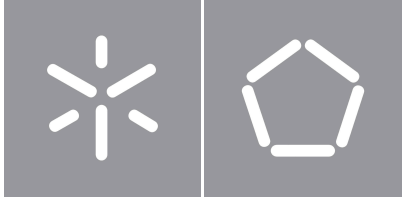
University of Minho
School of Engineering

Tomás Rodrigues Alves de Sousa

Classification and Clustering using Swap Test as distance metric

july 2023



University of Minho
School of Engineering

Tomás Rodrigues Alves de Sousa

Classification and Clustering using Swap Test as distance metric

Masters Dissertation
Integrated Master's in Physics Engineering

Dissertation supervised by
Luís Paulo Santos
André Sequeira

july 2023

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

I want to convey my heartfelt appreciation to the following people, without whom this thesis would not have been possible:

First and foremost, I want to thank my peers and girlfriend for their unwavering encouragement and support throughout the process. Your faith in me and willingness to listen to me talk about my studies kept me going even when things got tough.

I am also grateful to Luis Paulo Santos, my professor, for accepting my thesis proposal and offering invaluable advice and feedback throughout the writing process. Your knowledge and views were valuable in assisting me in refining my ideas and developing a more substantial thesis.

Lastly, I'd like to thank my mentor, Andre Sequeira. Your unwavering support and desire to go above and beyond to assist me in completing this thesis have been truly unique. I can't convey how much your advice and mentoring have meant to me.

Thank you so much for your motivation, support, and advice. This thesis is as much yours as mine, and I am thankful for everything you have done to assist me in finishing it.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, july 2023

Tomás Rodrigues Alves de Sousa

Abstract

This master's thesis explores the advantages of using a quantum-based distance metric in a Machine Learning (ML) algorithm. It compares the performance of such a hybrid algorithm with an entirely classical algorithm. Quantum Machine Learning (QML) has been growing in recent years. Some studies suggest that QML may even provide a polynomial speed-up for data categorization compared to traditional ML. However, analyzing the benefits is not straightforward, as QML algorithms often rely on abstract, oracle (black-box) models that frequently rely on Quantum Random Access Memory (QRAM). Furthermore, loading classical data onto quantum registers limits the applicability of QML, imposing a bottleneck. We used the Swap Test to measure the overlap between two quantum states to achieve our objective. Then we replaced the classical distance metric in a distance-based machine learning algorithm with the quantum-based distance metric. Our research showed that the Swap Test could be used as a distance metric in classical algorithms, despite the fact that the results obtained are not better than the classical metrics. In the final discussion, we present some ways that can improve the obtained results.

Keywords Quantum Machine Learning, Machine Learning, Classification, Clustering, Swap Test, distance metric

Resumo

Esta dissertação de mestrado visa explorar as potenciais vantagens de usar uma métrica de distância baseada em quantum num algoritmo de Machine Learning (ML) e comparar o desempenho de um algoritmo híbrido com o de um algoritmo totalmente clássico. Quantum Machine Learning (QML) tem vindo a crescer nos últimos anos. Alguns estudos sugerem que o QML pode avir a contribuir para uma aceleração polinomial na categorização de dados em comparação com o ML tradicional Landman [2021]. No entanto, analisar os benefícios não é direto, pois os algoritmos QML geralmente dependem de modelos abstratos baseados em oráculos (caixa preta) que frequentemente dependem de Quantum Random Access Memory (QRAM). A aplicabilidade pode ser limitada devido á dificuldade imposta em carregar dados clássicos para registos quânticos. Para atingir o nosso objetivo, usamos o Swap Test para medir a sobreposição entre dois estados quânticos e, em seguida, substituímos a métrica de distância clássica num algoritmo de Machine Learning por uma métrica de distância baseada em quantum. A nossa pesquisa mostrou que o Swap Test pode ser usado como métrica de distância, em algoritmos clássicos, apesar de os resultados obtidos não serem melhores que as métricas clássicas. Na discussão final, apresentamos algumas formas que podem melhorar os resultados obtidos.

Palavras-chave Quantum Machine Learning, Machine Learning, Classification, Clustering, Swap Test, metrica de distancia

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Computation is an integral part of our daily lives, with computers being the primary tool for processing large amounts of information. However, as we continue to push the boundaries of computational power, the limitations of traditional computing methods are becoming increasingly apparent. For example, traditional computing methods struggle to handle large amounts of data, like big data, and they also have difficulty with complex problems like simulating quantum systems (Lloyd [1996]), breaking encryption (Aaronson [2017]), and modeling protein folding (Baker [2003]). With the increasing demand for more efficient and powerful computing systems, researchers have been exploring new approaches to computation, and one promising area of research is quantum computing.

Quantum computing relies on the principles of quantum mechanics, which govern the behavior of matter and energy at the atomic and subatomic levels. Unlike classical computing, which uses bits to represent data, quantum computing uses quantum bits or qubits. Qubits can exist in multiple states simultaneously, known as superposition. They can also become entangled, which allows for the creation of complex interactions between multiple qubits. These properties of quantum mechanics allow quantum computers to perform certain types of computations much faster than classical computers. The promise of computational advantage resulted in rapid growth in recent years for this field of research, with many scientists and engineers working to develop new quantum algorithms to use on real-world quantum computers.

A field that could potentially gain from computational speed up is Machine Learning, and therefore Quantum Machine Learning (QML) has become an emerging field. QML seeks to leverage the power of quantum computing to improve the performance of machine learning algorithms. Some algorithms use quantum-based techniques such as quantum entanglement, quantum superposition, and quantum interference to perform operations on data in a way that is impossible with classical algorithms. However,

the analysis of the benefits of QML is not straightforward, as QML algorithms often rely on abstract, oracle (black-box) models that frequently rely on Quantum Random Access Memory (QRAM), and the applicability may suffer due to the bottleneck imposed by loading classical data onto quantum registers, as reported in studies such as (Landman [2021]).

This thesis explores how we can incorporate quantum algorithms, such as the Swap test, in classical machine learning by replacing some components of classical algorithms. These may not convey immediate improvements but showcases the possibility of incorporating quantum components in classical algorithms. We first provide an overview of the fundamental concepts of classical machine learning and then delve into how quantum computing can perform tasks in classical machine learning algorithms. Specifically, we introduce quantum algorithms for distance estimation and techniques for encoding classical information as quantum states. These can represent feature vectors in the quantum realm. Then we use the swap test to measure the overlap between two quantum states as a quantum-based distance metric, which we use to replace a classical distance metric on a classical ML algorithm. This approach, which we call the hybrid machine learning routine, is helpful in testing the performance of a quantum-based distance metric with real-world classical data.

Our research results determine if the swap test can be used as a distance metric in distance-based classical machine learning algorithms, enhancing the already available distance metrics.

1.2 Motivation

This thesis explores the potential advantages of using quantum computing in machine learning. The rapid advancements in quantum computing have led researchers to investigate new ways to apply quantum algorithms to real-world problems and machine learning is a rapidly growing field that has the potential to revolutionize many aspects of society.

One of the main motivations of this thesis is to investigate the use of the swap test as a quantum-based distance metric. The swap test is a quantum algorithm that can measure the overlap between quantum states. In this thesis, we will explore using the swap test as a quantum-based distance metric in a machine learning algorithm, specifically the K-means algorithm, and compare its performance to a classical distance metric. The choice of the K-means algorithm was motivated by its simplicity and the fact that it is a distance-measure-based algorithm. Furthermore, this thesis aims to showcase the implementation of a hybrid quantum machine learning algorithm. By combining quantum and classical computing, we aim to demonstrate the potential benefits of using quantum computing in machine learning

and provide a practical example of how quantum computing can incorporate machine learning problems. This thesis also aims to contribute to the growing body of research in quantum machine learning.

By understanding the foundations and principles of classical machine learning, we can better appreciate the potential benefits and limitations of using quantum computing in this field. Therefore, this thesis will also focus on evaluating the performance of the hybrid quantum machine learning algorithm using real-world data to compare with classical machine learning algorithms and demonstrate the use of the swap test as an alternative distance metric to current machine learning algorithms such as the K-means.

Chapter 2

Machine Learning

2.1 Introduction

Machine Learning (ML) is often associated with robots and artificial intelligence, but it is much more than that. It is a powerful tool with many applications, such as computer vision, pattern recognition, forecasting, anomaly detection, and text-to-speech.

A straightforward example of ML is a spam filter. A spam filter is a program that uses ML algorithms to identify and filter out unwanted emails based on specific criteria. For instance, a simple spam filter can classify an email as "spam" if it contains the words "winning" and "prize" and "not spam" otherwise. Of course, modern spam filters are much more complex than this, but the basic principle remains the same.

ML is changing our world, from business to self-driving cars. It pushes humanity to the next level, and the potential for future applications is limitless. Despite this, many people still have misconceptions about ML. This section provides a clear and straightforward introduction to the fundamentals of ML to help readers better understand this field.

What Is Machine Learning

Machine Learning (ML) is a branch of computer science that develops algorithms and models that can learn from data and make predictions or decisions without being explicitly programmed. It sits on the idea that a computer program can improve its performance on a given task, T, by learning from experience, E, as measured by a performance measure P, (Mitchell [1997]) .

One of the critical aspects of ML is the use of data. Most ML models require large amounts of data to perform a task, compare it with the actual result, and then improve over time based on error, which means that good quality data that is well-understood and cleaned is a must before being fed into the ML process.

To "make" a machine learn, we must understand various concepts such as supervised and unsupervised ML, models, cost functions, and data normalization and generalization. The choice of algorithms

depends on the type of data and task we are trying to automate.

A classic example of ML is a machine learning algorithm used to play a game of checkers. In this case, the experience E is playing many games of checkers, task T is playing checkers with many players, and the performance measure P is the probability that the algorithm will win in the game. Board games have been groundbreaking for ML because they can give precise details about the environment to the model so it can learn from experience. Many are associated with high intelligence, like checkers, chess, and Go.

It is important to note that ML is a subset of Artificial Intelligence (AI) and differs from AI. AI is a general concept that creates human-like critical thinking capabilities and reasoning skills for machines. At the same time, ML is specific and aims to create machines that can learn autonomously from data and make predictions or take decisions on a specific problem using pattern recognition and predictions in unknown situations.

Machine Learning has been a growing field in all industries and has changed how we approach complex problems. However, it has some limitations, one being the computational power and time required to perform some tasks.

2.2 Types of Machine Learning

Machine learning highly depends on data Tom Mitchell 2.1. This sub-field of computer science aims to build algorithms based on examples of some phenomenon (E), used to teach algorithms to learn from Data and interactions. In this field of computer science, it is possible to find four main categories of learning - supervised, semi-supervised, unsupervised, and reinforcement learning. We discuss each learning type in the following sections.

Supervised Learning

In supervised learning, we have a data set that maps $Xi \rightarrow Yi$. Here **X** is the collection of all the n-dimensional feature vectors. Each feature vector is a vector in which the dimension is $j = \{1, \ldots, N\}$ containing a value in each feature j that describes the data point. In the same way, **Y** is the collection of all the **labels** that represent the set. These labels can be real numbers data structures like vectors and graphs or simply a class (binary/multi). The primary aim of a supervised algorithm is to induce a relationship (linear or nonlinear) between the inputs and outputs and predict the output for yet unobserved input values. For example, we want to be able to predict the output Yi for any input Xi.

An excellent example of a supervised task is classifying a binary class set of emails as spam or not.

The classifier then labels unseen emails as either spam or not spam. A loss function quantifies the quality of the prediction made. The goal is to minimize this loss function $f(Y'(i), Y(i))$, where $Y'i$ represents the predicted class and Yi represents the actual value of the data point *i*.

There are some steps to consider to reach the end goal of supervised learning.

1. **Data analysis and model Selection:** We assume that the probability distribution belongs to a family of functions parameterized by some vector θ. It can also be called inductive bias. We could have two models representing the distribution functions of the data: generative and discriminative models. In a generative model, the algorithm learns how to generate new samples from the data distribution, while in discriminative models, the model learns how to identify newly generated data.
2. **Learning:** Given a training set, we optimize the learning parameters to minimize the loss function.
3. **Inference:** Here, the trained model is used to predict the output Yi'(Xi).

Supervised learning is vastly utilized in various subjects, such as object detection, semantic segmentation, panoptic segmentation, keypoint detection, regression problems, and language modeling.

Unsupervised Learning

In Unsupervised learning, we have a data set of unlabeled data, meaning that we only have, **X**, all of the n-dimensional feature vectors that compose the collection. The lack of **Y**, the labels is the main difference between supervised and unsupervised learning. Unsupervised learning algorithms aim to extract properties from all the feature vectors to transform the data into another feature vector or a value that solves other problems, such as clustering, returning an id that represents the group.

These types of models can be used for:

1. **Density estimation:** Using only the data set, we directly estimate the probability distribution $p(X)$.
2. **Clustering:** Separate the data into various clusters based on their similarities and differences. The notion of similarity depends on the case at hand, especially in the strategy used to define such similarity.
3. **Dimensionality Reduction:** The idea behind dimensionality reduction is to represent the feature vectors X_i that compose the data into a different vector space, either to visualize the data better or to combine features that bring high correlations. The output of the model is, as expected, a feature vector that has fewer features than the input X_i;

4. **Generation of new data points:** With the data set, we could generate new data points that would follow the probability distribution, allowing us to replicate or forecast the data.

Like in supervised learning, the same steps take place in unsupervised learning.

Semi-Supervised Learning

With this model, the idea is to use a data set with labeled and unlabeled data points. We shall that machine learning is highly dependent on the volume of data. More generally means a generalization of the model. So instead of adding more uncertainty to the problem, we add more information, meaning a better probability distribution.

Reinforcement learning

These machine learning models differ from the previous ones as a result of the shape of the data set, previously we had a data set mapped a $X_i \rightarrow Y_i$ in the supervised learning or an unlabeled data set composed only by feature vectors X. Now we have a model that stays in a particular environment and can perceive that environment's state as a vector of features, meaning that during the learning process, there is no correct answer, but rather a sub-optimal answer for those features at that time. So the end goal of the model is to produce a sequence of sub-optimal decisions that better suit the optimizer. Typically the optimization process resorts to the use of a reward system. The reward system is a feedback loop that tells the algorithm if the decisions helped or harmed the end goal. The environment with which the algorithm interacts has a Markov Decision Process. Some application areas for this algorithm are game playing, robotics, resource management, and logistics.

2.3 Key Machine Learning Concepts

This section aims to set the mark for some of the most important concepts regarding machine learning. Of course, it is difficult to mention all since Machine Learning is a vast field of study, and every Machine Leaning problem has its learning curve. Nonetheless, the concepts in this section aim to address the problem we want to tackle throughout the dissertation.

2.3.1 Data

Data is essential to every machine learning algorithm; the more data we can give an ML model, the better it can generalize. As mentioned above, if we are talking about supervised or unsupervised learning, ML

data is generally represented in feature vectors. There are different types of data:

- **Numerical data** refers to any data points with a numerical value. Numerical data might be continuous or discrete. While discrete data have a specific value, continuous data can have any value within a certain range. For instance, the number of doors on cars will be two, four, or five, while the house cost will be continuous and might range from 100K to 500K. Numerical data can be of the int or float data types.

- **Categorical data** When we talk about attributes, commonly, they are represented through categorical data, for instance, the color of a car or the year it was. It may also take the form of a number so long as the number designates a class. For instance, 1 may represent a gas vehicle, and 0 for a diesel vehicle. Some ML algorithms, such as those that derive from decision Trees (Günlük et al. [2021]), can handle categorical data without any data engineering.

- **Time Series** consists of numbers gathered at regular intervals over a predetermined period. Like in the stock market, knowing a stock's price is crucial ahead of time. The type of data has a temporal feature associated with it so that it is simple to track the timestamp of the data. There are specific ML algorithms that deal with time-series data, such as *ARIMA*, *SARIMA*, *ARIMAX*, and others (Parmezan et al. [2019]).

- **Text data** Literals are the only thing in text data. As the model is mathematical and requires data to inform numbers, the first step in managing data is to convert them into numbers. We might utilize functions as a collection of word formulations to do this. The field of ML associated with text data is normally NLP, and there is a particular way to deal with this type of data, often requiring neural-network algorithms, such as **BERT algoritms** and **LSTM**.

Model Parameters and Hyper-parameters

Hyper-parameterA hyper-parameter is an attribute of a learning algorithm that typically, but not always, has a numerical value. The setting of those values impacts the algorithm's performance. The method does not learn hyper-parameters from data. Instead, the programmer must set them before running the algorithm.

Parameter The variables known as parameters are what the learning process uses to define the model. The learning algorithm alters parameters directly in response to training data. Finding parameter values that make the improve performace is the aim of learning.

Model-Based Learning and Instance-Based Learning

Model-based techniques for supervised learning are the most common. SVM is one such algorithm. A model with parameters learned from the training data uses model-based learning algorithms using the training data. Algorithms for instance-based learning employ the entire dataset as the model. The k-Nearest Neighbors instance-based algorithm is often used in practice (KNN). For example, in classification, the kNN algorithm looks at the input example's immediate vicinity in the space of feature vectors to predict a label for it. It then outputs the label that it frequently encounters in this immediate vicinity.

The difference between Classification and Regression

Classification The challenge of classification is to label an unlabeled example automatically. One well-known application of classification is spam detection. The classification problem is resolved in machine learning by a classification learning algorithm that takes a set of labeled examples as inputs and creates a model that can take an unlabeled example as input and either directly output a label or output a number that the programmer can use to deduce the label. Probability is an illustration of such a number. A label in a classification problem belongs to one of a limited number of classes. Binary classification is used when there are only two classes (e.g., "approved"/"not approved," "spam"/"not spam," etc.). Multi-class classification is a classification problem involving three or more classes. While some learning algorithms are, by definition, binary classification methods, others naturally support more than two classes. Some techniques allow a binary classification learning algorithm to become a multi-class method. Clustering algorithms can also solve classification problems, since clustering algorithms group similar instances, these groups can be used as classes in a classification problem.

Regression Regression is a problem of predicting a real-valued label from an unlabeled sample (often referred to as a target). A well-known regression example is estimating home value based on attributes like square footage, the number of bedrooms, location, and other factors or predicting the stock market value based on previous events. A regression learning technique uses a set of labeled instances as inputs to create a model that can accept an unlabeled example as input and output a target to solve the regression problem.

2.3.2 Normalization

The process of normalization involves transforming a numerical feature's actual range of values into a standard range of values, often in the range [-1, 1] or [0, 1]. Let us say, for instance, that a specific feature's natural range is 100 to 1200. One can normalize the values into the range [0, 1] by taking 100 out of each feature value and dividing the result by 1100. The normalization formula is typically expressed as follows:

$$\overline{x} = \frac{x^i - min^i}{max^i - min^i} \tag{2.1}$$

where min^i and max^i are the minimum and maximum values of all the feature vectors in the dataset. Normalizing data is not always a must. But, it may result in better performances, especially when talking about regression algorithms, and we have a very small number and a very large number in the same set.

2.3.3 Standardization

The process of standardization (also known as z-score normalization) involves rescaling the feature values to give them the characteristics of a standard normal distribution with $\mu = 0$ and $\sigma = 1$, where μ is the mean (the average value of the feature, averaged over all examples in the dataset), and σ is the standard deviation from the mean. Z-scores are calculated as follows:

$$\hat{x} = \frac{x^i - \mu i}{\sigma i} \tag{2.2}$$

When to apply standardization and when to employ normalization needs to be clarified. Usually, testing both and determining which performs better for the task is usually the best procedure. For example, unsupervised learning algorithms benefit more from standardization than from normalization, and when having a high range of values since it shrinks the gap between those values. Standardization can also outperform normalization when the dataset has a distribution close to the normal distribution. In other cases, normalization is preferable.

2.3.4 Underfitting and Overfitting

When discussing machine learning results, it is inevitable to talk about **bias**. Bias is, by definition, a "deviation of the expected value of a statistical estimate from the quantity it estimates." In machine learning, a model has low bias if it predicts the labels on the training set well. On the other hand, if the model fails to predict under the training set, we say that it has a high bias or that the model **underfits**. Bias can also refer to a value that systematically causes the algorithm to have bad or excellent results. So the term "underfitting" refers to the situation in which a model cannot accurately forecast the labels of the data on which it was trained. Underfitting can be caused by several factors, the most prominent of which are as follows:

- The model is too simple for the data
- The features available are not representative enough

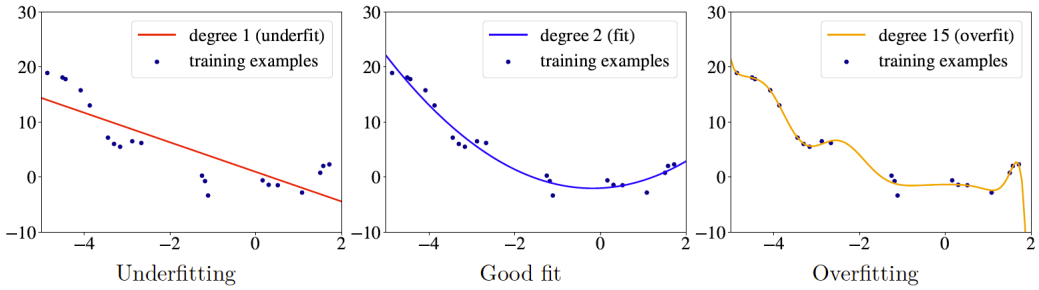


Figure 1: Examples of underfitting (linear model), good fit (quadratic model), and overfitting the data (polynomial of degree 15).

Another issue a model could have is one in which it **overfits** the data. The model with an overfitting problem makes accurate predictions based on the training data, but it performs poorly when applied to at least one of the two holdout data sets. Several factors can cause overfitting, the most prominent of which are the following:

- The model is far too complicated for the available information (for instance, an extremely tall decision tree or an extremely deep or wide neural network frequently overfits);
- There is an excessive number of features yet inadequate training examples.

2.3.5 Distance and Similarity measures

In machine learning, distance means a concrete way of describing what it means for data points of some space to be close to or far away from each other. How we calculate distance can change dramatically depending on our dimensional space. Nonetheless, the sense that distance represents the similarity between data points remains.

Let us have a **distance function** between two vectors a and b, $d(a, b)$ that defines the distance between both vectors. For d to be a function of distance, it must follow the following properties (Deza and Deza [2012]) :

- **Non-negativity:** The distance between x and y is always a value greater than or equal to zero.

$$d(a, b) \geq 0$$

- **Identity of indiscernible :** The distance between a and b is equal to zero if and only if a is equal to b.

$$d(a, b) = 0 \iff a = b$$

- **Symmetry:** The distance between a and b is equal to the distance between b and a.

$$d(a, b) = d(b, a)$$

- **Triangle inequality:** Considering the presence of a third point d, the distance between a and b is always less than or equal to the sum of the distance between a and d and the distance between b and d.

$$d(a, b) \leq d(a, d) + d(d, b)$$

When the distance is a range $[0, 1]$, the **similarity measure** $s(a, b)$ can be calculated as follows:

$$s(a, b) = 1 - d(a, b)$$

Now let us state a few well-known distance metrics below, where a_i represents the i^{th} value of the vector a, and the same goes for b.

- **Minkowski distance measure:** Also known as L_p *norm*, the Minkowski can be defined as:

$$Mink_{distance}(a,b) = \sqrt[p]{\sum_{i=1}^{n} |a_i - b_i|^p} \tag{2.3}$$

where p is a positive value. Depending on the value that is given to p, we can derive three more distances:

1. **Manhattan Distance** *When* $p = 1$. Also known as the l_1 *norm* or rectilinear distance, it represents the sum of the absolute differences between the opposite values in the vector.

$$Man_{distance}(a,b) = \sum_{i=1}^{n} |a_i - b_i| \tag{2.4}$$

2. **Euclidean Distance** *When* $p = 2$. Also known as the l_2 *norm* (Wen et al. [2016]), the Euclidean distance is an extension of the Pythagorean Theorem. Representing the root of the sum of the square of differences between the opposite (Liberti et al. [2012]).

$$Eucl_{distance}(a,b) = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2} \tag{2.5}$$

The Euclidean distance is a case called **Square Euclidean distance** where the sum of the squared differences is without taking the square root.

$$SquaredEucl_{distance} = \sum_{i=1}^{n} (a_i - b_i)^2 \tag{2.6}$$

- **Inner product distance measures** the inner product is some product of pairwise values from vector a and b. This gives a similarity measure between the two vectors. Many well-known metrics derive from this, such as:

1. **Jacard Distance:** This distance metric measures the dissimilarity between sample sets. It complements the Jaccard similarity coefficient (Cesare and Xiang [2012]).

$$Jac_{distance}(a,b) = \frac{\sum_{i=1}^{n} (a_i - b_i)^2}{\sum_{i=1}^{n} a_i^2 + \sum_{i=1}^{n} b_i^2 - \sum_{i=1}^{n} a_i b_i} \tag{2.7}$$

2. **Cosine Distance** Also called angular distance, the cosine distance derives from the cosine similarity that measures the angle between 2 vectors:

$$Cos_{distance}(a,b) = 1 - \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{2.8}$$

2.3.6 Dimensionality Reduction

Dimensionality reduction is a technique that aims to reduce the number of dimensions representing a feature vector, a well-known algorithm for this is the Principal Component Analysis (PCA). First introduced by *Karl Pearson in 1901* (F.R.S.) as an analog of the principal axis theorem, the **Principal Component Analysis (PCA)** is a technique vastly used in Machine Learning, especially when dealing with large datasets, that contain a high number of dimensions/features since it allows for a dramatic dimensionality reduction of the dataset.

Despite many formulations on calculating the **PCA**, the dimensionality reduction can be interpreted as a sequence of p unit vectors of a set of points located in a real coordinate space. The i^{th} vector in this sequence represents the direction of a line that best fits the data while remaining orthogonal to the first $i-1$ vectors in the sequence.

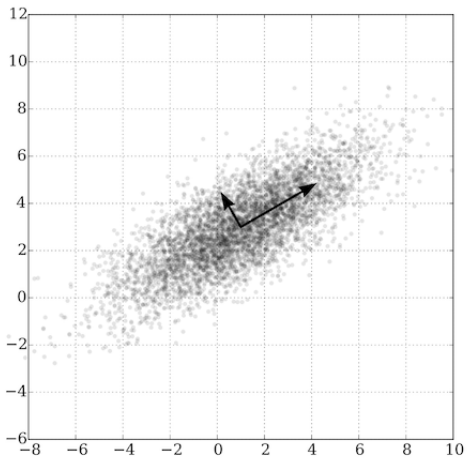


Figure 2: The scattered points represent a multivariable Gaussian distribution, and the vectors that are displayed are the eigenvectors of the covariance matrix, but they have been scaled by the square root of the respective eigenvalue and shifted to position their tails at the mean

In this context, "best-fitting line" refers to the line that achieves the goal of reducing the average squared perpendicular distance between the points and the line as much as possible. These directions make up an orthonormal basis, ensuring that the data's many dimensions are not linearly connected. The dimensionality reduction is achieved by using only the first few principal components to change the basis to the initial vector.

2.4 Clustering Algorithms

Clustering is grouping similar objects based on specific characteristics or features. It is a fundamental technique in machine learning, as it allows for discovering patterns and relationships within a dataset. One typical example of a dataset used for clustering is the digits dataset, which contains images of handwritten digits. Using a clustering algorithm, we can group similar images of digits based on their visual features.

Clustering algorithms can be applied for various end goals, such as data compression, anomaly detection, and feature selection. One of the most common applications of clustering algorithms is classification. Clustering algorithms can create classes for a classification problem by grouping similar instances. This approach is known as cluster-based classification, where clusters are the classes for the classification task. Clustering can also be used as a preprocessing step for classification, where the clusters are used to create a more informative feature representation of the data.

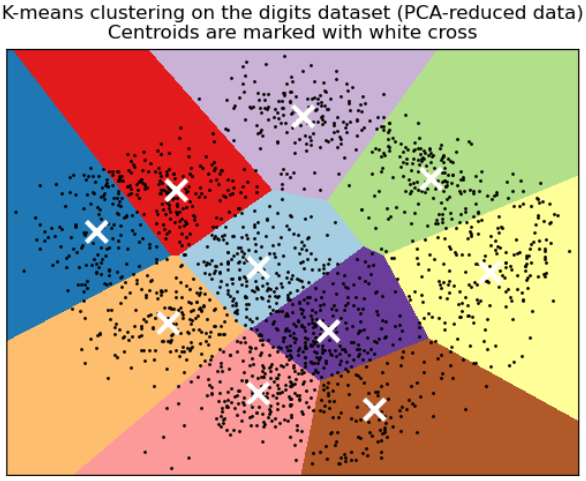


Figure 3: clustering on digits dataset. The withe cross represents the centroids

In this section, we will delve into the various clustering algorithms and their specific use cases and discuss how they can be applied to classification problems. Understanding the different clustering algorithms and their specific use cases can better identify and solve clustering problems in various fields. For example, in computer vision, clustering can group similar images for image classification tasks. Clustering can group similar documents for text classification tasks in natural language processing.

Figure 3 illustrates an example of clustering on the digits dataset. The white crosses represent the centroids of each cluster. Here the clustering algorithm groups similar images together based on their visual features. This can be useful for tasks such as image classification, where the clusters can be used

as classes for the classification task.

To begin grouping similar examples, one must first locate comparable examples. Then, determine the degree of similarity between two or more instances by merging the feature data of the examples into a metric referred to as a **similarity measure**. In the following sections, we will look into different types of clustering and a centroid-based algorithm, the KMeans algorithm.

2.4.1 Types of Clustering

As the size of the dataset increases, the running time of many clustering algorithms becomes infeasible. The computational complexity of these algorithms is often measured by the number of pairwise similarity comparisons that need to be made between the instances in the dataset. For example, a naive implementation of the k-means algorithm has a time complexity of $\mathcal{O}(n^2)$, where n is the number of instances in the dataset. This means that as the number of instances increases, the algorithm's running time increases at a rate of n^2.

In contrast, some algorithms, such as the k-means algorithm, have a linear time complexity of $\mathcal{O}(n)$, meaning they scale well with larger datasets.

In addition to the time complexity, it is also essential to consider the algorithm's suitability for the specific dataset characteristics. For example, some algorithms are better suited for datasets with many features, while others are better for datasets with many instances. The data distribution can also play a role in the choice of algorithm (Xu and Tian [2015]). For example, density-based algorithms are well suited for non-linearly distributed data, while centroid-based algorithms are better for datasets with linearly distributed data.

Centroid-based Clustering

Centroid-based Clustering groups the data in a way that is not hierarchical, as opposed to the hierarchical clustering method, which is explained further down. K-means is the centroid-based grouping technique that sees the most widespread application. Algorithms based on the centroid are practical, although they are susceptible to beginning conditions and outliers. The k-means clustering technique is the primary topic of study in this class because it is powerful and very straightforward.

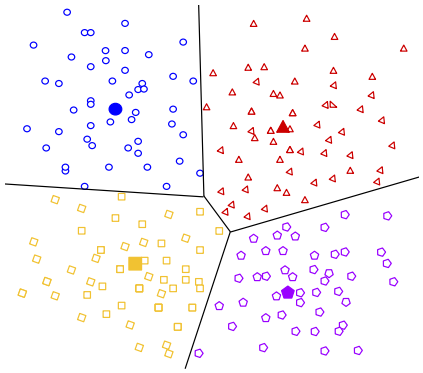


Figure 4: Centroid-based clustering

Density-based Clustering Clusters are formed by connecting regions with a high example density using density-based clustering. This paves the way for distributions of various shapes, provided that dense areas can be connected. These algorithms struggle when presented with data with a wide range of densities and a high dimension count. In addition, the architecture of these algorithms makes it so that they do not place outliers in any of the clusters.

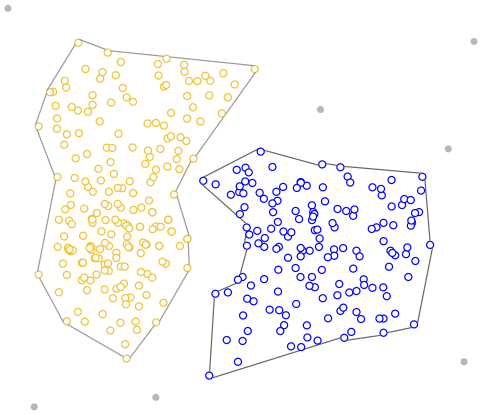


Figure 5: Density-based clustering

Distribution-based Clustering

This clustering approach assumes data is composed of distributions, such as Gaussian distributions. In 6, the distribution-based algorithm clusters data into three Gaussian distributions. As the distance from the distribution's center increases, the probability that a point belongs to the distribution decreases. The bands show a decrease in probability. When the type of distribution is unknown, one should use a different algorithm.

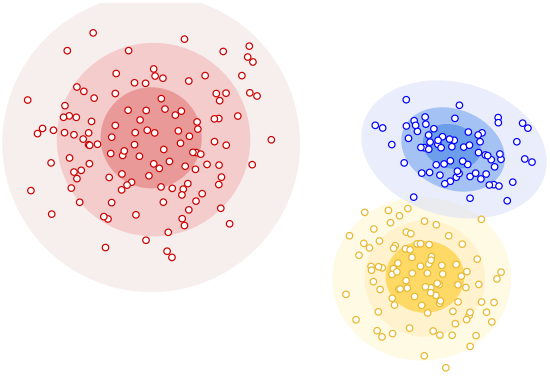


Figure 6: Distribution-based Clustering

Hierarchical Clustering Hierarchical clustering is a very different type of clustering that forms a tree of clusters (Fowlkes and Mallows [1983b]). It should come as no surprise that hierarchical clustering works particularly well with hierarchical data, such as taxonomies.

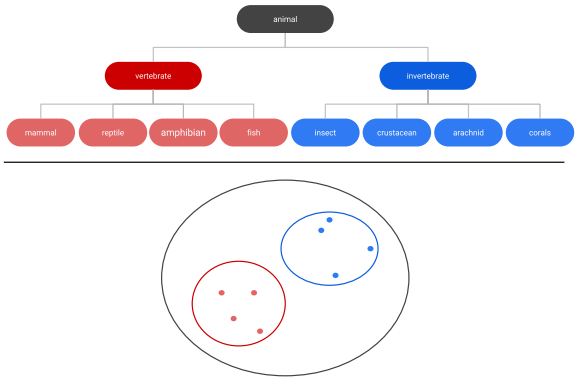


Figure 7: Hierarchical Clustering

Cluster-based Classification

Cluster-based classification is a method of classification that utilizes clustering algorithms as a preprocessing step. Cluster-based classification aims to group similar instances and assign class labels to the clusters. This approach is based on the assumption that instances within the same cluster are more likely to belong to the same class.

One of the main advantages of cluster-based classification is that it can handle datasets with many classes and instances where traditional classification methods may struggle. It can also be helpful when the class labels are unknown or difficult to obtain. By using clustering to similar group instances, we can uncover hidden patterns in the data and assign class labels based on these patterns.

There are several approaches to cluster-based classification, such as using the cluster centroid as the

class representative or a majority voting scheme. Another approach is to use a probability-based method, where the class label of an instance is determined by the class label of the cluster it belongs.

It is worth mentioning that using clustering algorithms for classification problems has its limitations. One of them is that the clusters created by the algorithm sometimes align with the actual class labels, which can lead to misclassification.

Another essential aspect of cluster-based classification is the choice of the clustering algorithm. Different algorithms have different strengths and weaknesses, and the choice of algorithm will depend on the specific characteristics of the dataset and the problem at hand.

For example, k-means is a popular choice for clustering continuous variables, while hierarchical clustering is well-suited for datasets with many instances. In addition, density-based clustering algorithms, such as DBSCAN, help detect clusters with non-convex shapes, which can be helpful in datasets with complex patterns.

2.4.2 KMeans Algorithm

Now that we have listed the types of clustering, let us dive deeper into a specific example of **Centroid-base clustering** the **KMeans Algoritm**.

This algorithm is used to cluster data by first attempting to divide samples into K groups with the same variance and then minimizing a criterion known as the within-cluster sum-of-squares or the inertia 2.4.2. This approach requires the cluster count to be supplied before it can be used. It works well with many samples and has been implemented in various application areas and disciplines of study.

The k-means algorithm creates K distinct clusters C from a given set of samples X_n, with each cluster being defined by the average value μ_j of the samples contained within it. The means are often referred to as the *centroids* of the cluster. Nevertheless, it is essential to remember that they are not, in most cases, points from the initial set X, even though they exist in the same space.

The algorithm aims to minimize the **inertia**

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (\|x_i - \mu_j\|^2) \tag{2.9}$$

The equation above can be recognized as a measure of how internally coherent the clusters are. Nonetheless, this can bring drawbacks, like the assumption that the clusters are convex and isotropic, which is only sometimes the case. Moreover, this leaves the algorithm vulnerable to elongated and irregularly shaped clusters.

Another problem from the inertia equation is the need for more normalization. Since the result of the equation is not bounded, it can lead to wrong results depending on the distance metric used. Running a **Principal Component Analysis (PCA)** before the k-means can mitigate this and speed up the computation.

Lloyd's algorithm is another name for the K-means clustering method. The algorithm can be broken down into three distinct phases, as stated in the pseudocode below:

Algorithm 1 KMeans

1: Selecting the initial K centroids
2: Define a threshold
3: **for** difference between the old and the new centroids $<$ threshold **do**
4: assigns each sample to its nearest centroid
5: creates new centroids by taking the mean value of all of the samples assigned to each previous centroid
6: **end for**

The selection of the initial centroids can be made in many different ways. The simplest method is to choose K random samples from the set X_n. However, this selection can significantly impact the results of the algorithm.

The K-means algorithm will always converge if given enough time, although this convergence can be to a local minimum. However, this is hugely reliant on the centroids being initialized correctly initially. Consequently, the computation is frequently carried out multiple times, each with a unique initialization of the centroids. The **k-means ++** (Arthur and Vassilvitskii [2007]) is a particular case of k-means with a different initialization strategy. This method sets the initial centroids (usually) far apart, leading to better results than those obtained from random initialization. There is a complete discussion on this matter in (Celebi et al. [2013]).

There are many variants of the k-means algorithm, but we need to get into more detail. We will mention some of this algorithm's advantages and disadvantages.

Advantages

- Relatively simple to implement;
- Scales to large data sets;
- Guarantees convergence;
- Can warm-start the positions of centroids;
- Easily adapts to new examples;
- Generalizes to clusters of different shapes and sizes, such as elliptical clusters;

Disadvantages

- Choosing the number of clusters k manually

  There are strategies, like the elbow curve in the "Loss vs. Clusters" plot, to help us choose the number of clusters.

- Dependency on the initial values;
- Clustering data of varying sizes and densities.

  When trying to cluster data where the clusters are of variable sizes and densities, k-means need help. This can be mitigated by generalizing K-means with the help of weights for each cluster;

- Clustering outliers
- Scaling with number of dimensions

  Outliers can pull centroids, or outliers may be given their own cluster so they are not disregarded. A good practice is to remove them before the algorithm execution.

- Scaling with number of dimensions

  A distance-based similarity measure tends to converge to a constant value between any two supplied examples as the number of dimensions it considers rises. Reduce the problem's dimensionality by applying **Principal Component Analysis (PCA)** to the feature data.

2.4.3 Evaluation metrics

Let us see how a classification algorithm evaluates compared to a clustering algorithm. We see up front that it is different since the typology of the problem is entirely different. On the one hand, we want to see if the classification was successful, and for that, we can use metrics such as recall or precision. (section 3 subsection metrics). On the other hand, we have a much more complex problem than counting the number of errors/wrong classifications.

When looking at a clustering evaluation problem, we should not consider the absolute values of the cluster labels but rather whether the clustering defines separations of the data that are similar according to some ground truth or some similarity metric. They assume that members of the same class are more similar than members of different classes.

With this in mind, let us take a look at some evaluation metrics used:

- **Rand Index** The Rand index is a function initially proposed by (Hubert and Arabie [1985]) that gauges the similarity of the two assignments, ignoring permutations, given knowledge of the ground truth class assignments and our clustering method assignments of the same samples. This method is bounded, meaning lower values indicate different labeling, as higher values indicate similar clustering compared to the ground truth. The score ranges from 0 to 1. This method can compare all kinds of clustering algorithms since it is not dependent on the structure of the clusters.

  Being C the ground truth and K the clustering, let us define a and b as

  - **a** the number of pairs of elements that are in the same set in C and the same set in K;
  - **b** the number of pairs of elements that are in different sets in C and different sets in K

  The Rand index can be formulated as follows:

$$RI = \frac{a+b}{C_2^{n_{samples}}} \tag{2.10}$$

  In the above equation, $C_2^{n_{samples}}$ is the total of possible pairs in the dataset.

- **Fowlkes-Mallows scores**

  This evaluation method is presented by (Fowlkes and Mallows [1983a]) in Bell Labs and can be used when a ground truth class assignment is available. This method is defined as the geometric mean of the pairwise precision and recall:

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}} \tag{2.11}$$

Where TP is the number of **True Positive** (i.e., number of labeled points that genuinely belong to the ground truth cluster), FP is the number of **False Positive** (i.e., number of points that belong to the same clusters in the actual labels and not in the predicted labels) and FN is the number of **False Negative** (i.e., the number of points that belongs in the same clusters in the predicted labels and not in the actual labels).

This evaluation method gives a single score that is upper bounded by 1 and lower bonded by 0 It also is independent of the structure of the cluster.

- **Contingency Matrix**

  Similar to the Confusion Matrix for classification, there is the square matrix in which the order of rows and columns corresponds to a list of classes.

  The contingency matrix reports every true/predicted cluster pair's intersection cardinality. When the samples are independent and have an identical distribution, the contingency matrix gives enough data for all clustering metrics, eliminating the need to consider instances that do not cluster in all cases.

  Let us imagine that we have a 2-cluster problem, and the output of the Contingency Matrix is

$$CT = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

  At a glance, we can already tell there are two actual clusters, one for each row. The first row of the matrix indicates that there are 3 data points in the first cluster. Of those, two are predicted to be in cluster 0, one in a cluster on, and none in cluster 2. The same goes for the second row.

  This evaluation method allows for examining the spread of each actual cluster across predicted clusters and vice versa. Nonetheless, analyzing when it starts to increase the number of clusters can become a challenge.

- **Silhouette Coefficient**

  If the labels for the ground truth are unknown, the evaluation must be done with the model it-self. One such evaluation is the Silhouette Coefficient, where a model with better-defined clusters receives a higher Silhouette Coefficient score.'

  Each sample's Silhouette Coefficient is defined, and it consists of two scores:

  - **a:** The mean distance between a sample and all other points in the same class.

- **b:** The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient s for a single sample can be calculated by:

$$s = \frac{b - a}{\max(a, b)} \tag{2.12}$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample. This approach is highly utilized, especially when working with lots of data and clusters, since it can give a sense of how the data is distributed, scores near -1 indicate incorrect clustering, and scores averaging $+1$ indicate dense clustering or separated clusters. If the score is close to 0, we have overlapping clusters. (Rousseeuw [1987])

- **Homogeneity, completeness and V-measure**

  With the aid of conditional entropy analysis, understandable metrics may be defined based on the samples' ground truth class assignments.

  (Rosenberg and Hirschberg [2007]) proposed two ideal goals for cluster assignments:

  1. **Maximum Intra-cluster Similarity**: Aim to maximize similarity within each cluster, ensuring samples in the same cluster share similar characteristics.
  2. **Minimum Inter-cluster Similarity**: Aim to minimize similarity between different clusters, ensuring distinct characteristics for samples in different clusters.

  These goals can be assessed using conditional entropy analysis, which measures the uncertainty in class assignments given cluster assignments. By defining understandable metrics based on conditional entropy, we can evaluate the quality of cluster assignments in terms of meeting these goals.

2.5 Summary

In this chapter, we have discussed the topic of clustering and the various types of clustering algorithms available. We have also provided a detailed explanation of the k-means algorithm and its implementation and examples of its use in real-world applications. Additionally, we have discussed various evaluation metrics, such as the Fowlkes-Mallows score, Rand index, and silhouette score, that can evaluate the performance of clustering algorithms.

In the following chapters, we will delve further into clustering and Quantum Machine Learning (QML) by exploring a hybrid approach to the k-means algorithm. Specifically, we will use the swap test algorithm to replace traditional distance metrics such as Euclidean distance and Cosine distance with the overlap measure calculated on a quantum simulator.

We will then compare the results obtained using the traditional k-means algorithm with those obtained using the hybrid approach. We will also use the evaluation metrics discussed in this chapter, such as the Rand index and Fowlkes-Mallows score, to evaluate the performance of the hybrid approach.

Overall, this chapter has provided a solid foundation for the understanding of clustering algorithms and how they can be used to extract valuable insights from data. The further chapters will provide more in-depth information on QML and lay the foundation for the hybrid algorithm.

Chapter 3

Quantum Machine Learning

In this chapter, we will talk about Quantum Machine Learning (QML) has gained significant attention in the last decade due to its potential to leverage the power and speedups of quantum computation for solving complex Machine Learning problems. This interdisciplinary field has attracted interest from scholars and institutions who see QML as an exciting avenue for combining two significant research areas. First, the rapid growth of research in QML can result from the availability of quantum hardware and the development of quantum algorithms that outperform their classical counterparts in some specific tasks (Biamonte et al. [2016]).

According to (Wiebe [2020]), QML has two fundamental definitions. The first is that *"QML involves using a quantum device to solve a machine learning task with greater speed or accuracy than its classical analog would allow."*, this comes from algorithms proposed that promise polynomial speedups, such as (fir) and (Wiebe et al. [2014]) and many others, and focuses on the use of quantum machines to solve the same tasks as classical machines. The second definition is that it *"involves using a quantum device to classify or extract features from quantum states."*, this statement comes from the possibility of gaining speedups in problems where the data itself is hard to generate or store classically.

Despite the differences in definition, QML has flourished in both directions, either by speeding up linear algebra calculations essential to many Machine Learning algorithms or by providing new and robust models, depicted in figure 8. According to (Seaar Al-Dabooni [2019]), different approaches to QML have led to faster computation and the development of new machine-learning models that take advantage of quantum properties. The authors provide examples of how quantum computing has been applied to linear algebra and other fundamental components of machine learning algorithms and how it has enabled the development of new models.

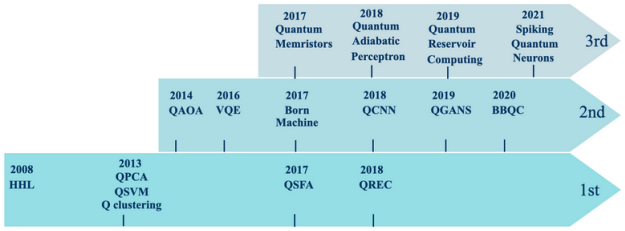


Figure 8: QML development timeline

To help make sense of the different methods and approaches in QML, (Aïmeur et al. [2006]) introduced a typology that identifies four distinct methods for combining quantum computing and machine learning, each of which is dependent on whether or not one assumes the data to have been produced by a quantum (Q) or classical (C) system, as well as whether or not the information processing device in question is quantum (Q) or classical (C). (see Figure 9).

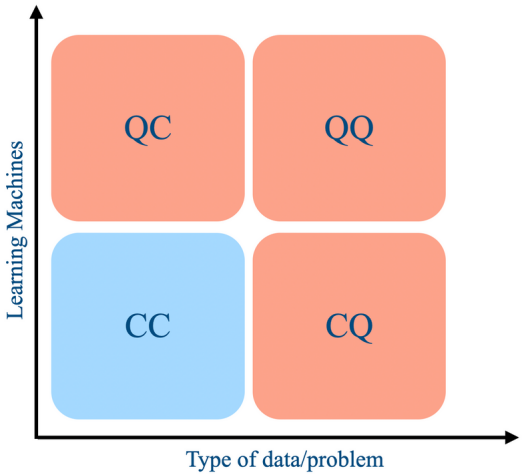


Figure 9: Four Approaches to Combine Quantum Computing and Machine Learning

The **CC** approach is the complete classical way, where classical data is processed by a classical machine, nonetheless the. Classical implementations derive from quantum ideas/counterparts, the so-called "quantum-inspired" algorithms. The **CQ** graph section refers to classical machine learning operating on quantum data. Orthogonal to this, we have the **QC** that uses quantum algorithms on classical feed data. Finally, we have the **QQ** that looks at quantum data processed by quantum computers.

As a result of this evolution, Quantum Machine Learning (QML) has established itself as an active sub-discipline of quantum computing research. We will look at some challenges and possible solutions to **QC** regime.

3.1 Quantum Computing Key Concepts

The field of quantum computing has seen remarkable advancements in recent years, with the development of practical algorithms and the construction of experimental quantum computers.

In this section, we will be providing a brief introduction to the fundamental concepts of quantum computing. To serve as a foundation for the upcoming chapters where we will discuss the application of quantum computing in machine learning.

Quantum computing is a field that utilizes the principles of quantum mechanics. The fundamental component is quantum bits, or qubits, to store and process information instead of classical bits. A qubit achieves a linear combination of two states by utilizing the quantum mechanical phenomenon of superposition. For example, a traditional binary bit can only represent a single binary value, such as 0 or 1, and can only be in one of two states. Conversely, a qubit can represent a 0, a 1, or any percentage of 0 and 1 in a superposition of both states, with a specific likelihood of being a 0. The standard notation used for these states in quantum mechanics is the *dirac notation*, '$|\rangle$.'

A fundamental component in quantum computing is quantum gates or operators, which manipulate qubits. These gates are the quantum equivalent of classical logic gates, including the NOT gate, the Hadamard gate, Pauli matrices, and the identity operator.

This section will also provide a brief overview of the main principles of quantum mechanics that form the basis for quantum computing. These include superposition, entanglement, and interference. By understanding these key concepts, we can better understand the underlying principles of quantum computing and how it differs from classical computing.

This section is not an exhaustive explanation of quantum mechanics or quantum computing but rather a brief reminder of the basic concepts to understand the chapters' ahead. To gain a deeper understanding of the subject, we recommend visiting (Nielsen and Chuang [2010]).

3.1.1 Fundamental Principles of Quantum Mechanics

The principles of quantum mechanics form the basis for quantum computing and are fundamental to understanding the behavior of quantum systems. Three of the main principles are superposition, entanglement, and interference.

Superposition is a fundamental concept in quantum mechanics that allows a quantum system to exist in multiple states simultaneously. It is typically represented using the notation $|\psi\rangle = a_1 |0\rangle + a_2 |1\rangle$, where $|\psi\rangle$ is the state of the system, and $|0\rangle$ and $|1\rangle$ are the basis states when measuring in

the computational basis, also known as the *z* basis. The complex coefficients a_1 and a_2 describe the probability amplitudes of the quantum system in the states $|0\rangle$ and $|1\rangle$, respectively. This mathematical representation of the physical state of a quantum system is known as a *quantum state*.

Entanglement is a phenomenon in which two or more quantum systems become correlated so that the state of one system is dependent on the state of the other. Entanglement is often represented using the notation $|\psi\rangle = a\,|00\rangle + b\,|11\rangle$, where the first and second qubits are entangled. Entanglement is, therefore crucial principle in quantum computing as it allows for the creation of quantum states that cannot be represented classically and is the basis for many quantum algorithms, such as quantum teleportation and quantum key distribution.

Interference is a fundamental principle of quantum mechanics that involves adding or canceling the amplitudes of quantum states. This phenomenon can occur constructively or destructively and is determined by the relative phase of the complex coefficients. Interference is the foundation of various quantum algorithms, including the quantum Fourier transform and the quantum phase estimation algorithm, as it plays a crucial role in determining the probabilities of the outcomes of a quantum system.

These principles of quantum mechanics are the foundation for quantum computing and are essential to understanding the behavior of quantum systems. For a more detailed explanation of these principles and the mathematical formalism of quantum mechanics, we recommend reading (Nielsen and Chuang [2010]) and (Sakurai and Napolitano [2017]). Quantum circuits are a universal language for describing complex quantum computations. Just like programming languages allow the user to code without manually manipulating the $0's$ and $1's$ that make a classical computation, quantum circuits are the first analog to a first and primitive programming language for quantum computing. A well-known language that provides this type of circuit is qiskit, a Python SDK provided by IBM. It is possible to find more information about this package in qiskit.org.

3.1.2 Quantum bit

A single single quantum bit $|\psi\rangle$ and can be represented in following form:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.1}$$

In the equation above the $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are known as the *computational basis states*, forming an orthogonal basis in the complex vector space.

Where α and β are complex numbers where $|\alpha|^2 + |\beta|^2 = 1$.

A straightforward deduction is the complex conjugate of $|\psi\rangle$:

$$|\psi\rangle^{\dagger} = \langle\psi| = \alpha^* \langle 0| + \beta^* \langle 1| \tag{3.2}$$

In quantum mechanics, measurement is the way to obtain information about a quantum property, this is called an *observable*, and it is represented by a quantum *operator* that acts on quantum states. When a measurement is performed on 3.1, the state will collapse to one of the *eigenstates* of the corresponding observable with some probability. In quantum circuits, the eigenstates for measurement are typically the computational basis $|0\rangle$ and $|1\rangle$ and the probability that $|\psi\rangle$ collapses to state $|0\rangle$ is given by $|\alpha|^2$.

3.1.3 Quantum Gates

Here we will show some of the quantum gates used in this paper. There are two types of quantum gates: single-qubit and multi-qubit. In both cases, something to bear is that qubits in quantum circuits are usually initialized in the state $|0\rangle$.

Pauli gates X, Y and Z are single qubit gates represented by:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{3.3}$$

Pauli gates define *rotation gates* that can then be used to create more complex rotations like $Rot(\theta) \equiv e^{i\frac{\theta}{2}\sigma}$, where σ is the generator that defines the direction for the rotation, i.e., $\sigma = \sigma_x, \sigma_y, \sigma_z$. We can see in the Bloch Sphere (fig. 10) a state $|\psi\rangle$ as a result of combining two rotations, on the initial state $|0\rangle$, a $Rot(\theta)$ on the y axis and $Rot(\phi)$ on the x axis:

$$|\psi\rangle = e^{i\lambda}(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\alpha}\sin\left(\frac{\theta}{2}\right)|1\rangle) \tag{3.4}$$

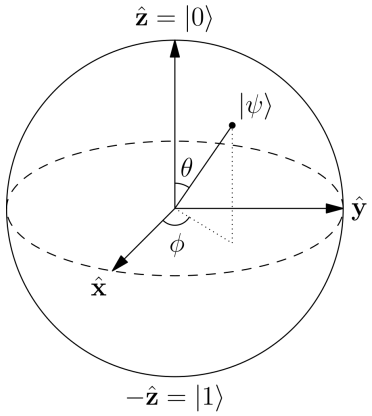


Figure 10: Qubit $|\psi\rangle$ represented in the Bloch Sphere.

Applying these rotations to a state makes it rotate around the x, y, or z axis by an angle of θ:

$$R_X(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad R_Y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \tag{3.5}$$

Hadamard is a single qubit gate represented by H:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.6}$$

Used to create superpositions in qubits. This gate has the following result when applied to a basis state.

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \end{aligned} \tag{3.7}$$

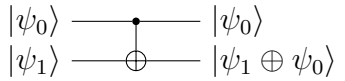
CNOT or controlled-NOT gate is a 2-qubit gate. It can be represented by:

$$CNOT \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.8}$$

The example above is of a controlled gate, where the state of the qubit is changed based on the value of another qubit, called the control qubit:

$$\begin{aligned} CNOT|00\rangle &= |00\rangle, \quad CNOT|01\rangle = |01\rangle \\ CNOT|10\rangle &= |11\rangle, \quad CNOT|11\rangle = |10\rangle \end{aligned} \tag{3.9}$$

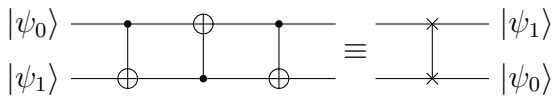
The representation of this multi-qubit gate in a circuit is given by:



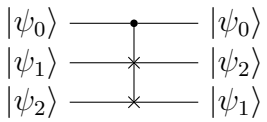
SWAP gate is another multi-qubit gate that swaps the state of two qubits:

$$SWAP \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

It has the following circuit representation:



Fredkin or controlled SWAP (C-SWAP) gate is a swap gate, but with a control qubit it can be represented by:



3.2 Data Encoding

When tackling a QML problem is natural that the first question we ask ourselves is how to load data into a quantum computer. To load classical data into a quantum computer, we must first implement a process called *state preparation*. This process is responsible for the data encoding strategy. Different strategies can have a significant impact on the runtime performance of the quantum algorithm, especially in the number of qubits used.

It is important to note that the best encoding strategy will depend on the specific problem and the nature of the data used.

Different strategies can significantly impact the quantum algorithm's runtime performance, particularly in the number of qubits required. Here, we will provide a brief overview of three commonly used data encoding techniques: *Basis Encoding*, *Angle Encoding*, and *Amplitude Encoding* (Lloyd et al. [2020]). Let us consider a dataset of M entry points each with N features, $D = \{x^1, \ldots, x^m, \ldots, x^M\}$ where $x^m = [x_1^m, \ldots, x_N^m]$ is a N-dimensional vector for $m = 1, \ldots, M$

- *Basis Encoding* involves a process known as basis embedding. Each input of a qubit system is linked to a computational basis state. Since this is the case, traditional data must be binary strings. The embedded quantum state may be thought of as the bit-wise translation of a binary string into the states of the quantum subsystems that correspond to those states. For instance, the 4-qubit quantum state $|1001\rangle$ corresponds to the value $x = 1001$ in decimal. As a result, one quantum subsystem is equivalent to one bit of classical information.

  Let us have a look at the traditional dataset that was stated earlier. To do basis encoding, each example has to be a string of binary digits with N bits; $x^m = (b_1, \ldots, b_N)$ where $b_i \in \{0, 1\}$ for $I = 1, \ldots, N$. Assuming that all characteristics are stored with unit binary precision (one bit), it

is possible to directly transfer each input example $x^{(m)}$ to the quantum state $\left|x^{(m)}\right\rangle$. It is possible to describe an entire dataset as a superposition of several computational basis states.

$$|\psi\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} \left|x^{(m)}\right\rangle \tag{3.11}$$

This equation means that for N bits, there are 2^N possible basis states.

- *Angle Encoding*

  Angle encoding is a strategy that uses the angle of a vector in high-dimensional space to encode classical data into a quantum system. Encoding a datapoint with N features into N qubits, using one qubit per feature. To encode a real-world data set X, the data should first be normalized between 0 and 2π, resulting in $\tilde{X}$, to ensure that the data can be accurately represented using angle encoding. To better distinguishability between the points, the encoding could be chosen between 0 and π, ensuring that the furthest points are opposite when we are encoding using only a single rotation.

  The $\tilde{X}$ vectors can be represented by:

$$|\psi_x\rangle = \bigotimes_{j=1}^{n} e^{-i\tilde{x}_j \sigma_x} |0^n\rangle \tag{3.12}$$

  where $\sigma \in \{I, \sigma_x, \sigma_y, \sigma_z\}$. The main advantage of this method is the ability to preserve the high-dimensional structure of the data, as the angles of the vectors are preserved in the encoding process. However, it requires more qubits to encode the same data than amplitude encoding. Additionally, the state preparation of the angle-encoded vectors can be costly in terms of time complexity, especially for high-dimensional data sets. It is $\Theta(1)$ per feature if we consider an encoding with a depth of 1, meaning using a single rotation.

  Furthermore, angle encoding requires an angle generator $\sigma \in I, \sigma_x, \sigma_y, \sigma_z$, which defines the direction of the rotation used in the encoding process. The angle generator's choice can affect the encoding's performance, as different generators may result in different distributions of the encoded data.

- *Amplitude Encoding* Amplitude encoding uses only $\log_2 N$ qubits for encoding a N dimensional data point x. Nonetheless, it brings some limitations to the table due to the association between real classical information vectors and quantum amplitudes. To encode the classical vector into a

quantum state, we need to ensure that it is normalized, i.e., for a data set X so that $\sum_k |x_k|^2 = 1$, resulting in $\tilde{X}$. The superposition over states that encodes the features into its amplitudes can be represented as:

$$|\psi_x\rangle = \sum_{j=1}^{2^n} x_j |j\rangle \tag{3.13}$$

The need for normalization imposes the first limitation to this method since the quantum states represent the data in one fewer dimension or with one fewer degree of freedom. A classical two-dimensional vector $(x_1, x_2)^T$ can only be associated with an amplitude vector $(0, 1)^T$ of a qubit that fulfills the equation $|\alpha_0|^2 + |\alpha_1|^2 = 1$, meaning that a two-dimensional vector is represented in a unit circle, which is a manifold with one dimension that exists in a space with two dimensions. When three-dimensional vectors are stored in three amplitudes of a two-qubit quantum system, the three-dimensional space is reduced to the surface of a sphere. This process continues until the space is reduced to a two-dimensional plane. A workaround to this restriction can be adding one element to the classical vector with one value and then normalizing the resulting vector, allowing the new element to carry on information about the normalization constant. Another limitation of this encoding method is the time cost of the state preparation of the dense amplitude vectors.

The following table presented by (Schuld and Petruccione [2021]) provides a comparison of the run-times for the three data-encoding algorithms described, where M represents the number of inputs or data points, N represents the number of features for each data point, and τ represents the number of bits in a binary representation of the data point. The basis, amplitude, and angle encoding methods provide a recipe for encoding a single input, which can be applied in superposition to encode an entire dataset.

Encoding	#qubits	Runtime	Input type
Basis	$N\tau$	$O(N\tau)$	Single input (binary)
Amplitude	$\log N$	$O(N)/O(\log(N))^a$	Single input
Angle	N	$O(N)$	Single input

Table 1: Qubit Complexity analysis

[a]Note that the complexity notation for amplitude encoding is given as $O(N)$ or $O(\log(N))$ based on the specific application.

These are only a tiny sample of the possible data encoding strategies. To learn more, see (Schuld and Petruccione [2021]) Chapters 3 and 4.

Data encoding is one of the essential steps in the road map for developing a quantum machine learning algorithm and therefore is crucial for us to mention some problems that arrive from creating real-world algorithms. Despite many notorious encoding strategies and ingenious solutions, one of the biggest problems is the need for qubits. This problem puts a bottleneck in the development of QML algorithms since, in real-world ML scenarios, data has many entry points and features, and therefore is difficult to compare algorithms and showcase a quantum advantage. Another problem comes from the lifespan of quantum data. After the encoding process, we need to use it, but there is no place to store it. There are some articles mentioning *q-rams* like (Landman [2021]) and (Schuld et al. [2018]), but there is no actual proof of concept at the time. The encoding process must be repeated multiple times since quantum computing is probabilistic. The model needs to be run multiple times to ensure president results, increasing the computation time even further.

3.3 Measuring the overlap between quantum states

In classical machine learning, the similarity between two vectors is crucial in various algorithms, as discussed in Section 2.3.5. When we shift to the quantum realm, comparing two states changes, but it is still possible to have a notion of similarity. Suppose we recall the similarity measures based on the inner product. In that case, we can have a quantum analog with quantum states $\langle a|b\rangle$ or the absolute square value $|\langle a|b\rangle|^2$ called the *overlap*. A family of tiny quantum circuits may accomplish this goal by using interference between several branches of a superposition. The swap test, which provides the absolute value of the inner product of the quantum states of two distinct quantum systems, is the interference procedure that is the most well-known and widely used today.

3.3.1 SWAP Test

First mentioned in (Barenco et al. [1996]) and then rediscovered by (De Wolf [2019]), the swap test has since found applications in various quantum machine learning algorithms. This method for quantifying the similarity or overlap between two quantum states is typically represented by density matrices ρ_1 and ρ_2. The basic idea behind the swap test is to extract the absolute square of the inner product of two-qubit registers, $|a\rangle$ and $|b\rangle$, from the probability of measuring an ancilla qubit in a particular state (Schuld and Petruccione [2021]). The controlled swap operation is represented mathematically by the unitary operator

$SWAP_{12}$, which acts on the two-qubit system and is defined as:

$$SWAP_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

To perform the Swap Test method, we start with the quantum state $|0\rangle |a\rangle |b\rangle$, where the ancilla qubit is in state $|0\rangle$, and $|a\rangle$ and $|b\rangle$ are two-qubit registers. Then, applying a Hadamard gate to the ancilla qubit leads to the state.

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |a\rangle |b\rangle \tag{3.15}$$

Next, we apply a swap operator on the two registers, conditioned on the ancilla qubit in state $|1\rangle$. This operation swaps the states $|a\rangle |b\rangle$ to $|b\rangle |a\rangle$ in the corresponding branch, leading to the state

$$\frac{1}{\sqrt{2}}(|0\rangle |a\rangle |b\rangle + |1\rangle |b\rangle |a\rangle). \tag{3.16}$$

Applying another Hadamard gate to the ancilla qubit results in the state

$$\frac{1}{2}(|0\rangle \otimes (|a\rangle |b\rangle + |b\rangle |a\rangle) + |1\rangle \otimes (|a\rangle |b\rangle - |b\rangle |a\rangle)). \tag{3.17}$$

This prepares two branches of a superposition, one containing a sum between the "unswapped" and "swapped" states of the two registers and the other containing their difference. The probability of measuring the ancilla qubit in state $|0\rangle$, denoted as $p_0 = |\langle 0| | \otimes \mathbb{1} |\psi\rangle|^2$, where $\langle 0|$ acts on the ancilla and $\mathbb{1}$ is the identity operator acting on the remainder of the qubits, is given by

$$p_0 = \frac{1}{2} + \frac{1}{2}|\langle a|b\rangle|^2, \tag{3.18}$$

and reveals the overlap of the two states via

$$|\langle a|b\rangle|^2 = 2p_0 - 1. \tag{3.19}$$

Note that in the more general case where the two input states are mixed a and b, the same routine can be applied, and the success probability of the post-selective measurement is given by $p_0 = \frac{1}{2} - \frac{1}{2}tr\{ab\}$, where ab is the matrix product of the two mixed states.

In quantum machine learning, the Swap Test is often used in contexts where $|a\rangle$ and $|b\rangle$ represent normalized and real-valued N-dimensional data vectors $a = (a_1, \ldots, a_N)$ and $b = (b_1, \ldots, b_N)$,

respectively. In this case, the overlap $|\langle a|b\rangle|^2$ equals the cosine similarity between the two vectors, a commonly used metric in machine learning tasks such as classification and clustering. The Swap Test can thus be used as a subroutine to calculate the cosine similarity between two vectors in a quantum machine learning algorithm, such as quantum support vector machines (Lloyd et al. [2020]).

To replicate this method in a quantum computer, we can use a pseudo-code such as the one in Algorithm 2:

Algorithm 2 SWAP TEST

Inputs: M copies each of the n qubits quantum states $|\psi\rangle$ and $|\phi\rangle$, and N_0 is the number of shots.

1: **for** j ranging from 1 to M state **do**
2: initialize an ancilla in state $|0\rangle$ and apply a Hadamard gate to the ancilla qubit
3: **for** i ranging from 1 to n qubit of state j **do**
4: apply **CSWAP** to $|\psi_i\rangle$ and $|\phi_i\rangle$ with the control in the ancilla qubit, where i represent the i^th qubit of state j.
5: apply a Hadamard gate to the ancilla qubit
6: measure the ancilla in the Z basis and record the measurement as 0 or 1
7: **end for**
8: compute $2p_0 - 1$, where the probability $p_0 = \frac{N_0}{M}$, where N_0 is the number of times $|0\rangle$ was measured.
9: **end for**

return Overlap

The algorithm above is going to encode the Swap test routine that can be represented in a quantum circuit as: **Fredkin** or controlled SWAP (C-SWAP) gate is a swap gate, but with a control qubit it can be represented by:

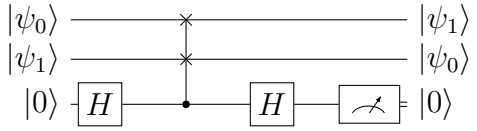


Figure 11: Swap Test for two qubits

3.3.2 Inversion Test

Another algorithm for calculating the overlap between two quantum states, $|a\rangle$ and $|b\rangle$, is the inversion test, which Nielsen and Chuang first introduced in (Nielsen and Chuang [2010]). This routine can be used to reduce the number of qubits required to a bare minimum when computing state overlaps of the form $|\langle a|b\rangle|^2$.

Suppose we have a quantum circuit A that prepares $|a\rangle = A|0\rangle$ and another circuit B that prepares $|b\rangle = B|0\rangle$. The idea is to run the circuit $B^\dagger A|0\rangle$ and measure the state of each qubit. The probability of observing the quantum computer back in the initial state $|0\rangle$ is given by the Born rule as $|\langle 0|(B^\dagger A|0\rangle)|^2$, which is just the desired overlap.

Mathematically, we can see this by writing out the expectation value of the projective measurement $M = |0\rangle\langle 0|$, which is given by

$$\langle 0|A^\dagger B(|0\rangle\langle 0|)B^\dagger A|0\rangle = \langle 0|A^\dagger B|0\rangle\langle 0|B^\dagger A|0\rangle \qquad = |\langle 0|B^\dagger A|0\rangle|^2 = |\langle a|b\rangle|^2.$$

To implement the inversion test, the quantum computer must be able to implement the inverse of one of the state preparation circuits. The inverse of a quantum circuit $U = U_L...U_1$ is given by $U_1^\dagger...U_L^\dagger$. Thus, the quantum computer must be able to implement the complex-conjugate transpose version of every gate in the circuit.

For many data-encoding strategies, this is straightforward because many of the fundamental quantum gates are their inverse or can be inverted by feeding the parameter times a factor of -1. For example, a Pauli rotation fulfills $R_z^\dagger = R_{-z}$. However, for near-term quantum computers, it may not be possible to invert a routine exactly, in which case one can revert to the previous method.

Overall, estimating the inner product or overlap of two n-qubit quantum states via measurements requires, at most, $2n + 1$ qubits and a number of gates that is linear in the number of qubits.

The inversion test can be represented in a quantum circuit as:

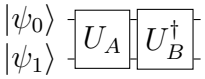


Figure 12: Inversion Test for two quibits

3.3.3 Functions after the Overlap

As seen, the overlap provides a way to measure the similarity between quantum states, which can be used in various quantum machine learning tasks, such as quantum classification and quantum clustering. This process is called a feature map that maps classical data to high-dimensional quantum states. The similarity between the quantum states is proportional to the similarity between the original data points. This subsection explores the concept of data encoding as a feature map and the results generated by the quantum overlap, focusing on the distance between the encoded data points and the functions that the encoding represents.

Data Encoding and the Overlap

The choice of data encoding strategy plays a crucial role in determining the properties of the resulting functions. As we showed, three common data encoding strategies are basis encoding, amplitude encoding, and angle encoding (Schuld and Petruccione [2021]). Table 2 summarizes these encoding strategies and their corresponding functions after the overlap.

Encoding	Kernel $\kappa(x, x')$
Basis encoding	$\delta_{x,x'}$
Amplitude encoding	$\lvert x^\dagger x' \rvert^2$
Angle encoding	$\prod_{k=1}^{N} \lvert \cos(x'_k - x_k) \rvert^2$

Table 2: Overview of data-encoding strategies and their functions (Schuld, 2021)

The input domain is assumed to be $X \subseteq \mathbb{R}^N$. The quantum functions are defined as the dot product or inner product of the data points in the quantum state. The basis encoding has a kernel equal to 1 if the two inputs are the same and 0 otherwise. The amplitude encoding has a kernel proportional to the square of the dot product between the two inputs. The angle encoding has a kernel equal to the product of the cosine similarity between each feature of the two inputs.

As we see in the above resulting functions, the overlap output differs for different encodings because each encoding has a different representation of the input data. The overlap measures the similarity between the quantum states that represent the inputs. For example, suppose the data is represented in the basis encoding. In that case, the result of the overlap will depend on the presence or absence of the computational basis states in the superposition. On the other hand, if the data is represented in the angle encoding, the result of the overlap will depend on the relative phase between the quantum states. These

differences in the outputs of the overlap for different encodings highlight the importance of choosing the appropriate encoding strategy based on the specific requirements of the quantum algorithm, especially when comparing quantum results to classical ones.

Below it is possible to see in detail how to derive these functions.

Basis encoding

The data-encoding feature map of basis encoding maps a binary string to a computational basis state,

$$\phi : x \mapsto |i_x\rangle \langle i_x| \tag{3.20}$$

The Kronecker delta gives the quantum kernel:

$$\kappa(x, x') = |\langle i_{x'}|i_x\rangle|^2 = \delta_{x,x'} \tag{3.21}$$

Which is a rigorous similarity measure on input space and arguably not the best choice of data encoding for quantum machine learning tasks, do to the high number of qubits needed.

Amplitude encoding

The data-encoding feature map of amplitude encoding associates each input with a quantum state whose amplitudes in the computational basis are the elements in the input vector,

$$\phi : x \mapsto |x\rangle \langle x| = \sum_{i,j=1}^{N} x_i x_j^* |i\rangle \langle j| \tag{3.22}$$

The quantum kernel is the absolute square of the linear kernel

$$\kappa(x, x') = |\langle x'|x\rangle|^2 = |x^\dagger x'|^2 \tag{3.23}$$

This quantum kernel does not add much power to a linear model in the original feature space. It is more of interest for theoretical investigations that aim to eliminate the effect of the feature map.

Rotation encoding

The data-encoding feature map of this time-evolution encoding executed by Pauli rotations is given by

$$\phi : x \mapsto |\phi(x)\rangle\langle\phi(x)| \tag{3.24}$$

where, if we use Pauli-Y rotations,

$$\phi(x)\rangle = \sum_{q_1,\ldots,q_n=0}^{1} \prod_{k=1}^{n} \cos(x_k)^{q_k} \sin(x_k)^{1-q_k} |q_1, \ldots, q_n\rangle \tag{3.25}$$

, and the corresponding quantum kernel is related to the cosine kernel:

$$\kappa(x, x') = \prod_{k=1}^{n} |\sin(x_k)\sin(x'_k) + \cos(x_k)\cos(x'_k)|^2 = \prod_{k=1}^{n} |\cos(x_k - x'_k)|^2 \tag{3.26}$$

3.4 Quantum-Clustering

Quantum Clustering is a subset of machine learning that utilizes quantum computing to perform clustering algorithms. Clustering, as seen in 2.4, groups similar data points in a dataset. It is a crucial step in many machine learning applications. With the advent of quantum computing, quantum clustering has emerged as a promising approach to performing clustering algorithms more efficiently.

A popular choice is the q-means algorithm, offering a quantum-inspired alternative to the classical K-means algorithm. Others like the *Quantum Spectral Clustering* (Landman [2021]), a graph-based machine learning algorithm. Here, we will only cover one quantum clustering algorithm, the q-means, since it also uses the Swap Test and is a direct match in the quantum realm to the classical k-means algorithm.

The q-means algorithm The **q-means** algorithm is a quantum-based adaptation of the classical k-means algorithm. It uses the Swap Test as a distance metric to perform clustering on quantum data. Unlike classical k-means, q-means can handle quantum data more efficiently.

This algorithm uses quantum subroutines instead of the traditional k-means methods of performing tasks such as efficient tomography, finding the minimal value among a group of components, and distance estimation. In this quantum analog, we first choose k randomly chosen centroids or utilize an initialization method similar to k-means++. Then, in Steps 1 and 2, all data points are allocated to clusters in superposition rather than sequentially, and in Steps 3 and 4, we update the centroids of the clusters. Until convergence, the operation is repeated.

The steps below are a brief overview of the quantum algorithm defined in (Landman [2021]) Chapter 8. For a more detailed understanding of the implementation, we recommend an integral look at this chapter.

1. **First Step - Distance Estimation** To perform the distance estimation, we first need to perform a superposition of states where we transform a set of states $|a\rangle\,|b\rangle$ in

$$|\psi\rangle = \frac{1}{2}\,|0\rangle \otimes (|a\rangle\,|b\rangle + |b\rangle\,|a\rangle) + \frac{1}{2}\,|1\rangle \otimes (|a\rangle\,|b\rangle - |b\rangle\,|a\rangle)$$

(it is important to notice that at this step, it is assumed to have two quantum states $|a\rangle$ and $|b\rangle$, probably stored in a QRAM). We can use a subroutine like the *Swap Test* mentioned in the previous section 3.3.1 to perform this operation. The runtime of this subroutine is $\mathcal{O}(\log Nd)$ for a data matrix $V \in \mathbb{R}^{Nxd}$ (see Landman [2021] *Theorem 8.1*)

2. **Second Step - Cluster Assignment** We have calculated the squared inner product between each point and the k centroids, then selected the new clusters.

3. **Third Step - Centroid State Creation** The index of the data points is stored in the first register of this state generated in the previous step, and the label for the data point being iterated through is stored in the second register. Both registers are part of this state. Given these conditions, we must locate the new centroids, representing the average data points with the same label. (once more, this step requires a QRAM)

4. **Fourth Step - Centroids Update** To complete the update phase, it is necessary to transition from the quantum states that represent the centroids to a description of the centroids that adhere more closely to the classical model. The algorithm for vector state tomography will be used in this endeavor.

There are many implementations of the **q-means** algorithm, one slightly different from another, either by the steps used to reach the final goal or the runtime complexity calculation. Regardless, these all have a common point: the need for a quantum random access memory (QRAM), a theoretical device to store quantum data. Some possible implementations can be found in (Park et al. [2019]).

The need for a quantum random access memory and the current limitations of quantum computing have made it challenging to implement this algorithm on real-world data. This limitation is why in the next chapter, we will introduce the Hybrid k-means algorithm, which combines the strengths of both quantum and classical algorithms to overcome these challenges and provide a more practical solution for real-world data and motivates our desire to create a practical solution for clustering tasks using a quantum computer.

Chapter 4

Experimentation of Swap Test as a distance metric

This chapter comes as a result of the investigation made on both Classical and Quantum Machine Learning. Despite the current evolution in the field of *QML*, redefining the current Machine Learning state is still challenging, which means that in order for *QML* to tackle the same problems as *ML*, we first need to build more robust and less noisy machines and possibly build a more high-level quantum programming language that allows users to leverage the power of quantum towards more challenging problems known in *ML* such Forecasting, Clustering, and Computer Vision.

These limitations do not mean we cannot make algorithms and POCs (Proofs of Concept) that showcase some quantum advantages. Nonetheless, with this master thesis, we wanted to learn more about classical Machine Leaning and test the currently available Quantum Computer simulators in real-world ML problems. Therefore, we decided to join forces and test a simple **Hybrid Classical Quantum** algorithm approach that would allow us to use classical data and leverage some quantum advantages in the overall process.

The idea behind the Hybrid Classical Quantum algorithm developed is to create a new distance metric for the K-Means algorithm since there are some immediate difficulties in implementing a full quantum q-means presented in 3.4, such as the lack of qubits in a quantum machine and the need for a Q-RAM.

Our proposed distance metric, based on the **Quantum Overlap** concept, will serve as the "quantum block" in our algorithm. We aim to introduce a quantum approach to existing classical machine learning algorithms, going beyond just K-means.

To create this quantum metric, we will be using the **Swap Test** circuit (as shown in Figure 3.3.1) to evaluate its feasibility as a distance metric in distance-based classical machine learning algorithms.

Algorithm 3 refers to the pseudo-code for the **Hybrid** quantum-classical approach.

Algorithm 3 Hybrid KMeans

Input N size data set
Selecting the initial K centroids
Define a threshold
for each point i in N **do**:
    for each K centroid **do**:
        call the **SWAP func** 2 and calculate the difference between the old and the new centroids
        if difference < threshold **then**
            assigns each sample to its nearest centroid
        end if
    end for
    creates new centroids by taking the mean value of all of the samples assigned to each previous centroid
end for

The implementation of this algorithm uses the **K-Means** algorithm from Pyclustering a python library that has a collection of cluster analysis, graph coloring, neural network models and result analysis, implemented in **C++** that besides very fast allows the user to create its distance metric. We also used for the *Swap Test* a self-made class to do algebraic matrix calculations, simulating quantum operations.

4.1 Classification

To test the algorithm in a real-world situation, we first decided to tackle two classification problems, one using a small data set for that we use the **Iris Dataset** (Fisher [1988]) a commonly used dataset, the other is the **Dry Bean Dataset** (de Wolf [2019]), both donated to **UC Irvine University** by the authors and available in Machine Learning University Repository.

Iris Dataset

The Iris flower dataset is balanced and represents the three iris plant varieties, Iris setosa, Iris versicolor, and Iris virginica. Each variety represents a class in this data collection by a categorical feature. There are three classes with a total of fifty occurrences each. One class can be linearly separated from the others, whereas the remaining cannot.

These three classes are represented by one dimension vector composed of 4 numerical features:

1. The length of the sepal in centimeters;
2. The breadth of sepal in centimeters;
3. The length of the petal in centimeters;
4. The breadth of each petal, measured in centimeters

We chose this data set for its widespread use for tutorial and teaching purposes.

We have tested the hybrid algorithm against this dataset using two different classical measures, the *Cosine Distance* [2] and the *Euclidean Distance* [2]. We also used a Quantum distance measure using the **Swap Test**, implemented using algebra calculation in Python.

In the table 3 are represented two evaluation metrics for the classification task performed on the iris dataset.

Distance Metric and Evaluation Metrics		
Distance Metric	Fowlkes-Mallows	Rand Index
Euclidean	0.81	0.87
Cosine	0.96	0.97
SWAP Test (Rotation Encoding)	0.83	0.88
SWAP Test (Amplitude Encoding)	0.78	0.87

Table 3: Fowlkes-Mallows and Rand Index evaluation metrics obtained for Euclidean, Cosine, and Swap test as distance metrics in the iris dataset (Fisher [1988])

Both metrics reported in table 3 assess the obtained classification concerning the actual classification. We can see that the SWAP test with rotation encoding performs better than Euclidean classical distance evaluation (for both quality metrics). Still, all other comparisons between classical and quantum indicate that quantum performs worse than classical. The results do not allow for a robust conclusion on the merits of the proposed quantum distance metric. Further experimentation is required.

Dry Beans Dataset

The beans dataset (de Wolf [2019]) comprises seven distinct kinds of dry beans and their characteristics, such as their form, shape, strain, and structure, used for a computer vision system to differentiate them. For that purpose, a high-resolution camera captured photos of 13,611 distinct grains to construct the classification model.

Since we have a limited number of qubits for our implementation and are dealing with a high-dimension dataset, we decided to use a dimensionality reduction method known as PCA 2.3.6 to reduce the number of features that constitute this dataset. We decided to use the same number of features as the previous dataset - Iris Dataset - to evaluate the time performance of the algorithm against a dataset that has $10\times$ more data points than the previous one.

As before, we have tested the hybrid algorithm against this dataset using two different classical measures, the *Cosine Distance* [2] and the *Euclidean Distance* [2]. In addition, we also used a Quantum distance measure using the **Swap Test**, implemented using algebra calculation in Python.

In the table 4 are represented as two evaluation metrics for the classification task performed on the iris dataset.

Distance Metric and Evaluation Metrics		
Distance Metric	Fowlkes-Mallows	Rand Index
Euclidean	0.81	0.87
Cosine	0.96	0.97
SWAP Test (Rotation Encoding)	0.59	0.86
SWAP Test (Amplitude Encoding)	0.41	0.78

Table 4: Fowlkes-Mallows and Rand Index evaluation metrics obtained for Euclidean, Cosine, and Swap test as distance metrics in the dry beans dataset (de Wolf [2019])

According to both quality metrics, the proposed quantum distance metric performs worse than the classical approaches. This result suggests that the SWAP test with the evaluated data encoding strategies might need to be revised for classification tasks. However, there was a case where the SWAP test performed better for the IRIS dataset. Therefore, more extensive experimental evaluations are required, eventually including different data encoding strategies.

4.2 Clustering

As shown in the above chapters, clustering involves distinguishing objects into groups or clusters based on their similarities. To evaluate the performance of our hybrid algorithm in clustering, we have selected unlabeled generic datasets from (Pedregosa et al. [2011]). These datasets have different geometric structures, as shown in figure 13, which can help us analyze the algorithm and compare it with other classical algorithms. We will use these clusters as the reference for further analysis.

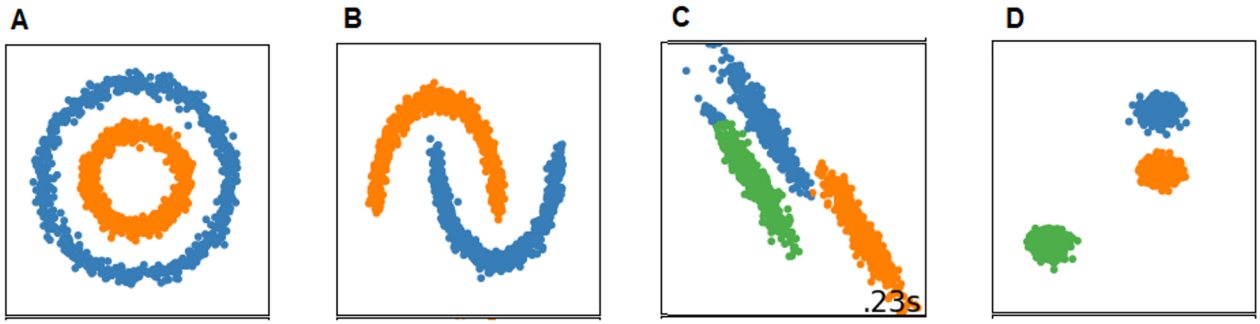


Figure 13: (A) two 1D non-flat manifolds (circles) which are non-convex, (B) two 1D non-flat manifolds (arcs) which are non-convex, (C) three 1D flat manifolds (segments) which are convex, (D) three 0D flat manifolds (centers as points) which are convex.

To explore the effectiveness of our proposed hybrid algorithm, using the Swap Test to compute quantum distance measures, we compare its performance against two commonly used classical distance measures: the Euclidean distance metric and the Cosine distance, allowing us to estimate the similarity between the two quantum states.

Figure 14 presents the results of the K-means algorithm using four distance metrics: A - Classical Euclidean distance, B - Classical cosine distance, C - Quantum SWAP test with rotation encoding, and D - Quantum SWAP test with amplitude encoding.

The first thing to note is that the obtained clusters are significantly different from the reference ones (see Figure 13), except for the data set presented in Figure 13 D and on the 5th row of figure 14. The results are different from the reference for the majority of the datasets, suggesting that the problem be revisited, and an appropriate feature map has to be found, such that results are identical to the reference (at least for the classical Euclidean distance metric).

Limiting the comparisons to figure 14, note that the quantum swap test with rotation encoding (column C) obtains results similar to both classical cases (columns A and B), whereas using amplitude encoding results in a completely different clustering pattern. These results reinforce the conclusion drawn from the iris

dataset that the quantum SWAP test does not necessarily perform worst than the classical metrics. First, however, a vast experimentation program has to be performed to understand under what circumstances (and why) these classification/clustering differences occur. In particular, it is known that data encoding corresponds to feature mapping therefore, it plays a crucial role in such machine learning algorithms.

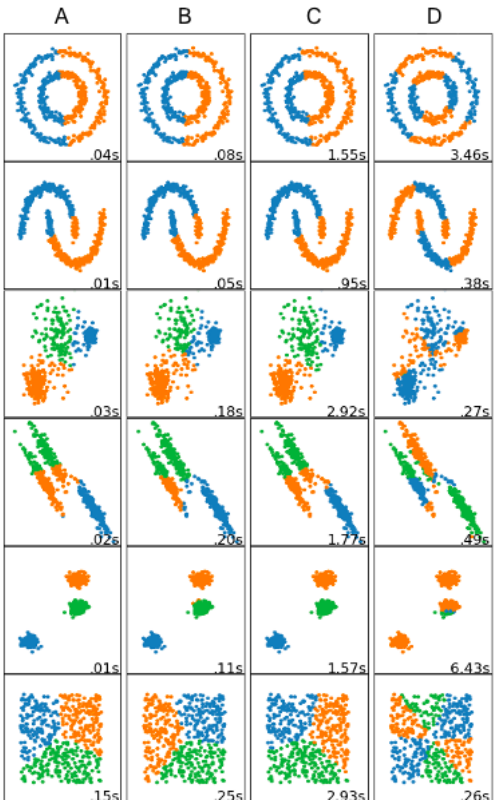


Figure 14: A - Euclidean Distance, B - Cosine Distance, C - **SWAP Test** (Rotation Encoding), D - **SWAP Test** (Amplitude Encoding)

Chapter 5

Conclusions

This dissertation showed the potential of quantum distance measures in classical Machine Learning (ML) algorithms and the challenges and limitations of current quantum computing technology for Quantum Machine Learning (QML). Despite these challenges, quantum computing can potentially solve problems intractable with classical computers, such as simulating complex quantum systems and optimizing large-scale combinatorial problems. However, the technology is still in its early stages and faces significant challenges such as overhead in loading classical data, and a reduced number of qubits.

To address these challenges, we proposed a Hybrid approach that integrates a quantum routine in classical machine learning algorithms. Our routine was the Swap test, which replaces the classical distance metric in distance-based classical algorithms. The use of quantum distance measures provides an alternative way to represent the similarity between data points. Our results demonstrate the potential of quantum distance measures in both Classification 4.1 and Clustering 4.2 tasks.

While our experiments show that combining classical and quantum techniques can lead to similar performance compared to classical approaches,*e.g.* fig 14 and table 3, it is essential to note that the results obtained using quantum distance measures were not superior to those obtained using classical distance measures in all cases. In some datasets, the quantum distance measures performed worse than the Euclidean and Cosine distances, like in the classification task of the dry beans dataset 4. These results can be attributed to various factors, such as the choice of encoding strategy or the specific characteristics of the dataset being analyzed.

Therefore, further research is required to develop more robust and efficient hybrid clustering algorithms that effectively utilize quantum computing capabilities. Overall, this dissertation contributes to the growing body of research in quantum machine learning and provides a promising avenue for future exploration. Furthermore, by combining classical and quantum techniques, we can overcome the limitations of current QML algorithms and pave the way for developing more powerful and efficient machine learning algorithms.

Bibliography

Quantum algorithms for supervised and unsupervised machine learning. pages 1–11.

Scott Aaronson. The future of quantum computing. *Nature*, 546(7659):195–202, 2017.

Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. In Luc Lamontagne and Mario Marchand, editors, *Advances in Artificial Intelligence*, pages 431–442, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07*, 2007.

David Baker. Computational challenges in protein folding. *Current opinion in structural biology*, 13(1): 169–176, 2003.

Adriano Barenco, Andre' Berthiaume, David Deutsch, Artur Ekert, Richard Jozsa, and Chiara Macchiavello. Stabilisation of quantum computations by symmetrisation, 1996.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. 2016. doi: 10.1038/nature23474.

M. Emre Celebi, Hassan A. Kingravi, and Patricio A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013. ISSN 09574174. doi: 10.1016/j.eswa.2012.07.021.

S. Cesare and Y. Xiang. *Software Similarity and Classification*. SpringerBriefs in Computer Science. Springer London, 2012. ISBN 9781447129097. URL https://books.google.pt/books?id=Fy_mNhg2lK4C.

Ronald de Wolf. Quantum computing: Lecture notes. July 2019.

Ronald De Wolf. Quantum computing: Lecture notes. *arXiv preprint arXiv:1907.09415*, 2019.

M.M. Deza and E. Deza. *Encyclopedia of Distances*. SpringerLink. Springer Berlin Heidelberg, 2012. ISBN 9783642309588. URL https://books.google.pt/books?id=QxX2CX5OVMsC.

R.A. Fisher. Iris. UCI Machine Learning Repository, 1988.

E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983a. doi: 10.1080/01621459.1983.10478008.

E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983b. doi: 10.1080/01621459.1983.10478008.

Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 1*, 2:559–572.

Oktay Günlük, Jayant Kalagnanam, Minhan Li, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, 81:233 – 260, 2021.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. ISSN 1432-1343. doi: 10.1007/BF01908075. URL https://doi.org/10.1007/BF01908075.

Jonas Landman. Quantum algorithms for unsupervised machine learning and neural networks. *arXiv preprint arXiv:2111.03598*, 2021.

Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56, 05 2012. doi: 10.1137/120875909.

Seth Lloyd. The limitations of classical simulation of quantum systems. *Physical Review A*, 53(5):1786, 1996.

Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. pages 1–11, 2020. URL http://arxiv.org/abs/2001.03622.

T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN 9780071154673. URL https://books.google.pt/books?id=EoYBngEACAAJ.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. doi: 10.1017/CBO9780511976667.

Daniel K. Park, Francesco Petruccione, and June Koo Kevin Rhee. Circuit-Based Quantum Random Access Memory for Classical Data. *Scientific Reports*, 9(1):1–8, 2019. ISSN 20452322. doi: 10.1038/s41598-019-40439-3.

Antonio Parmezan, Vinícius Alves de Souza, and Gustavo Batista. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. *Information Sciences*, 01 2019. doi: 10.1016/j.ins.2019.01.076.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Conference on Empirical Methods in Natural Language Processing*, 2007.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

J.J. Sakurai and J. Napolitano. *Modern Quantum Mechanics*. Cambridge University Press, 2017. ISBN 9781108422413. URL https://books.google.pt/books?id=010yDwAAQBAJ.

Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. 2021. ISBN 978-3-030-83097-7.

Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. 2018. doi: 10.1103/PhysRevA.101.032308.

Donald Wunsch Seaar Al-Dabooni. Ieee transactions on neural networks and learning systems (tnnls). *Current opinion in structural biology*, 30(7):1928–1942, 2019.

Jiajun Wen, Zhihui Lai, Yinwei Zhan, and Jinrong Cui. The l2,1-norm-based unsupervised optimal feature selection with applications to action recognition. *Pattern Recognition*, 60:515–530, 2016. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2016.06.006. URL https://www.sciencedirect.com/science/article/pii/S0031320316301248.

Nathan Wiebe. Key questions for the quantum machine learner to ask themselves. *New Journal of Physics*, 22(9), 2020. ISSN 13672630. doi: 10.1088/1367-2630/abac39.

Nathan Wiebe, Ashish Kapoor, and Krysta Svore. Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning. *Quantum Information and Computation*, 15(3-4):318–358, jan 2014. ISSN 15337146. URL http://arxiv.org/abs/1401.2142.

Dongkuan Xu and Yingjie Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2):165–193, 2015. ISSN 2198-5812. doi: 10.1007/s40745-015-0040-1. URL https://doi.org/10.1007/s40745-015-0040-1.