodes, b) four nodes, c) five nodes.

With the present study, a QUBO formulation of the GED problem between undirected and unlabelled graphs with same number of nodes was presented. A quantum annealing approach to solve the GED QUBO on few test graphs was performed using the D-Wave 2000Q device.

Results on test graphs have shown that an exact solution has been found by the D-Wave up to graphs with five nodes. Moreover, among the three values selected for the QUBO parameter A , the intermediate one ($A = 1.5$) has reached the best performances in all cases.

Results obtained using the quantum annealer could be improved applying reverse annealing techniques for local refinement of the solution [160] or adding a pause in the annealing schedule [161]. A hybrid approach [162] could be explored for dealing with graphs with more nodes. Finally, a possible evolution of the QUBO formulation for GED could be directed to take into account graphs with different number of nodes. A further expansion of this study could consider weighted and direct graphs.

Machine Learning with IBM Quantum Computer

Graphs are discrete objects that allow us to schematise a large variety of problems and processes spanning among many research areas [165].

In ML, facial expressions recognition is an application that employs graph theory to construct a mathematical model of a human face and usually a supervised algorithm to identify the correct expression.

Here we propose the use of a quantum computer, in particular the architecture developed by IBM [166], in order to obtain a classification of graphs, and thereby of human facial expressions. In this endeavour we will use some techniques that have been developed in the new growing field of Quantum Machine Learning [15]. We introduce a method for representing graphs as quantum states, that make use of the amplitude encoding technique. In order to define a graph classification we use quantum circuit to train a model that performs a classification similar to the nearest neighbours classification algorithm.

8.1 Dataset and Preprocessing

The dataset used is the freely available Extended Cohn-Kanade (CK+) database which is composed of multiple photos of people labelled with their facial expression, an example is shown below in Fig.8.1



Fig. 8.1: Examples of elements of the dataset: happy face on the left and sad face on the right.

Furthermore, each photo is identified with a point cloud of 68 point of the kind (x, y) , i.e. $v_i \in \mathbb{R}^2$ from which we select only those 20 points associated to the mouth as shown in Fig.8.2. The idea is to associate a weighted graph to each point cloud and the use a graph classifier to distinguish different human expressions.

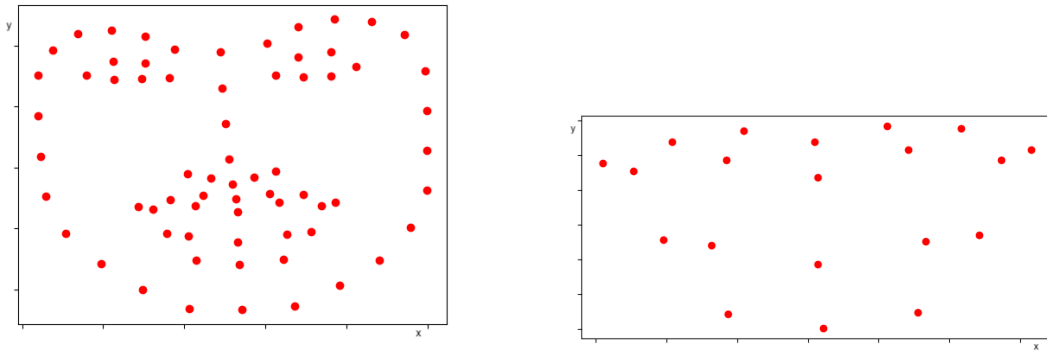
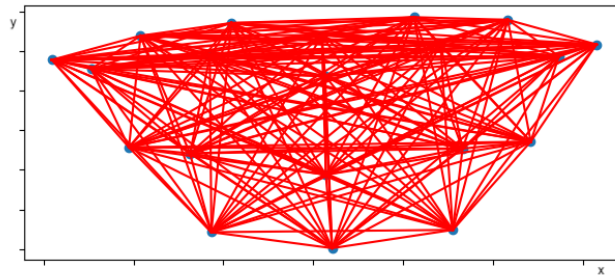


Fig. 8.2: Landmark points (left) and selected mouth points (right) in the (x,y) plane.

In order to obtain such graphs, we opted for two different strategies. The first one considers the weighted complete graph whose vertices are the n landmark points of the mouth and whose edge-weights $w_{i,j}$ are equal to the Euclidean distance between vertices i and j . The second strategy used the Delaunay triangulation between points of the mouth in order to obtain a meshed weighted graph, where weights are the same used for the previous method. The complexity of the triangulation algorithm is $O(n \log n)$. Given a mouth landmark point cloud as in Fig.8.2, the output of these two strategies is shown below in Fig.8.3

Complete graph



Meshed graph

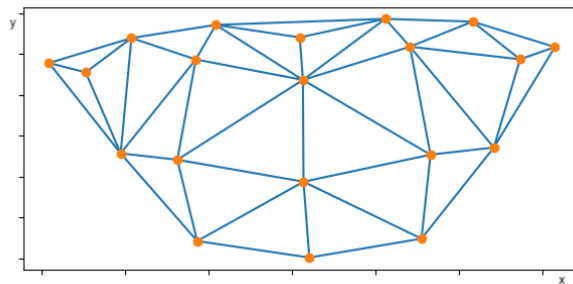


Fig. 8.3: Complete graph and meshed graph obtained from the mouth landmark points, using all the 20 nodes that identify the mouth

8.2 Encoding Graphs into Quantum States

Consider an undirected simple graph $G = (V_G, E_G)$, where V_G is the set of vertices and E_G the set of edges in G . By fixing an ordering of the vertices $\{v_i\}_{i=1..n}$, a graph G is uniquely identified by its adjacency matrix A_G with generic element

$$a_{ij} = \begin{cases} 1 & \text{if } e_{ij} \in E_G, \\ 0 & \text{if } e_{ij} \notin E_G. \end{cases} \quad (8.1)$$

Therefore if the cardinality of V_G is n , i.e. the graph G has n vertices, then A_G is a $n \times n$ square matrix with zeros on the diagonal. Since we are dealing with undirected simple graphs, A_G is also symmetric and this means that the meaningful information about graph G is contained in the $d = \frac{n^2-n}{2}$ elements of upper triangular part of A_G . We can now vectorize those elements and rename them as follows

$$\mathbf{a}_G = (a_{1,2} \ a_{1,3} \ \dots \ a_{1,n} \ a_{2,3} \ a_{2,4} \ \dots \ a_{2,n} \ \dots \ a_{n-1,n})^T = \left(g_1^{(G)} \ g_2^{(G)} \ g_3^{(G)} \ \dots \ g_d^{(G)} \right)^T, \quad (8.2)$$

where

$$g_{n(i-1) - \frac{i(i+1)}{2} + j}^{(G)} \equiv a_{i,j} \quad (8.3)$$

From vector \mathbf{a}_G we can construct a quantum state $|G\rangle$ associated to graph G by encoding the elements of the adjacency vector into the amplitudes of the quantum state, as done in [118]:

$$|G\rangle = \frac{1}{\gamma} \sum_{k=1}^d g_k |k\rangle \quad (8.4)$$

where γ is a normalization constant given by

$$\gamma = \sqrt{\sum_{\{k \text{ s.t. } g_k \neq 0\}} |g_k|^2} \quad (8.5)$$

This encoding can be extended to the case of weighted graphs $G = (V_G, E_G, w_{ij})$ where $w_{ij} \in \mathbb{R}_{\geq 0}$ is the weight associated to the edge e_{ij} . In this case, the adjacency matrix is defined as

$$a_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E_G, \\ 0 & \text{if } e_{ij} \notin E_G. \end{cases} \quad (8.6)$$

Quantum state encoding is then performed as in Equations 8.2-8.4-8.5. This allows us to represent a classical vector of d elements into a quantum state of $N = \lceil \log(d) \rceil$ qubits via amplitude encoding.

On the IBM Qiskit framework [166], states expressed by Equation 8.4 are realized following the method proposed by Shende et al. [167]. The purpose of the algorithm is to find an initialization circuit that takes a qubit register to an arbitrary target state specified by a vector of amplitudes. In our case this target state is $|G\rangle$. The basic idea lays in the assumption that the quantum register already starts in the desired target state, and then we construct a circuit that takes it to the $|00\dots 0\rangle$ state. The initialization circuit is then the reverse of such circuit. In our case such initialization circuit is identified by the unitary \mathbf{G} and it is such that

$$|G\rangle = \mathbf{G}|00\dots 0\rangle \quad (8.7)$$

where \mathbf{G} has the form

$$\mathbf{G} = \begin{bmatrix} R_y(-\theta_0)R_z(-\phi_0) & & & & \\ & R_y(-\theta_1)R_z(-\phi_1) & & & \\ & & \ddots & & \\ & & & R_y(-\theta_{2^N-1-1})R_z(-\phi_{2^N-1-1}) & \\ & & & & \ddots \end{bmatrix}^\dagger \quad (8.8)$$

and can be decomposed (not efficiently, with a runtime of $O(2^{N+1})$) as a circuit constituted by a sequence of the elementary gates CX , $R_y(\theta)$ and $R_z(\phi)$.

8.2.1 Example

Imagine to select $n = 4$ random vertices among those belonging to the mouth landmark points. The complete graph constructed using those points is shown in Fig.8.4

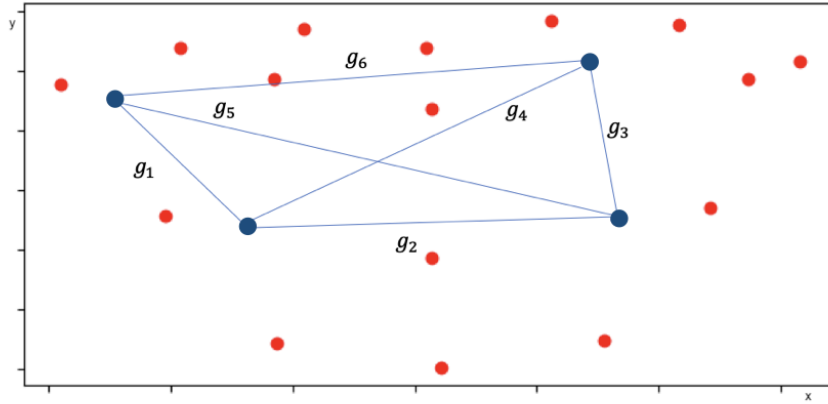


Fig. 8.4: Complete graph constructed using 4 randomly selected mouth landmark points

Such graph is encoded into the quantum state $|G_4\rangle$

$$|G_4\rangle = \frac{1}{\gamma} (g_1|000\rangle + g_2|001\rangle + g_3|010\rangle + g_4|011\rangle + g_5|100\rangle + g_6|101\rangle)$$

where γ is a normalization constant. The quantum circuit G_4 that realizes $|G_4\rangle$ i.e. $G_4|000\rangle = |G_4\rangle$ and that is obtained as explained in the previous section using the method proposed by Shende et al. [167] is shown below in Fig.8.5.

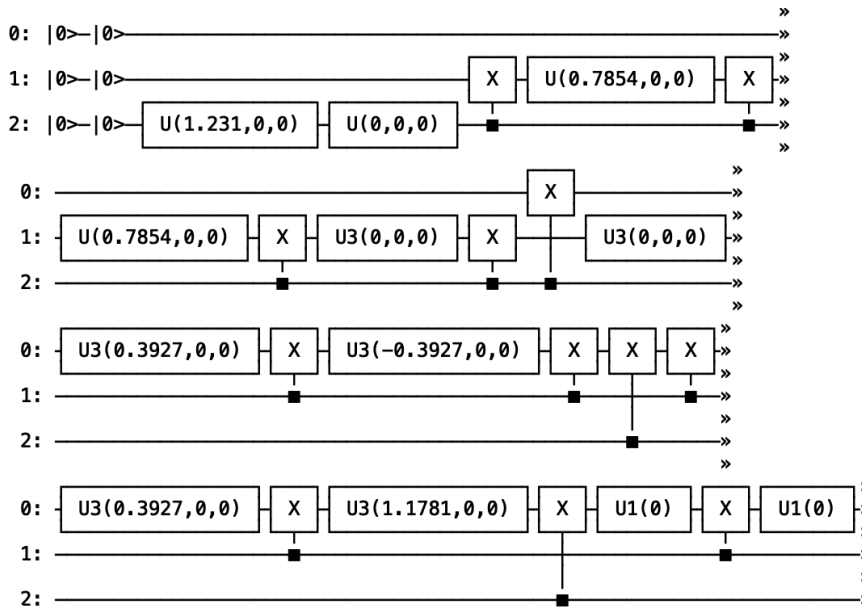


Fig. 8.5: Quantum circuit realizing the state G_4 . Three qubits identified as 0, 1 and 2 are employed in the circuit.

As claimed before this algorithm is exponential in the number of qubits used and represent a bottleneck to the speed of the whole algorithm that could be in principle be overcome both with a faster algorithm and with a more complex connection between physical qubits in the hardware.

A Quantum Random Access Memory (QRAM) could in principle construct efficiently a quantum state like $|G\rangle = \frac{1}{\gamma} \sum_{k=1}^d g_k |k\rangle$, provided that the classical vector (g_1, g_2, \dots, g_d) is relatively uniform, i.e. without a few g_i 's that are vastly larger than the others [168]. This might be the case for states $|G\rangle$ associated to meshed graphs. For further considerations on the possible caveats of QRAM see [80].

8.3 Graphs Quantum Classifier

The first implementation of a quantum classifier realized on the IBM quantum computer is due to [164] where input data is expressed in the form a single qubit state. Here we extend their circuit in order to deal with a dataset whose elements are represented by states of multiple qubits.

Given a dataset \mathcal{D} , e.g. coming from the facial expression recognition problem, consider the training set

$$\mathcal{D}_{train} = \{(G^{\{1\}}, y^{\{1\}}), \dots, (G^{\{M\}}, y^{\{M\}})\} \quad (8.9)$$

of M pairs $(G^{\{m\}}, y^{\{m\}})$, where $G^{\{m\}}$ are graphs (e.g. representing the face of an individual) while $y^{\{m\}} \in \{c_l\}_{l=1}^L$ identify which of the L possible class labels is associated to the graph. Classes partition graphs into groups sharing common features, in our case sad and happy faces.

Given a new unlabelled input G_{test} , the task is to assign a label y_{test} to G_{test} using the distance [169]

$$d(G_1, G_2) = \sqrt{|\mathbf{a}_{G_1} - \mathbf{a}_{G_2}|^2} = \sqrt{\sum_{i=1}^d |g_i^{(G_1)} - g_i^{(G_2)}|^2} \quad (8.10)$$

and then classifying according to

$$\min_{c_l} \left[\sum_{m \text{ s.t. } y^{\{m\}} = c_l} d(G_{test}, G^{\{m\}}) \right] \quad (8.11)$$

Note that in the case of binary classification there are only two classes i.e. $y^{\{m\}} \in \{+1, -1\}$ and the classifier of Equation 8.11 can be expressed as

$$y_{test} = -\text{sgn} \left[\sum_{m=1}^M y^{\{m\}} d(G_{test}, G^{\{m\}}) \right] \quad (8.12)$$

The quantum circuit that implements such a classification requires four multi-qubit quantum registers. In the initial configuration:

- $|0\rangle_a$ is an ancillary qubit that is entangled to the qubits encoding both the test graph and training graphs;
- $|00\dots 0\rangle_m$ is a register of $\lceil \log(M) \rceil$ qubits that stores the index m of the training graphs $G^{\{m\}}$;
- $|00\dots 0\rangle_g$ is a register of $\lceil \log(d) \rceil$ qubits used to encode both test and training graphs;
- $|00\dots 0\rangle_c$ is a register of $\lceil \log(L) \rceil$ qubits that stores the classes $y^{\{m\}} \in \{c_l\}_{l=1}^L$ of the $G^{\{m\}}$.

In the case of a training set of two graphs, one per class, the circuit is implemented as shown in Fig.8.6.

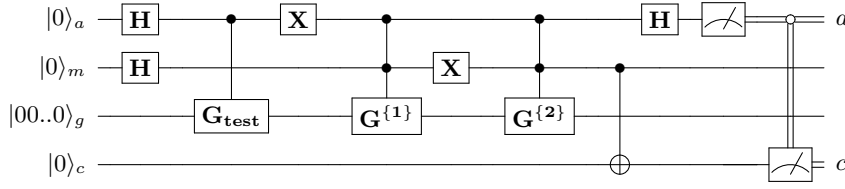


Fig. 8.6: Graph quantum binary classifier circuit

After the first two Hadamard gates the state of the circuit is the following

$$\frac{1}{2} (|0\rangle_a + |1\rangle_a) (|0\rangle_m + |1\rangle_m) |00..0\rangle_g |0\rangle_c \quad (8.13)$$

The **Control- G_{test}** gate, followed by an **X** operator on the ancilla produces the state

$$|G_{test}\rangle = \frac{1}{\gamma_{test}} \sum_{k=1}^d g_k^{test} |k\rangle \quad (8.14)$$

and entangles it with the $|0\rangle_a$ state of the ancilla. Then a $G^{\{1\}}$ gate is controlled by both the first qubit a and by the second one m , which is also subjected to an **X** gate afterwards. This has the effect of creating the state associated to the training graph $G^{\{1\}}$

$$|G^{\{1\}}\rangle = \frac{1}{\gamma^{\{1\}}} \sum_{k=1}^d g_k^{\{1\}} |k\rangle \quad (8.15)$$

and entangle it with both $|1\rangle_a$ and $|0\rangle_m$. Then, the double controlled $G^{\{2\}}$ gate has a similar effect since it stores the second training graph state $|G^{\{2\}}\rangle$ entangling it with $|1\rangle_a$ and $|1\rangle_m$. At this stage, the **Control-X** gate entangles the m index of the two test graphs with the correct class state $|0\rangle_c$ (or $|1\rangle_c$). The state of the circuit at this point can be written as

$$\frac{1}{\sqrt{2M}} \sum_{m=0}^{M-1} \left(|0\rangle_a |G_{test}\rangle_g + |1\rangle_a |G^{\{m\}}\rangle_g \right) |m\rangle_m |y^{\{m\}}\rangle_c \quad (8.16)$$

Finally the Hadamard gate applied to the ancilla generates the state

$$\frac{1}{2\sqrt{M}} \sum_{m=0}^{M-1} \left[|0\rangle_a \left(|G_{test}\rangle_g + |G^{\{m\}}\rangle_g \right) + |1\rangle_a \left(|G_{test}\rangle_g - |G^{\{m\}}\rangle_g \right) \right] |m\rangle_m |y^{\{m\}}\rangle_c \quad (8.17)$$

Measuring the ancilla to be in state $|0\rangle_a$ produces the state

$$\frac{1}{\sqrt{\sum_m \sum_i |g_i^{(G_{test})} + g_i^{(G^{\{m\})}|^2}} \sum_{m=0}^{M-1} \sum_{i=1}^d \left(g_i^{(G_{test})} + g_i^{(G^{\{m\})} \right) |i\rangle_g |m\rangle_m |y^{\{m\}}\rangle_c \quad (8.18)$$

a subsequent measurement of the class qubit $|y^{\{m\}}\rangle$ in the state $|0\rangle_c$ is obtained with probability

$$\begin{aligned}
p(y^{\{m\}} = 0) &= \frac{1}{\sum_m \sum_i |g_i^{(G_{test})} + g_i^{(G^{\{m\})}|^2} \sum_{m \text{ s.t. } y^{\{m\}}=0} \sum_{i=1}^d |g_i^{(G_{test})} + g_i^{(G^{\{m\})}|^2} = \\
&= 1 - \left(\frac{1}{\sum_m \sum_i |g_i^{(G_{test})} + g_i^{(G^{\{m\})}|^2} \sum_{m \text{ s.t. } y^{\{m\}}=0} d(G_{test}, G^{\{m\}})^2 \right) \quad (8.19)
\end{aligned}$$

The probability of obtaining $|y^{\{m\}}\rangle$ in the state $|0\rangle_c$ depends on the distance between the test graph G_{test} and all those training graphs belonging to the class $c = 0$. Therefore if it is valued to be lower than 0.5, then the test set is classified as $y_{class} = -1$ while if higher, $y_{class} = +1$. The quantum circuit hence realizes the classification expressed in Eqn. 8.12 .

8.4 Experimental Results and Comments

The number of qubits used by the quantum classifier scales with n which is the number of vertices in the graphs, selected randomly among those of the mouth landmark points, see the following table. The number of elements in the adjacency matrix that identify the graphs is denoted as $d = \frac{n(n-1)}{2}$.

n	$d \approx O(n^2)$	# qubits used by the algorithm
3	9	7
4	16	8
5	25	8
10	100	10
20	400	12
50	2500	15
100	10000	17
500	250000	21
1×10^7	1×10^{14}	50

Therefore the encoding allows for an exponential compression of resources since only a number $O(\log_2(n^2))$ of qubits are needed to represent a complete graph of n nodes. The quantum classifier is evaluated on both strategies using the `qasm_simulator` and the real 14 qubits quantum computer `ibmq_16_melbourne` available through the IBM cloud platform. We compared the results with a classical binary classifier based on the Frobenius norm between graph adjacency matrices defined as:

$$||A|| = \left[\sum_{i,j} abs(a_{i,j})^2 \right]^{1/2} \quad (8.20)$$

so that the distance used in Eq. 8.12 becomes

$$d(G_{test}, G^{\{m\}}) = \left[\sum_{i,j} abs(g_i^{test} - g_i^{\{m\}})^2 \right]^{1/2} \quad (8.21)$$

In more details, the dataset we used is composed of:

- 250 samples of the form $\{G_{test}, G^{\{1\}}, G^{\{2\}}\}$ have been used for the classical classifier and the quantum IBM simulator `qasm_simulator`
- 16 samples of the form $\{G_{test}, G^{\{1\}}, G^{\{2\}}\}$ have been used for the real 14 qubits quantum computer `ibmq_16_melbourne`

Moreover, the number of vertices n used by the algorithm was gradually increased by randomly selecting them among the 20 landmark points of the mouth.

The metric used to evaluate the correctness of the classifier is the accuracy which is calculated as the number of samples correctly classified over the number of total samples.

$$\text{accuracy} = \frac{\text{correctly classified samples}}{\text{total samples}} \quad (8.22)$$

An accuracy of 1 indicates that all samples were correctly classified, 0 indicates that no sample has been classified correctly.

The values of the accuracy obtained with `qasm_simulator` and a classical binary classifier are shown in the plot below

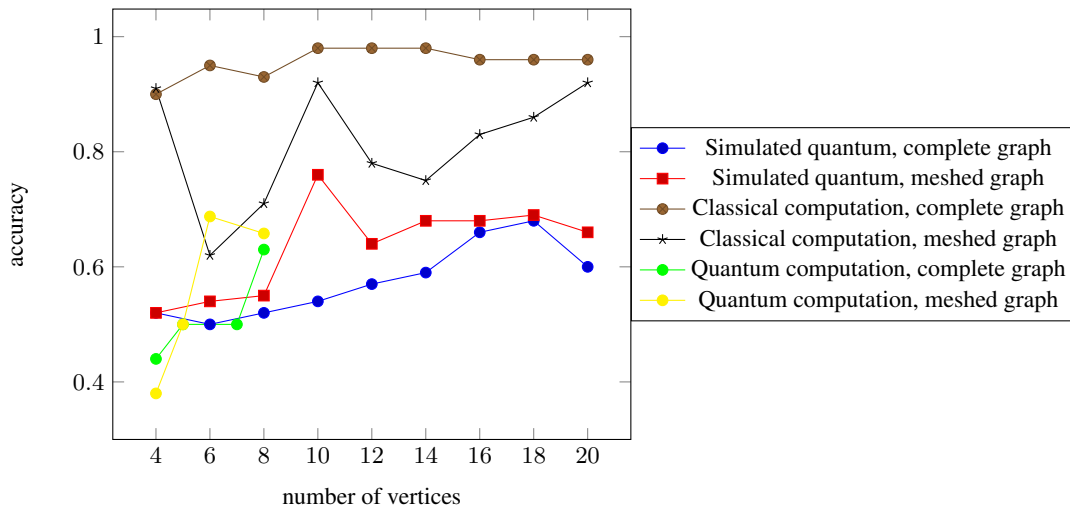


Fig. 8.7: Accuracy varying number of vertices obtained with the `qasm_simulator`, the classical binary classifier and the `ibmq_16_melbourne` quantum device

The highest accuracy is obtained with the classical classifier of Eq. 8.21, using the complete graph strategy. After, we find the classical classifier with meshed graphs. For the quantum simulator, the meshed strategy performs better than the complete one. This is due to the fact that the complete graph encoding into a quantum state is much more complex since there are always $n(n-1)/2$ amplitudes which are different from zero. In the case of meshed graph, the graph is sparse and much less amplitudes are needed in the quantum state encoding the graph.

The actual quantum computation was only possible for graphs with up to 8 vertices; after that we couldn't get any outcome from the quantum computer. One reason why this happened could be that the number of gates used exceeded the coherence time of the qubits. Moreover, results of the accuracy obtained on the quantum computer are biased by the fact that the dataset is composed only of 16 samples.

Finally it is worth noticing that both the classical and simulated meshed have a spike at $n = 10$ which seems to be a good compromise between number of vertices used and sparsity of the graph.

Interesting extensions to this work are the study of different graph encodings into quantum states [171], the usage of other quantum classification schemes [172], and finally the use of the latest 53 qubit IBM quantum computing chip.

Conclusions

In recent years, quantum computing and machine learning have gained a lot of attention in their respective areas of research for their great potentials in solving hard computational tasks. This success pushed towards the birth of a new interdisciplinary field called Quantum Machine Learning that merges in different ways quantum computing and machine learning techniques in order to achieve improvements in both fields. It was the aim of this doctoral thesis to expand the range of applicability of QML to new scenarios.

The first proposed idea involves the application of Topological Data Analysis in order to characterize entanglement. In particular, we have introduced a homology-based technique for the analysis of multi-qubit entangled state vectors. In this approach, a quantum state was associated to a dataset by introducing a metric-like measure defined in terms of bipartite entanglement. Hence, by analysing the persistence of homologies at different scales, we achieved a novel classification of multi-qubit entanglement.

Secondly, we have proposed two theoretical approaches for realizing quantum kernel methods on device specific hardware have been proposed. The first one deals with Error Correcting Output Codes (ECOC), which is a standard method in Machine Learning employing an ensemble of binary classifier, such as Support Vector Machine, for a multi-class classification problem. Appropriate choice of error correcting codes further enables incorrect individual classification decisions to be effectively corrected in the composite output. We have proposed a quantum algorithm based on the quantum Support Vector Machine that is able to reproduce the ECOC process with a speedup associated with efficient QRAM calls. The other approach explores the relation between TQC (see Section 3.3) and kernel methods in ML. We have presented a method for computing a Hamming distance based kernel between binary strings via TQC. This is obtained by an encoding of binary strings as some special braiding and deriving their Hamming distance as the Jones polynomial of a particular link. Another contribution of this thesis is a proposal of using quantum annealing for solving

- Minimum Spanning Tree Problem with Δ -Degree Constraint, which is a hard subroutines involved in the ML algorithm of minimum spanning tree clustering;
- Graph Edit Distance, which could be used to construct an edit distance based kernel function.

Both problems are formalized as combinatorial optimization problems in terms of QUBO and, after an analysis of the structure of the model, solved using the D-Wave 200Q quantum annealing device. We have shown that in both cases the quantum hardware is able to solve problems of small size and that future improvements on the hardware density of connections is essential in order to claim any advantage with respect to classical solvers.

Finally, we have considered the problem of graph classification in the context of facial expression recognition and showed how to solve it using a fully quantum classifier implemented on the IBM quantum computer available through the cloud platform IBM Quantum Experience. We have applied the method introduced in [164] which consists in exploiting quantum interference to define an efficient classifier for a given dataset. The quantum circuit implementing our graph classifier is based on an encoding

of a graph adjacency matrix into the amplitudes of a quantum state. A test on the IBM quantum hardware revealed interesting properties of the quantum classifier with respect to a classical k-NN algorithm.

The results presented in this thesis all imply further investigation. In particular, improvements of the experimental part are conditioned by the evolving progress of the physical devices currently in use or the construction of new ones based on new and possibly more powerful approaches.

References

- [1] C. Bishop, *Pattern Recognition and Machine Learning*, 738, Springer-Verlag New York (2016)
- [2] T. Mitchell, *Machine Learning*, McGraw Hill, New York (1997)
- [3] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach* (2nd ed.). Prentice Hall., (2003)
- [4] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*. 46 (3): 175–185 (1992)
- [5] B. S. Everitt, S. Landau, M. Leese, D. Stahl, *Miscellaneous Clustering Methods, Cluster Analysis*, 5th Edition, John Wiley & Sons, Ltd, Chichester, UK (2011)
- [6] D. MacKay, An Example Inference Task: Clustering, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press., 284-292 (2003)
- [7] P. S. Bradley, U. M. Fayyad, Refining Initial Points for k-Means Clustering, *Proceedings of the Fifteenth International Conference on Machine Learning* (1998)
- [8] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering, *Machine Learning*, 75, 245-249 (2009)
- [9] R.M. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations*, ed. R.E. Miller, J.W. Thatcher and J.D. Bohlinger, 85 (1972)
- [10] S. Theodoridis, *Pattern Recognition*, 984, Elsevier Academic Press (2008)
- [11] J. Mercer, Functions of positive and negative type and their connection the theory of integral equations, *Philosophical Transactions of the Royal Society of London* 209, 415-446 (1909)
- [12] M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of Machine Learning*, 432, MIT Press (2012)
- [13] C. Cortes, V. Vapnik, Support-vector networks, *Machine Learning*, 20, 273–297 (1995)
- [14] J. A. K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.*, 9, 293–300 (1999)
- [15] P. Wittek, *Quantum Machine Learning*, 176, Elsevier Academic Press (2014)
- [16] T. Asano, B. Bhattacharya, M. Keil, F. Yao, Clustering algorithms based on minimum and maximum spanningtrees, *Proceedings of the 4th Annual Symposium on Computational Geometry*, 252-257 (1988)
- [17] Y. Xu, V. Olman, D. Xu, Minimum spanning trees for gene expression data clustering, *Genome Informatics*, 12, 24-33 (2001)
- [18] C. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, *IEEE Transactions on Computers* (1971)
- [19] O. Grygorash, Y. Zhou, Z. Jorgensen, Minimum Spanning Tree Based Clustering Algorithms, 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), 73-81 (2006)
- [20] A. Zomorodian, G. Carlsson, Computing persistent homology. *Discrete Comput. Geom.*, 33(2), 249-274 (2005)
- [21] G. Carlsson, Topology and data. *AMS Bulletin*, 46(2), 255-308 (2009)
- [22] A. Hatcher, *Algebraic Topology*, Cambridge University Press (2002)

- [23] H. Edelsbrunner, *A Short Course in Computational Geometry and Topology*, Springer Publishing Company (2014)
- [24] K. Borsuk, *Fundamenta Mathematicae* 35(1), 217-234, (1948).
- [25] R. Ghrist, Barcodes: the persistent topology of data *Bulletin of the American Mathematical Society*, 45 (2008)
- [26] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, Stability of persistence diagrams, *Proc. 21st Sympos. Comput. Geom.*, 263–271 (2005)
- [27] J. Reininghaus, S. Huber, U. Bauer, R. Kwitt, A stable multi-scale kernel for topological machine learning, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4741-4748 (2015)
- [28] R. Horodecki, P. Horodecki, K. Horodecki, Quantum Entanglement *Review of Modern Physics*, 81, 865–942 (2009)
- [29] A. Einstein, B. Podolsky, N. Rosen, Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.* 47, 777 (1935)
- [30] E. Schrödinger, M. Born. Discussion of probability relations between separated systems. *Mathematical Proceedings of the Cambridge Philosophical Society* (1935)
- [31] E. Schrödinger, P.A.M. Dirac. Probability relations between separated systems. *Mathematical Proceedings of the Cambridge Philosophical Society* (1936)
- [32] J. S. Bell. On the Einstein-Podolsky-Rosen paradox, *Physics Vol. 1, 3*, 195-290, Physics Publishing Co. (1964)
- [33] S. J. Freedman, J. F. Clauser, Experimental Test of Local Hidden-Variable Theories, *Physical Review Letters* 28 (14), 938-941 (1972)
- [34] A. Aspect, P. Grangier, G. Roger, Experimental Realization of Einstein-Podolsky-Rosen-Bohm Gedankenexperiment: A New Violation of Bell’s Inequalities, *Physical Review Letters* 49 (2), 91-94 (1982)
- [35] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge (2000)
- [36] P.A.M. Dirac, A new notation for quantum mechanics, *Mathematical Proceedings of the Cambridge Philosophical Society*, (1939)
- [37] A. Peres, Separability Criterion for Density Matrices, *Physical Review Letters* 77, 1413–1415 (1996)
- [38] G. Vidal, R.F. Werner, A computable measure of entanglement, *Phys. Rev. A* 65, 032314 (2002)
- [39] G.H. Golub, C. F. Van Loan, *Matrix Computations* (3rd ed.), Johns Hopkins University Press. (1996)
- [40] W. K. Wootters, *Quantum Information & Computation* 1, 1 (2001)
- [41] M. Walter, D. Gross, J. Eisert, Multi-partite entanglement arXiv:1612.02437 (2017)
- [42] M. A. Nielsen, Conditions for a Class of Entanglement Transformations, *Phys. Rev. Lett.* 83, 436-439 (1999)
- [43] G. Vidal, Entanglement monotones, *Journ. of Mod. Opt.* 47, 355 (2000)
- [44] W. Dur, G. Vidal, and J. I. Cirac, Three qubits can be entangled in two inequivalent ways, *Physical Review A* 62, 062314 (2000)
- [45] F. Verstraete, J. Dehaene, B. De Moor, H. Verschelde, Four qubits can be entangled in nine different ways *Phys. Rev. A* 65, 052112 (2002)
- [46] G. Gour, N. R. Wallach, All Maximally Entangled Four Qubits States, *Journal of Mathematical Physics* 51, 112201 2010
- [47] R. Cleve, A. Ekert, C. Macchiavello, M. Mosca, Quantum Algorithms Revisited, *Proceedings: Mathematical, Physical and Engineering Sciences* 454, 339-354 (1998)
- [48] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, H. Neven, What is the Computational Value of Finite-Range Tunneling? *Phys. Rev. X* 6, 031015 (2016)
- [49] M. Born, V. A. Fock, Beweis des Adiabatsatzes, *Zeitschrift für Physik, A*, 51, 165-180 (1928)
- [50] O. Masayuki, N. Hidetoshi, Quantum annealing: An introduction and new developments, arXiv e-prints, 1006.1696 (2010)

- [51] J. Cai, W. G. Macready, A. Roy, A practical heuristic for finding graph minors, eprint arXiv:1406.2741 (2014)
- [52] H. Satō, Algebraic Topology: An Intuitive Approach, Iwanami series in modern mathematics, American Mathematical Society (1999)
- [53] C. C. Adams, The Knot Book, W.H. Freeman (1994)
- [54] L. H. Kauffman, Knots and physics, 4th ed. World Scientific, Singapore, Series on Knots and Everything (2013)
- [55] K. Reidemeister, Knoten und Gruppen, Knotentheorie, Springer Berlin Heidelberg, Berlin, Heidelberg, 41-69 (1932)
- [56] K. Reidemeister, Elementare Begründung der Knotentheorie, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg., 5, 1, 24-32 (1927)
- [57] V. F. R. Jones, A polynomial invariant for knots via von Neumann algebras, Bulletin (New Series) of the American Mathematical Society, Bull. Amer. Math. Soc. (N.S.), 01, 1, 103–111, American Mathematical Society, 12 (1985)
- [58] L. H. Kauffman, State models and the Jones polynomial, Topology, 26, 3, 395 - 407 (1987)
- [59] L. H. Kauffman, New Invariants in the Theory of Knots, Am. Math. Monthly, March 1988, 95, 3 (1988)
- [60] J. K. Pachos, Cambridge, Introduction to Topological Quantum Computation, Cambridge University Press (2012)
- [61] J. W. Alexander, Proceedings of the National Academy of Sciences of the United States of America, 3, 93-95, National Academy of Sciences, A Lemma on Systems of Knotted Curves, 9 (1923)
- [62] A. Markoff, Über die freie Äquivalenz der geschlossenen Zöpfe, Rec. Math. [Mat. Sbornik] N.S. (1936)
- [63] A. Yu. Kitaev, Fault-tolerant quantum computation by anyons, Annals of Physics, 303, 1, 2 - 30 (2003)
- [64] Freedman, Michael H., P/NP, and the quantum field computer, 95, 1, 98-101, Proceedings of the National Academy of Sciences (1998)
- [65] A. Kitaev, J. Preskill, Topological Entanglement Entropy, Phys. Rev. Lett., 96, 11, 110404, 4, American Physical Society (2006)
- [66] F. Wilczek, Quantum Mechanics of Fractional-Spin Particles, Phys. Rev. Lett., 49, 14, 957–959, 0, American Physical Society (1982)
- [67] R. W. Hamming, Error detecting and error correcting codes, Bell System Tech J., 29, 147–160 (1950)
- [68] D. Aharonov, V. Jones, Z. Landau, A polynomial quantum algorithm for approximating the Jones polynomial, Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 427–436 (2006)
- [69] M. H. Freedman, A. Kitaev, Z. Wang, Simulation of Topological Field Theories by Quantum Computers, Commun. Math. Phys., 227, 3, 587–603 (2002)
- [70] C., Corinna, V. Vapnik, Support-Vector Networks, Machine Learning, 20, 3, 273–297 (1995)
- [71] E. Aïmeur, G. Brassard, S. Gambs, Quantum speed-up for unsupervised learning, Machine Learning, 90, 2, 261–287 (2013)
- [72] M. V. Altaisky, N. N. Zolnikova, N. E. Kaputkina, V. A. Krylov, Y. E. Lozovik, N. S. Dattani, Towards a feasible implementation of quantum neural networks using quantum dots, Applied Physics Letters, 108, 10, 103108 (2016)
- [73] J. Barry, D. T. Barry, S. Aaronson, Quantum partially observable Markov decision processes, Phys. Rev. A, 90, 3, 032311, 11, American Physical Society (2014)
- [74] S. Lu, S. L. Braunstein, Quantum decision tree classifier, Quantum Information Processing, 13, 3, 757–770 (2014)
- [75] N. Wiebe, A. Kapoor, K. M. Svore, , Quantum Algorithms for Nearest-neighbor Methods for Supervised and Unsupervised Learning, Quantum Info. Comput., 15, 3-4, Rinton Press, Incorporated (2015)
- [76] C. Bishop, Pattern Recognition and Machine Learning, 738, Springer-Verlag New York (2016)

- [77] Thomas Mitchell, *Machine Learning*, McGraw Hill, New York (1997)
- [78] M. Schuld, I. Sinayskiy, F. Petruccione, An introduction to quantum machine learning, *Contemporary Physics*, 56:2, 172-185 (2015)
- [79] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, *Nature*, 549, 195–202 (2017)
- [80] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, L. Wossnig, Quantum machine learning: a classical perspective, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474 (2018)
- [81] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, *Arxiv 1307.0411* (2013)
- [82] E. Tang, Quantum-inspired classical algorithms for principal component analysis and supervised clustering, *ArXiv 1811.00414* (2018)
- [83] V. Dunjko, H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: a review of recent progress, *Reports on Progress in Physics*, 81 (2018)
- [84] S. Arunachalam, R. De Wolf, A Survey of Quantum Learning Theory, *ArXiv 1701.06806* (2017)
- [85] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gomez, R. Biswas, Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers, *Quantum Science and Technology*, 3 (2018)
- [86] M. Schuld, F. Petruccione, *Supervised Learning with Quantum Computers*, 287, Springer International Publishing (2018)
- [87] S. Arunachalam, V. Gheorghiu, T. Jochym-O’Connor, M. Mosca, P. V. Srinivasan, On the robustness of bucket brigade quantum RAM, *New J. Phys.*, 17, 123010, (2015)
- [88] V. Dunjko et al., Quantum-Enhanced Machine Learning, *Phys. Rev. Lett.*, 117, 6 (2016)
- [89] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, R. Melko, Quantum Boltzmann Machine, *Phys. Rev. X*, 8, 11 (2018)
- [90] D. Crawford, A. Levit, N. Ghadermarzy, J. S. Oberoi, P. Ronagh, Reinforcement Learning Using Quantum Boltzmann Machines, *ArXiv 1612.05695* (2016)
- [91] E. Stoudenmire, D. J. Schwab, Supervised Learning with Tensor Networks *Advances in Neural Information Processing Systems*, 29, 4799-4807 (2016)
- [92] G. Sergioli, E. Santucci, L. Didaci, J. I. A. Miszczak, R. Giuntini, A quantum-inspired version of the nearest mean classifier, *Soft Computing*, 22, 691-705 (2018)
- [93] Y Levine, Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design, *International Conference on Learning Representations* (2018)
- [94] E. Aïmeur, G. Brassard, S. Gambs, Quantum speed-up for unsupervised learning, *Machine Learning*, 90, 261-287 (2013)
- [95] M V. Altaisky, Towards a feasible implementation of quantum neural networks using quantum dots, *Appl. Phys. Lett.* 108, 103108 (2016)
- [96] N. Wiebe, A. Kapoor, K. Svore, Quantum algorithms for nearest-neighbours methods for supervised and unsupervised learning, *Quantum Info. Comput.*, 15, 316-356 (2015)
- [97] J. Barry, D. T. Barry, S. Aaronson, Quantum partially observable Markov decision processes, *Physical Review A*, 90, 032311 (2014)
- [98] S. Lu, S. L. Braunstein, Quantum decision tree classifier, *Quantum information processing*, 13, 757–770 (2014)
- [99] B. Heim., Quantum versus classical annealing of Ising spin glasses, *Science*, 348, 215-217 (2015)
- [100] L. Bottarelli, M. Bicego, M. Denitto, A. Di Pierro, A. Farinelli, R. Mengoni, Biclustering with a quantum annealer, *Soft Computing*, 22 6247–6260 (2018)
- [101] I. Agresti, N. Viggianiello, F. Flamini, N. Spagnolo, A. Crespi, R. Osellame, N. Wiebe, F. Sciarrino, Pattern Recognition Techniques for Boson Sampling Validation, *Phys. Rev. X*, 9, 14 (2019)
- [102] P. Huembeli, A. Dauphin, P. Wittek, Identifying quantum phase transitions with adversarial neural networks, *Phys. Rev. B* 97, 134109 (2018)
- [103] P. Huembeli, A. Dauphin, P. Wittek, Automated discovery of characteristic features of phase transitions in many-body localization, *Phys. Rev. B* 99, 104106 (2019)

- [104] A. Canabarro, F. F. Fanchini, A. L. Malvezzi, R. Pereira, R. Chaves, Unveiling phase transitions with machine learning, *Phys. Rev. B* 100, 045129 (2019)
- [105] J. Gray, L. Bianchi, A. Bayat, S. Bose, Machine Learning Assisted Many-Body Entanglement Measurement, *Phys. Rev. Lett.*, 121, 6 (2018)
- [106] L. Bianchi, E. Grant, A. Rocchetto, S. Severini, Modelling non-markovian quantum processes with recurrent neural networks, *New Journal of Physics*, 20, 12 (2018)
- [107] J. Bang, J. Ryu, S. Yoo, M. Pawłowski, J. Lee, A strategy for quantum algorithm design assisted by machine learning, *New Journal of Physics*, 16 (2014)
- [108] M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, A. Zeilinger, Automated Search for new Quantum Experiments, *Phys. Rev. Lett.* 116, 090405 (2016)
- [109] A. Seif, K. A. Landsman, N. M. Linke, C. Figgatt, C. Monroe, M. Hafezi, Machine learning assisted readout of trapped-ion qubits, *Journal of Physics B: Atomic, Molecular and Optical Physics*, 51, 17 (2018)
- [110] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, *Rev. Mod. Phys.* 91, 045002 (2019)
- [111] M. Benedetti, E. Grant, L. Wossnig, S. Severini, Adversarial quantum circuit learning for pure state approximation, *New Journal of Physics*, 21 (2019)
- [112] A. Di Pierro, S. Mancini, L. Memarzadeh, R. Mengoni. Homological analysis of multi-qubit entanglement, *Europhysics Letters*, 123 (2018)
- [113] L. O'Driscoll, R. Nichols, P. A. Knott, A hybrid machine learning algorithm for designing quantum experiments, *Quantum Mach. Intell.*, 1-11(2019)
- [114] R. Iten, T. Metger, H. Wilming, L. del Rio, R. Renner, Discovering physical concepts with neural networks, *ArXiv* 1807.10300 (2018)
- [115] S. Theodoridis, *Pattern Recognition*, 984, Elsevier Academic Press (2008)
- [116] J. A. K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.*, 9, 293–300 (1999)
- [117] D. Anguita, Quantum optimization for training support vector machines, *Neural Networks*, 16, 763-770 (2003)
- [118] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, *Physical Review Letters*, 113, 5 (2014)
- [119] V. Giovannetti, S. Lloyd, L. Maccone, Quantum Random Access Memory, *Phys. Rev. Lett.*, 100, 4 (2008)
- [120] H. Buhrman, R. Cleve, J. Watrous, R. De Wolf, Quantum fingerprinting. *Physical Review Letters*, 87, 4 (2001)
- [121] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum Algorithm for Linear Systems of Equations, *Physical Review Letters*, 103, 4 (2009)
- [122] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum principal component analysis, *Nature Physics*, 10, 631–633 (2014)
- [123] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, New York, NY, USA (2011)
- [124] Z. Li, X. Liu, N. Xu, J. Du, Experimental Realization of a Quantum Support Vector Machine, *Phys. Rev. Lett.*, 114, 5 (2015)
- [125] P. J. Coles, *Quantum Algorithm Implementations for Beginners*, *ArXiv* 1804.03719 (2018)
- [126] D. Windridge, R. Mengoni, R. Nagarajan, Quantum error-correcting output codes, *International Journal of Quantum Information*, 16 (2018)
- [127] J. R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18 (2016)
- [128] J. A. Nelder, R. Mead, A Simplex Method for Function Minimization, *The Computer Journal*, 7, 308-13 (1965)
- [129] J. C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Trans. Automat. Contr.*, 37, 332-41 (1992)

- [130] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum Circuit Learning Phys. Rev. A, 98, 032309 (2018)
- [131] M. Schuld, N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces Phys. Rev. Lett., 122, 6 (2019)
- [132] V. Havlicek, A. D. Corcoles, Supervised learning with quantum-enhanced feature spaces, Nature, 567, 2019–212 (2019)
- [133] L. A. Goldberg, H. Guo, The complexity of approximating complex-valued ising and tutte partition functions, Computational complexity, 26, 765–833 (2017)
- [134] W. Van Dam, S. Hallgren, L. Ip, Quantum algorithms for some hidden shift problems. SIAM Journal on Computing 36, 763-778 (2006)
- [135] M. Rotteler, Quantum algorithms for highly non-linear boolean functions. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms, 448-457 (2010)
- [136] V. N. Smelyanskiy, E. G. Rieffel, S. I. Knysh, C. P. Williams, M. W. Johnson, M. C. Thom, W. G. Macready, K. L. Pudenz, A Near-Term Quantum Computing Approach for Hard Computational Problems in Space Exploration arXiv:1204.2821 [quant-ph] (2012)
- [137] A. Di Pierro, S. Mancini, L. Memarzadeh, R. Mengoni, Homological analysis of multi-qubit entanglement, Europhysics Letters 123, 30006 (2018)
- [138] R. Mengoni, A. Di Pierro, L. Memarzadeh, S. Mancini, Persistent homology analysis of multiqubit entanglement arXiv:1907.06914 [quant-ph] (2019)
- [139] H. Edelsbrunner, D. Letscher and A. Zomorodian, Topological Persistence and Simplification, Discrete Computational Geometry, 28, 511 (2002)
- [140] K. Zyczkowski, and H. J. Sommers, Induced measures in the space of mixed quantum states, Journal of Physics A: Mathematical and General 34, 35 (2001)
- [141] L. Memarzadeh, and S. Mancini, Minimum output entropy of a non-Gaussian quantum channel, Physical Review A 94, 022341 (2016)
- [142] W. K. Wootters, Entanglement of formation and concurrence, Quantum Information & Computation 1, 1 (2001)
- [143] S. Hill, and W. K. Wootters, Entanglement of a Pair of Quantum Bits, Physical Review Letters 78, 26 (1997)
- [144] E. Lieb, T. Schultz, D. Mattis, Two soluble models of an antiferromagnetic chain, Annals of Physics 16, 407 (1961);
S. Katsura, Statistical mechanics of the anisotropic linear Heisenberg model, Physical Review 127, 1508 (1962);
P. Pfeuty, The one-dimensional Ising model with a transverse field, Annals of Physics 59, 79 (1970);
E. Barouch, B. McCoy, Statistical Mechanics of the XY model. II. Spin-Correlation Functions, Physical Review A 3, 786 (1971)
- [145] H. Jaffali, F. Holweck, Quantum entanglement involved in Grover’s and Shor’s algorithms: the four-qubit case ,arXiv:1811.08894 (2019)
- [146] L. Memarzadeh, S. Mancini, Stationary entanglement achievable by environment-induced chain links, Physical Review A 83, 042329 (2011)
- [147] D. Beckman, A. N. Chari, S. Devabhaktuni, J. Preskill, Efficient networks for quantum factoring. Phys. Rev. A, 54: 1034-1063 (1996)
- [148] G. Valentini, T. G. Dietterich, Low bias bagged support vector machines. In Accepted for publication, International Conference on Machine Learning, ICML, 752-759 (2003)
- [149] L. Breiman, Bagging predictors. Machine Learning, 34:123-140 (1996)
- [150] D. Deutsch, R. Jozsa, Rapid Solution of Problems by Quantum Computation, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 439: 553-558 (1992)
- [151] V. Korepin, L. Grover, Simple algorithm for partial quantum search, Quantum Information Processing, 5: 5-10 (2006)

- [152] P. Shor., Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493-R2496 (1995)
- [153] J. Vitri, P. Radeva, O. Pujol, Discriminant ECOC: A Heuristic Method for Application Dependent Design of Error Correcting Output Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28: 1007-1012 (2006)
- [154] S. Escalera, D. M. Tax, O. Pujol, R. P. Duin, P. Radeva, Subclass Problem-Dependent Design for Error-Correcting Output Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1041-1054 (2008)
- [155] E. L. Allwein, R. E. Schapire, Y. Singer, Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *J. Mach. Learn. Res.*, 1: 113–141 (2001)
- [156] E. B. Kong and T. G. Dietterich, Error-Correcting Output Coding Corrects Bias and Variance. *Machine Learning Proceedings*, 313 - 321 (1995)
- [157] D. Windridge, R. Nagarajan, Quantum Bootstrap Aggregation. *Quantum Interaction*, 115-121 (2017)
- [158] M. Ali Bagheri, G. A. Montazer, S. Escalera, Error correcting output codes for multiclass classification: Application to two image vision problems. *AISP 2012 - 16th CSI International Symposium on Artificial Intelligence and Signal Processing*, 508-513 (2012)
- [159] https://docs.dwavesys.com/docs/latest/c_handbook_9.html#embedding-complete-graphs
- [160] D. Venturelli, A. Kondratyev, Reverse quantum annealing approach to portfolio optimization problems, *Quantum Mach. Intell.*, 1, 17-30 (2019)
- [161] J. Marshall, D. Venturelli, I. Hen, E. G. Rieffel, Power of Pausing: Advancing Understanding of Thermalization in Experimental Quantum Annealers, *Phys. Rev. Applied*, 11, 044083 (2019)
- [162] M. Booth, S. P. Reinhardt, A. Roy, Partitioning Optimization Problems for Hybrid Classical/Quantum Execution, D-Wave Technical report (2017)
- [163] D.S. Johnson, The NP-completeness column, *ACM Transactions on Algorithms*, 1, 160 (2005)
- [164] M. Schuld, M. Fingerhuth, F. Petruccione, Implementing a distance-based classifier with a quantum interference circuit, *EPL Europhysics Letters*, 119, 6 (2017)
- [165] J.A. Bondy, *Graph Theory With Applications*, Elsevier Science Ltd. (1976)
- [166] , H. Abraham, et al., Qiskit: An Open-source Framework for Quantum Computing (2019)
- [167] V. V. Shende, S. S. Bullock, I. L. Markov, Synthesis of quantum-logic circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1000-1010 (2006)
- [168] S. Aaronson, Read the fine print. *Nature Phys* 11, 291-293 (2015)
- [169] D. Banks, K. Carley, Metric inference for social networks, *Journal of Classification*, 11, 1 (1994)
- [170] M. Schuld, N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, *Phys. Rev. Lett.* 122, 040504 (2019)
- [171] F. Tacchino, C. Macchiavello, D. Gerace, et al., An artificial neuron implemented on an actual quantum processor, *npj Quantum Inf* 5, 26 (2019)
- [172] D. K. Park, C. Blank, F. Petruccione, The theory of the quantum kernel-based binary classifier, *Physics Letters A*, 126422, 0375-9601 (2020)

