UNIVERSITÉ SORBONNE PARIS NORD

DOCTORAL THESIS

Quantum Enhanced Machine Learning

Author:
Kaoutar BENLAMINE

Ph.D. directors:
Younès BENNANI, Professor, Université Sorbonne Paris Nord
Basarab MATEI, Senior Lecturer-HDR, Université Sorbonne Paris Nord

Doctoral Committee:

- Younès BENNANI, Professor, Université Sorbonne Paris Nord
- Nistor GROZAVU, Senior Lecturer, HDR, Université Sorbonne Paris Nord
- Charlotte LACLAU, Senior Lecturer, Télécom Saint Etienne
- Philippe LERAY, Professor, Polytech'Nantes
- Basarab MATEI, Senior Lecturer, HDR, Université Sorbonne Paris Nord
- Toshiaki OMORI, Professor, Kobe University
- Rosanna VERDE, Professor, Università della Campania
- Cédric WEMMERT, Professor, Université de Strasbourg

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Computer Science

in the

Université Sorbonne Paris Nord
Laboratoire d'Informatique de Paris Nord
UMR 7030 CNRS-USPN

November 3, 2020

"If you are not completely confused by quantum mechanics, you do not understand it."

John Wheeler

"I think I can safely say that nobody understands quantum mechanics."

Richard Feynman

UNIVERSITÉ SORBONNE PARIS NORD

Abstract

Institut Galilée
Laboratoire d'Informatique de Paris Nord
UMR 7030 CNRS-USPN

Doctor of Computer Science

Quantum Enhanced Machine Learning

by Kaoutar BENLAMINE

Cette thèse se situe à la frontière entre l'informatique quantique et l'apprentissage artificiel et a pour objectif le traitement des données en grandes dimensions.

L'apprentissage artificiel quantique est un nouveau domaine d'étude avec de récents travaux sur les versions quantiques d'algorithmes supervisés et non supervisés. Ces dernières années, de nombreux algorithmes d'apprentissage artificiel quantique ont été proposés afin d'améliorer le temps de traitement des algorithmes classiques.

La première contribution de cette thèse est de donner un aperçu sur le domaine quantique en apprentissage artificiel. Ensuite, la quantification des algorithmes classiques.

Nous avons proposé une analyse et une comparaison des différentes distances pour les algorithmes de clustering basés sur des protoptypes. Pour un ordinateur conventionnel, calculer les distances euclidiennes est facile, mais pour le faire de la même manière sur un ordinateur quantique serait beaucoup plus compliqué et exigerait plus de qubits que ce que nous pouvons nous permettre. Pour définir une distance quantique, nous utiliserons la nature probabiliste des qubits pour mesurer les amplitudes probabiliste. Pour les algorithmes de clustering quantique, les distances sont nécessaires mais ne doivent pas nécessairement être proportionnelles à la distance réelle, elles doivent avoir qu'une corrélation positive avec elle.

Tous les algorithmes de clustering basés sur des prototypes pourraient être résolus en utilisant ces distances. Comme application, nous avons présenté la version quantique de K-means qui est un algorithme de clustering. L'algorithme K-means quantique donne une bonne classification tout comme sa version classique, la seule différence réside dans la complexité: alors que la version classique de K-means prend du temps polynomial, la version quantique ne prend que du temps logarithmique. Nous avons également proposé la version quantique de la factorisation matricielle semi-non négative ainsi que la version collaborative de ces algorithmes à savoir: K-means quantique collaboratif et la factorisation matricielle semi-non négative quantique collaborative qui se basent sur la combinaison de plusieurs solutions de clustering pour obtenir une meilleure solution en termes de clustering et de complexité.

UNIVERSITÉ SORBONNE PARIS NORD

Abstract

Institut Galilée
Laboratoire d'Informatique de Paris Nord
UMR 7030 CNRS-USPN

Doctor of Computer Science

Quantum Enhanced Machine Learning

by Kaoutar BENLAMINE

The goal of this thesis is on the borderline between quantum computing and machine learning and deals with data processing in very large dimensions.

Quantum machine learning is a new area of study with the recent work on quantum versions of supervised and unsupervised algorithms. In recent years, many quantum machine learning algorithms have been proposed providing a speed-up over the classical algorithms.

The first contribution of this thesis is to give an overview of the field. Afterwards, the quantization of classical algorithms.

We proposed an analysis and a comparison of different distances for protoptypes-based clustering algorithms. For a conventional computer, calculating Euclidean distances is easy, but doing it in the same way on a quantum computer would be much more complicated and would require more qubits than we can afford. To define a quantum distance, we will use the probabilistic nature of qubits to measure phase differences and probability amplitudes. For quantum clustering algorithms, distances are necessary but does not need to be proportional to the real distance, but it must only have a positive correlation with it.

All prototypes based clustering algorithms could be solved using these distances. As an application of this task, we presented a quantum K-means version which is a prototype based clustering algorithm. This quantum K-means algorithm gives a good classification just like its classical version, the only difference resides in the complexity: while the classical version of K-means takes time polynomial, the quantum version takes only time logarithmic especially in large datasets. We also proposed the quantum version of Semi Non-negative Matrix Factorization. Likewise, we proposed the quantum collaborative version of these algorithms; quantum collaborative K-means and quantum collaborative Semi Non-negative Matrix Factorization. These collaborative algorithms are based on combining several clustering solutions to get a better solution in terms of clustering and also complexity.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my director Prof. Younès Bennani for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. He never hesitated to give me new ideas and show me the right way to follow. His guidance helped me throughout the research and writing this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. I think I am lucky to meet such a person who is not only a good director but also a kind person who provided me guidance for work related subjects and for personal life as well.

My sincere thanks also goes to my co-director Dr. Basarab Matei who helped me not only in the work but also personally, he was all the time encouraging me in the hard moments. Also my grateful goes to my supervisor Dr. Nistor Grozavu who encouraged me during my work.

I want to thank Prof. Rosanna Verde (Università della Campania), Prof. Philippe Leray (Polytech'Nantes) and Prof. Toshiaki Omori (Kobe University) who have reported my thesis and for their constructive remarks. I thank also Prof. Cédric Wemmert (Université de Strasbourg), Dr. Charlotte Laclau (Télécom Saint Etienne) for accepting to be part of my thesis jury and for their interesting and relevant comments.

I would also like to thank all my friends in Computer Science Laboratory of Paris North University (LIPN-UMR 7030 CNRS).

Last but not least, I would like to thank my family especially my parents who supported me and encouraged me during my studies. My grateful goes also to my brother and sisters particularly Majda who was all the time listening to my problems and encouraging me to do my best.

Kaoutar BENLAMINE
Université Sorbonne Paris Nord
November 2020

Contents

List of Figures

List of Tables

Dedicated to my parents . . .

Introduction

Les techniques d'apprentissage artificiel sont appliquées pour résoudre une grande variété de problèmes tels que : le tri, la régression et la classification. En apprentissage artificiel supervisé, l'apprenant reçoit un ensemble d'exemples avec des caractéristiques présentées sous la forme de vecteurs en haute dimension et avec des étiquettes correspondantes pour marquer les catégories. L'objectif est de classer de nouveaux exemples en utilisant les connaissances extraites à partir de ces ensembles de données. En apprentissage non supervisé, la machine essaie de trouver une structure cachée dans des données non étiquetées.

La quantité de données générées dans notre société augmente de plus en plus, c'est pour cela qu'il est nécessaire de disposer de moyens plus puissants pour traiter ces informations. C'est la raison pour laquelle des études et applications récentes se concentrent sur le problème de l'apprentissage artifiel à grande échelle. De nombreux travaux ont été consacrés à l'apprentissage artificiel quantique. Par exemple, le développement de procédures quantiques pour l'algèbre linéaire comme : la multiplication matricielle, le calcul des vecteurs propres et des valeurs propres et l'estimation des distances entre des états quantiques. Des efforts ont également été faits pour résoudre le problème de la reconnaissance des formes et pour développer des versions quantiques des réseaux de neurones artificiels largement utilisés en apprentissage automatique. Le clustering pourrait poser un défi pour les ordinateurs classiques, en particulier avec la croissance de la taille des données. Si les données sont dans des espaces vectoriels de dimension N, cela prendrait du temps $\mathcal{O}(polyN)$ sur les ordinateurs classiques, alors qu'avec des ordinateurs quantiques, cela prendrait un temps $\mathcal{O}(\log N)$. Ainsi, le clustering quantique peut fournir des accélérations exponentielles pour les problèmes impliquant de grandes masses de données en haute dimensionnalité.

Dans cette thèse, nous nous sommes concentrés sur la transformation d'algorithmes classiques non supervisés en algorithmes quantiques, nous détaillons les différentes étapes qu'un algorithme quantique devrait suivre, en particulier l'algorithme QK-means, le clustering collaboratif et la factorisation matricielle semi non négative quantique. Et comme la tâche principale de l'algorithme d'apprentissage automatique est d'analyser la similitude entre les données, nous proposerons une estimation de la distance pour le clustering à base de prototypes quantiques qui se fait à travers l'évaluation de la distance et des produits scalaires entre les données.

Guide de la thèse

Cette thèse est structurée en six chapitres et organisée comme suit:

Chapter 1: Aperçu sur le calcul quantique

Les ordinateurs quantiques et classiques essayent de résoudre des problèmes, mais la manière dont ils manipulent les données pour obtenir des réponses est fondamentalement différente. C'est pourquoi nous avons fourni ce chapitre afin de donner une explication de ce qui rend les ordinateurs quantiques uniques. Nous introduisons

les concepts de base de la mécanique quantique, afin de comprendre comment fonctionnent les algorithmes quantiques et comment la mécanique quantique sera utilisée pour accélérer les algorithmes d'apprentissage artificiel. Ce chapitre présentera les concepts et les éléments les plus utiles en calcul quantique comme : le qubit, la superposition, l'intrication, ... Et aussi la description des algorithmes quantiques les plus connus.

En effet, la mécanique quantique semble contre-intuitive même pour les physiciens qui la traitent quotidiennement. Et cela est dû au fait que tout dans l'univers est à la fois une nature de particules et d'ondes. Comme l'écrivait Albert Einstein: «Il semble que nous devions parfois utiliser l'une et parfois l'autre, alors que parfois nous pouvons utiliser l'une ou l'autre. Nous sommes confrontés à un nouveau type de difficulté. Nous avons deux images contradictoires de la réalité ; séparément aucune d'elles n'explique complètement les phénomènes de la lumière, mais ensemble elles le font ".

Chapter 2: Etat de l'art des algorithmes d'apprentissage quantiques

Dans ce chapitre, nous présentons les algorithmes d'apprentissage classiques et quantiques et nous faisons une comparaison entre ces différents algorithmes. Nous décrivons les algorithmes les plus utilisés en apprentissage artificiel tels que : *K*-means, *K*-median, l'analyse en composantes principales (ACP), les séparateurs à vastes marges (SVM) et les réseaux de neurones. Nous présentons pour chaque algorithme sa version quantique. Enfin, nous comparons les performances des algorithmes d'apprentissage artificiel classiques et quantiques en termes de complexité.

Chapter 3: Clustering quantique à base d'un seul modèle

Les principales étapes des algorithmes de clustering à base de prototypes sont bien connues, mais quand nous en arrivons à la version quantique cela nécessite des adaptations. C'est pourquoi, nous optons pour l'introduction d'un chapitre qui montre en détail les principales étapes des algorithmes de clustering quantiques afin de facilter la lecture des chapitres qui suivent, après on a présenté la version quantique de *k*-means et la factorisation matricielle semi non-négative.

Dans ce chapitre, nous avons décrit les différentes manières de transformer les données à un état quantique. Ensuite, nous avons présenté la fidélité qui est une mesure de similarité. Cette dernière va permettre de calculer la distance entre les états quantiques. Cette distance est totalement différente de la version classique et ceci est dû à l'instabilité de la distance dans le cas quantique.

Nous présentons l'algorithme de Grover qui est un algorithme quantique qui effectue une recherche sur des données non ordonnées et offre une accélération quadratique par rapport à la version classique. Ce dernier nous aidera dans la classification quantique car nous pouvons rechercher le centroïde le plus proche d'une donnée.

Ensuite, nous montrons l'évaluation de l'indice quantique que nous avons adapté afin d'évaluer la qualité du clustering dans le cas quantique "Quantum Davies Bouldin".

Nous complétons le chapitre par une série d'évaluations empiriques afin d'évaluer les distances quantiques et leur comportement par rapport à la stabilité durant la recherche des centres les plus proches dans le bon ordre.

Au sein de ce chapitre, une analyse de la distance quantique sera effectuée ; car cette distance est instable (elle change d'une itération à une autre entre les mêmes données). Cette analyse est cruciale puisqu'elle permettra de développer tout algorithme de clustering à base de prototypes. La version quantique de *K*-means donne

une bonne classification tout comme sa version classique. La principale différence réside dans la complexité ; alors que la version classique de K-means prend un temps polynomial, la version quantique ne prend qu'un temps logarithmique, en particulier dans le cas des grands ensembles de données.

Nous décrivons aussi la version classique et quantique de la factorisation matricielle semi non-négative (SNMF). La version quantique de SNMF pourrait être mise en œuvre en utilisant la descente de gradient quantique. La factorisation matricielle semi non-négative quantique (QSNMF) donne une accélération exponentielle par rapport à sa version classique. L'idée principale de cette descente de gradient quantique est d'améliorer la fonction objectif afin d'avoir une forme algébrique. En effet, les fonctions homogènes et non homogènes sur de grands espaces dimensionnels peuvent être optimisées en utilisant la descente de gradient quantique [Reb+16]. Ensuite, nous utilisons plusieurs copies de l'état quantique actuel $|current\rangle$ pour produire plusieurs copies d'un autre nouvel état quantique $|new\rangle$ en utilisant la fonction objectif du gradient qui a été optimisé.

Chapter 4: Clustering quantique à base de plusieurs modèles (approche collaborative)

Ce chapitre est consacré à l'approche collaborative. En particulier, nous présenterons la version classique et la version quantique de collaborative K-means et La factorisation matricielle semi non-négative collaborative.

K-means quantique collaboratif vise à collaborer avec différents modèles de K-means afin d'obtenir un meilleur clustering par rapport aux QK-means et de gagner en vitesse par rapport à sa version classique.

Nous abordons également dans ce chapitre la factorisation matricielle semi non-négative collaborative quantique qui consiste à faire collaborer les différentes méthodes de clustering. Cette version de SNMF collaborative quantique n'a pas seulement l'avantage de rapidité mais aussi en collaborant plusieurs solutions de clustering, chacune avec ses propres biais et imperfections, on obtient une meilleure solution en termes de qualité de clustering.

Chapter 5: Autres directions de recherche: Apprentissage collaboratif pour améliorer la non-unicité de NMF

Dans ce chapitre, nous étudions la version classique de NMF collaborative afin d'améliorer son unicité. Nous proposons une nouvelle stratégie pour contrôler la variabilité des résultats dans les modèles NMF. Cette stratégie repose sur l'apprentissage collaboratif entre plusieurs modèles NMF suivi d'un consensus entre les différentes solutions obtenues. Les poids de la procédure de collaboration de plusieurs NMF sont définis en utilisant Karush-Kuhn-Tucker (KKT), de cette façon nous réduisons l'erreur de reconstruction standard et nous évitons également la collaboration négative entre les modèles. En effet, la solution NMF n'est pas unique : une matrice et son inverse peuvent être utilisés pour changer la factorisation. C'est à dire que l'algorithme NMF est instable : si nous exécutons NMF pour les mêmes données, nous obtenons une factorisation différente pour chaque exécution. Pour faire face à ce problème, nous avons proposé une nouvelle approche qui se base sur la collaboration entre plusieurs NMF et puis un consensus entre ces différentes collaborations. Enfin, les résultats expérimentaux ont montré l'efficacité de notre approche proposée qui est la réduction de l'erreur de reconstruction standard dans le modèle NMF. En effet, les résultats obtenus montrent que l'erreur de reconstruction standard tend vers zéro lorsque le nombre de modèles NMF impliqués tend vers l'infini. Comme futur travail, nous avons l'intention d'étudier la stabilité de NMF dans la version quantique.

Introduction

Machine learning techniques are applied for solving a large variety of problems such as: sorting, regression and classifying information. In supervised machine learning, the learner is provided a set of training examples with features presented in the form of high-dimensional vectors and with corresponding labels to mark its category. The aim is to classify new examples based on these training sets. In unsupervised learning, the machine tries to find a hidden structure in unlabeled data.

As the amount of data generated in our society is increasing more and more, it is necessary to have more powerful ways of handling information. That's why recent studies and applications are focusing on the problem of large-scale machine learning. A lot of works have been devoted to quantum machine learning. For example, the development of quantum procedures for linear algebra as: matrix multiplication, eigenvectors and eigenvalues of matrices and estimating distances between quantum states. Efforts have also been made to solve the problem of pattern recognition and to develop the quantum version of artificial neural networks widely used in machine learning. Clustering could pose a challenge for classical computers especially with the current speed growing data size. If our vectors are in N dimensional vector spaces, this would take time $\mathcal{O}(polyN)$ on classical computers, while with quantum computers this would take time $\mathcal{O}(\log N)$. Thus, quantum clustering can provide exponential speed-ups for problems involving large vectors.

In this dissertation, we were focused on transforming unsupervised classical algorithms to quantum one, we give in details the different steps that a quantum algorithm should follow, in particular QK-means, quantum collaborative K-means and quantum Semi Non negative matrix factorization. And as the main task of machine learning algorithm is to analyze the similarity between the vectors, we give the estimation of the distance for quantum prototypes based clustering that is done through the evaluation of the distance and the inner products between the large vectors.

Guide to the Dissertation

This dissertation is structured in six chapters and organized as follows:

Chapter 1: Scope and overview of quantum computation

Quantum and classical computers both try to solve problems, but the way they manipulate data to get answers is fundamentally different. That's why we provided this chapter in order to give an explanation of what makes quantum computers unique by introducing the basic concepts of quantum mechanics, so as to understand how quantum algorithms work and how quantum mechanics is used to speed-up machine learning algorithms. In particular, this chapter will include the most useful concepts in quantum computation as: qubit, superposition, entanglement, ... And also the description of the most known quantum algorithms.

Indeed, quantum mechanics seems to be counter-intuitive even for the physicists who deal with it every day. And this is due to the fact that everything in the universe

has both particle and wave nature, at the same time. As Albert Einstein wrote: "It seems as though we must use sometimes the one theory and sometimes the other, while at times we may use either. We are faced with a new kind of difficulty. We have two contradictory pictures of reality; separately neither of them fully explains the phenomena of light, but together they do".

Chapter 2: State of the art of Quantum Machine Learning algorithms

In this chapter, we present the classical and quantum machine learning algorithms and we make a comparison between these different algorithms. We explained the most used algorithms in machine learning such as: K-means, K-median, principal component analysis, support vector machine and neural networks. We try to explain the quantum version of these algorithms. Finally, we compare the performance of classical and quantum machine learning algorithms in terms of complexity.

Chapter 3: Quantum one model clustering

The main steps for prototype based clustering are well known but when we come to the quantum version of prototype based clustering it's yet unclear. That's why, we opt for introducing a chapter that shows in detail the main steps of quantum prototype based clustering algorithms and then we present the quantum version of K-means and Semi Non-negative matrix Factorization.

We showed the different manner to transform the data to quantum states. Then, we describe fidelity which is a similarity measure. This latter will allow to compute the distance between states. This distance is totally different from the classical version and this is due to the instability of the quantum distance.

Therefore, we present Grover's algorithm which is a quantum algorithm that performs a search over unordered data and gives a quadratic speed-up over the classical version. This latter will help us in quantum clustering as we can search for the closest centroid to a given data.

Then, we showed the quantum index evaluation that we have adapted in order to evaluate the clustering of the quantum algorithm "quantum Davies bouldin".

We complete the chapter with a series of empirical evaluations in order to evaluate the quantum distances and its behavior versus the stability to find the nearest centers in the right order.

During this chapter, an analysis of quantum distance will be done because the notion of distance in quantum computing is different from the classical one; as the distance in quantum mechanics is unstable (it changes from an iteration to another). This analysis is so crucial as it will help to solve any prototype based clustering algorithm.

The quantum version of K-means has given a good classification just like its classic version, the experimental results show the effectiveness of that. Indeed, the main difference resides in complexity; while the classic version of K-means takes polynomial time, the quantum version only takes logarithmic time, especially in large data sets.

We also described the Semi Non-negative matrix factorization and we showed also its quantum version. The quantum version of SNMF could be implemented using quantum gradient descent. Quantum Semi Non-negative Matrix Factorization (QSNMF) gives an exponential speedup compared to its classical version. The main idea of this quantum gradient descent is to improve the objective function in order to have an algebraic form. Indeed, both homogeneous and inhomogeneous function over large dimensional spaces can be optimized by using quantum gradient descent [Reb+16]. Then, we use multiple copies of the current quantum state $|current\rangle$ to

produce multiple copies of another new quantum state $|new\rangle$ by using the objective function of the gradient that has been optimized.

Chapter 4: Quantum Multi-models clustering (collaborative approach)

This chapter will be devoted to describe the quantum collaborative approach more specifically; we will describe the collaborative version of *K*-means and SNMF in both the classical and the quantum version. Finally, we made a comparison between the classical and the quantum version of these algorithms in terms of computational time complexity.

Collaborative quantum *K*-means aims to collaborate different *K*-means models in order to get a better clustering compared to *QK*-means and to get a speed up compared to its classical version.

We also tackle in this chapter the quantum collaborative semi non-negative matrix factorization which consists on the collaboration of different clustering methods, in order to reach an agreement on the partitioning of a common dataset. This latter doesn't have only the advantage of speed up but also by collaborating several clustering solutions, each one with its own bias and imperfections, one will get a better overall solution in terms of classification.

Chapter 5: Other research direction: Collaborative learning to improve the non uniqueness of NMF

In this chapter, we study the classical and quantum version of collaborative NMF in order to improve its uniqueness. We propose a novel strategy to control the variability results in the NMF models. This strategy is based on the collaborative learning between several NMF models followed by a consensus between the different solutions obtained. The weights of the collaboration procedure of several NMF is defined by using Karush-Kuhn-Tucker (KKT), in this way we reduce the standard reconstruction error and also we avoid the negative collaboration between the models. Indeed, NMF solution is not unique: a matrix and its inverse can be used to transform the two factorization matrices. That's to say that NMF algorithm is unstable: if we run NMF for the same data we get different factorization between each run. To tackle that problem, we have proposed a new approach which is adapted for collaboration between several NMF and finding a consensus between these different collaborations. Finally, the experimental results showed the effectiveness of our proposed approach which is the reducing of the standard reconstruction error in NMF model. The obtained results show that the standard reconstruction error go to zero when the number of involved NMF models go to infinity.

As a future work, we are intending to study the non-uniqueness of NMF in the quantum version.

Chapter 1

Scope and overview of Quantum Computation

Contents

1.1 Context

Despite the success of machine learning in different tasks as clustering, data generation and reinforcement learning, the increasing size of datasets makes the treatment of all that information very tough by taking a long time.

Recently, the availability of a large amount of data and also the increased computational power makes machine learning (ML) algorithms achieve a remarkable success in different areas such as face recognition, self driving, entertainment services and finance investment. However, this revolution starts to face increasing challenges. With the increasing size of datasets constantly, we might soon reach a point where

the current computational tools will no longer be sufficient. Although tailored hardware architectures, like graphics processing units (GPUs) and tensor processing units (TPUs), can significantly improve performance, they might not offer a structural solution to the problem. That's why a number of researchers are interested in the field of quantum computation which is a computational paradigm based on the laws of quantum mechanics, in order to use its power to speed up classical machine learning algorithms. Therefore, quantum computer can be the solution for that problem; it can efficiently solve selected problems that are believed to be hard for classical machines.

The power of quantum computers resides on manipulating large numbers of high-dimensional vectors in a time faster than classical computers. Indeed, the operations like dot products, overlaps, norms, etc., that take a polynomial time in the classical machine learning algorithms, take only a logarithmic time in the quantum version. That's why quantum computers are a powerful tool for manipulating large amounts of data.

- A quantum algorithm must be faster, computationally less complex than any known classical algorithm to the same purpose.
- Privacy of the owners of the data.
- Quantum machine learning also provides advantages in terms of privacy: the data base itself is of size O(MN), but the owner of the data base supplies only O(log MN) quantum bits to the user who is performing the quantum machine learning algorithm.
- Quantum neural networks exploit the superposition of quantum states to accommodate gradual membership of data instances.
- Gain in terms of complexity and speed.
- These clustering algorithms offer a speedup compared to their classical counterparts, but they do not improve the quality of the resulting clustering process.

  This is based on the belief that finding the optimal solution for a clustering problem is NP-hard, then quantum computers would also be unable to solve the problem exactly in polynomial time.

Example:
The rate of generation of electronic data generated per year is estimated to $10^{18} \approx 2^{60}$ bits. This entire data set could be represented by a quantum state using 60 qubits, and the clustering analysis could be performed using a few hundred operations.

Even if the number of bits to be analyzed were to expand to the entire information content of the universe within the particle horizon, $O(10^{90} \approx 2^{300})$ bits.

1.2 A rough description of a quantum system

We will start a bit abruptly by defining what a quantum system is. To keep it easy as possible, so that every one can understand; we will try not to use mathematical notions in this first description of a quantum system and its state. Let's roughly say that a quantum system is a system (think of it as a computational unit), characterized by:

- The number of qubits n (for the moment, think of computer bits, but they are quite different),

- Its initial state, which is represented as a complex vector in $\mathbb{C}^{2^n}$.
- A collection of registers, each one of which is associated with a qubit (thus, we must have at least n in our architecture), and each one directly interacts with all or a subset of the rest registers (qubits), based on a couple mapping of the quantum computer architecture.

When we say "quantum computing", we very roughly mean that:

- We have access to a quantum computer.
- We start from a specific state, that we choose. Essentially, the initial state is the input to the quantum computer.
- We carefully select a series of operations, that often correspond to multiplying a quantum state with a unitary matrix. The result of each multiplication corresponds to an intermediate quantum state.
- The combination of the initial state and the set of operations we choose correspond roughly to a definition of an algorithm, through quantum tools and nomenclature.
- The final state is the output of the system, that we measure and that we hope is the answer to the problem we want to solve.

1.3 Quantum computation

In order to understand how quantum algorithms work and how quantum mechanics is used to speed-up machine learning algorithms, we provide this section to describe the basic concepts of quantum mechanics, we tried to explain it mathematically. We follow [YM08].

A long time ago, physicists thought that the universe is made of tiny particles or consists of waves. But after, they discovered that all matter in the world has both particle-like and wave-like behavior. This is a somewhat counter-intuitive approach to the reality that's why quantum mechanics is considered to be counter-intuitive even for the physicists who deal with it every day.

In the macroscopic world, we know that:
(i) electricity may go or not through a circuit,
(ii) a proposition is true or false, etc.

The above can be easily described through a "bit": a bit quantify the information that we are in a particular state among two states. There are different ways to represent this information. Here, we will use the following representation of a bit:

$$|\text{state}\rangle = \begin{bmatrix}\text{state is } 0\\ \text{state is } 1\end{bmatrix}$$

Observe that in our macroscopic world, the state cannot be in both situations. More rigorously, let us represent the state 0 as:

$$|\mathbf{0}\rangle = \begin{bmatrix}1\\0\end{bmatrix}$$

and the state 1 as:

$$|\mathbf{1}\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

We can see that these two representations are orthogonal: Therefore, we are exclusively in one of the basic states 0 and 1. This situation change in the quantum context.

1.3.1 Qubit

The basic unit of quantum computation is the qubit, which is an elementary quantum system such as an atom or a nuclear spin with two basic physical states.

As we will see shortly, a quantum bit is actually a generalization of the classical bit. Using the notation above, a quantum bit can be described as:

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$$

where, as always, $|c_0|^2 + |c_1|^2 = 1$.

Here is some examples:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \cdot |\mathbf{0}\rangle + \frac{1}{\sqrt{2}} \cdot |\mathbf{1}\rangle = \frac{|\mathbf{0}\rangle + |\mathbf{1}\rangle}{\sqrt{2}}$$

and

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \cdot |\mathbf{0}\rangle - \frac{1}{\sqrt{2}} \cdot |\mathbf{1}\rangle = \frac{|\mathbf{0}\rangle - |\mathbf{1}\rangle}{\sqrt{2}}$$

Therefore the two states $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$ form a computational basis and any other state $|\psi\rangle$ of the qubit can be written as linear combination:

$$|\psi\rangle = c_0 \, |\mathbf{0}\rangle + c_1 \, |\mathbf{1}\rangle$$

where c_0 , $c_1 \in \mathbb{C}$ and $|c_0|^2 + |c_1|^2 = 1$. When the state $|\psi\rangle$ is measured, either $|\mathbf{0}\rangle$ or $|\mathbf{1}\rangle$ is observed, with probability $|c_0|^2$ or $|c_1|^2$ respectively. Equivalently, $|\psi\rangle$ is a column vector representing a quantum state living in some Hilbert space, here the space $\mathbb{C}$.

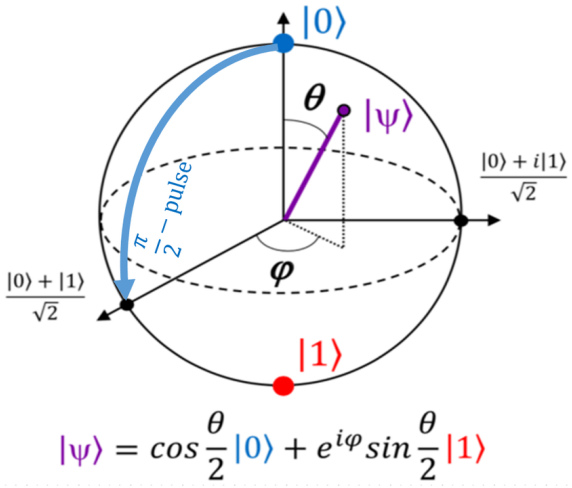


FIGURE 1.1: The Bloch sphere

By definition the qubits are connected to complex numbers, by representing the complex numbers on the unit complex sphere in three dimensions, the called the

Bloch sphere [Blo46], see Figure 1.1, we give a geometric explanation to single qubit operations.

Remark 1 *Parameters representation*

Consider the case of a quantum state: $|\psi\rangle = c_0 \cdot |\mathbf{0}\rangle + c_1 \cdot |\mathbf{1}\rangle$.

Reminder: $|c_0|^2 + |c_1|^2 = 1$, *where* $c_i \in \mathbb{C}$. *Using the polar representation of a complex number, we have the following equivalent representations:*

$$|\psi\rangle = r_0 e^{i\varphi_0} \cdot |\mathbf{0}\rangle + r_1 e^{i\varphi_1} \cdot |\mathbf{1}\rangle$$

where r_i *are the amplitudes and* φ_i *are the angles. Until here to describe the state* $|\psi\rangle$ *we need* 4 *parameters. As complex numbers, in the case of quantum bits, we know that a quantum state does not change if we multiply it with any number of unit norm; i. e.,* $|\psi\rangle = e^{-\varphi_0} |\psi\rangle$ *Then, our (equivalent) state is:*

$$e^{-i\varphi_0} \cdot |\psi\rangle = e^{-i\varphi_0} \cdot \left(r_0 e^{i\varphi_0} \cdot |\mathbf{0}\rangle + r_1 e^{i\varphi_1} \cdot |\mathbf{1}\rangle \right) = r_0 \cdot |\mathbf{0}\rangle + r_1 e^{j(\varphi_1 - \varphi_0)} \cdot |\mathbf{1}\rangle \tag{1.1}$$

From 4 parameters, now we end up with 3 parameters, which are: r_0, r_1 *and* φ *where* $\varphi := \varphi_1 - \varphi_0$.

We know that $|c_0|^2 + |c_1|^2 = 1$, *so we deduce that:* $r_0^2 + r_1^2 = 1$, *which allow to reduce the number of parameters to 2.*

Using the angle representation and setting $r_0 = \cos\frac{\theta}{2}$ *and* $r_1 = \sin\frac{\theta}{2}$, *we get:*

$$|\psi\rangle = \cos\frac{\theta}{2} \cdot |\mathbf{0}\rangle + e^{i\varphi} \sin\frac{\theta}{2} \cdot |\mathbf{1}\rangle ,$$

where $0 \leq \theta \leq \pi$ *and* $0 \leq \phi \leq 2\pi$, *these two parameters define a point on the surface of the Bloch sphere. In other words, a different interpretation of the above is that we care only about the relative phase between the two states, not their actual phase; moreover, without loss of generality, we can assume that one of the states has real coefficient.*

The notion of qubit has a natural extension, which is the quantum register. To define a quantum register of n-different qubits, let us denote $|\mathbf{e}_i\rangle$ the i-th vector of the canonical basis in $\mathbb{C}^n$. For suitable known complex weights $c_i, 1 \leq i \leq n$, satisfying $\sum_{i=1}^{n} |c_i|^2 = 1$, the quantum register register $|\psi\rangle$ has the following form:

$$|\psi\rangle = \sum_{i=1}^{n} c_i \, |\mathbf{e}_i\rangle ,$$

Since each state is described by two complex numbers c_i, the total number required to describe a n register qubit system is $\mathbb{C}^{2^n}$, a 2^n dimensional Hilbert space. Therefore, a quantum register from a mathematical point of view is a vector in $\mathbb{C}^{2^n}$.

To conclude this paragraph note that generally the classical computer store the information in the form of bits which can store either 1 or 0. From the remark above a qubit can store both 1 and 0 at the same time. So, 1 qubit = 2 bits, 2 qubits = 4 bits and n qubits = 2^n bits. Thus, n qubits can store 2^n bits information which is very high. This shows that quantum computer code information 2 times higher than classical computer.

1.3.2 Dirac notation

The state in quantum physics contains statistical information about a quantum system. As already mentioned it is represented by a vector in $\mathbb{C}^{2^n}$. In quantum mechanics

we use the Dirac notation [Dir39]: the vector ψ is denoted by $|\psi\rangle$ (ket notation) and represents column vectors. The dual of a ket is a bra. Mathematically, it represents a Hermitian conjugate of a ket, and thus it is a row vector defined by:

$$|\psi\rangle = \langle\psi|^T$$

The standard notation for inner product in the bra-ket notation is:

$$\langle\varphi'|\psi\rangle \equiv \left(\langle\varphi'|\right) \cdot \left(|\psi\rangle\right) = \begin{bmatrix} \varphi_1^T & \varphi_2^T & \dots & \varphi_n^T \end{bmatrix} \cdot \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{bmatrix}.$$

Similarly, the standard notation for the outer product is: $|\varphi\rangle\langle\psi| \in \mathbb{C}^{n\times n}$. Some properties: $\langle\varphi|^T = |\varphi\rangle$ and $|\varphi\rangle^T = \langle\varphi|$. Note that bras and kets can added and bras and kets can be multiplied by scalars following the same rules as complex vectors.

1.3.3 Superposition

In quantum physics, that is to say on the scale of the atom and the electron, certain particles can be in different states at the same time: it is the ability of a quantum system to be in multiple states at the same time until it is measured.

To be more precise let us introduce the following simplistic scenario: consider a set of discrete points as possible positions of some particle. Each point position determined by it's cartesian coordinates corresponds to a state of the particle. The particle being in the state $\mathbf{x}_i$ is going to be denoted with the Dirac "ket" notation $|\mathbf{x}_i\rangle$. Moreover, let us make the following association of states with vectors in the n-dimensional complex space:

$$|\mathbf{x}_i\rangle \longrightarrow \mathbf{e}_i^\top$$

where $\mathbf{e}_i$ is the canonical basis in this space. From the classical mechanics perspective, this is what we all need in order to be able to characterize where exactly the states of the particle lies in the macroscopic world, by observing and determining it's Cartesian coordinates. While in the microscopic quantum world, this is no longer true: we have to define all possible combinations of states as possible states. In particular, let $|\psi\rangle$ be an arbitrary state, that can be described as a linear combination of all the above states. For suitable known complex weights $c_1, c_2, \dots, c_n$, $|\psi\rangle$ can be written as:

$$|\psi\rangle = c_1|\mathbf{x}_1\rangle + c_2|\mathbf{x}_2\rangle + \cdots + c_n|\mathbf{x}_n\rangle. \tag{1.2}$$

In this case, given that we know the "basis" with respect to $\mathbf{x}_i$'s, we can represent the state $|\psi\rangle$ by just knowing the coefficients c_i. This state is called the "superposition" of all the basic states $|\mathbf{x}_i\rangle$. In the quantum context, we can say that $|\psi\rangle$ is on all states simultaneously.

Note that once we observe (measure) $|\psi\rangle$, the state collapses to one of the basic states. This state will be the one selected with probability $\frac{|c_i|^2}{||\psi\rangle|_2}$, for some i. In other words, the (normalized) modulus squares of the complex numbers c_i play the role of the probabilities of the state being collapsed to that basic state i.

From a mathematical point of view it's quite true: an electron is mathematically in several places at once, because we use probability calculations to know its position.

Each position is associated with a probability coefficient. From a physical point of view, there are two schools: the first one is the Copenhagen School, which says that quantum superposition should not be illustrated, because the quantum superposition must not be interpreted physically. The phenomenon must remain a mathematical concept. the second one is Everett's School which tells us that concretely there may be parallel universes for each a superposed state.

Note that in order to write n qubit system 2^n complex numbers are required, whereas on classical computer only n zeros or ones are required to fully describe n bit system. The effect increases exponentially with the number of qubits, much more n is big much complex numbers are needed to emulate quantum state on a classical machine. Superposition of states is one of the two quantum mechanical effects that quantum algorithms exploit to outperform classical algorithms. The second one effect is the so called entanglement which is described in the next paragraph.

1.3.4 Entanglement

Entanglement is a type of correlation that has no classical analog, that's why, it's not easy to simulate it classically and therefore it creates a quantum advantage comparing to the classical version.

In the previous paragraph, we limited the presentation to the case of a single particle which is in one of n different states in n-dimensional space. We can use this simple case to assembly and model multi-particle quantum systems. For example, let's consider the case where we have particles instead of only one particle. Suppose that we have two particles that lies in two different sets of points in the space.

Assuming we have the basic states $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathcal{Y} = \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m\}$ we can create the assembly of states by using the tensor product symbol $\mathcal{X} \otimes \mathcal{Y}$, where the possible states are in the set:

$$\{|\mathbf{x}_i\rangle \otimes |\mathbf{y}_j\rangle;\ \text{forall } 0 \leq i \leq n, \text{and for all } 0 \leq j \leq m\}$$

Here, $|\mathbf{x}_i\rangle \otimes |\mathbf{y}_j\rangle$ means that the first particle is in state i of $\mathcal{X}$ and the second particle is in state j of $\mathcal{Y}$. Using all possible combinations, we can write down the state of two particles as a superposition of the basic states such as:

$$|\psi\rangle = c_{0,0} \cdot |\mathbf{x}_0\rangle \otimes |\mathbf{y}_0\rangle + \cdots + c_{n,m} \cdot |\mathbf{x}_n\rangle \otimes |\mathbf{y}_m\rangle. \tag{1.3}$$

So far we indicate that the state of an assembly of simpler quantum systems can be described as the tensor product of the states of each individual system. i. e., we can describe the overall system by looking into the constituents. However, this is not always true for quantum systems. Here, we will just describe this via an example.

Example 1 *Let us consider $n = m = 2$ and assume that the current state $|\psi\rangle$ satisfies:*

$$\begin{aligned} |\psi\rangle &= |\mathbf{x}_0\rangle \otimes |\mathbf{y}_0\rangle + |\mathbf{x}_1\rangle \otimes |\mathbf{y}_1\rangle \\ &= 1 \cdot |\mathbf{x}_0\rangle \otimes |\mathbf{y}_0\rangle + 0 \cdot |\mathbf{x}_0\rangle \otimes |\mathbf{y}_1\rangle + 0 \cdot |\mathbf{x}_1\rangle \otimes |\mathbf{y}_0\rangle + 1 \cdot |\mathbf{x}_1\rangle \otimes |\mathbf{y}_1\rangle . \end{aligned}$$

Let us see whether we can describe the above state as the tensor product of the basic states of single particles. Remember that the state using only the first particle can be described as: $c_0 \cdot |\mathbf{x}_0\rangle + c_1 \cdot |\mathbf{x}_1\rangle$, and the state using only the second particle can be described as: $c'_0 \cdot |\mathbf{y}_0\rangle + c'_1 \cdot |\mathbf{y}_1\rangle$. Let us create the states of the two particle system by "assembling the simple systems: i. e., we compute the tensor product of their states".

In particular, we have:

$$(c_0 \cdot |\mathbf{x}_0\rangle + c_1 \cdot |\mathbf{x}_1\rangle) \otimes (c'_0 \cdot |\mathbf{y}_0\rangle + c'_1 \cdot |\mathbf{y}_1\rangle) = \\ c_0c'_0 \cdot |\mathbf{x}_0\rangle \otimes |\mathbf{y}_0\rangle + c_0c'_1 \cdot |\mathbf{x}_0\rangle \otimes |\mathbf{y}_1\rangle + c_1c'_0 \cdot |\mathbf{x}_1\rangle \otimes |\mathbf{y}_0\rangle + c_1c'_1 \cdot |\mathbf{x}_1\rangle \otimes |\mathbf{y}_1\rangle .$$

We need to choose the coefficients in order to create $|\psi\rangle$. However, this means that:

$$c_0c'_0 = c_1c'_1 = 1$$

and

$$c_0c'_1 = c_1c'_0 = 0.$$

These two equations have no solution: therefore we cannot use the basic states and the tensor product to generate the current state.

Following this example, Let us consider the state using only the first data vector, it can be described as:

$$|\mathbf{x}\rangle = c_0 \cdot |\mathbf{x}_0\rangle + c_1 \cdot |\mathbf{x}_1\rangle + \cdots + c_n \cdot |\mathbf{x}_n\rangle,$$

and the state using only the second data vector can be described as:

$$|\mathbf{y}\rangle = c'_0 \cdot |\mathbf{y}_0\rangle + c'_1 \cdot |\mathbf{y}_1\rangle + \cdots + c'_m \cdot |\mathbf{y}_m\rangle.$$

Let us compute the tensor product of the states.

In particular, we have:

$$(c_0 \cdot |\mathbf{x}_0\rangle + c_1 \cdot |\mathbf{x}_1\rangle + \cdots + c_n \cdot |\mathbf{x}_n\rangle) \otimes (c'_0 \cdot |\mathbf{y}_0\rangle + c'_1 \cdot |\mathbf{y}_1\rangle + \cdots + c'_m \cdot |\mathbf{y}_m\rangle) \quad (1.4)$$

In order to determine the states of the two data vectors by "assembling" the simple systems we need to choose the coefficients in order to create $|\psi\rangle$. However, this means that the system of equations formed by identification of the coefficients in Equation (1.3) with the coefficients in Equation (1.4) is completely determined and has always a solution. This is not always the case, as in our Example, therefore we cannot use the basic states and the tensor product to generate the state $|\psi\rangle$. In this case, we say that the two data states $|\mathbf{x}\rangle$ and $|\mathbf{y}\rangle$ are "entangled". In other words: an entangled state can not be written as a tensor product of its constituent qubits states or an entangled state is any state that cannot be tensor-factorized. On the other hand, if two $|\mathbf{x}\rangle$ and $|\mathbf{y}\rangle$ are viewed as a single quantum state, this combined state is equal to the tensor product of the constituent qubits. Any state that can be written as a tensor product from the constituent subsystems are called separable states.

How is entanglement used

To see what it means in practice, let us consider the following: Assume we measure the first particle and we find it in position $\mathbf{x}_0$. Can we use this information to infer the state of the other particle?

Yes. We can see that the component $|\mathbf{x}_0\rangle \otimes |\mathbf{y}_1\rangle$ has coefficient 0 in the expression of the state of the two particles. This means that, if we found first particle in $\mathbf{x}_0$, we will find the second particle in state $\mathbf{y}_0$. The surprising fact is that the two particles could be light years away and they can still infer the state of the individual particles by observing one of them. To wrap it up: if we can infer some information of some particles by looking some others, we say that the particles are entangled. If this is not the case, then the particles are separable.

Therefore entanglement describes the way that particles can become correlated to predictably interact with each other regardless of how far apart they are. Particles, such as photons, electrons, or qubits that have interacted with each other retain a type of connection and can be entangled with each other in pairs, in the process known as correlation. As we already mentioned, knowing the spin state of one entangled particle allows one to know that the spin of its mate is in the opposite direction. For example, when two spinning particles are entangled their spins are correlated not only in one direction but in all directions. Then no matter how great the distance between the correlated particles, they will remain entangled as long as they are isolated.

1.3.5 Dynamics of a quantum system

Let us introduce the notion of time in a quantum system, through the sequence of operators we apply to it. The evolution of a quantum system over time is necessarily unitary. Geometrically, a unitary transformation is a rigid body rotation of the Hilbert space, therefore resulting in a transformation of the state vector that doesn't change its length. We differentiate the notion of observables from the notion of evolution of a quantum system.

As an example: if our system lies at state $|\psi_t\rangle$ at the time t and at state $|\psi_{t+1}\rangle$ at the time $t+1$, this transition can be described as:

$$|\psi_{t+1}\rangle = \mathbf{U}|\psi_t\rangle$$

for some unitary matrix $\mathbf{U}$ with the dimensions are given by the context. The difference between observables and generic unitary transformations is that the latter satisfy:

- the product of two unitary matrices is always unitary (in any order);
- the inverse of a unitary matrix is unitary.

These properties are important because they allow "traversing" the quantum system states over time. That is, we can apply sequentially:

$$\mathbf{U}_0|\psi\rangle,\ \mathbf{U}_1\mathbf{U}_0|\psi\rangle, \ldots,\ \prod_{i=0}^{T}\mathbf{U}_i|\psi\rangle$$

to get the evolution. The key property of unitary transformations is that they preserve the energy of the quantum system. Let us revisit the rough description of quantum computing: we start at a state and we apply a sequence of carefully selected unitary transformations that reflects the algorithm we want to implement. Then, when we are done, we measure the output to get the final state.

The discussion of how to select the unitary transformations involve the introduction of the Schrodinger equation.

1.3.6 Schrodinger equation

A basic postulate of quantum mechanics is that a closed system evolves in time via the Schrodinger equation. It is the most fundamental equation in quantum mechanics, the equation is named after Erwin Schrodinger, who won the Nobel Prize along with Paul Dirac in 1933 for their contributions to quantum physics.

Schrodinger's equation describes the wave function of a quantum mechanical system, which gives probabilistic information about the location of a particle and other observable quantities such as its momentum.

The Schrodinger equation describes the evolution of a quantum state in a similar way to Newton's laws in classical mechanics. It is considered as a wave equation for the wave function of the particle in question, and so the use of the equation to predict the future state of a system is sometimes called wave mechanics. The wave function is one of the most important concepts in quantum mechanics, because every particle is represented by a wave function ψ and it depends on position and time.

The equation itself derives from the conservation of energy and is built around an operator called the Hamiltonian. In its simplest form, it's written like that:

$$i\hbar \frac{d\,|\psi(t)\rangle}{dt} = H\,|\psi(t)\rangle \tag{1.5}$$

where H is the Hamiltonian of the system, which is some Hermitian operator and $\hbar$ is the reduced Planck constant ($\hbar = \frac{h}{2}\pi$ with h the Planck constant, allowing conversion from energy to frequency units). We know that the global energy is preserved in an isolated system, throughout its evolution. Energy is an observable and, for a quantum system, we can use a Hermitian matrix to represent it. This matrix is also known as the hamiltonian matrix. Then, the Schrodinger equation describes that the rate of variation from a state to another is proportional to the energy expressed via the hamiltonian matrix. Solving this equation with some initial conditions, we get the evolution.

The wave function is one of the most important concepts in quantum mechanics, because every particle is represented by a wave function ψ and it depends on position and time. When you have an expression for the wave function of a particle, it tells you everything that can be known about the physical system, and different values for observable quantities can be obtained by applying an operator to it.

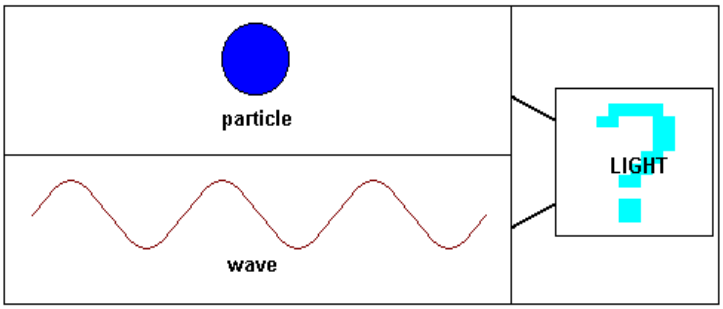


FIGURE 1.2: Particle as a wave

For a time-independent Hamiltonian the solution to the Schrodinger equation is

$$|\psi(t)\rangle = e^{-iHt}\,|\psi(0)\rangle$$

with $|\psi(0)\rangle$ the initial state of the system. This is a unitary evolution, which follows easily from one of the defining properties of a Hermitian operator: $\overline{H}^T = H$

1.3.7 Quantum gates

A quantum gate or quantum logic gate is an elementary quantum circuit that operates on a small number of qubits. For quantum computers, these are analogous to the classical logic gates for conventional computers.

For example: consider the classic NOT gate. i.e., the input is a single classical bit; the output is its inverse. Using ket notation, let us use as input $|\mathbf{0}\rangle$. Then, the NOT gate can be represented as:

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{\text{NOT gate}} \cdot |\mathbf{0}\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |\mathbf{1}\rangle$$

Similar logic applies for other gates such as AND, NOR, XOR, NAND. In all cases, we can represent the application of the gate on an input as a matrix-vector product. For the gates with an input more than a single bit, we can represent the input as the vertical concatenation of bits. Finally, when we have more than one gate in sequence, we can represent the sequence of matrices as a sequence of matrix-matrix multiplications, before we apply the input. Also, gates in parallel can be represented as Kronecker products of matrices.

A qubit does more than just sit around in the $|\mathbf{0}\rangle$ ground state. To put it into the $|\mathbf{1}\rangle$ excited state, we need a quantum gate.

Quantum gates, or operations, are typically represented as matrices. A gate that acts on one qubit is represented by a 2×2 unitary matrix. Since quantum operations must be reversible and preserve probability amplitudes, the matrices must be unitary. The result of the quantum gate is found by multiplying the matrix representing the gate with the vector representing the quantum state.

$$|\psi'\rangle = U\,|\psi\rangle \ , \quad \text{where } UU^T = 1$$

Pauli matrices are a set of three 2×2 complex matrices which are Hermitian and unitary.

$$X \equiv \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \ Y \equiv \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \ Z \equiv \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Note that since each Pauli matrix is a Hermitian matrix this ensure the information preservation, and together with the identity matrix form a basis for the space of 2×2 Hermitian matrices.

The Pauli X gate is known as an $X_\pi$rotation. It flips the zero to a one, or vice versa (this is why it is also commonly referred to as a bit-flip). Y gave you an excited state and Z did not do anything.

Here, we will present the most useful quantum gates (that we use it in our algorithms): Hadamard gate for superposition and Control-SWAP gate for entanglement.

Hadamard gate

This is one of the most used quantum gates. It can be used to convert the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, thus creating a superposition of the two states. The schematic representation of Hadamard gate is presented in Figure 1.3.

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

and

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

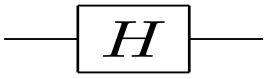


FIGURE 1.3: Hadamard gate

Note that two Hadamard gates in series restore a qubit to its original state.

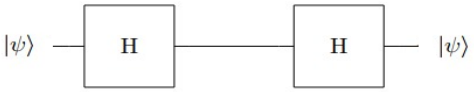


FIGURE 1.4: Hadamard gate in series

By using 3 Hadamard gates in parallel we define the Hadamard transform

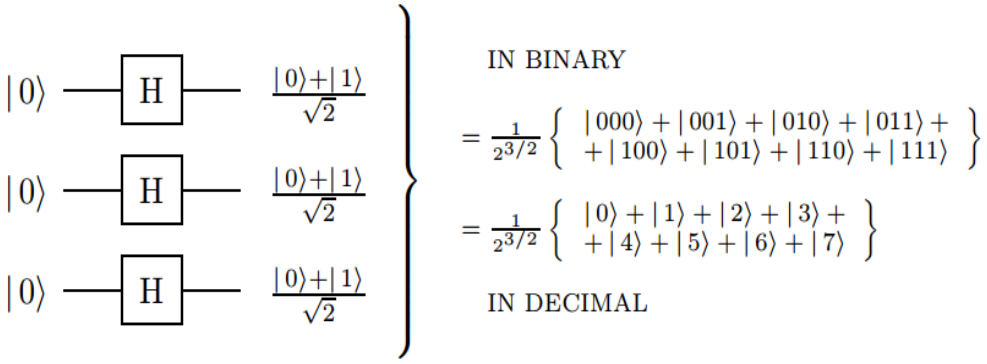


FIGURE 1.5: Hadamard gate in parallel

U3 gate

Let us define the generator matrix $U_3(\theta, \phi, \lambda)$, where:

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix}\cos(\theta/2) & -e^{j\lambda}\sin(\theta/2)\\ e^{j\phi}\sin(\theta/2) & e^{j(\phi+\lambda)}cos(\theta/2)\end{bmatrix}$$

The U_2 gate is the U_3 gate with $\theta = \pi/2$. The U_1 gate has $\theta = \phi = 0$. Then, the Hadamard matrix is $H = U_3(\pi/2, 0, \pi)$, $S = U_3(0, 0, \pi/2)$, and $T = U_3(0, 0, \pi/4)$. All the above operations are connected through the following properties:

- $X^2 = Y^2 = Z^2 = I$,
- $H = \frac{1}{\sqrt{2}}(X + Z)$,
- $X = HZH$,
- $Z = HXZ$,
- $S = T^2$.

Controlled-NOT gate

The C-Not gate operates on a quantum register consisting of two qubits (Figure 1.6), the C-Not gate flips the second qubit if and only if the first qubit is in the state $|1\rangle$. The schematic representation of C-Not is as follows:

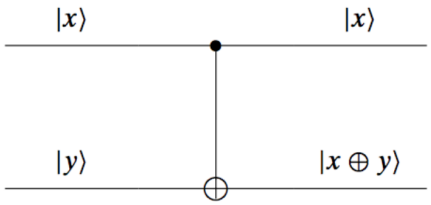


FIGURE 1.6: C-Not gate

This gate operates as follows: the top input goes through the gate as it is, and we observe it at the output. However, the top signal "controls" the bottom signal: if the top signal is $|0\rangle$, then the input $|y\rangle$ will be output as it is; if the top signal is $|1\rangle$, then the gate reverses the bottom signal. We can see that the operation on the bottom signal is similar to an XOR gate, that's why we use the symbol $\bigoplus$. The C-Not gate can be represented by the matrix :

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

In general, the inputs are allowed to be a linear superposition of $\{|0\rangle, |1\rangle\}$. The C-Not gate transforms the quantum state:

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle \quad \text{into} \quad a|00\rangle + b|01\rangle + c|11\rangle + d|10\rangle$$

Swap gate

The Swap gate represented in Figure 1.7 is two-qubit operation. Expressed in basis states, the Swap gate swaps the state of the two qubits involved in the operation. With respect to the basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, it is represented by the matrix:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

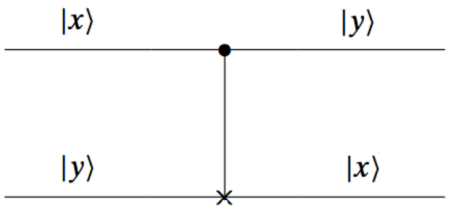


FIGURE 1.7: Swap Gate

CSWAP gate (Fredkin gate)

The Fredkin gate has three input qubits, but only one is for control. If the control bit is set to one, the target qubits are swapped, otherwise the state is not modified.

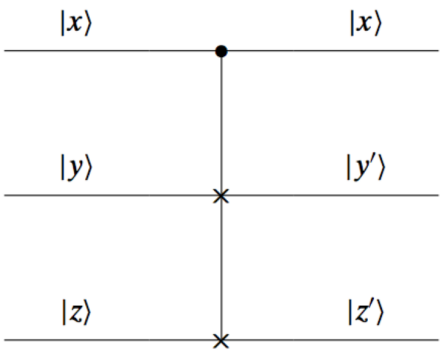


FIGURE 1.8: Fredkin gate

Only the top input operates as a controller:
if it is $|0\rangle$, then the other two inputs output their values as they are;
if it is $|1\rangle$, then the outputs of the two other signals are "switched": we see $|y'\rangle = |z\rangle$ and $|z'\rangle = |y\rangle$.

1.3.8 EPR paradox and Bell states

Bell states

We can generate the Bell states $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ with the following quantum circuit consisting of a Hadamard and CNOT gate [SL09]:

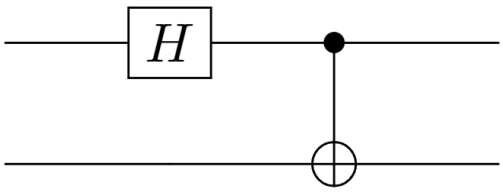


FIGURE 1.9: Quantum circuit of Bell states

The first qubit is passed through a Hadamard gate and then both qubits are entangled by a CNOT gate. If the input to the system is $|0\rangle \otimes |0\rangle$, then the Hadamard gate changes the state to

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

And after the CNOT gate the state becomes the Bell state: $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

Notice that the action of the CNOT gate is not so much copying, as our classical intuition would suggest, but rather to entangle. The four Bell basis states are maximally entangled states on two qubits, they form an orthonormal basis for $\mathbb{C}^4$:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$
$$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$
$$\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

EPR Paradox

Created in 1935 by Albert Einstein, Boris Podolsky, and Nathan Rosen, hence "EPR" [Sch35]. The Einstein's famous quote "God does not play dice with the Universe", is a summary of the following passage from Einstein's 1926 letter to Max Born: "Quantum mechanics is certainly imposing. But an inner voice tells me that it is not yet the real thing. The theory says a lot, but does not really bring us any closer to the secret of the Old One. I, at any rate, am convinced that He does not throw dice." Even until the end of his life, Einstein maintained the idea that quantum physics is an incomplete theory and that one day we would learn a more complete and satisfying theory that describes nature.

In what sense did Einstein consider quantum mechanics to be incomplete? To better understand this, imagine that we formulate a theory which would explain the act of throwing a coin. A simple heads or tails model is that its result is random (head 50 % of the time and head 50 % of the time). This model seems to be in perfect harmony with our experience of throwing a coin, but it is incomplete. A more complete theory would say that if we could determine the initial conditions of the coin with perfect precision (position, momentum), we could solve Newton's equations to determine with certainty the final result of the coin. The coin flip amplifies our lack of knowledge about the initial conditions and makes the result completely random. Likewise, Einstein believed that the randomness of the quantum measurement results reflected our lack of knowledge about the additional degrees of freedom of the quantum system.

Einstein refined this line of reasoning in an article he wrote with Podolsky and Rosen in 1935, where they presented the famous Bell states. Remember that for Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, when you measure the first qubit, the second qubit is determined. However, if two qubits are distant, the second qubit must have had a determined state within a certain time interval before the measurement, as the speed of light is finite. By the rotational symmetry of the Bell state, this fact is verified on all basis. This seems analogous to the example of the coin flipping. Therefore, EPR suggested that there is a more complete theory where "God does not throw dice". Until his death in 1955, Einstein tried to formulate a more complete "theory of local hidden variables" that would describe the predictions of quantum mechanics, but without reclassifying it as probabilistic results. However in 1964, almost three decades after the publication of the EPR, John Bell showed that the properties of Bell states (EPR) were not just fodder for philosophical discussion, but had verifiable consequences: the local hidden variables are not the answer. He showed that there exists a particular experiment which could be carried out on two qubits entangled in Bell states, so that no theory of the local hidden variables could not correspond to the result envisaged by quantum mechanics. Bell's experiment was performed with increasing precision, and the results have always been consistent with the predictions of quantum mechanics and incompatible with local theories of hidden variables.

1.3.9 The non-cloning theorem

The non cloning theorem [WZ82] is a result of quantum mechanics which forbids the creation of identical copies of an arbitrary unknown quantum state. It was stated by Wootters, Zurek, and Dieks in 1982, and has profound implications in quantum computing and related fields. The theorem follows from the fact that all quantum operations must be unitary linear transformation on the state (and potentially an ancilla).

The idea is that a cloning machine can accept an arbitrary input state and copy it with the same fidelity for all possible states.

Quantum cloning is the process that takes an arbitrary, unknown quantum state and makes an exact copy without altering the original state in any way. Quantum cloning is forbidden by the laws of quantum mechanics as shown by the no cloning theorem.

In contrast, the no cloning theorem is a vital ingredient in quantum cryptography, as it forbids eavesdroppers from creating copies of a transmitted quantum cryptographic key.

Fundamentally, the no-cloning theorem protects the uncertainty principle in quantum mechanics. If one could clone an unknown state, then one could make as many copies of it as one wished, and measure each dynamical variable with arbitrary precision, thereby bypassing the uncertainty principle. This is prevented by the non-cloning theorem.

Though perfect quantum cloning is not possible, it is possible to perform imperfect cloning, where the copies have a non-unit fidelity with the state being cloned. This can be done by coupling a larger auxiliary system to the system that is to be cloned, and applying a unitary transformation to the combined system. If the unitary transformation is chosen correctly, several components of the combined system will evolve into approximate copies of the original system. Imperfect cloning can be used as an eavesdropping attack on quantum cryptography protocols, among other uses in quantum information science.

Cloning means making an exact copy, where two identical systems result, the original and the copy. Consider a single particle in state $|\phi\rangle$ that is cloned, leaving the two-particle state $|\phi\rangle\,|\phi\rangle$. This state does not make sense because two particles are prohibited from occupying the same state.

Proof

Suppose we have a quantum bit in an unknown state, and we wish to make a copy of this sate:

$$|\psi\rangle = c_1\,|\mathbf{x}_1\rangle + c_2\,|\mathbf{x}_2\rangle\,. \tag{1.6}$$

Is it possible to make a copy of this quantum state? In other words, we are asking whether it is possible to start with two qubits in state $|\psi\rangle \otimes |0\rangle$ and transform them to the state $|\psi\rangle \otimes |\psi\rangle$.

To do so, we need to have a unitary transformation U such that:

$$U\,|\psi\rangle \otimes |0\rangle = |\psi\rangle \otimes |\psi\rangle\,.$$

The no cloning theorem says that this isn't physically possible. Only sets of mutually orthogonal states can be copied by a single unitary operator. We will show that no unitary transformation can achieve this simultaneously for two orthogonal states $|\psi\rangle$ and $|\phi\rangle$.

Recall that a unitary transformation is a rotation of the Hilbert space, and therefore necessarily preserves angles.

Consider that we have the two quantum states $|\psi\rangle$ and $|\phi\rangle$, such that:

$$|\psi\rangle = c_1 |\mathbf{x}_1\rangle + c_2 |\mathbf{x}_2\rangle$$

and

$$|\phi\rangle = c_1' |\mathbf{x}_1\rangle + c_2' |\mathbf{x}_2\rangle .$$

The cosine of the angle between them is given by (the absolute value of) their inner product: $c_1 c_1' + c_2 c_2'$.

Thus, the inner product of $\langle\psi|\phi\rangle^2$ is $(c_1 c_1' + c_2 c_2')^2$.

Assume we have a unitary operator U and the two quantum states $|\psi\rangle$ and $|\phi\rangle$:

$$|\psi\rangle \otimes |0\rangle \longrightarrow |\psi\rangle \otimes |\psi\rangle$$

$$|\phi\rangle \otimes |0\rangle \longrightarrow |\phi \otimes |\phi\rangle\rangle$$

Then $\langle\psi|\phi\rangle$ is 0 or 1.

$$\langle\psi|\phi\rangle = (\langle\psi| \otimes \langle 0|)(|\phi\rangle \otimes |0\rangle) = (\langle\psi| \otimes \langle\psi|)(|\phi\rangle \otimes |\phi\rangle) = \langle\psi|\phi\rangle^2 \tag{1.7}$$

1.3.10 Quantum parallelism

In the world of classical computers, parallelism refers to parallel (simultaneous) processing of different information in different processors. Quantum parallelism refers to simultaneous processing of different information or inputs in the same processor.

The quantum speed-up of algorithm comes from the quantum parallelism obtained by the superposition of qubits. The quantum parallelism is a feature of many quantum algorithms and in simple words it allows to evaluate the function on many inputs simultaneously. Quantum parallelism arises from the ability of a quantum memory register to exist in a superposition of base states. Each component of this superposition may be thought of as a single argument to a function. A function performed once on the register in a superposition of states is performed on each of the components of the superposition. Since the number of possible states is 2^N where N is the number of qubits in the quantum register, you can perform in one operation on a quantum computer what would take an exponential number of operations on a classical computer, which is amazing.

Suppose that f is a binary function:

$$f : \{0,1\}^n \to \{0,1\}$$

and also suppose that O is a reversible quantum operator that performs the following operation:

$$O|x\rangle|0\rangle = |x\rangle|f(x)\rangle$$

Applying O to a uniform quantum superposition performs the following:

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|0\rangle \rightarrow \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|f(x)\rangle$$

in a single computational step.

A quantum computer is able to encode all input strings of length N simultaneously into a single computational step. In other words, the quantum computer is able to pursue simultaneously 2^N classical paths, indicating that a quantum computer is significantly more powerful than a classical one.

1.3.11 Quantum Random Access Memory

Memory consists of cells that can store information. Each cell has an address, or word-line, to find the cell location. A memory with n cells has an address of 0 to $n-1$ bits. All cells contain the same amount of bits for the address.

A random access memory allows addressing memory cells in a classical computer: it is an array in which each cell of the array has a unique numerical address.

In order to access memory in a quantum system there needs to be a quantum equivalent to classical RAM. This quantum equivalent is referred to as QRAM. A quantum random access memory (QRAM) serves a similar purpose [GLM08]. A random access memory has an input register to address the cell in the array, and an output register to return the stored information. In a QRAM, the address and output registers are composed of qubits. The address register contains a superposition of addresses $\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle$, and the output register will contain a superposition of information, correlated with the address register: $\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle\,|v_j\rangle$.

$$\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle \rightarrow \frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle\,|v_j\rangle$$

The output register is entangled with input register.

While a random access memory (RAM) uses n bits to randomly address $N = 2^n$ distinct memory cells, a quantum random access memory (QRAM) uses n qubits to address any quantum superposition of N memory cells.

1.3.12 Measurements

Until so far we discussed that the collection of states of a system could be at some fixed time. We recall now the notion of observables which correspond to the physical quantities that can be observed in each state of the state space. More precisely assuming that the system is in the state $|\psi\rangle \in \mathbb{C}^n$, let us define an observable operator $\mathbf{O} \in \mathbb{C}^{n\times n}$. The action of the observable is denoted as $\mathbf{O}|\psi\rangle$ and it is by itself a state.

In quantum mechanics, each observable-linear operator is Hermitian. Therefore as Hermitian operator has real eigenvalues. Then, the eigenvalues of an observable are the only possible values the observable can take, as a result of measuring it on a given state. In other words, given a state and assuming the observable is seen as a an action on the system, the eigenvalues of the observable are the possible outputs.

The act of measuring is about an action that gives a definite answer. So far, through observables, we can only assume that one of the eigenvalues will be the result of an observable operator. Further, even for this case, we are not sure which of the eigenvalues we will observe. Finally, our framework so far does not tell us anything about what happens if we actually observe the eigenvalues.

Unlike the macroscopic world, quantum systems get perturbed and modified by the time we measure them. Further, only the probability of observing specific values can be calculated. It turns out that the following holds: Let **O** be an observable and $|\psi\rangle$ be a state of the system. If the result of measuring **O** is its eigenvalue λ, then the state, after the measurement, is collapsed to the eigenvector of **O** that corresponds to the eigenvalue λ, that we will "land" (as the next state). For the specific eigenvector $|\mathbf{e}\rangle$ the probability is given by $|\langle \mathbf{e}_i|\psi\rangle|^2$.

The state vector evolves deterministically as the continuous solution of the wave equation. Meanwhile, the state vector is in a superposition of component states. However, once we measure the state it looses this superposition and we return to the classical version.

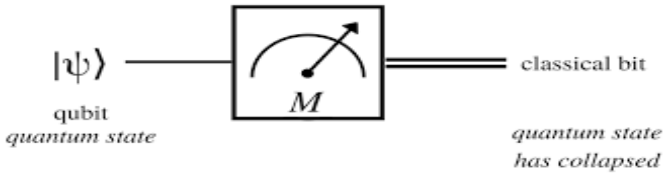


FIGURE 1.10: Measurement

Therefore measurements are irreversible actions because when we perform a measurement the state of the system collapses to whichever value ($|\mathbf{0}\rangle$ or $|\mathbf{1}\rangle$) has been observed, thus losing all memory of former amplitudes c_0 and c_1 [FF13].

With the exception of measurements, all other operations allowed by quantum mechanics are unitary operations on the Hilbert space in which our qubits live. They are represented by quantum gates, much as in a classical circuit.

1.4 Quantum Algorithms

Quantum algorithm is an algorithm which can be performed on a quantum computer and achieve a speedup, or other efficiency improvement, over any possible classical algorithm. In principle it is possible to run all classical algorithms on a quantum computer. Quantum algorithms use at least one of the steps is distinctly "quantum", using superposition or entanglement.

The advantage of a quantum computer is that it can do things which cannot be done by any machine based only on the laws of classical physics. Eventual applications of quantum computing range from breaking cryptographic systems to the design of new medicines.

Although large-scale quantum computers do not yet exist, the theory of quantum algorithms has been an active area of study for over 20 years. Here we aim to give a broad overview of quantum algorithmics, by giving some examples of quantum algorithms such as: Shor, Deutsch Josza, Grover and Simon's algorithm. In what follows, we will explain each of these algorithms in details.

1.4.1 Shor's algorithm

In the early 1990s, it was known that a quantum computer could be more efficient than any conventional computer for certain tasks in the Complexity-Theoretical Church-Turing thesis. However, surveys in these areas have been largely motivated

by academic curiosity. There were not many economic reasons for people to spend a lot of money or time trying to build a quantum computer.

That has been changed in 1994, when Peter Shor, a scientist working for Bell Labs, devised a polynomial time algorithm to factor large numbers on a quantum computer. The algorithm was considered as an important task as the difficulty of factoring large numbers is relied upon for many cryptographic systems. If an efficient method of factoring large numbers is implemented, most of many encryption schemes would be practically useless to protect their data from prying eyes. Although it has not been proven that factoring large numbers can not be archived on a conventional computer in polynomial time, in 2015 the fastest publicly available algorithm for factoring large numbers runs in $\mathcal{O}(\exp(\frac{64}{9}n^{/3}(\log n)^{2/3})$ operations, where n is the number of bits used to represent the number: this runtime exceeds polynomial time. On the contrary of Shor's algorithm which runs in $\mathcal{O}((\log n)^2 \log n)$ on a quantum computer, and then must perform $\mathcal{O}(\log n)$ steps of post processing on a classical computer. Overall then this time is polynomial. This discovery propelled the study of quantum computing forward, because such an algorithm is in high demand.

Shor's algorithm [Sho94] is based on a result of number theory. This result is:

The function $\mathcal{F}(a) = x^a \mod n$ is a periodic function, where x is an integer coprime to n. In the context of Shor's algorithm n will be the number we wish to factor. When two numbers are coprime it means that their greatest common divisor is 1. Calculating this function for an exponential number of a would take exponential time on a classical computer. Shor's algorithm utilizes quantum parallelism to perform the exponential number of operations in one step.

The reason why this function is useful for factoring large numbers is as follows:

Since $\mathcal{F}(a)$ is a periodic function, it has some period r. We know that $x^0 mod \text{n}= 1$, so $x^r mod \text{n}= 1$, and $x^{2r} mod \text{n}= 1$ and so on since the function is periodic. Given this information and through the following algebraic manipulation:

$$x^r \equiv 1 \mod n \tag{1.8}$$

$$(x^{r/2})^2 = x^r \equiv 1 \mod n \tag{1.9}$$

$$(x^{r/2})^2 - 1 \equiv 0 \mod n \tag{1.10}$$

and if r is an even number

$$(x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \mod n \tag{1.11}$$

We can see that the product $(x^{r/2} - 1)(x^{r/2} + 1)$ is an integer multiple of n, the number to be factored. So long as $x^{r/2}$ is not equal to ± 1, then at least one of $(x^{r/2} - 1)$ or $(x^{r/2} + 1)$ must have a nontrivial factor in common with n. So by computing $gcd(x^{r/2} - 1, n)$, and $gcd(x^{r/2} + 1, n)$, we will obtain a factor of n, where gcd is the greatest common denominator function.

Shor's algorithm tries to find r, the period of $x^a \mod n$, where n is the number to be factored and x is an integer coprime to n. To do this, Shor's algorithm creates a quantum memory register with two parts. In the first part, the algorithm places a superposition of the integers which are to be a's in the $x^a \mod n$ function. We will choose our a's to be the integers 0 through $q - 1$, where q is the power of two such that $n^2 \leq q < 2n^2$. Then the algorithm calculates $x^a \mod n$, where a is the superposition of the states, and places the result in the second part of the quantum memory register.

Next the algorithm measures the state of the second register, the one that contains the superposition of all possible outcomes for $x^a \mod n$. Measuring this register has the effect of collapsing the state into some observed value, say k. It also has the

side effect of projecting the first part of the quantum register into a state consistent with the value measured in the second part. Since we have partitioned our quantum register into two parts measurement of the second part collapses that part into exactly one value, while the other partition collapses into a state consistent with the observed value in the other portion of the register. It is still possible for the non measured part of the register to exist in a superposition of base states, as long as each of those base states are consistent with the measured value in the other part of the register. What this means in this instance is that after this measurement the second part of the register contains the value k, and the first part of the register contains a superposition of the base states which when plugged into $x^a \mod n$ produce k. Since we know $x^a \mod n$ is a periodic function, we know that the first part of the register will contain the values $c, c+r, c+2r...$ and so on, where c is the lowest integer such that $x^c \mod n = k$

The next step is to perform a discrete Fourier transform on the contents of the first part of the register (the one containing all integers such that $x^a \mod n = k$) and to put the result back into register one. The application of the discrete Fourier transformation has the effect of peaking the probability amplitudes of the first part of the register at integer multiples of the quantity q/r.

Now measuring the first part of the quantum register will yield an integer multiple of the inverse period. Once this number is retrieved from the quantum memory register, a classical computer can do some analysis of this number, make a guess as to the actual value of r, and then compute the possible factors of n.

1.4.2 Deutsch-Jozsa algorithm

A simple problem

Let us consider the following problem:

There is a univariate function f, defined over the binary alphabet $\{0,1\}$, with output range in the same alphabet $\{0,1\}$. i. e., $f : \{0,1\} \longrightarrow \{0,1\}$. Without assumptions about f, there are four possible configurations for the function f:

- Both inputs $0,1$ are mapped to the output 0: $f(0) = f(1) = 0$.
- Both inputs $0,1$ are mapped to the output 1: $f(0) = f(1) = 1$.
- Inputs pass through f unchanged: $f(0) = 0$ and $f(1) = 1$.
- Inputs are exchanged after passing through f: $f(0) = 1$ and $f(1) = 0$.

We will consider that the first two cases f as being "constant" (i. e., whatever input we insert, the function behaves the same way), and the last two cases of f as being "balanced". The problem is the following [Cle+98]:

Given a function $f : \{0,1\} \longrightarrow \{0,1\}$ and without knowing anything more than that, determine whether f is a constant or a balanced function with the minimum number of function evaluations.

Let us consider the following circuit in the diagram:

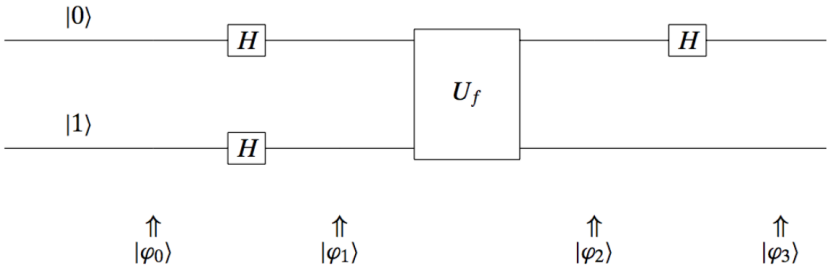


FIGURE 1.11: Deutsch Jozsa circuit

We start with the state:

$$|\varphi_0\rangle = |0,1\rangle.$$

Both inputs go through a Hadamard gate, which gives:

$$|\varphi_1\rangle = |H\cdot 0, H\cdot 1\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$

In the next step, we go through the function f (remember that this function uses the top input to define the output). So far, the top input has the form:

$$\frac{|0\rangle+|1\rangle}{\sqrt{2}}.$$

By using the abstract description of the output of f, a different way to write that output is:

$$(-1)^{f(x)}\frac{|x,0\rangle-|x,1\rangle}{\sqrt{2}} = (-1)^{f(x)}|x\rangle \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$

Then, using this information in our new circuit, we obtain altogether:

$$|\varphi_2\rangle = \frac{(-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$

Let's see what happens when we take the cases of f: in our case, $f(0)=1$ and $f(1)=0$.

Thus:

$$|\varphi_2\rangle = \frac{(-1)^1|0\rangle+(-1)^0|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}} = (-1)\frac{|0\rangle-|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$

Repeating the above procedure for all cases of f (constant or balanced) and using the same circuit architecture, we get the following behavior for $|\varphi_2\rangle$:

$$|\varphi_2\rangle = \begin{cases} (\pm 1)\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}, & \text{if } f \text{ is constant,} \\ (\pm 1)\frac{|0\rangle-|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}, & \text{if } f \text{ is balanced.} \end{cases}$$

Now, as a final step, we apply again the Hadamard gate to transform the top qubit as follows:

$$|\varphi_3\rangle = \begin{cases} (\pm 1)|0\rangle \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}, & \text{if } f \text{ is constant,} \\ (\pm 1)|1\rangle \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}, & \text{if } f \text{ is balanced.} \end{cases}$$

Then by just measuring the first qubit we can distinguish whether the function is balanced or constant and only by one function evaluation. This example shows why a quantum algorithm can achieve exponential acceleration, compared to classical computers.

The algorithm

The above example can be extended to multivariate functions. Consider functions $f : \{0,1\}^n \rightarrow \{0,1\}$. Notice that in this case, the cardinality of input values is exponential: there are 2^n different input configurations.

As before, we are interested in distinguishing whether f is constant or balanced. Here, constant f is defined as the case where all the inputs are mapped to either 0 or 1. Balanced f is denoted to the case where the half of the input go to 0, and the other half goes to 1.

Let's assume that we are assured that the function is either balanced or constant. We will briefly describe without much detail, how we would solve this problem using quantum systems with a single function evaluation. Similarly to the univariate input case, we will consider a circuit where the top input is a n-dimensional binary string, say $\mathbf{x}$, and the top input is a single qubit y. The function evaluation is again denoted with its matrix U_f and the outputs are represented in the following circuit:

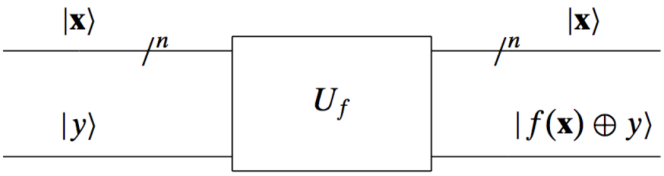


FIGURE 1.12: Circuit of Deutsch Josza algorithm

Similarly to the univariate case, we will need Hadamard matrices to put inputs into superposition. We can define multidimensional Hadamard matrices, by combining single qubit Hadamard matrices. The pattern of defining a n-dimensional Hadamard matrix, say $H^{\otimes n}$, is through its Kronecker product with itself. e.g., given

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

we can compute:

$$H^{\otimes 2} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Now, we have all the required tools to solve the problem. We will just represent the final circuit that allows to distinguish between a constant vs. balanced binary valued function in just one function evaluation. Here it is:

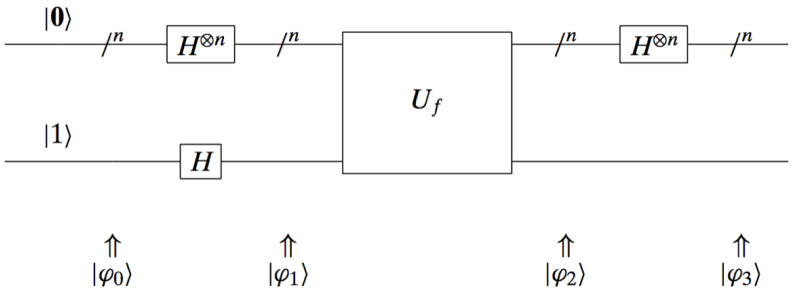


FIGURE 1.13: Deutsch Josza algorithm

This circuit defines the Deutsch-Jozsa algorithm [DJ92]: i. e., the generalization of the Deutsch algorithm for the multivariate binary case. The final outcome is that if we measure the top qubit in the circuit above, we will only get $|\mathbf{0}\rangle$ if the function is constant. In any other case, it is balanced. And we did so with only one function evaluation, an exponential speed-up is gained.

1.4.3 Grover's Algorithm

In mathematical terms, the standard way to find a specific element in a given set of n elements, is to do an exhaustive search. which means to look into each individual entries in the list and report the solution when we find that particular entry. This exhaustive search has $O(n)$ computational complexity. Indeed, this search can be done in a time much more faster [Gro96].

Grover's algorithm [Gro97] claims to solve the problem "with high probability" in $O(\sqrt{n})$ time. Note the "high probability" statement, since the algorithm is not deterministic.

To achieve our goals, there are two main steps: which are "phase inversion" and "inversion about the mean". At first sight, these may be irrelevant to the problem we discuss, but their combination is crucial for Grover's algorithm.

Phase inversion

The main idea behind the phase inversion is to change the phase of the (complex) weight/probability that corresponds to the element we are looking for.

Inversion about the mean

The "inversion about the mean" is exactly what it implies: we are asked to find a different sequence of numbers such that:

(i) the new numbers (summed up together) have the same distance as before from the mean, but inverted around the mean,

(ii) the new sequence has the same mean.

As can be easily proved, the second part is automatically proven if the first part is true.

We can easily observe that the combination of phase and mean inversion in sequence stretches the distance between the entry of interest and the rest of the entries in the vector. i. e., in a way we are forcing the element of interest to stand out compared to the rest of the entries.

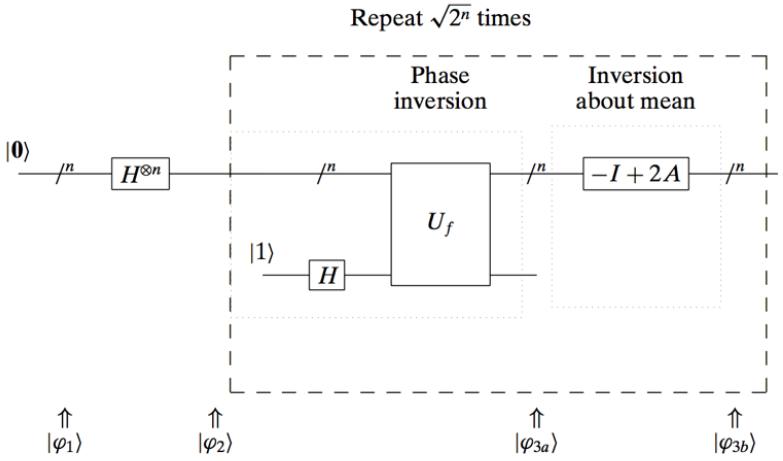


FIGURE 1.14: Grover's algorithm

Phase and mean inversion as quantum gates

In what follows, we will describe these two main steps, let's see how we can represent them in Grover algorithm. Remember that in the initial definition of the function f, we take as an input binary strings with length n and we output a binary number, with 1 only in the case where the input sequence matches the number we look for. One way to represent the phase and mean inversions together is as follows:

$$(-I+2A) = \begin{bmatrix} -1+\frac{2}{2^n} & \frac{2}{2^n} & \dots & \frac{2}{2^n} \\ \frac{2}{2^n} & -1+\frac{2}{2^n} & \dots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \dots & -1+\frac{2}{2^n} \end{bmatrix}$$

where $A = \frac{1}{2^n} \cdot \mathbf{1}_{2^n \times 2^n}$ is the matrix with all entries equal to $\frac{1}{2^n}$. The matrix A is the matrix that computes the mean. We can also easily prove that the transformation $(-I+2A)$, is actually unitary; thus it could be implemented as a quantum gate.

Putting all the ingredients together: the algorithm

Grover's algorithm can be described with a quantum circuit as follows Fig 1.14 : The algorithm performs the following motions:

- Set top input to $|\mathbf{0}\rangle$
- Apply the Hadamard transform to put in superposition all possible outcomes, with equal probability.
- Repeat $\sqrt{2^n}$ times:
  - Apply phase inversion
  - Apply mean inversion about the mean
- Measure output: with high probability, the result will be the entry required.

Example

We will close this paragraph with an example, in order to show the power of Grover's algorithm.

Let's consider that $n = 3$. The possible states are:

$$[000, 001, 010, 011, 100, 101, 110, 111].$$

Our function f is equal to 1 only at the state 101. Following the algorithm above we have:

- Initially: $|\varphi_1\rangle = [0\,0\,0\,0\,0\,0\,0\,0]^\top$.
- After applying the Hadamard transfrom, we get:

$$|\varphi_2\rangle = \left[\tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\right]^\top.$$

- In the inner loop:
  - The phase inversion gives: $|\varphi_{3a}\rangle = \left[\tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ -\tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\ \tfrac{1}{\sqrt{8}}\right]^\top$.
  - The mean of the vector of probabilities is:

$$\frac{7 \cdot \frac{1}{\sqrt{8}} - \frac{1}{\sqrt{8}}}{\sqrt{8}} = \frac{3}{4\sqrt{8}}.$$

  - After inversion around the mean, our vector of probabilities becomes: $|\varphi_{3b}\rangle = \left[\tfrac{1}{2\sqrt{8}}\ \tfrac{1}{2\sqrt{8}}\ \tfrac{1}{2\sqrt{8}}\ \tfrac{1}{2\sqrt{8}}\ \tfrac{1}{2\sqrt{8}}\ \tfrac{5}{2\sqrt{8}}\ \tfrac{1}{2\sqrt{8}}\ \tfrac{1}{2\sqrt{8}}\right]^\top$.
  - ... (after one more iteration in the inner loop).
- Measuring the top output, we get:

$$|\varphi_{3b}\rangle = \left[-\tfrac{1}{4\sqrt{8}}\ -\tfrac{1}{4\sqrt{8}}\ -\tfrac{1}{4\sqrt{8}}\ -\tfrac{1}{4\sqrt{8}}\ -\tfrac{1}{4\sqrt{8}}\ \tfrac{11}{4\sqrt{8}}\ -\tfrac{1}{4\sqrt{8}}\ -\tfrac{1}{4\sqrt{8}}\right]^\top.$$

We can notice that the probability to measure 101 is: $|\frac{11}{4\sqrt{8}}|^2 = 0.9453125$, while any other entry appears in the output with probability: $|\frac{1}{4\sqrt{8}}|^2 = 0.0078125$. Thus, even with two iterations, we get the solution we want with probability close to 1.

Grover's algorithm is probabilistic in the sense that it gives the correct answer with a probability of less than 1. Though there is technically no upper bound on the number of repetitions that might be needed before the correct answer is obtained, the expected number of repetitions is a constant factor that does not grow with n.

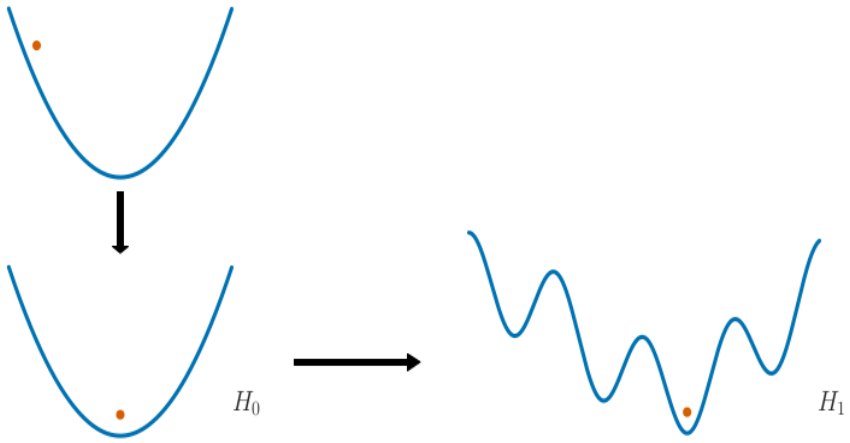
1.4.4 Adiabatic algorithm

In this paragraph, we follow the lines of the P. Wittek's book [Wit14] on "Quantum machine learning". As Wittek mentionned, an adiabatic process changes conditions gradually so as to allow the system to adapt its configuration. If the system starts in the ground state of an initial Hamiltonian, after the adiabatic change, it will end in the ground state of the final Hamiltonian. Consider a Hamiltonian H_0 whose unique ground state can be easily constructed, and another Hamiltonian, H_1 , whose

ground state energy we are interested in. To find this ground state, one considers the Hamiltonian:

$$H(\lambda) = (1-\lambda)H_0 + \lambda H_1$$

with $\lambda = \lambda(t)$ as a time-dependent parameter with a range in $[0,1]$. The quantum adiabatic theorem states that if we start in the ground state of $H(0) = H_0$ and we slowly increase the parameter λ, then the system will evolve to the ground state of H_1, if $H(\lambda)$ is always gapped, that is, there is no degeneracy for the ground state energy.



In adiabatic quantum computing [KN98], the aim is to find the global minimum of a given function $f : \{0,1\}^n \to (0,\infty)$, where $min_x f(x) = f_0$ and $f(x) = f_0$ if and only if $x = x_0$, that is, there is a unique minimum. We seek to find x_0. To do so, we consider the Hamiltonian:

$$H_1 = \sum f(x) \left|x\right\rangle \left\langle x\right|$$

We take an initial Hamiltonian H_0 and the Hamiltonian equation $H(\lambda) = (1-\lambda)H_0 + \lambda H_1$ to adiabatically evolve the system.

Thus, if we measure the system in the computational basis at the end of this process, we obtain the minimum x_0

1.4.5 The Harrow-Hassidim-Lloyd Algorithm

The quantum algorithm designed by Aram Harrow, Avinatan Hassidim, and Seth Lloyd were formulated in 2009, [HHL09], in order to solve linear systems. The key idea of HHL algorithm is to find an estimate of the result of a scalar measurement on the solution vector of a fixed linear system of equations [HHL09].

The Harrow-Hassidim-Lloyd (HHL) algorithm is one of the main important algorithms believed to provide an acceleration over their classical counterparts.

Solving systems of linear equations is one of the most basic problems you can imagine, as well as one of the most important. The HHL algorithm allows us to approximate this problem in $polylog(N)$ time, since several hypotheses are met. It is exponentially faster than any conventional algorithm for solving linear systems (which must take at least $poly(N)$ time), but it is not, strictly speaking, a completely fair comparison, because the way the data is represented. In addition, the HHL algorithm also requires some assumptions to be satisfied in order to reach the poly-logarithmic running time.

The problem statement of our linear system can be written as:

$$A\vec{x} = b$$

Where A is a Hermitian matrix, i.e. $\overline{A^T} = A$ and b is a unit vector, and our purpose is to find the solution of the vector $\vec{x}$ satisfying $A\vec{x} = b$.

The problem reduces to invert A. To solve this problem on a quantum machine, we should represent both the inputs and outputs in terms of a quantum state. Thus we assume that b is given in the form $|b\rangle$ as a $\log N$ qubit state, and the goal is to produce an approximation of the following quantum state $|x\rangle$ which is the quantum-mechanical representation of the desired solution vector x:

$$|x\rangle = \frac{A^{-1}|b\rangle}{||A^{-1}|b\rangle||}$$

This is the solution to the linear system $Ax = b$ in the quantum form.

Let's transform the vectors in terms of a quantum state as:

$$A|x\rangle = |b\rangle$$

Additionally, A is required to be sparse. This is not necessary for the algorithm to give results, but it is required for the speedup with respect to the best classical alternative holds.

This algorithm assumes that the user is not interested in the values of $\vec{x}$ itself, but rather the result of applying some operator M onto x, $\langle x|M|x\rangle$.

- A is Hermitian

  First of all, we represent the vector b as a quantum state of the form:

  $$|b\rangle = \sum_{i=1}^{N} b_i |i\rangle ,$$

  For a superposition of different times t, we use Hamiltonian simulation techniques to apply the unitary operator e^{iAt} to $|b\rangle$. We use the quantum phase estimation in order to decompose $|b\rangle$ into the eigenbasis of A so as to find the corresponding eigenvalues λ_j.

  As A Hermitian, we can write:

  $$A = \sum_{j=1}^{N} \lambda_j |\mu_j\rangle \langle\mu_j| ,$$

  where $\lambda_j \in \mathbb{R}$ are eigenvalues, and $|\mu_j\rangle \in \mathbb{R}^{\mathbb{N}}$ the respective eigenvectors.

  The state of the system after this decomposition is approximately:

  $$\sum_{j=1}^{N} \beta_j |\mu_j\rangle |\lambda_j\rangle ,$$

  where $|\mu_j\rangle$ is the eigenvector basis of A, and $|b\rangle = \sum_{j=1}^{N} \beta_j |\mu_j\rangle |\lambda_j\rangle$.

  Then, we will perform the linear map taking $|\lambda_j\rangle$ to $C\lambda_j^{-1}|\lambda_j\rangle$, where C is a normalizing constant. The linear mapping operation is not unitary, so that

would require a number of repetitions as it has some probability of failing. After it succeeds, we uncompute the $|\lambda_j\rangle$ register:

$$\sum_{j=1}^{N} \beta_j \lambda_j^{-1} |\mu_j\rangle = A^{-1} |b\rangle = |x\rangle ,$$

where $|x\rangle$ is a quantum-mechanical representation of the desired solution vector x.

- A is not Hermitian

  Normally, the algorithm requires that the matrix A is Hermitian so that it can be converted into a unitary operator. In the case where A is not Hermitian, we define a Hermitian matrix C as:

  $$C = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$$

  the algorithm can be solved using:

  $$Cy = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

  to obtain $y = \begin{pmatrix} 0 \\ x \end{pmatrix}$

  Then, the algorithm requires an efficient procedure to prepare $|b\rangle$. We suppose that there exists a linear operator B which can take an arbitrary quantum state $|initial\rangle$ to $|b\rangle$ efficiently or that this algorithm is a subroutine in a larger algorithm and receives $|b\rangle$ as input. Any error in the preparation of state $|b\rangle$ is ignored.

The algorithm can be also used as a subroutine in another quantum algorithm. Generally when we want to solve linear systems of equations we don't want just the solution, but we want to do something with the solution. For example after using HHL, we might want to calculate $\langle x|F|x\rangle$ for another matrix F - if F is an efficiently implementable measurement, then we could estimate this quadratic form efficiently.

It would be like using the Quantum Fourier Transform. Even if the Fourier transform is given as a quantum state in which it is difficult to extract a particular Fourier coefficient, it is still remarkably useful for solving problems like factorization of integers. If the algorithm using HHL also has an efficient way to generate the $|b\rangle$ needed during HHL then the representation of the data is no longer a problem and could be in fact considered as an advantage.

1.5 Conclusion

As most researchers in computer science don't know the basis of quantum physics, this chapter was devoted to give an explanation of the most basic notions in quantum computing, namely the notion of qubit, superposition, entanglement and so on. We also described the well known algorithms in quantum computing like Shor's algorithm, Deutsch-Jozsa algorithm and Grover's algorithm that we will use it in our contributions.

Chapter 2

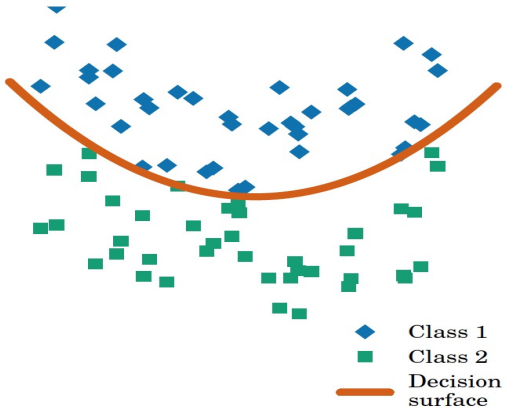
State of the art of Quantum Machine Learning algorithms

Contents

Machine learning is a branch of artificial intelligence that aims to learn from data to give the correct predictions. It studies the techniques that allow the machine to learn from past experiences. Unlike traditional computing where the machine is explicitly programmed on how it should react according to each situation, machine learning learns from examples of the past situations in order to be able to generalize in the future to the new situations. It is used in a wide variety of applications to solve problems like: clustering, text mining, regression, etc. There are two main types of machine learning tasks: supervised and unsupervised machine learning.

In supervised machine learning, the learner is provided a set of training examples with features presented in the form of high-dimensional vectors and with corresponding labels to mark its category. The aim is to classify new examples based on these training sets.

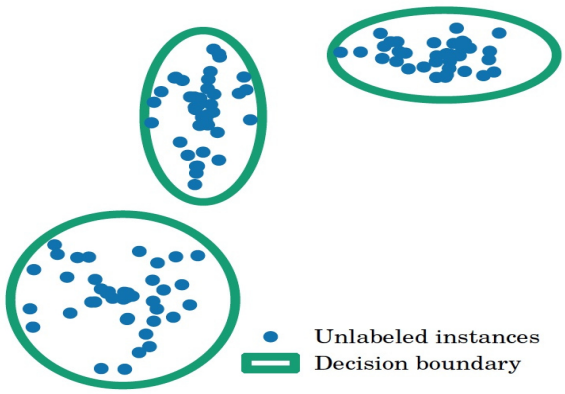
Given training set (x,y) that associates classifier labels y to each data point x. For each new data point x assign the correct class y.



Schematic diagram of supervised learning. Given labeled training instances, the goal is to identify a decision surface that separates the classes.

In unsupervised machine learning, the system aims to classify the data into different groups without prior information.

Given data x, find classes y that naturally partition the data.



Schematic diagram of unsupervised learning. The training instances do not have a label. The learning process identifies the classes automatically, often creating a decision boundary.

Machine learning is fundamentally different from rote learning, where the machine would simply seek to memorize the examples presented without learning anything from them. In fact, in machine learning the machine will not only be able to recognize most of the examples that have been presented to it, but it will be also able to generalize to cases not previously observed.

The increased computational power and data availability, as well as algorithmic advances, have led machine learning techniques to impressive results in regression, classification, data generation and reinforcement learning tasks.

Despite these successes,

with the ever increasing size of data sets, we may soon reach a point where current computational tools will no longer be sufficient. While custom hardware architectures, such as graphics processing units (GPUs) and tensor processing units (TPUs), can dramatically improve performance, they may not offer a structural

solution to the problem. That's why a growing number of researchers want to explore the possibility of harnessing the power of quantum computing to accelerate classical machine learning algorithms. Quantum computers could be the solution as they can effectively solve these problems which are considered to be difficult for classical machines, in particular Quantum Machine Learning (QML) which is the intersection between machine learning and quantum computation.

Here, we will present the most major advances in quantum machine learning; we will explain each quantum machine learning algorithm which has been done in literature.

Computational complexity studies the resources required to perform a given computational task. One of the major objectives of this theory is to classify the problems according to their time complexity, which roughly corresponds to the number of elementary steps necessary to solve the problem according to the size of the input. We define quantum acceleration as the runtime advantage obtained by a quantum algorithm over classical methods for the same task. We quantify the runtime with the asymptotic scaling of the number of elementary operations used by the algorithm with respect to the size of the input.

One of the roots of the accelerations theorized in quantum computing is the ability to process information in quantum superposition. Since machine learning is ultimately about analyzing large amounts of data, it is important to ask how the data is transformed into quantum superposition. There are two types of algorithms: those which work on quantum data and those which seek to process data stored in a classical memory. The first case is ideal for quantum machine learning. The data is ready for analysis and we don't have to spend computational resources to convert the data into quantum form. The second case is more elaborate because it requires a procedure which encodes classical information in a quantum state.

2.1 Quantum machine learning algorithms

2.1.1 Quantum K-means

K-means [Llo82] is a type of unsupervised algorithm that groups similar data into clusters.It calculates the centroids of K clusters and assigns a data point to that cluster having least distance between its centroid and the data point.

Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters, we assume that we have K clusters.

1. Choose a value of K, we randomly assign each data point to any of the K clusters.
2. Compute cluster centroid for each of the clusters, and assign each point to the closest cluster centroid.
3. Update the centroid of each cluster.
4. Repeat steps 2 and 3. Find the closest distance for each data point from new centroids and get associated with new K clusters. Repeat this process until convergence occurs which means until centroids does not change.

The quantum version of K-means [Ben+19b] algorithm provides an exponential speed-up comparing to its classical version. The algorithm is basically based on computing the distance between the quantum states using swap test and using

Grover's algorithm already described in the previous section to assign vectors to the closest centroid of cluster.

In the quantum algorithm to calculate the distance between two vectors $\vec{x}$ and $\vec{w}$, we need to do the following:

- Generate two states, $|\psi\rangle$ and $|\phi\rangle$ with an ancilla variable. Where

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle\,|x\rangle + |1\rangle\,|w\rangle)$$

$$|\phi\rangle = \frac{1}{\sqrt{Z}}(|\vec{x}|\,|0\rangle - |\vec{w}|\,|1\rangle)$$

  And $Z = |\vec{x}|^2 + |\vec{w}|^2$ is the sum of the squared norms of the two instances.

- Perform a projective measurement on the ancilla alone.

After generating the two states $|\psi\rangle$ and $|\phi\rangle$, we perform a swap test on the ancilla qubit. The swap test is composed of a Hadamard gate, Fredkin gate and another Hadamard gate using the ancillary qubit $|0\rangle$.

After applying the swap test circuit, the distance is evaluated by:

$$2Z(2P(|0\rangle) - 1)$$

Which implies that if we get a probability $P(|0\rangle) = \frac{1}{2}$ the states $|\psi\rangle$ and $|\phi\rangle$ are orthogonal, whereas the probability $P(|0\rangle) = 1$ indicates that the states are identical. The subroutine should be repeated several times to obtain a good estimator of probability. The advantage of using swap test is that the states $|\psi\rangle$ and $|\phi\rangle$ can be unknown before procedure and simple measurement is performed on the control qubit.

we can move on to the algorithm retrieving the euclidean distance $|\vec{x} - \vec{w}|^2$ between the two real valued vectors $\vec{x}$ and $\vec{w}$.

As $|x\rangle = \frac{\vec{x}}{|\vec{x}|}$ and $|w\rangle = \frac{\vec{w}}{|\vec{w}|}$, so we have $|\vec{x} - \vec{w}|^2 = 2Z|\,\langle\phi|\psi\rangle\,|^2$

To sum up, we prepare the two states and we apply swap test circuit, we repeat that procedure to obtain an acceptable estimate of probability. The classical algorithms require $O(N)$ to compute the euclidean distance between two vectors, while in the quantum version we have an exponential speed-up. After computing the distance between each centroid and data, we are ready to use Grover's algorithm to choose the closest centroid of cluster.

The quantum K-means algorithm provides an exponential speed-up compared to classical K-means algorithm. The quantum parallelism is achieved thanks to the clever calculation of distances between large dimensional vectors. Therefore, an exponential speed-up is gained.

2.1.2 Quantum K-medians

K-medians [RK87] is a clustering algorithm that represent the variation of K-means clustering where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median. This has the effect of minimizing the error over all clusters with respect to the 1-norm distance metric, as opposed to the squared 2-norm distance metric which K-means does.

K-medians algorithm attempts to create K disjoint cluster that must be selected as the cluster centers. The olgorithm follows the steps below:

- Assign each point in the data set to its closest center. The points assigned to the same center are then said to be in the same cluster, therefore they are added to the same disjoint-set.
- Calculate the median to determine the updated value of the cluster's center.
- Repeat the process until there isn't a great change.

The K-medians algorithm is similar to the K-means algorithm, but instead of defining new cluster centroid by computing the mean, we compute the median, therefore, the representative element of a cluster is always a data instance. K-medians is a variation of K-means. Indeed, the K-means algorithm has the disadvantage that in some cases the calculated mean may lie outside the desired region, while the cluster centroid calculated as the median still belongs to a set of training vectors. In other words, the centroid in K-means rarely coincides with a data instance: it lies in between the data points. Unlike K-means, K-medians may not converge: it can oscillate between the medians. K-medians is more robust and less sensitive to noise than K-means. The quantum version of K-medians algorithm requires the same quantum subroutines as quantum K-means algorithm: swap test circuit, computing the distance and Grover's algorithm [ABG07].

Algorithm 1: Quantum K-median algorithm

Input: Set of vectors $x_n \in \mathbb{R}^M, n = \{1, 2, ..., N\}$, initial centroids $w_1, w_2, ..., w_K \in \mathbb{R}^M$.

Output: The set of K clusters C_k, $|C_k|$ is the number of vectors within the cluster k.

repeat

Assignment step (clustering):
Compute the distance between each data and centroid, then use Grover algorithm to assign each data to the nearest cluster.

Update step:
For all $k = \{1, 2, ..., K\}$, update the centroid w_k of each cluster C_k by computing the median.

until a stopping condition is satisfied.

The algorithm of K-medians is similar to quantum K-means algorithm, they have approximately the same time complexity except the time complexity of the calculation of medians and means. Comparing to its classical counterpart, we gain an exponential speed with quantum computers.

2.1.3 Quantum Principal Component Analysis

Principal Component Analysis (PCA) [WEG87] is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. It is used to make data easy to explore and visualize by reducing the number of variables.

The components are orthogonal and each component is a linear combination of the original variables. This orthogonality between components indicates that the correlation between these components is zero.

The first principal component captures the direction of the maximum variability in the data. The second principal component captures the remaining variance in the data but has variables uncorrelated with the first component. Similarly, all successive

principal components (PC3, PC4 and so on) capture the remaining variance while being uncorrelated with the previous component.

Principal component analysis is used in several applications like: compression, simplifying data for easier learning, visualization. We should choose whether to go forward with PCA or not because the domain of knowledge is very important. For example, if the data is noisy, it is not suitable to use it as all the components of PCA have quite a high variance.

PCA reduces the dimensionality by transforming the data to a new set of uncorrelated variables (the principal components) of which the first few retain most of the variation present in the original dataset. The standard way to calculate the principal components boils down to finding the eigenvalues of the data covariance matrix.

The different steps of Principal component analysis are:

- Construct the covariance matrix C.
- Diagonalize C.
- Cite the principal component.

These steps should be done in a quantum manner. To do so, first of all we will present some properties of the density matrix that we will use it after.

Properties of density matrix

- A density matrix is Hermitian, that's to say:

$$\rho = \overline{\rho}^T$$

-

$$tr(\rho) = 1$$

Proof

$$\begin{aligned} tr(\rho) &= tr(\sum p_j |\psi_j\rangle \langle\psi_j|) \\ &= \sum p_j tr(|\psi_j\rangle \langle\psi_j|) \\ &= \sum p_j \\ &= 1 \end{aligned}$$

The quantum version of PCA [LMR14] uses the same steps of the classical version but working with quantum states:

- Transform the data into the quantum state.
- Construct the density matrix (covariance matrix).
- Find eigenvectors and eigenvalues.

  To diagonalize a matrix A which is Hermitian we use the ability to perform $e^{-iAt} |\psi\rangle$ which gives us the eigenvectors+eigenvalues (using quantum simulation).

  – Turn data into Hamiltonian $e^{-i\rho t} |\psi\rangle$ (Exponentiate the matrix).
  ρ is Hermitian we should turn it into Hamiltonian

  – Apply quantum phase algorithm

The underlying idea is that, doing a lot of partial application of a swap gate (for a small amount of time) and trace operations on two states and σ allow one to apply $e^{-i\sigma t}$ to ρ.

Recall: 1 *In the Schrödinger representation, the evolution over time of a quantum system is characterized (at the infinitesimal level) by the Hamiltonian operator.*

$$i\hbar\frac{d\left|\psi(t)\right\rangle}{dt} = H\left|\psi(t)\right\rangle$$

where ψ is the system's wave function, and H is the Hamiltonian operator.

In a stationary state

$$\left|\psi(t)\right\rangle = e^{-i\frac{Et}{\hbar}}\left|\psi(0)\right\rangle$$

where E is the energy of the stationary state.

We can easily see that a stationary state is an eigenvector of the Hamiltonian operator, with energy as its eigenvalue.

Quantum principal component analysis uses multiple copies of an unknown density matrix to construct the eigenvectors corresponding to the large eigenvalues of the state. In particular, we show that multiple copies of the state ρ can be used to implement the unitary operator $e^{-i\rho t}$: that is, the state functions as an energy operator or Hamiltonian, generating transformations on other states.

1. Transform data into quantum states

   We assume that our data sets consists of arrays of vectors stored in a form of complex vectors of dimension $N = 2^n$.

   We encode classical information into the norm of a quantum state, suppose we have a vector $v_j \in \mathbb{C}^d$

   $$\left|v\right\rangle = \frac{1}{|\vec{v}|}\sum_{i=1}^{N} v_i\left|i\right\rangle$$

2. Construct density matrix

   In the classical version, we have:

   $$C = \frac{1}{2^n}\sum_{j=1}^{N}\vec{v}_j\vec{v}_j^T$$

   The quantum version is:

   $$\rho = \frac{1}{2^n}\sum_{j=1}^{N}\left|v_j\right\rangle\left\langle v_j\right|$$

3. Find eigenvectors and eigenvalues

   Note that if we can perform $e^{-iAt}\left|\psi\right\rangle$ using quantum phase or model measurement of Von Newman, we can find the eigenvectors and the eigenvalues. Thus the question is how to perform $e^{-i\rho t}\left|\psi\right\rangle$? Wich means how to take a density matrix and turn it into Hamiltonian. We will try to make the density matrix exponentiation.

$$e^{-iS_{12}\Delta t}\rho_1 \otimes \sigma_2 e^{-iS_{12}\Delta t}$$

Where:

S_{12} is the swap, that's to say that: $S_{12}\left|\phi\right\rangle_1 \otimes \left|X\right\rangle_2 = \left|X\right\rangle_1 \otimes \left|\phi\right\rangle_2$,
$S_{12}^2 = \mathbb{1}_{12}$
and $e^{-iS_{12}\Delta t} = \cos\Delta t\mathbb{1} - i\sin\Delta t S_{12}$

We have

$$\cos\Delta t\mathbb{1}_{12} - i\sin\Delta t S_{12}\rho_1 \otimes \sigma_2 \cos\Delta t\mathbb{1} + i\sin\Delta t S_{12} =$$
$$\cos^2\Delta t\rho_1 \otimes \sigma_2 - i\sin\Delta t[S_{12}\rho_1 \otimes \sigma_1] + \sin^2\Delta t S_{12}\sigma_1 \otimes \rho_2$$

So,

$$\begin{aligned} tr_1 &= \cos^2\Delta t\sigma - i\sin\Delta t[\rho,\sigma] + \sin^2\Delta t\rho \\ &= e^{-i\rho\Delta t}\sigma e^{i\rho\Delta t} + o(\Delta t^2) \end{aligned}$$

Density matrix exponentiation now allows us to apply the quantum phase algorithm to find the eigenvectors and eigenvalues of an unknown density matrix. If we have n copies of ρ, we use the ability to apply $e^{-i\rho t}$ to perform the quantum phase algorithm. In particular, the quantum phase algorithm uses conditional applications of $e^{-i\rho t}$ for varying times t to take any initial state $\left|\psi\right\rangle\left|0\right\rangle$ to $\sum_i \psi\left|v_i\right\rangle\left|r_i\right\rangle$, where v_i are the eigenvectors of ρ and r_i are the corresponding eigenvalues.

2.1.4 Quantum Support vector machine

Support vector machines (SVM) [Hea+98] is a supervised learning methods used for classification, regression and outliers detection. SVM constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, it presents one of the most robust prediction methods based on the statistical learning framework. Intuitively, support vector machines search for the hyperplanes that divide a dataset into classes such that the margin around the hyperplane is maximized, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

The disadvantages of support vector machines are:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

A support vector machine tries to find the optimal separating hyper-plane. It maximizes the distance between the hyper-plane and closest data points called support vectors. The objective function could be convex or non-convex depending on the kernel used in SVM algorithm. The non-convex functions tend to converge to a local optimum, thus the classical SVM balances between optimization effectiveness and the prediction accuracy. Quantum version of support vector machines [RML14] using minimization quantum subroutine like Grover algorithm and provides convergence to a global optimum for non-convex cost function [Ang+03].

Consider that we have M data points of $\{(\vec{x_j}, y_j) : j = 1, 2, ..., M\}$, where $\vec{x_j}$ is a N-dimensional vector in data feature space, and $y_j \in \{-1, +1\}$ is the label of the data, which is +1 or -1. A support vector machine tries to find the optimal separating hyper-plane $\vec{w}.\vec{x} + b = 0$ that divides the data points into two categories as follows:

$$\vec{w}.\vec{x_j} + b \geq 1 \qquad when \;\; y_j = +1$$

$$\vec{w}.\vec{x_j} + b \leq -1 \qquad when \;\; y_j = -1$$

Indeed, the mathematical formulation of the optimisation problem of support vector machines contains a kernel matrix K containing the inner product of the feature vectors $(K)_{pk} = \vec{x_p}.\vec{x_k}$, where $p, k = 1, ..., N$ as entries. Therefore, the evaluation of the inner product in support vector machines is very important. That's why, the quantum version of SVM is based on calculating the inner product between the states.

The evaluation of the inner product can be done faster in quantum computers [RML14]. The quantum evaluation of a classical inner product relies on the fact that the quantum states are normalised as:

$$\langle x^p | x^k \rangle = \frac{\vec{x}^p.\vec{x}^k}{|\vec{x}^p||\vec{x}^k|}$$

Therefore, the kernel matrix can be calculated as follows:

$$tr[|\xi\rangle \langle\xi|] = \frac{1}{\sum_{i=1}^{2^n} |\vec{x}^i|^2} \sum_{p,k=1}^{2^n} \langle x^p | x^k \rangle \, |\vec{x}^p||\vec{x}^k| \, |p\rangle \langle k|$$

Where

$$|\xi\rangle = \frac{1}{\sqrt{\sum_{i=1}^{2^n} |\vec{x}^i|^2}} \sum_{i=1}^{2^n} |\vec{x}^i| \, |i\rangle \, |x^i\rangle$$

To summarize, the quantum version of SVM computes the kernel matrix using the inner product and perform the classification of the data. The complexity of the quantum SVM is $O(log(NM))$, so we gain an exponential speed-up comparing the classical version of SVM.

2.1.5 Quantum neural networks

The simplest definition of a neural network is provided by the inventor of one of the first neuro-computers, Dr. Robert Hecht-Nielsen. He defines a neural network as: "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs" [Cau89].

Neural networks are a series of algorithms that imitate the operations of a human brain to recognize relationships between vast amounts of data. It's based on a set of

connected units or nodes called artificial neurons, which loosely model neurons in a biological brain. Each connection, like synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The output of each neuron is calculated by a nonlinear function of the sum of its inputs. During learning process, neurons adjust the weight by increasing or decreasing the strength of the signal at a connection. Different layers may perform different transformations on their inputs.

Types of neural networks Neural networks are sometimes described in terms of their depth, including how many hidden layers they have. This is why the term neural network is used as a synonym of deep learning. They can also be described by the number of hidden nodes the model has or in terms of how many inputs and outputs each node has. Variations in the classic neural network design allow various forms of forward and backward propagation of information between levels.

There are different types of artificial neural networks, like: feed-forward neural networks, recurrent neural networks, convolutional neural networks, ...

Feed-forward neural networks are one of the simplest variants of neural networks. They pass information in one direction, through various input nodes, until it reaches the output node. The network may or may not have layers of hidden nodes, which makes their functioning more interpretable. It is ready to handle large amounts of noise. This type of neural network is used in technologies such as facial recognition and computer vision.

Recurrent neural networks (RNN) are more complex. They save the output of processing nodes and feed the result back into the model. This is how the model is said to learn to predict the outcome of a layer. Each node in the RNN model acts as a memory cell, continuing the computation and implementation of operations. This neural network starts with the same front propagation as a feed-forward network, but then goes on to remember all processed information in order to reuse it in the future. If the network's prediction is incorrect, then the system self learns and continues working towards the correct prediction during backpropagation. This type of neural network is frequently used in text-to-speech conversions.

Convolutional neural networks (CNN) are one of the most popular models used today. This neural network uses a variation of multilayer perceptrons and contains one or more convolutional layers that can be either entirely connected or pooled. These convolutional layers create feature maps that record a region of image which is ultimately broken into rectangles and sent out for nonlinear processing. The CNN model is particularly popular in the realm of image recognition; it has been used in many of the most advanced applications of artificial intelligence, such as: facial recognition, natural language processing and image classification.

The advantages of artificial neural networks include:

- The network can perform more than one job at a time thanks to parallel processing abilities.
- The ability to learn and model nonlinear, complex relationships helps model the real life relationships between input and output.
- The ability to predict the output of unseen data.
- The ability to learn from events and make decisions based on the observations.

The disadvantages of neural networks are:

- The lack of rules for determining the appropriate neural network structure, which means some tests should be done to find the right structure.

- All problems must be translated into numerical values before they can be presented to the neural network.
- The inability of explanations of how we get the results.

Artificial neural networks have been widely studied in the literature of quantum machine learning and they are used in several fields such as: pattern recognition, classification problems and financial time series prediction. They try to learn the non-linear dependencies between an input and output data. The classical neural networks algorithm uses the layers of connected neurons and selects the weights in each layer so that the cost function is minimized. The neurons are represented by activation functions such as sigmoid or softmax functions, which introduce the non-linearity to the algorithm.

An artificial neural network is a graph where the nodes are called neurons and their connections are the weights that represent the synaptic strengths between neurons. An activation function defines the value of a neuron according on the current value of all other neurons weighted by their weights, and the dynamics of the neural network is given by successively updating the value of neurons via the activation function. Therefore, an artificial neural network can be seen as a computational device, the input is the initial values of the neurons and the output is a stable state of the entire network. To program a neural network, we need to select the weights and an activation function encoding a certain input-output relationship. The power of artificial neural networks is that they can learn their weight from training data, a fact that neuro-scientists consider to be the basic principle of how our brains work by processing information. For pattern classification, we generally consider what are known as feed-forward neural networks in which the neurons are arranged in layers, and each layer feeds its values into the next layer. An input is presented to a feed-forward neural network by initializing the input layer, and after each layer successively updates its nodes. Indeed, feed-forward neural networks are able to classify input models extremely well if an appropriate set of weighting parameters is given. To evoke the desired generalization, the network is initialized with training vectors, the output is compared to the correct output, and the weights adjusted by gradient descent to minimize the classification error, this procedure is called back-propagation. A challenge for classifying models with neural networks is the computational cost of the back-propagation algorithm, even though we consider improved training methods such as deep learning.

Major research direction have been focused on accelerating the training of classical models and on the development of networks where all the building blocks, from single neurons to training algorithms, are executed on a quantum computer a so-called quantum neural network. The first works on quantum neural networks appeared in the 1990s and a number of articles have been published on the subject. However, it should be noted that the field of neural networks has not reached a level of scientific maturity comparable to the other areas of quantum machine learning already explained above. This lack of progression is due to the linearity of quantum mechanics and the non-linearity of elements in neural network.

The quantum version of neural networks is based on the principles of quantum mechanics. A lot of proposals try to find a quantum equivalent to the perceptron unit from which neural networks are built. One problem is that nonlinear activation functions do not immediately correspond to the mathematical structure of quantum theory, since a quantum evolution is described by linear operations and leads to probabilistic observation. Ideas to imitate the activation function of the perceptron with a quantum mechanical formalism reach from special measurements to postulating

non-linear quantum operators. A direct implementation of the activation function using the circuit-based model of quantum computation has recently been proposed by Schuld, Sinayskiy and Petruccione based on the quantum phase estimation algorithm [SSP14].

Recently several works attempt to build the quantum version of neural networks whose elements and updating rules are only based on the laws of quantum mechanics, however, there is no consensus on the defining its characteristics. Indeed, most of the articles failed to reproduce the basic characteristics of neural networks, because the biggest challenge for a quantum artificial neural network is that quantum mechanics is linear but artificial neural networks require non-linearity [Cyb89].

Recently,some efforts [Wan+17] [ROAG17] have been done to encounter the problem of non linearity by using measurements and introducing a number of overhead qubits into the input and output of each network node. However, these models don't afford a fully quantum neural network as it still lack some important features,such as: the parameters of the model that are still remaining classic. The authors of the papers recognize that, in their current forms, the most likely applications of these models appear to be learning quantum objects rather than enhancing classical data learning. We note that despite the several works done in quantum neural networks [GZ01], [PK97] and [PM11] there is no attempt to model the non-linearity directly on the amplitudes, which means that until now there is no proposal of a fully efficient quantum neural network.

Finally, the benefits of quantum neural networks in comparison to the classical algorithm are basically based on the speed-up that come from superposition.

2.2 Comparison of the performance of classical and quantum machine learning algorithms

It is noticeable that in our days the theory of quantum computations are among the most rapidly developing scientific areas. Quantum machine learning algorithms require to encode the classical data into a quantum computer to make it accessible for quantum information processing. Then, quantum information processing routines are applied to finally make a measurement to get the final results. Recently, researchers investigated if quantum computing can help to improve classical machine learning algorithms.

Indeed, this quantization is done in order to improve the performance which means to speed up the learning algorithms. Quantum computers has the natural advantage of reducing the time complexity of a computation process.

Computational complexity studies the resources required to perform a given computational task. One of the major goals of this theory is to classify problems according to their time complexity, which roughly corresponds to the number of elementary steps required to solve the problem according to the size of the input [AB09]. The advantage obtained by a quantum algorithm over the classical ones is the speed-up that we gain for the same tasks.

Quantum computers use effects such as superposition and entanglement to process information in a way that classical computers don't. recently, we have seen a great progress in building more powerful quantum computers in order to take advantage of quantum computing.

Quantum machine learning software uses quantum algorithms to process information. Quantum algorithms can in principle surpass the most well-known classical algorithms to solve certain problems especially in terms of acceleration.

For example, quantum computers can search an unsorted database in exponential time while in classical computer a polynomial time is required. Likewise, quantum computers can perform Fourier transforms, invert sparse matrices, and find their eigenvalues and eigenvectors in logarithmic time, while the most well-known algorithms for classical computers take a polynomial time: the quantum computer has an exponential speed compared to the best classical computer algorithms.

2.3 Conclusion

To recap, in this chapter we tried to give the state of the art of the quantum version of the most important machine learning algorithms for data science, namely: quantum K-means, QPCA, quantum SVM and quantum neural networks. It is noticeable that the quantum version of these algorithms follow the same steps as the classical version but these steps are done in a quantum manner.

Chapter 3

Quantum One-model clustering

Contents

3.1 Components of quantum prototype based clustering algorithm

Clustering is an unsupervised learning method which aims to divide the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

In Data Science, we can use clustering analysis to gain valuable insights from our data by seeing which groups the data points are in when we apply a clustering algorithm.

It has a large number of applications spread across various domains, like: anomaly detection, social network analysis, image segmentation and medical imaging.

In general, clustering can be divided into two groups :

Hard Clustering: In hard clustering, each data point either belongs to a cluster completely or not. As an instance, we want the algorithm to read all of the tweets and determine if a tweet is a positive or negative tweet.

Soft Clustering: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.

For example: The data points in the graph below clustered together can be classified into one single group. In the figure below, we can easily distinguish the clusters, and we can identify that there are 3 clusters.

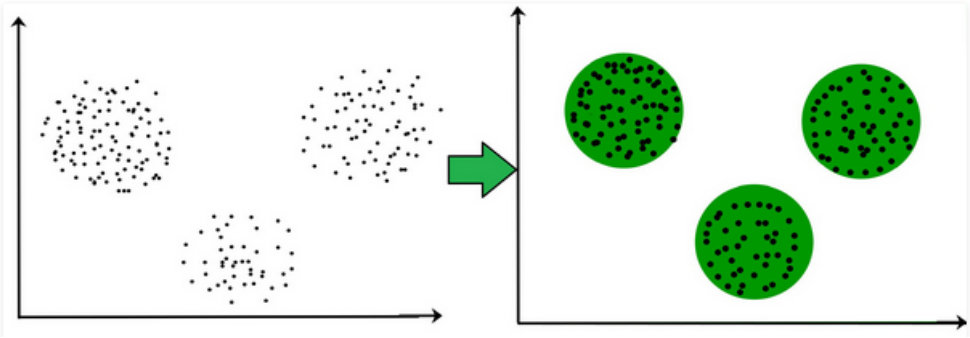


FIGURE 3.1: Clustering

Clustering is very important as it determines the intrinsic structures among the unlabeled data by forming clusters of similar groups.

Indeed, a clustering algorithm requires 5 elements, in this chapter we show the different steps of clustering in a quantum manner. This table summarizes the main difference between classical and quantum clustering.

Elts	Classical Clustering	Quantum Clustering
1	Data	States
2	Similarity/distance	Fidelity
3	Nearest centroid search	Grover's algorithm
4	Centroid update procedure	Quantum update rule
5	Quality index	Quantum quality index

TABLE 3.1: Classical & Quantum Clustering

In this chapter, we present our first contribution. We show the main steps of prototype based clustering algorithms in quantum case:

First of all, we explain how to compute the distance between each state and centroids; for this task we describe the fidelity which is a similarity measure and we present how to transform the data to quantum states in different ways. Due to the instability of the quantum distance, the estimation of the distance in the quantum context is a totally different way of thinking about the way we estimate it between points in the classical context.

Therefore, we explain also how to search for the closest centroid to a given data using Grover's algorithm which is a quantum algorithm that perform a search over unordered data and gives a quadratic speed-up over the classical version.

We also adapted a quantum criterion to the classical case which is quantum Davies Bouldin. Because, we can not evaluate quantum algorithms by classical indexes.

Finally, a series of empirical evaluations will be done to evaluate the quantum distances and its behavior versus the stability of finding the nearest centers in the right order.

3.1.1 How to estimate the distances between the different data and centroids?

Distance measurements to the different centroids are necessary for a clustering algorithm. To do this, how can we translate the idea of measuring distances into something that can be easily and efficiently done on a quantum computer? For a conventional computer, calculating Euclidean distances is easy, but doing it in the same way on a quantum computer would be much more complicated and would require more qubits than we can afford. On the other hand, the probabilistic nature of qubits makes it easier to measure phase differences and probability amplitudes.

For clustering algorithms, distances are necessary just to assign data points to different clusters. The objective is then to know which cluster is closest to a data point. This measure does not need to be proportional to the real distance, but only in positive correlation with it. Indeed, we only need the nearest centroid, not the exact values of the real distances.

In quantum computing, many distance-type measurements are available when we process qubits, such as the inner product between two (normalized) vectors and the probabilities of measuring a qubit in the states $|0\rangle$ or $|1\rangle$.

Indeed, the estimation of the distance between states in the context of quantum learning is totally different from the classical learning.

While in the classical version the distance computation involve coordinates, in quantum learning we need to define the distance with respect to the probabilistic nature of qubits. In quantum learning notions as phase differences or amplitudes of different probabilities are easy to measure and quantify, unfortunately the estimation of the distance between two states can not be directly defined because of its instability.

More precisely, let's consider that w and x are two position vectors corresponding to some centroid cluster in the available data and to some new data point for which we want to decide the cluster assignment, respectively. In the quantum context, we associate to those two vectors two quantum states denoted by $|\phi\rangle$ and $|\psi\rangle$, we will therefore estimate the so-called distance between the states $|\phi\rangle$ and $|\psi\rangle$. As already mentioned this can not be done directly.

In a nutshell, our strategy detailed below, consist to associate the inner product of x and w to the probability that some ancillary qubit is measured on the state 0.

Then we starts by considering an ancillary qubit $|0\rangle$ and the two quantum states $|\phi\rangle$ and $|\psi\rangle$. In these two quantum states are stored the normalized position vector x and w. Since we are in the quantum context the distance between the state $|\phi\rangle$ and the state $|\psi\rangle$ entangled with the ancillary qubit $|0\rangle$ will give us the distance between x and w.

To this end, we apply a Hadamard gate to the ancillary qubit $|0\rangle$, to define the superposition. After that, we use the entanglement between the states $|\phi\rangle$ and $|\psi\rangle$ and the concerning ancillary qubit, and then we use a swap gate controlled on the ancillary qubit between the quantum state $|\phi\rangle$ and $|\psi\rangle$.

After that, we apply another Hadamard gate to the ancillary qubit, and finally we measure the ancillary qubit. In this way we recover the inner product of x and w as the ancillary probability.

Once, we have computed the distance between each state and centroids using swap test circuit, we assign each state to the closest cluster centroid using Grover's algorithm [Gro98]. Finally, we update the centroids of each cluster. We detail the described strategy in the sections below.

Fidelity as a similarity measure

The history of the quantum fidelity (or the so-called Uhlmann-Jozsa Fidelity) can be traced back to the late 70's and early 80's to the works of Uhlmann [Uhl76] and Alberti [Alb83], who were studying generalizations of the quantum mechanical transition probability between two states. In later years the fidelity was studied by Richard Jozsa (who coined the term "Fidelity") [Joz94] and then by Benjamin Schumacher [Sch95] in the context of quantum communications.

Most of the current applications of quantum fidelity take place in the context of quantum technology. Since fidelity is a relative measure, the most appropriate fidelity depends on the application proposed. These technologies generally have a well-defined purpose. For example, it can be used to measure gravitational waves in a more sensitive way as in Einstein-Poldosky-Rosen (EPR). As well as other applications such as cryptography, quantum computing, quantum data processing and storage, and the growing field of quantum thermodynamics.

Each of these areas has its own criteria for success. However, the components of the technology ultimately depend on the achievement of certain quantum states and their processing. Therefore, knowledge of fidelity can help to measure how close one is to reaching the required quantum states that are used at a given stage of a quantum process. This concept is also applicable more widely, in fundamental physics problems such as quantum phase transitions. Fidelity is a vital concept of quantum information. It provides a mathematical prescription to quantify the degree of similarity of a pair of quantum states. Indeed, this comparison is useful in many situations. For example, as any experimental preparation for a quantum state is limited by imperfections and noise, we are generally interested in the proximity of the state actually produced with the state for which the production was intended. This is a common problem in quantum computing that arises in the context of the entanglement quantification [GT09]: the closer a given quantum state is to the set of the separable states, the less entangled it is and vice versa.

Measuring and calculating the fidelity between quantum states is at the heart of various quantum information tasks. Recently, fidelity measure has also been widely applied to study quantum phase transitions, see [Gu10].

In this chapter, we are using fidelity in order to measure the similarity between two quantum states. In the case of two pure states $|\psi\rangle$ and $|\phi\rangle$ fidelity can be defined as:

$$Fid(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|^2$$

Fidelity is similar to a measure commonly used in classical information, called cosine similarity. The properties of fidelity include:

Symmetry: $Fid(|\psi\rangle, |\phi\rangle) = Fid(|\phi\rangle, |\psi\rangle)$.

Bounded values: The fidelity varies between 0 and 1, $0 \leq Fid(|\psi\rangle, |\phi\rangle) \leq 1$.

If the states are orthogonal $Fid(|\psi\rangle, |\phi\rangle) = 0$ (that's to say, perfectly distinguishable), and $Fid(|\psi\rangle, |\phi\rangle) = 1$ if the states are identical.

Invariance under unitary transformation: $Fid(|U\psi\rangle, |U\phi\rangle) = Fid(|\phi\rangle, |\psi\rangle)$.

This means that if we apply the same unitary transformation U to two quantum states this does not change their fidelity.

Convexity: For any three quantum states $|\psi\rangle$, $|\phi\rangle$ and $|\gamma\rangle$ and given p_1 and p_2 such that $p_1 + p_2 = 1$, then $Fid(|\psi\rangle, p_1|\phi\rangle + p_2|\gamma\rangle) = p_1 Fid(|\psi\rangle, |\phi\rangle) + p_2 Fid(|\psi\rangle, |\gamma\rangle)$.

Multiplicativity: $Fid(|\psi_1\rangle \otimes |\psi_2\rangle, |\psi_3\rangle \otimes |\psi_4\rangle) = Fid(|\psi_1\rangle, |\psi_3\rangle) Fid(|\psi_2\rangle, |\psi_4\rangle)$.

The fidelity $|\langle\psi|\phi\rangle|$ [ABG06] of two quantum states $|\psi\rangle$ and $|\phi\rangle$ as a similarity measure can be obtained through the quantum circuit Swap test presented in Figure 3.2.

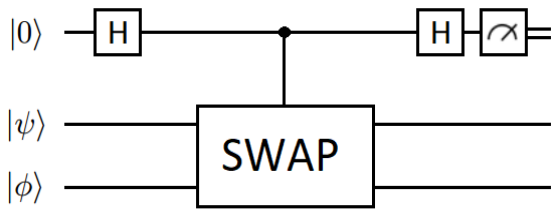


FIGURE 3.2: Swap test circuit

The circuit of swap test allows to compare two quantum states. It is composed of two Hadamard gates and a Control-Swap gate.
The control qubit is on the state :

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

After applying the Controlled Swap test, we get:

$$CSWAP\,|+\rangle\,|\psi\rangle\,|\phi\rangle \rightarrow \frac{|0\rangle\,|\psi\rangle\,|\phi\rangle + |1\rangle\,|\phi\rangle\,|\psi\rangle}{\sqrt{2}}$$

By applying the second Hadamard gate, we obtain:

$$CSWAP\,|+\rangle\,|\psi\rangle\,|\phi\rangle \rightarrow \frac{|0\rangle\,(|\psi\rangle\,|\phi\rangle + |\phi\rangle\,|\psi\rangle) + |1\rangle\,(|\psi\rangle\,|\phi\rangle - |\phi\rangle\,|\psi\rangle)}{2}$$

While measuring the ancillary qubit, we get:

$$\begin{aligned} P(|0_A\rangle) &= \left|\frac{1}{2}\,\langle 0|0\rangle\,|\psi\rangle\,|\phi\rangle + |\phi\rangle\,|\psi\rangle\right|^2 \\ &= \frac{1}{4}\,||\psi\rangle\,|\phi\rangle + |\phi\rangle\,|\psi\rangle|^2 \\ &= \frac{1}{2} + \frac{1}{2}\,|\langle\psi|\phi\rangle|^2 \end{aligned}$$

After measurement, we have $P(|0_A\rangle) = \frac{1}{2} + \frac{1}{2}|\,\langle\psi|\phi\rangle\,|^2$. A probability of $1/2$ consequently shows that the two quantum states $|\psi\rangle$ and $|\phi\rangle$ do not overlap at all (they are orthogonal), while a probability of 1 indicates that they have maximum overlap.

Destructive Swap test

Indeed, the swap test presented above could be presented in a form of another circuit that contains no ancilla qubit and uses less gates.

In what follows, we will prove that the Swap test is equivalent to destructive Swap test [GECP13]. This equivalence is useful to optimize the circuit: it reduces the number of gates and we don't need the ancillary qubit for measurement.

For that purpose, we will first of all explain some gates:

XOR: The XOR of two binary values is only true if one, and only one of them is true. We have $x \oplus x = 0$ and $x \oplus 0 = x$.

CNOT: The CNOT gate is a controlled NOT operation that flips the target value if the control is in state $|1\rangle$. We have $CNOT\,|x\rangle\,|y\rangle = |x\rangle\,|x \oplus y\rangle$.

Toffoli gate (CCNOT): The Toffoli gate is a controlled-controlled-NOT. The target is only flipped if both control qubits are $|1\rangle$. We have $CNOT\,|x\rangle\,|y\rangle\,|z\rangle = |x\rangle\,|y\rangle\,|(x.y) \oplus z\rangle$.

After explaining these gates, we will now give the equivalence between the swap test circuit and other circuits explained below: It's very simple to show that the CSwap gate can be replaced with a series of CNOTs (SWAP $\equiv$ CNOT) :

$|x\rangle\,|y\rangle \rightarrow |x \oplus y\rangle\,|y\rangle \rightarrow |x \oplus y\rangle\,|y \oplus x \oplus y\rangle = |x \oplus y\rangle\,|x\rangle \rightarrow |x \oplus y \oplus x\rangle\,|x\rangle = |y\rangle\,|x\rangle$

So, we can replace the CSwap gate in the Swap test by series of CNOT, we get

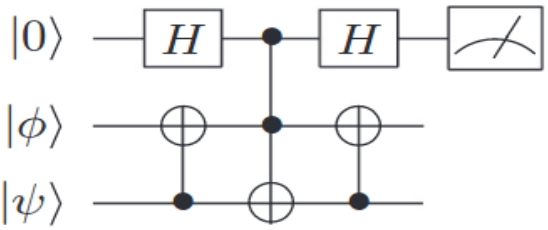


FIGURE 3.3: Cswap as CNOT

We know that a NOT gate is just a Z gate on another rotation axis. The rotation axis can be changed by two surrounding Hadamard gates. By replacing it in the circuit above we get:

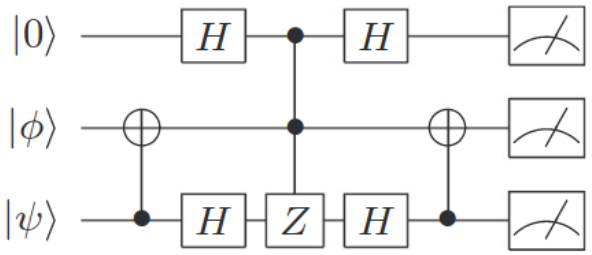


FIGURE 3.4: CCNOT as CCZ

We observe that the controlled controlled Z operation defined by:

$$CCZ\,|x\rangle\,|y\rangle\,|z\rangle = (-1)^{xyz}\,|x\rangle\,|y\rangle\,|z\rangle$$

Therefore it can be implemented transversely, which means we can put the Z rotation on any of the control qubit. Thus, we get the circuit below:

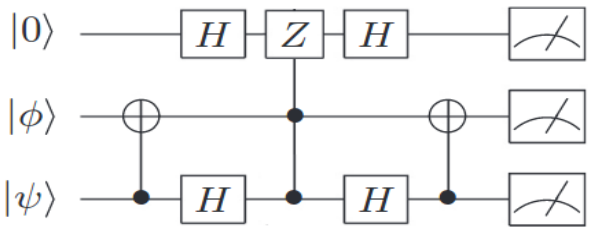


FIGURE 3.5: Symmetry of CCZ

We can notice in the circuit above that there are some unnecessary gates for the measurement on the ancilla qubit, that we can get rid of the circuit. By doing so, we get the circuit below:

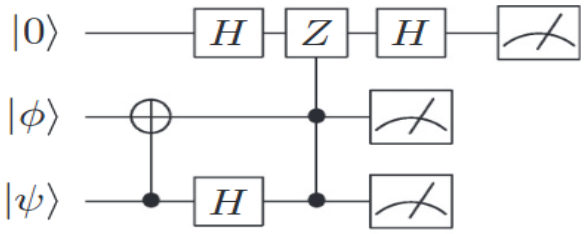


FIGURE 3.6: Removing useless gates

For another time, we use the equivalence between the X gate and HZH gate:

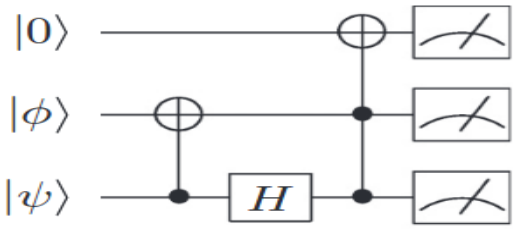


FIGURE 3.7: Rewriting CCZ as CCNOT

Thanks to the principle of deferred measurement [NC00], we can remove the ancillary qubit, and measure the other two qubits instead. The probability of reading 1 in the ancillary qubit is equivalent to the probability of reading 1 in both the qubit $|x\rangle$ *and* $|y\rangle$. Thus, here we get the final circuit:

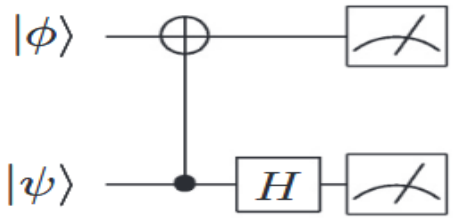


FIGURE 3.8: Final circuit

The order of the input states is not relevant. The SWAP test should give the same results for $|\phi\rangle\,|\psi\rangle$ and for $|\psi\rangle\,|\phi\rangle$, giving us two equivalent circuits.

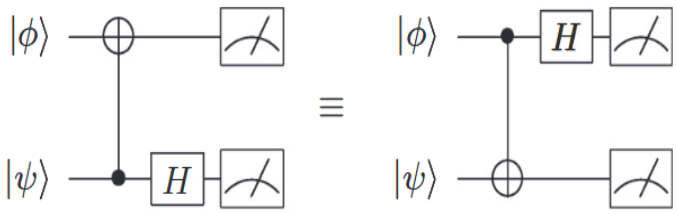


FIGURE 3.9: Destructive Swap test circuit

States construction to estimate the distance-type measurements

Several works have been made to compute the fidelity between two quantum states, all these works use the swap test circuit to obtain the similarity measure but they use different data preparation and construction of the states. In this section, we present the preparation and the construction of the states $|\psi\rangle$ and $|\phi\rangle$ of each method. Let's consider that we want to compute the distance between the two quantum states $|x\rangle$ and $|w\rangle$.

Wiebe et al. approach

1. Data preparation

   Given $N = 2^n$ dimensional complex vectors $\vec{x}$ and $\vec{w}$ with components $x_j = |x_j|e^{-i\alpha_j}$ and $w_j = |w_j|e^{-i\beta_j}$ respectively. Assume that $\{|x_j|, \alpha_j\}$ and $\{|w_j|, \beta_j\}$ are stored as floating point numbers in quantum random access memory.

2. Construction of the states

   An alternative representation of the states was suggested by Wiebe, Kapoor and Svore [WKS18]. It aims to write the parameters into amplitudes of the quantum states.

$$|\psi\rangle = \frac{1}{\sqrt{d}} \sum_j |j\rangle \left(\sqrt{1 - \frac{|x_j|^2}{r_{max}^2}} e^{-i\alpha_j} |0\rangle + \frac{x_j}{r_{max}} |1\rangle \right) |1\rangle$$

$$|\phi\rangle = \frac{1}{\sqrt{d}} \sum_j |j\rangle |1\rangle \left(\sqrt{1 - \frac{|w_j|^2}{r_{max}^2}} e^{-i\beta_j} |0\rangle + \frac{w_j}{r_{max}} |1\rangle \right)$$

   Where $j = \{1, ..., n\}$, and r_{max} is an upper bound on the maximum value of any feature in the dataset. The input vectors are d-sparse, i.e., contain no more than d non-zero entries.

   Using the swap test, the inner product is evaluated by:

$$d_{q_1}(|x\rangle, |w\rangle) = d^2 r_{max}^4 (2P(|0\rangle) - 1) \tag{3.1}$$

Lloyd et al. approach

1. Data preparation

   For the future of quantum machine learning, it may be vital to find quantum ways of representing and extracting information. To use the forces of quantum mechanics without being limited to the classical ideas of data encoding; Lloyd, Mohseni and Rebentrost [LMR13] proposed a way to encode the classical vectors into a quantum state.

   Consider $N = 2^n$ dimensional complex vectors $\vec{x}$ and $\vec{w}$, we have:

$$|x\rangle = \frac{\vec{x}}{|\vec{x}|} \quad , \quad |w\rangle = \frac{\vec{w}}{|\vec{w}|}$$

2. Construction of the states

   Seth Lloyd and his co-workers proposed a way to construct the state $|\psi\rangle$ and $|\phi\rangle$. The idea is to adjoin an ancillary qubit to states creating an entangled state

$|\psi\rangle$. The greater the difference between the states $|x\rangle$ and $|w\rangle$, the more the resulting state is entangled [Cai+15].

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle |x\rangle + |1\rangle |w\rangle)$$

$$|\phi\rangle = \frac{1}{\sqrt{Z}}(|\vec{x}| |0\rangle - |\vec{w}| |1\rangle)$$

Where $Z = |\vec{x}|^2 + |\vec{w}|^2$

After applying the swap test circuit, the distance is evaluated by:

$$d_{q_2}(|x\rangle, |w\rangle) = 2Z(2P(|0\rangle) - 1) \tag{3.2}$$

We can verify that the desired distance $|\vec{x} - \vec{w}| = \sqrt{2Z|\langle\phi|\psi\rangle|^2}$.

$$\begin{aligned}
|\langle\psi|\phi\rangle|^2 &= \frac{1}{2Z}|(|\vec{x}|\langle 0| - |w|\langle 1|)(|0\rangle|x\rangle + |1\rangle|w\rangle)|^2 \\
&= \frac{1}{2Z}||\vec{x}|\langle 0|0\rangle|x\rangle + |\vec{x}|\langle 0|1\rangle|w\rangle - |w|\langle 1|0\rangle)|x\rangle - |w|\langle 1|1\rangle)|w\rangle|^2 \\
&= \frac{1}{2Z}|(|\vec{x}||x\rangle - |\vec{w}||w\rangle)|^2 \\
&= \frac{1}{2Z}|\vec{x} - \vec{w}|^2
\end{aligned}$$

The desired distance is $d_{q_2} = 2Z(2P(0) - 1)$, we can see that if vector $\vec{x}$ is identical to the centroid of the cluster w the state $|\psi\rangle$ is orthogonal to $|\phi\rangle$ and the success probability of projective measurement is $P(0) = 1/2$, therefore the distance $d_{q_2} = 0$.

Anagolum approach

1. Data preparation

   For simplification, we assume that we are in 2-dimensional space. Let's consider that we have two vectors $\vec{x}(x_0, x_1)$ and $\vec{w}(w_0, w_1)$.

   We can map data values to θ and α values using these equations.

   For x we get:

   $$\alpha_0 = (x_0 + 1)\frac{\pi}{2} \ , \ \theta_0 = (x_1 + 1)\frac{\pi}{2} \tag{3.3}$$

   Similarly for w we get:

   $$\alpha_1 = (w_0 + 1)\frac{\pi}{2} \ , \ \theta_1 = (w_1 + 1)\frac{\pi}{2} \tag{3.4}$$

2. Construction of the states

   To construct the two states $|\psi\rangle$ and $|\phi\rangle$ Anagolum uses U gate as follows [Ana19]: (Note that this reference to Anagolum's work is only a blog post)

   $$|\psi\rangle = U(\theta_0, \alpha_0, 0)|0\rangle \tag{3.5}$$

   $$|\phi\rangle = U(\theta_1, \alpha_1, 0)|0\rangle \tag{3.6}$$

Indeed, U gate implement the rotations we need to perform to encode our data points.

$$U(\theta, \alpha, \lambda) = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\alpha}sin\frac{\theta}{2} & e^{i\lambda+i\alpha}\cos\frac{\theta}{2} \end{pmatrix}$$

This instruction would cause the qubit to move θ radians away from the positive z-axis, and α radians away from the positive x-axis.

Using the swap test, the distance is evaluated by:

$$d_{q_3}(|x\rangle, |w\rangle) = P(|1\rangle) \tag{3.7}$$

3.1.2 How to search for the closest centroid to a given data?

In quantum computing, Grover's algorithm allows to search for one or more element in an unsorted database with N entries in $O(\sqrt{N})$ time. Grover's algorithm begins with a quantum register of n qubit initialized to $|0\rangle$, when n is the number necessary to represent the search space, we have $2^n = N$ which means $|0\rangle^{\otimes n} = |0\rangle$.

Equal superposition: The first step is to apply the Hadamard transform $H^{\otimes n}$ to put the system into an equal superposition of states:

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$$

Quantum oracle O: An oracle is a black-box function and a quantum oracle is a quantum black-box, which means that it can observe and modify the system without collapsing it to a classical state. It will recognize if the system is in the correct state: if the system is in the correct state, then the oracle will rotate the phase by π radians, otherwise it will do nothing. To create the circuit that represents this oracle it is necessary to implement this transformation of $|x\rangle$:

$$|x\rangle \to (-1)^{f(x)}|x\rangle$$

where $f(x) = 1$ if x is the correct state, and $f(x) = 0$ otherwise.

Diffusion transform: It performs inversion about the average, transforming the amplitude of each state so that it is as far above the average as it was below the average prior to the transformation, and vice versa. This part consists of another application of the Hadamard transform $H^{\otimes n}$, followed by a conditional phase shift that shifts every state except $|0\rangle$ by -1, followed by another Hadamard transform. The *diffusion transform* can be represented by this equation, using the notation $|\psi\rangle$:

$$H^{\otimes n}[2|0\rangle\langle 0| - I]H^{\otimes n} = 2H^{\otimes n}|0\rangle\langle 0|H^{\otimes n} - I = 2|\psi\rangle\langle\psi| - I$$

Giving the entire Grover iteration:

$$[2|\psi\rangle\langle\psi| - I]O$$

The total runtime of a single Grover iteration is $\Theta(2n)$ from the two Hadamard transforms, plus the cost of applying $O(n)$ gates to perform the conditional phase shift, is $O(n)$. It follows that the runtime of Grover's entire algorithm, performing $O(\sqrt{N}) = O(2^{\frac{n}{2}})$ iterations each with a runtime of $O(n)$, is $O(2^{\frac{n}{2}})$.

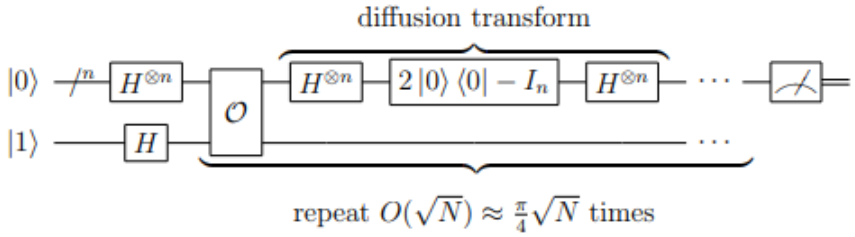


FIGURE 3.10: Circuit of Grover's algorithm

Algorithm 2: Grover's algorithm

Input: A quantum oracle O which performs the operation $O|x\rangle = (-1)^{f(x)}|x\rangle$, where $f(x) = 0$ for all $0 \leq x < 2^n$ except x_0, for which $f(x_0) = 1$, n qubits initialized to the state $|0\rangle$

Runtime: $O(\sqrt{N})$ operations, with $O(1)$ probability of success.

Output: x_0

Procedure:

Initial state

$$|0\rangle^{\otimes n}$$

apply the Hadamard transform to all qubits

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle = |\psi\rangle$$

apply the Grover iteration $R \approx \frac{\pi}{4}\sqrt{2^n}$ *times*

$$[(2|\psi\rangle\langle\psi| - I)O]^R|\psi\rangle \approx |x_0\rangle$$

measure the register x_0

3.1.3 Validation criteria

As we have already described, the notion of distance in quantum mechanics is not stable. Therefore, the estimation of the distance in the classical version it's not the same as the quantum version. For example, if we take the distance between two quantum states, after each measurement we get different values but the belonging of the state to a cluster it would be all the time the same. This is a totally different way of thinking about the way we estimate distances between points which can make the quantum clustering algorithms faster than its classical counterpart.

Our purpose is to efficiently define the validation criteria in the context of quantum learning in order to evaluate the goodness of clustering in quantum algorithms.

One of the most important considerations regarding the machine learning model is assessing its performance which means the quality of the model. In the case of supervised learning algorithms, evaluating the quality of the model is easy because we already have labels for every example. However, in the case of unsupervised learning algorithms, we are not that much blessed because we deal with unlabeled data. But still, we have some metrics that give the practitioner insight into the happening of change in clusters depending on the algorithm.

Indeed, there are several indices which are used to measure cluster validity. In our case, we chose the Davies Bouldin index which is an internal index that used clustering and the underlying data set to assess the quality of the clustering. It places

similar objects in the same cluster and dissimilar objects in different clusters. In what follows, we give the Davies-Bouldin (DB) index as a criteria of validation, and we transform the classical Davies Bouldin index in the quantum version; quantum Davies-Bouldin (QDB) index. This transformation is basically done by transforming the classical distance to the quantum version.

In fact, a good clustering algorithm aims to create clusters whose:

The intra-cluster similarity is high (The data that is present inside the cluster is similar to one another)

The inter-cluster similarity is less (Each cluster holds information that isn't similar to the other)

Classical Davies-Bouldin index

The Davies Bouldin index introduced by David L. Davies and Donald W. Bouldin in 1979 is a metric for evaluating clustering algorithms [DB79] that can be calculated with the following formula:

$$DB = \frac{1}{K}\sum_{k=1}^{K}\max_{k\neq k'}\frac{d_n(w_k)+d_n(w_{k'})}{d(w_k,w_{k'})}, \tag{3.8}$$

where K is the number of clusters, d_n is the average distance of all elements from the cluster C_k to their cluster centre w_k, $d(w_k, w_{k'})$ is the distance between clusters centres w_k and $w_{k'}$. Just like Silhouette score, Calinski-Harabasz index and Dunn index, Davies-Bouldin index provide an internal evaluation schema. I.e. the score is based on the cluster itself and not on external knowledge such as labels. This index well evaluates the quality of unsupervised clustering because it's based on the ratio of the sum of within-clusters scatter to between-clusters separation. More the value of DB is lower, means that we have a better cluster. The main objective is to evaluate how well the clustering has been done.

Quantum Davies-Bouldin index

As we have already mentioned before, the notion of distance in quantum approaches is different from the classical case. Quantum distance does not need to be proportional to the real distance, but only have a positive correlation with it. We need only the nearest centroid, not the exact value of the real distance. To evaluate the quality of quantum clustering with a Davies-Bouldin index-type based on intra- and inter-cluster distances, we propose to adapt it to the quantum case. To do this, we will define the Quantum Davies-Bouldin (QDB) quality index as follows:

$$QDB = \frac{1}{K}\sum_{k=1}^{K}\max_{k\neq k'}\frac{\delta_n(w_k)+\delta_n(w_{k'})}{\delta(w_k,w_{k'})}, \tag{3.9}$$

where

$$\delta_n(w_k) = \frac{1}{|C_k|}\sum_{i=1}^{|C_k|} d_q(|x_i\rangle_{x_i\in C_k}, |w_k\rangle) \quad \text{and} \quad \delta(w_k,w_{k'}) = d_q(|w_k\rangle, |w_{k'}\rangle)$$

This index has been done by transforming the classical distances to the quantum one as we have explained previously. Where K is the number of clusters, δ_n is the quantum version of the distance d_n and $\delta(w_k, w_{k'})$ is the quantum distance between clusters centres w_k and $w_{k'}$.

In fact, the estimation of the Davies Bouldin in the context of quantum learning is totally different from the classical learning.

While in the classical version, the Davies Bouldin index gives a fix value, in quantum learning the quantum DB will give different values as the distance change from an iteration to another. Therefore, instead of having only one value of DB we will get an interval of the QDB, this interval is due to the probabilistic nature of the qubits and it represents the different variation of this index while it is in the quantum state.

3.1.4 Experimental results

In this section, we compare the different quantum distances explained before in terms of relevance and stability by answering these two questions through some empirical evaluations:

- Which quantum distance has a high probability of finding the right nearest centre?
- Which quantum distance has a high probability of finding the nearest centres in the right order with a good stability?

Comparison of different quantum distances

Which quantum distance has a high probability of finding the right nearest centre? Because of the probabilistic nature of the qubits, the distance between two states is hard to compute as we will get a probabilistic result; the distance is unstable. However, it's easier to assign data points to different groups because we don't need the exact distances to each one but only the closest centroid. Thus, we can just put the new data point in the cluster associated with the smallest value that our parameter takes. To illustrate more our idea, we gave an example of two distributions. Fig. 3.11 and fig. 3.12 represent two distributions where the black dot X is the test data and the three crosses are the centers ($C1$, $C2$, $C3$).

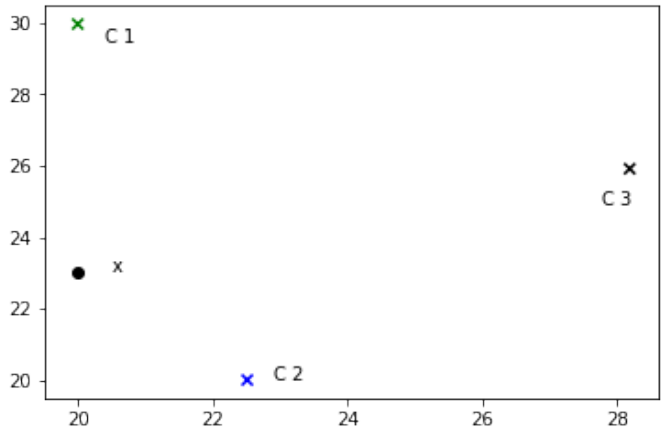


FIGURE 3.11: Distribution 1

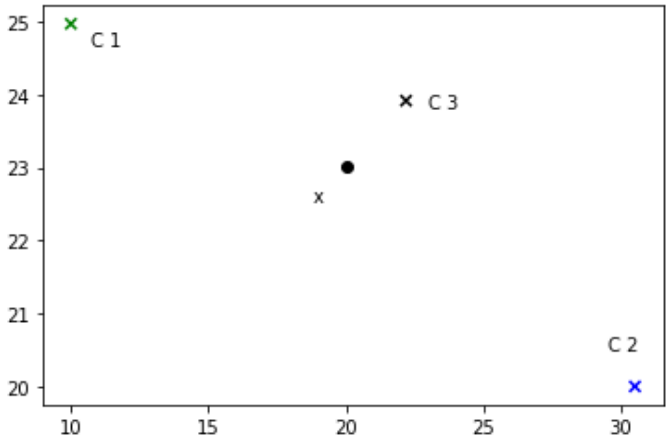


FIGURE 3.12: Distribution 2

From Table 3.2, we can notice that the distance changes from an iteration to another but the assignment to the closest centroid is correct. Comparing the three approaches in term of the confidence interval, we can notice that the Wiebe et al. approach gives a higher confidence interval in a time lower than other approaches.

Distance	Distribution	C1 (times)	C2 (times)	C3 (times)	Probability of success
Wiebe et al. approach	1	665	9322	13	[92.71%, 93.70%]
(d_{q_1})	2	0	0	10000	[99.96%, 100%]
Lloyd et al. approach	1	2190	7620	190	[75.35%, 77.02%]
(d_{q_2})	2	0	515	9485	[94.40%, 95.26%]
Anagolum approach	1	2722	6530	748	[64.36%, 66.22%]
(d_{q_3})	2	2486	2175	5339	[52.41%, 54.36%]

TABLE 3.2: Distance-types Comparison

Which quantum distance has a high probability of finding the nearest centres in the right order with a good stability? After analyzing the performance of the different quantum distances in terms of the stability of the values allowing the choice of the right center, we will study the behaviour of these quantum distances, but this time in terms of the stability of the order of the nearest centers. In other words, how far away is it possible to find the nearest centers in the right order whatever the iteration? To do this, we carried out 10,000 searches for the nearest centers for the two distributions studied. We analyzed the stability of the order of the nearest centers found by each quantum distance. The results show that the distance d_{q_1} is the best one which offers a very good stability in the order of the nearest centers in the case of the two distributions studied. As shown in Table 3.2, the distance d_{q_1} shows a very good stability in the order of the nearest centers compared to the other two quantum distances. For distribution 1, the correct order of the nearest centers is [C2 C1 C3]. The distance d_{q_1} finds this order with a probability of 85.32% (8532 times out of 10,000 searches), while the distance d_{q_2} and d_{q_3} find the order with a probability of 53.26% (5326/10,000) and 26.10% (261/10,000) respectively. In the case of distribution 2, the situation is more complicated because the test point is almost halfway between two centers. This situation is confirmed by the results obtained in fig. 3.14. Indeed, the distance d_{q_1} always finds the right order of the nearest centers [2 0 1]. Nevertheless, this distance continues to provide the right solution but the order

changes significantly [C3 C2 C1]. Compared to the other two quantum distances, the distance d_{q_1} seems much more stable in the order of the nearest centers. As can be seen in both fig. 3.13 and fig. 3.14, the other two quantum distances d_{q_2} and d_{q_3} change order quite often compared to the distance d_{q_1}. Order stability is a very relevant information on the behaviour of quantum distances.

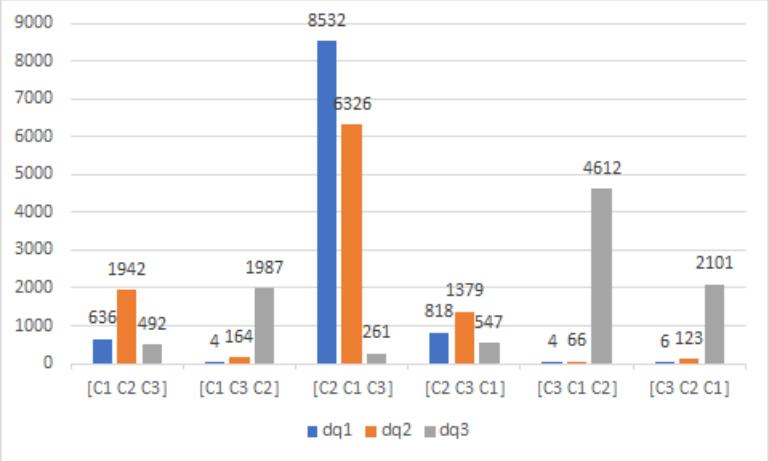


FIGURE 3.13: Order of belonging distribution 1

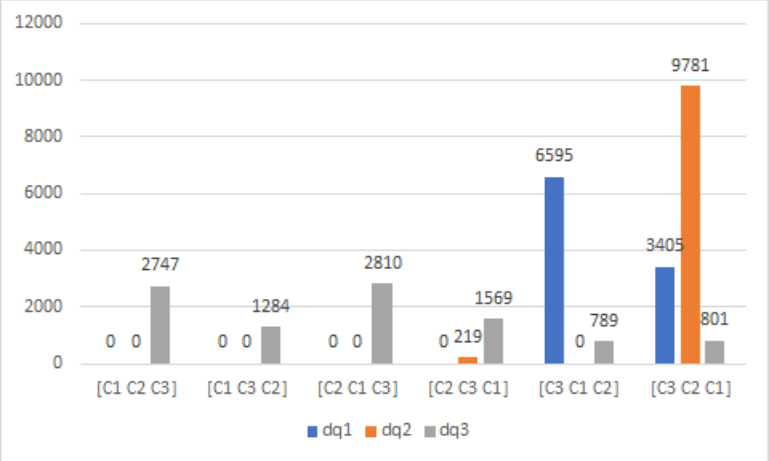


FIGURE 3.14: Order of belonging distribution 2

3.2 Quantum K-means

3.2.1 Classical K-means

The K-means clustering [Mac+67] is a type of unsupervised clustering, one of the most widely used clustering method early developed by Lloyd [LLo82]. Let $X = \{x_1, x_2, \ldots, x_N\}$ be the data set, each row vector $x_n \in \mathbb{R}^M$, $1 \leq n \leq N$ is composed of M attributes (features). The K-means clustering allows to divide the set of data X into K clusters $C = \{C_1, \ldots, C_K\}$ by minimizing the following sum of squared errors:

$$R = \sum_{k=1}^{K} \sum_{n \in C_k} \|x_n - w_k\|^2 = \sum_{k=1}^{K} \sum_{n=1}^{N} g_{nk} \|x_n - w_k\|^2, \tag{3.10}$$

where $w_k \in \mathbb{R}^M$ is the centroid of the data within the cluster C_k and $G \in \mathbb{R}_+^{N\times K}$ is the binary classification matrix defined by $g_{nk} = 1$, if the data $x_n \in C_k$, and 0 otherwise. Firstly, the K-means algorithm initialize randomly K centroids $w_1, w_2, ..., w_K \in \mathbb{R}^M$. Then the algorithm usually iteratively unfolds in two phases:

1. at first we go over every point and assign each to the cluster of the nearest centroid.

2. at the second phase, the centroid of each cluster is updated.

The algorithm converges when there is no further change in the assignment of instances to clusters. The idea behind K-means clustering is very natural. It just puts every new data point you ask it to classify into the group that it is closest to.

Algorithm 3: K-means algorithm

Input: Set of vectors $x_n \in \mathbb{R}^M, n = \{1, 2, ..., N\}$, initial centroids $w_1, w_2, ..., w_K \in \mathbb{R}^M$.

Output: The set of K clusters C_k, $|C_k|$ is the number of vectors within the cluster k.

repeat

Assignment step (clustering): Assign each data to the cluster C_{k^*}, k^* is computed by:

$$k^* = \underset{k\in\{1,2,...,K\}}{\operatorname{argmin}} \|x_n - w_k\|^2$$

Update step: For all $k = \{1, 2, ..., K\}$, update the centroid w_k of each cluster C_k by:

$$w_k = \frac{1}{|C_k|}\sum_{n=1}^{N} g_{nk}x_n$$

until a stopping condition is satisfied.

3.2.2 Quantum K-means

K-means clustering [Mac+67] is a type of unsupervised machine learning that aims to find groups in the data by dividing the dataset into K clusters. In this section, we give the algorithm of quantum K-means, this algorithm is adaptable for the three approaches explained above.

We show the different steps of QK-means: first of all, we transform the data to quantum states. Then, we compute the distance between each state and centroids. Indeed, the estimation of the distance between states in the context of quantum learning is totally different from the classical learning.

While in the classical version the distance computation involve coordinates, in quantum learning we need to define the distance with respect to the probabilistic nature of qubits. In quantum learning notions as phase differences or amplitudes of different probabilities are easy to measure and quantify, unfortunately the estimation of the distance between two states can not be directly defined because of its instability.

More precisely, let's consider that w and x are two position vectors corresponding to some centroid cluster in the available data and to some new data point for which we want to decide the cluster assignment, respectively. In the quantum context, we associate to those two vectors two quantum states denoted by $|\phi\rangle$ and $|\psi\rangle$, we will therefore estimate the so-called distance between the states $|\phi\rangle$ and $|\psi\rangle$. As already mentioned this can not be done directly.

In a nutshell, our strategy detailed below, consist to associate the inner product of x and w to the probability that some ancillary qubit is measured on the state 0.

Then we starts by considering an ancillary qubit $|0\rangle$ and the two quantum states $|\phi\rangle$ and $|\psi\rangle$. In these two quantum states are stored the normalized position vector x and w. Since we are in the quantum context the distance between the state $|\phi\rangle$ and the state $|\psi\rangle$ entangled with the ancillary qubit $|0\rangle$ will give us the distance between x and w.

To this end, we apply a Hadamard gate to the ancillary qubit $|0\rangle$, to define the superposition. After that, we use the entanglement between the states $|\phi\rangle$ and $|\psi\rangle$ and the concerning ancillary qubit, and then we use a swap gate controlled on the ancillary qubit between the quantum state $|\phi\rangle$ and $|\psi\rangle$.

After that, we apply another Hadamard gate to the ancillary qubit, and finally we measure the ancillary qubit. In this way we recover the inner product of x and w as the ancillary probability.

Once, we have computed the distance between each state and centroids using swap test circuit, we assign each state to the closest cluster centroid using Grover's algorithm [Gro98]. Finally, we update the centroids of each cluster. We detail the described strategy in the next paragraphs.

Data preparation and states construction

Normally, there are several methods to prepare the data and construct the states. According to [Ben+19b] the method of Wiebe, Kapoor and Svore [WKS18] gives good results in terms of clustering and also stability.

In what follows, we use this method for the preparation and the construction of the states $|\psi\rangle$ and $|\phi\rangle$ for the local phase and the states $|\gamma^{(l)}\rangle$ and $|\delta^{(l')}\rangle$ for the collaborative phase.

Given $N = 2^n$ dimensional complex vectors x and w with components $x_j = |x_j|e^{-i\alpha_j}$ and $w_j = |w_j|e^{-i\beta_j}$ respectively. Assume that $\{|x_j|, \alpha_j\}$ and $\{|w_j|, \beta_j\}$ are stored as floating point numbers in quantum random access memory.

Wiebe, Kapoor and Svore [WKS18] suggested a representation of the states that aims to write the parameters into amplitudes of the quantum states.

With the definitions of x and w, we define the quantum states:

$$|\psi\rangle = \frac{1}{\sqrt{d}} \sum_j |j\rangle \left(\sqrt{1 - \frac{|x_j|^2}{r_{max}^2}} e^{-i\alpha_j} |0\rangle + \frac{x_j}{r_{max}} |1\rangle \right) |1\rangle$$

$$|\phi\rangle = \frac{1}{\sqrt{d}} \sum_j |j\rangle |1\rangle \left(\sqrt{1 - \frac{|w_j|^2}{r_{max}^2}} e^{-i\beta_j} |0\rangle + \frac{w_j}{r_{max}} |1\rangle \right),$$

where $j = \{1, ..., n\}$, and r_{max} is an upper bound on the maximum value of any feature in the dataset. The input vectors are d-sparse, i.e., contain no more than d non-zero entries.

The idea behind this algorithm is to adjoin an ancillary qubit to the states creating an entangled state $|\psi\rangle$. The bigger difference between the states $|x_n\rangle$ and $|w_k\rangle$, the more entangled the resulting state is, and therefore we can use this entanglement to estimate the distance between vectors [Cai+15].

Cluster assignment

After computing the distance between each training state and each cluster centroid. We assign each state $|x_n\rangle$ to the closest centroid $|w_k\rangle$ by using Grover's algorithm [Gro98]. More precisely, we should find the solution of the following minimization problem:

$$\underset{w}{\operatorname{argmin}} D(|x\rangle, |w\rangle) = \underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{|x_n\rangle \in C_k} d_q^2(|x_n\rangle, |w_k\rangle) \tag{3.11}$$

While the best classical algorithms for a search over unordered data requires $\mathcal{O}(N)$ time, Grover's algorithm performs the search on a quantum computer in only $\mathcal{O}(\sqrt{N})$ operations, which means a quadratic speed-up over its classical version. This speed is done thanks to the superposition of states, in other words, the search is done globally, which means a significant improvement in optimization routines.

Grover's algorithm [Gro98] performs a search over an unordered set of 2^N items to find the unique element that satisfies some conditions. Its goal is to look in the function inputs to check if the function returns true for that input. This function can be represented as a quantum oracle and could be constructed from a large number of combined quantum gates.

Update the centroid

Assume that we have a set of quantum states $|X\rangle = \{|x_n\rangle \in \mathbb{C}^M, n = 1, ..., N\}$, and a set of K clusters C_k, $|C_k|$ is the number of vectors within the cluster k. K-means clustering aims to partition the N observations into K clusters C_k with $|W\rangle = |w_1\rangle, |w_2\rangle, ..., |w_K\rangle$ centroids, so as to minimize the within-cluster variance. Formally, the objective is to find:

$$\underset{W}{\operatorname{argmin}} D(|x\rangle, |w\rangle) = \underset{C}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{|x_n\rangle \in C_k} d_{q_i}^2(|x_n\rangle, |w_k\rangle) \tag{3.12}$$

We compute the distance between each training state and each cluster centroid using the swap test circuit 3.2. Then, we assign each state to the closest centroid using Grover's algorithm as explained before.

The second step of QK-means is updating the centroid of each cluster. To do so, the update centroid of each cluster is given by:

$$|w_k^{(t+1)}\rangle = |(Y_k^{(t)})^T X\rangle$$

where

$$|Y_k^{(t)}\rangle = \frac{1}{\sqrt{|C_k|}} \sum_{n=1}^{N} y_{nk} |n\rangle \quad \text{and} \quad y_{nk} = \begin{cases} 1 & \text{if } x_n \in C_k \\ 0 & \text{else} \end{cases}$$

We give the main steps of the proposed algorithm in the following. The distance $d_{q_i}(|x_n\rangle, |w_k\rangle)$ is at the user's choice, in our case we opt for the distance d_{q_1} as it gives the best result. For the stopping criterion, we use the relative distortion between two iterations with respect to a threshold ϵ.

The algorithm of QK-means [Ben+19b] is the following:

Algorithm 4: Quantum K-means algorithm

Input: $|X\rangle = \{|x_n\rangle \in \mathbb{C}^M, n = 1, ..., N\}$, K number of clusters C_k, initial centroids of the clusters at $t = 0$: $|w_1^{(0)}\rangle, |w_2^{(0)}\rangle, ..., |w_K^{(0)}\rangle$.

Output: K clusters C_k.

repeat

Assignment step (clustering): Each data is assigned to the cluster with the nearest center using Grover's search:

$$C_k^{(t)} \longleftarrow \{|x_n\rangle : d_{q_i}^2(|x_n\rangle, |w_k^{(t)}\rangle) \leq d_{q_i}^2(|x_n\rangle, |w_j^{(t)}\rangle), \forall j, 1 \leq j \leq K\}$$

where each $|x_n\rangle \in |X\rangle$ is assigned to exactly one $C_k^{(t)}$, even if it could be assigned to more of them.

Update step: The center of each cluster C_k is recalculated as being the average of all data belonging to this cluster (following the previous assignment step):

$$|w_k^{(t+1)}\rangle \longleftarrow |(Y_k^{(t)})^T X\rangle$$

until Convergence is reached

Convergence can be considered as achieved if the relative value of the distortion level $D(|x\rangle, |w^{(t)}\rangle)$ falls below a small prefixed threshold ϵ:

$$\frac{D(|x\rangle, |w^{(t-1)}\rangle) - D(|x\rangle, |w^{(t)}\rangle)}{D(|x\rangle, |w^{(t)}\rangle)} < \epsilon$$

3.2.3 Experimental results

Datasets

The classical and quantum version of K-means was tested on three real world datasets available for public use in the UCI Machine learning repository [DG17].

- *Iris* - Iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.
- *Wine* - Wine is a dataset that is related to a chemical analysis of wines grown in the same region in Italy but derived from different cultivars.
- *Wisconsin Diagnostic Breast Cancer (WDBC)* - This data has 569 instances with 32 variables (ID, diagnosis, 30 real-valued input variables). Each data observation is labeled as benign (357) or malignant (212).

Clustering through quantum K-means

We used three different datasets to show the experimental results of QK-means. Fig. 3.15, 3.16 and 3.17 represent the projection of the datasets iris, wine and breast cancer respectively using the principal component analysis. We can notice that the algorithm of QK-means has identify the different clusters (groups) which are significantly different (distant) from each other. Therefore, the quantum K-means gives a good classification just like it's classical version, but the advantage of the quantum version is that it can deal with high dimensional spaces in a time much more quicker than the classical version, which is crucial in nowadays.

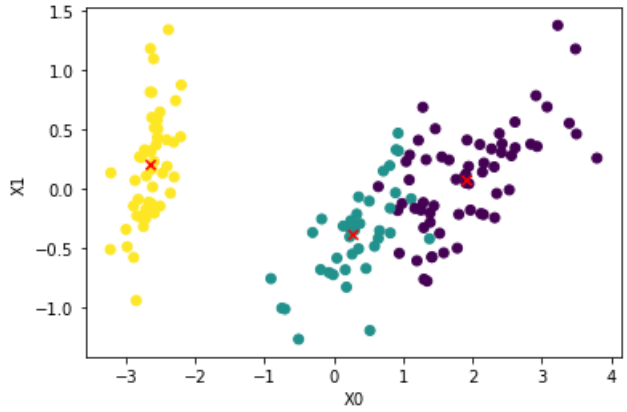


FIGURE 3.15: *QK*-means clustering on Iris data

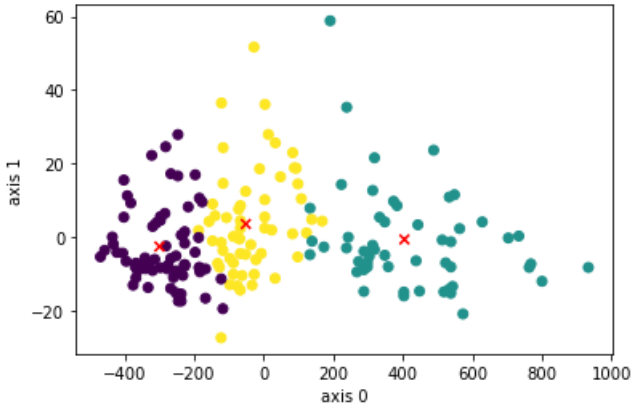


FIGURE 3.16: *QK*-means clustering on Wine data

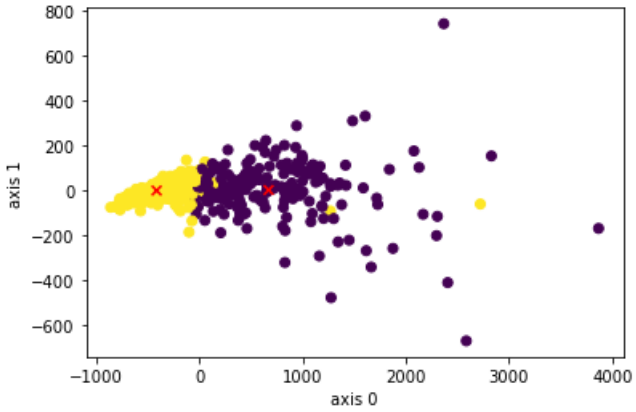


FIGURE 3.17: *QK*-means clustering on Breast Cancer data

For each data set we compare the Davies-Bouldin (DB) index for both the classical and the quantum version of *K*-means. These results are represented in Table 3.3. DB and QDB index are not calculated with the same distances. Direct comparison is therefore difficult, but we can see that QDB shows a decreasing behaviour during different iterations of learning process, indicating an improvement in the quality of

quantum clustering. We can therefore consider that QDB is a good quality indicator for quantum clustering.

Dataset	K-means (DB)	QK-means (QDB)
Iris	0.66	$[0.37, 0.56]$
wine	0.53	$[0.40, 0.59]$
Breast Cancer	0.50	$[0.38, 0.57]$

TABLE 3.3: K-means & QK-means using DB index

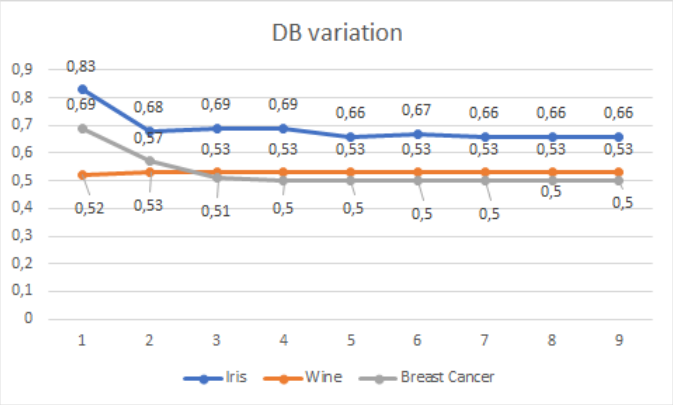


FIGURE 3.18: Davies-Bouldin Variation

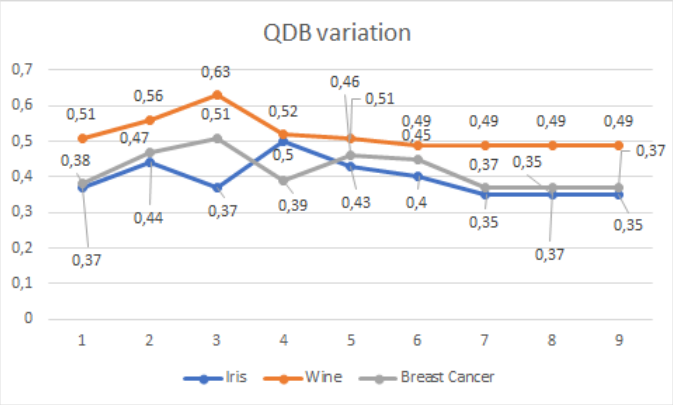


FIGURE 3.19: QDavies-Bouldin Variation

3.2.4 Computational time complexity of K-means

Let us consider the K-means problem of assigning N vectors to K clusters such that the average distance from each cluster centroid to all points of the cluster is minimized.

After randomly choosing the initial centroids, the standard method to solve this minimization problem suppose two different steps (i) assign each vector to the closest centroid and (ii) update all the centroids. This strategy is repeated until the assignment become stationary. The euclidian distance to the centroids in some N-dimensional is obtained in $\mathcal{O}(N)$. Therefore each iteration of the classical algorithm takes time $\mathcal{O}(K \times M \times N^2)$. The factor M arises since each vector is tested eventually for some reassignment. This complexity analysis is valid for classical K-means.

Let us now analyze the quantum version of the K-means algorithm. In our strategy the assignment step in quantum version is based on the application of several quantum gates. Instead to compute the euclidean distance based on the coordinates list of two different points in $\mathbb{R}^N$, in quantum version we directly compare the two quantum states. Also finding the problem of the closest centroid is optimized since is based on Grover's algorithm [Gro98].

The cluster assignment in the quantum context is no more a list of the different cluster assignments as in the classical K-means problem. The assignment is a quantum state which contains the different clusters labels correlated with the corresponding cluster assignments by using a quantum superposition.

Therefore the unsupervised quantum K-means has a complexity of order $\mathcal{O}(K \times \log(M \times N))$.

It is noticeable that this would take time polynomial time on classical computers, while with quantum computers this would takes time logarithmic.

	K-means	QK-means
Time computational complexity	$\mathcal{O}(K \times M \times N^2)$	$\mathcal{O}(K \times \log(M \times N))$

TABLE 3.4: Time computational complexity of classical & quantum K-means

3.3 Quantum Semi Non-negative Matrix Factorization

Introduction

Clustering has received a lot of attention as an important problem with many applications, and a wide range of different algorithms and methods have emerged over the years. Much interest has been shown using NMF for clustering of non-negative data [Sha+06] [CP09], it has been applied in different areas such as pattern recognition, segmentation, clustering and text mining. SNMF is an unsupervised algorithm for clustering where a data matrix is factorized into (usually) two matrices with the property that the matrix G is constrained to be non negative whereas the data matrix X and the basis vectors of F are unconstrained, in order to make the interpretation of the resulting matrices much more easier. This factorization leads to two matrices one of them contains the prototypes of a data set while the other one is a data partition matrix.

NMF [LS99] [PT94a] is equivalent to K-means clustering, more details in [DHS05]. Kim and Park [KP08] proposed sparse NMF algorithm for data clustering and they compare it with K-means. They showed that their algorithm outperforms the K-means and the ordinary NMF in terms of the consistency of the results.

Clustering could pose a challenge for classical computers especially with the current speed growing data size. If our vectors are in N dimensional vector spaces, this would take time $\mathcal{O}(polyN)$ on classical computers, while with quantum computers this would take time $\mathcal{O}(\log N)$. Thus, quantum clustering can provide exponential speed-ups for problems involving large vectors.

In this section, we are interested in the quantum version of SNMF that we can implement using quantum gradient descent. Quantum Semi Non-negative Matrix Factorization (QSNMF) gives an exponential speedup compared to its classical version. The main idea of this quantum gradient descent is to improve the objective function in order to have an algebraic form. Indeed, both homogeneous and inhomogeneous function over large dimensional spaces can be optimized by using quantum

gradient descent [Reb+16]. Then, we use multiple copies of the current quantum state $|current\rangle$ to produce multiple copies of another new quantum state $|new\rangle$ by using the objective function of the gradient that has been optimized.

In this part, we will show the different steps of Quantum Semi Non-negative Matrix Factorization (QSNMF).

Let $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \in \mathbb{R}_+^{M\times N}$, be a non-negative data matrix with M rows and N columns, here $\mathbf{x}_n \in \mathbb{R}_+^{M\times 1}$ represents the n^{th} column of X. In what follows, $\|\cdot\|$ stays for the Euclidean norm and $\|\cdot\|_F$ for the Frobenius norm. Let K be a fixed input parameter.

3.3.1 Semi Non-negative Matrix Factorization

The Semi Non-negative Matrix Factorization (SNMF) [DLJ08] relaxes the non-negativity constrains of NMF and allows the data matrix X and the loadings matrix F to have mixed signs, while restricting only the features matrix G to comprise of strictly non-negative components. More precisely we define the following optimization:

$$(F, G) = \underset{G\geq 0}{\operatorname{argmin}}\, \mathcal{L}(X, F, G), \text{ where } \mathcal{L}(X, F, G) = \|X - FG\|_F^2. \tag{3.13}$$

Note that the domain of the real valued functional $\mathcal{L}$ is $\mathbb{R}^{M\times N} \times \mathbb{R}^{M\times K} \times \mathbb{R}^{K\times N}$. The SNMF also allows to define a clustering algorithm that aims to factorize a given matrix into (usually) two matrices, where we consider one of the matrices as a matrix containing the prototypes of a data set and the other one as a data partition matrix.

The aim of clustering is to cluster the columns of X, so as to optimize the difference between X and the clustered matrix revealing significant block structure. More formally, we seek to partition the set of columns $\{1, \ldots, N\}$ into K clusters $C = \{C_1, \ldots, C_K\}$. The partitioning naturally induces clustering prototypes matrix $F \in \mathbb{R}^{M\times K}$, and a reduced partition matrix $G \in \mathbb{R}_+^{K\times N}$ specifying the cluster representation. Basically, we can interpret X to be a linear combination of some columns of F, where each column in F is a component, and each row in G contains the corresponding coefficient for each component.

To solve the optimization problem (3.13), with the only matrix constraint $G \geq 0$, we use gradient descent, which starts with a random initialization of F and after usually unfolds in two phases: at first the functional is minimized with respect to G while in the second phase, the functional is minimized with respect to F.

Let us point out several peculiar properties of the binary indicator matrix G. If the clusters $C_1, \ldots, C_K$ have distinct cluster centroids $\mathbf{f}_1, \ldots, \mathbf{f}_K$, each of the N columns of G will contain a single 1 and $K-1$ elements that are 0. Accordingly, the columns of G will sum to one

$$\sum_{k=1}^{K} g_{kn} = 1, \text{ for all column } n. \tag{3.14}$$

and its row sums will indicate the number of elements per cluster

$$\sum_{n=1}^{N} g_{kn} = n_k, \text{ for all row } k. \tag{3.15}$$

It follows that the rows of G are pairwise perpendicular because $g_{kn}g_{k'n} = \delta_{kk'}$, δ being the Kronecker symbol, which is then to say that the matrix GG^T is a diagonal matrix where: $(GG^T)_{kk'} = n_k\delta_{kk'}$ for all $k, k' = 1, \ldots, K$.

Our strategy is to write the problem in terms of polynomial optimization. Then we use several results of [Reb+16] to derive our quantum algorithm.

Problem statement

In this section, we formulate the gradient descent procedure which will be thereafter simulated in the quantum context. Here, we use some general linear algebra results and also some well known results about the NMF [LS01]. The gradient descent method unfolds in the two following phases:

$$F = F - \eta_F \circ [\nabla_F(\|X - FG\|_F^2)] \text{ and } G = G - \eta_G \circ [\nabla_G(\|X - FG\|_F^2)]. \tag{3.16}$$

Due to the symmetric role of F and G we will detail only the phase corresponding to the optimization with respect to the matrix F. Also, in order to include $G \geq 0$ at each step of the optimization process, all the negative entries of G are redefined as 0.

Lemma 1 *The gradient of $[\nabla_F(\|X - FG\|_F^2)]$ is composed of two terms: the gradient of the homogenous quadratic form with a diagonal matrix and the gradient of an inhomogenous affine form.*

Proof: 1 *Let us remark that as squared Frobenius norm of a matrix difference, the real valued functional $\mathcal{L}$ can be written as:*

$$\begin{aligned}\|X - FG\|_F^2 &= tr[(FG)^T(FG)] - 2tr[X^TFG] + tr[X^TX] \\ &= [vec(F)]^TB^TBvec(F) - 2vec(X)^TBvec(F) + tr[X^TX],\end{aligned}$$

where $B = (G^T \otimes I_M)$ is the $MN \times MK$ matrix the Kronecker product of the matrix G^T with I_M the M dimensional identity matrix. Also for any $M \times N$ matrix A, $vec(A)$ is the $MN \times 1$ column vector; the vectorization of A which obtained by stacking the columns of the matrix A on top of one another.

Note that $B^TB \in \mathbb{R}_+^{MK \times MK}$ symmetric matrix by construction. Moreover we have:

$$B^TB = (G^T \otimes I_M)^T(G^T \otimes I_M) = (G \otimes I_M)(G^T \otimes I_M) = (GG^T \otimes I_MI_M),$$

where we have used the mixed product property of the Kronecker product. Since GG^T is diagonal, see Equations (39) and (38), it follows that B^TB is diagonal. We also have:

$$\nabla_F[[vec(F)]^TB^TBvec(F)] = 2vec(F)B^TB = 2B^TBvec(F). \tag{3.18}$$

Therefore the gradient of the quadratic function acts as an operator on F. By construction $vec(X)^TBvec(F) + tr[X^TX]$ is an affine form with respect to F, we have:

$$\nabla_F[-2vec(X)^TBvec(F) + tr[X^TX]] = -2vec(X)^TB. \tag{3.19}$$

Note that $[vec(F)]^TB^TBvec(F)$ is a quadratic form with B^TB symmetric matrix and that $vec(X)^TBvec(F) + tr[X^TX]$ is an affine form which prove our claim.

The same conclusion stays for the gradient of $[\nabla_G(\|X - FG\|_F^2)]$ which will be composed of two terms: the gradient of the homogenous quadratic form with a diagonal matrix and the gradient of an inhomogenous affine form. There exists a difference between the two computations since FF^T is no more diagonal, nevertheless FF^T is diagonalizable as real symmetric matrix. Then to estimate G we will work with the diagonal form of FF^T.

3.3.2 Quantum gradient descent algorithm for SNMF

In this section, we present the gradient descent strategy to estimate F, the same strategy could be used to estimate G. Let us denote $y = vec(F) \in \mathbb{R}^{MK\times 1}$, $A = 2B^TB \in \mathbb{R}_+^{MK\times MK}$, $C^T = -2vec(X)^TB \in \mathbb{R}^{MK\times 1}$.

So the functional $\mathcal{L}(X,F,G)$ for fixed G could be written as $f : \mathbb{R}^{MK\times 1} \to \mathbb{R}$ where:

$$f(y) = \frac{1}{2}y^TAy + C^Ty + tr[X^TX], \tag{3.20}$$

where A is diagonal. To estimate G the functional $\mathcal{L}(X,F,G)$ for fixed F has the same form as the functional in Equation (3.20) with the matrix A is no more diagonal but sparse.

- *Data preparation: from a vector to a quantum state*

  To implement a quantum version of the gradient descent algorithm we assume that our data sets consist of N data vectors stored as M dimensional column vectors, i.e.: $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) \in \mathbb{R}_+^{M\times N}$. We also assume without loss of generality that $MK = 2^{n_0}$ where n_0 is an integer. A vector not satisfying this condition can be always padded to 2^{n_0} as long as the corresponding entries of A in Equation (3.20) are set to 0.

  Therefore the data matrix X elements are presented as the amplitudes of a n_0 qubit quantum state (see [RML13] [LMR13]) as follows:

$$|x\rangle = \frac{1}{|x|}\sum_{i=1}^{n_0} x_i\,|i\rangle\,,$$

  where $|i\rangle$ is the i^{th} computational basis state. In what follows, all the produced vectors y^0 by the quantum algorithm are normalized by $\sum_{i=1}^{n_0}|y_i^0|^2 = 1$, this constraint is known as spherical constraint. We make the following main assumptions:

  - A_1 We assume that there exists a quantum algorithm such that the operation $|\mathbf{0}\rangle \to |y^0\rangle$ performs n_0 qubits. Recall that the initial quantum state $|y^0\rangle$ is defined via amplitude encoding.
  - A_2 Let $j,k = 1,\ldots,MK$. We assume that there exists a quantum algorithm that performs the operation $|j,k\rangle\,|\mathbf{0}\rangle \to |j,k\rangle\,|A_{j,k}\rangle$ on $2n_0 + n_\alpha$ qubits where $A_{j,k}$ is encoded to accuracy $\alpha = 2^{-n0}$. We assume that the accuracy error is smaller than other errors involved in the optimization. To take advantage of sparsity, we also assume the following oracle, which allows to choose the non-zero matrix elements.
  - A_3 The input matrix is sparse, in order to use the sparsity of A, let $j = 1,\ldots,MK$ and $l = 1,\ldots,K$, and we assume that there exists a function $g_A(j,l)$ which compute the column index of the l-th non-zero element of row j of matrix A. Then we assume that the operation$|j,l\rangle \to |j,g_A(j,l)\rangle$ performs on 2^{n_0} qubits.

  The goal of these assumptions is to ensure an efficient simulation of e^{-iAt}, e.g.: [RML13] [LMR13].

- *Computing the gradient*

By using Lemma (1) we have:

$$f = f_{hom} + f_{inhom}, \text{ where } f_{hom} = \frac{1}{2} y^T A y \text{ and } f_{inhom} = C^T y + tr[X^T X]. \quad (3.21)$$

We want to simulate the time evolution under the respective gradient of the function f. More precisely we assume that we have multiple copies of the state $y^{(t)}$ at step t we want to define the state:

$$|y^{(t+1)}\rangle = |y^{(t)}\rangle - \eta \, |\nabla_y f(y)\rangle \quad (3.22)$$

with some given accuracy. Again by using Lemma (1) we have:

$$\nabla_y f = \nabla_y f_{hom} + \nabla_y f_{inhom}, \text{ where } \nabla_y f_{hom} = Ay \text{ and } \nabla_y f_{inhom} = C^T. \quad (3.23)$$

Therefore the simulation of the time evolution is based on quantum state exponentiation and the use of additional subroutines to simulate terms that contain inner products and outer products.

Computing $\nabla_y f_{hom}$:

Note that $f_{hom} = \frac{1}{2} y^T A y$ can be written as:

$$\langle y^T | \, A \, |y\rangle = tr(A\rho)$$

where $\rho = |y\rangle \langle y|$ is the corresponding density matrix, and tr is the trace operator. By using Lemma (1) the gradient $D = \nabla_y f_{hom}$ is an operator applied to y. Thus, the gradient operator D can be implemented as a Hamiltonian and it can be represented as:

$$D = tr\{(\rho \otimes I)A\}$$

where ρ is the joint quantum state of one copy of ρ. The matrix exponential $e^{-iA\tau}$ can be simulated efficiently on a quantum computer.

The idea is to implement the matrix exponentiation $e^{-iD\Delta t}$ [LMR14]. So, we can exponentiate D by the mapping from D to A and the exponentiation of A. For a short time Δt, multiple copies of ρ are used and matrix exponentiation of M is performed. In the reduced space of the last copy of ρ, we have:

$$tr\{e^{-iA\Delta t} \rho^{\otimes 2} e^{iA\Delta t}\} = e^{-iD\Delta t} \rho^{\otimes 2} e^{iD\Delta t} + \epsilon$$

Where ϵ is the error term that contains the error simulation of $e^{-iA\Delta t}$ and the error of the Hamiltonian simulation.

The multiplication $D \, |y\rangle = |\nabla f(y)\rangle$ can be implemented via phase estimation. We implement series of powers of the exponential in the equation above controlled on a time register in order to extract the eigenvalues of D. By conditional rotating and measuring an extra ancilla qubit [HHL09], the matrix multiplication with D is performed:

$$\sum_j \lambda_j(D) \beta_j \, |u_j(D)\rangle$$

Where $|y\rangle = \sum_j \beta_j \, |u_j(D)\rangle$ is the original state $|y\rangle$ written in the eigenbasis $\{|u_j(D)\rangle\}$ of D, and $\lambda_j(D)$ is the additional register encoding the corresponding eigenvalue in binary representation.

Computing $\nabla_y f_{inhom}$:

The optimization of functions containing terms of the form f_{inhom} can be performed via additional simulation terms and vector additions. In order to perform the inhomogeneous updates we require the following ingredients.

1. We perform additional vector additions related to the states C . For example in the gradient of the inhomogenous part requires one additional vector addition by C. Before adding the vectors we have to conditionally apply the gradient operator to the y and C^Trespectively. This requires in each iteration a number of copies of the c_{ij} and y_j. As we require in every state another copy of the states this adds another tree of states to the resource requirements, since we need to build these in parallel to the main algorithm.
2. We perform operations similar to the homogeneous step matrix-vector multiplications and matrix inversions via a conditional rotations on the eigenvalue registers and post selection. This will result in different success probabilities for the gradient operator than presented above.
3. Finally we perform a measurement in the yes/no-basis as before and perform the vector addition of the current solution and the step update.

- *Gradient descent procedure*

  Assume we have multiple copies of the state $y^{(t)}$ at step t with an accuracy $\epsilon^{(t)}$. Our purpose is to prepare the state $|y^{(t+1)}\rangle = |y^{(t)}\rangle - \eta\,|\nabla f(y)\rangle$ with an accuracy $\epsilon^{(t)} \geq \epsilon^{(t+1)}$. Let's begin by preparing the state:

$$|y^{(t)}\rangle\,(\cos\theta\,|\mathbf{0}\rangle - i\sin\theta\,|\mathbf{1}\rangle)$$

  where θ is an external parameter determining the gradient step size and an ancilla qubit is used for the gradient step in the second register.

  Conditioned on the first ancilla being in state $|\mathbf{1}\rangle$, we multiply the operator D by $|y^{(t)}\rangle$. After the conditional phase estimation, we obtain:

$$|\psi\rangle = \cos\theta\,|y\rangle\,|\mathbf{0}\rangle - i\sin\theta\sum_j \lambda_j(D)\,|\mathbf{1}\rangle\,\beta_j\,|u_j(D)\rangle$$

  Where $|u_j(D)\rangle$ are the eigenstates of D and $\lambda_j(D)$ are the eigenvalues.

  We use another ancilla initialized to $|\mathbf{1}\rangle$, and perform a joint measurement of the final two ancillas. The first ancilla is measured in the basis $|yes\rangle = \frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + i\,|\mathbf{1}\rangle)$ and $|no\rangle = \frac{1}{\sqrt{2}}(i\,|\mathbf{0}\rangle + |\mathbf{1}\rangle)$ and the second ancilla is measured in the state $|\mathbf{1}\rangle$:

$$\cos\theta\,|y^{(t)}\rangle - \xi_D\sin\theta D\,|y^{(t)}\rangle$$

  Where $\xi_D \geq \eta$ and θ can be chosen like:

$$\cos\theta = \frac{1}{\sqrt{1+\frac{\eta^2}{\xi_D^2}}}\,, \quad \sin\theta = \frac{\eta}{\xi_D\sqrt{1+\frac{\eta^2}{\xi_D^2}}}$$

  To obtain the state:

$$|y^{(t+1)}\rangle = \frac{1}{C^{(t+1)}}\,|y^{(t)}\rangle - \eta\,|\nabla f(y)\rangle$$

where η is the learning rate and:

$$(C_D^{(t+1)})^2 = 1 - 2\eta \langle y^{(t)}| D |y^{(t)}\rangle + \eta^2 \langle y^{(t)}| D^2 |y^{(t)}\rangle .$$

The success probability of the measurement of $|yes\rangle |\mathbf{1}\rangle$ is given by:

$$P = \frac{1}{2(1+\frac{\eta^2}{\xi_D^2})}(C^{(t+1)})^2$$

The quantum state $|y^{(t+1)}\rangle$ is an approximation to the classical vector $y^{(t+1)}$. The error of the updated step is $\mathcal{O}(\epsilon^{(t)} + \eta\epsilon_D^{(t)})$.

3.3.3 Complexity of Semi non-negative matrix factorization

We presented the quantum version of the gradient descent for polynomial optimization for SNMF. In order to analyze the complexity of the procedure, we first note that the sparsity factor (the number of non-zero matrix elements in each row and column) of order K of A is $\mathcal{O}(K)$ by definition of A. Moreover the L^2 norm of A is bounded by $\|A\| \leq 2KMN$, again by definition of A.

Assume the setting of optimization of polynomials. Let the task is to use the quantum gradient descent method for $T \geq 0$ steps to prepare a solution $|y^{(t+1)}\rangle$ to final accuracy $0 < \delta < 1$, with step-size schedule $\eta_>0$. Assume that a quantum algorithm exists at every step $0 < t < T$ and the error $\epsilon \geq 0$. We also assume that the space explored with the algorithms is bounded by a constant, i.e. $T\eta_t = \mathcal{O}(1)$. By using Result 3 (Multiple gradient steps) from [Reb+16], we have: that there exists a quantum algorithm for multiple gradient steps that requires $\mathcal{O}(poly(K, MN)^T)$ copies of the initial state $|y^{(0)}\rangle$. The gate requirement is $\mathcal{O}(poly(K, MN)^T n_0)$ quantum gates.

Recall that the computational complexity for the classic Semi-NMF is of order $T\mathcal{O}(K(MN))$ see [DLJ08], T recall is the maximum number of iterations. It follows that:

	SNMF	QSNMF
Time computational complexity	$\mathcal{O}(KMN)$	$\mathcal{O}(Kpolylog(MN))$

TABLE 3.5: QSNMF versus SNMF

3.4 Conclusion

In this chapter, we present the general skim of quantum prototype based clustering algorithm where we analyzed three different methods to estimate the distance for quantum prototypes based clustering algorithms. Through this analysis, we noticed that the notion of distance in quantum computing is different from the classical one. Because what counts in the quantum computation is the correlation not the real values of the distance. This analysis is so crucial as it can solve any prototype based clustering algorithm. We validate the comparison of the three different approaches by performing a series of empirical evaluations regarding the quantum distance estimation and its behavior versus the stability of finding the nearest centers in the right order.

We also implemented a new logarithmic time complexity quantum algorithm for K-means clustering. As well, we propose quantum SNMF using quantum gradient descent. We showed how an improvement in terms of complexity can be gained

through quantum computation. We used alternate optimization method for quantum gradient descent for both homogeneous and inhomogeneous part of the objective function. Our algorithm Quantum Semi Non-negative Matrix Factorization (QSNMF) leads to an exponential speed-up compared to its classical version: the complexity of the problem becomes $\mathcal{O}(Kpolylog(MN))$ instead of $\mathcal{O}(KMN)$ where M and N are the dimension of the matrix and K is the number of clusters.

Chapter 4

Quantum Multi-models clustering (collaborative approach)

Contents

4.1 Quantum Collaborative K-means

4.1.1 Classical Collaborative K-means

Clustering is one of the main exploratory task in Machine Learning. Indeed, there is a large number of existing clustering algorithms that can give very different results with the same data, and choosing between several clustering results is often difficult. This task could be solved by asking an expert to choose the most adapted method and the parameters that will work best for a specific data set. This kind of decision requires a deep knowledge of both the data to be analyzed, and also the huge number of algorithms that are available. Unfortunately, even with a good expert having a decent knowledge of both the data and the algorithms, it is still difficult to make the right choices when it comes to clustering. In order to resolve this problem, the scientific community has suggested several ways of combining the results of different algorithms. The goal was then to use the results of several methods to reach a consensus or a synthesis. These works are based on the intuition that combining results from several sources or experts can help finding a better solution for a given problem.

The general collaborative methods consists of two steps:

1. Local step: Each algorithm will process the data it has access to and produce a clustering result, e.g. a "summary" and a partition of the data set.

2. Collaborative step: The algorithms share their results and try to confirm or improve their models with the goal of finding better clustering solutions.

There are three types of collaborative clustering: vertical collaborative clustering, horizontal collaborative clustering and hybrid collaborative clustering.

- Vertical collaboration refers to the situation where several algorithms look for clusters in different data sets described by the same features but containing different objects. The collaboration is said to be vertical because the data set is split alongside the data and thus the information is exchanged vertically. It is closely related to knowledge transfer and transfer learning. It can also be applied to large data sets by splitting them and processing each subset with different algorithms exchanging their information.

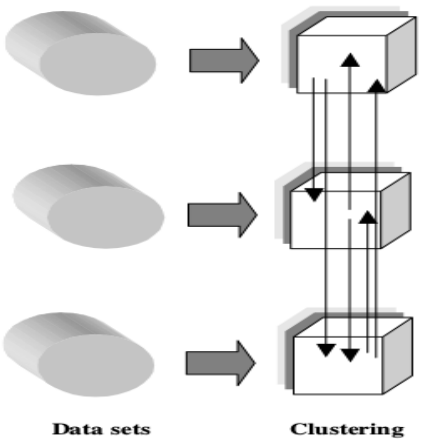


FIGURE 4.1: Vertical collaborative clustering

- Horizontal collaboration is about several algorithms working on the same objects that may be represented in different feature spaces. The collaboration is said to be horizontal because the data set is split alongside the features and thus the information is exchanged horizontally. It can be applied to multi-view clustering, multi-expert clustering, clustering of high dimensional data, or multi-scale clustering.

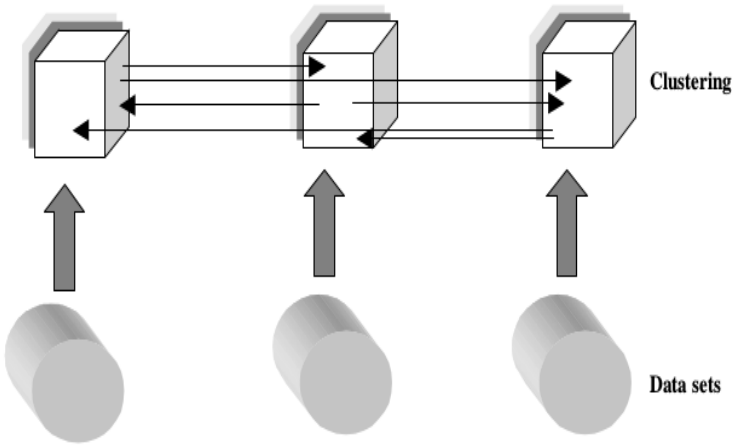


FIGURE 4.2: Horizontal collaborative clustering

- Hybrid collaborative clustering is when both horizontal and vertical approaches are used at the same time.

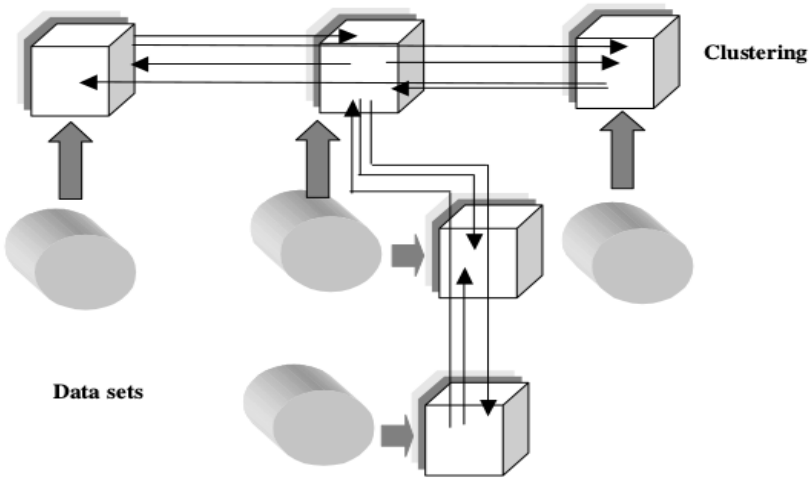


FIGURE 4.3: Hybrid collaborative clustering

Generally speaking, horizontal collaboration and vertical collaboration are both difficult but in different ways: for horizontal collaboration, the difficulty comes from the fact that the algorithms may not be in the same feature space and thus can not exchange prototypes or information based on the features of the data. However, since it is the same data that are split between different sites, comparing the partitions found by different algorithms is a little bit easy. In the case of vertical collaboration, the difficulty lies precisely in the fact that without any common data, comparing the partitions of different algorithms is impossible and voting systems can not be used. But, given that all collaborators are prototype-based algorithms and that the data distribution between the different sites are similar enough, vertical collaboration makes it very easy to exchange prototypes and other information on the structures of the clusters between the different collaborators. However, the clusters must first be mapped, and this task can be difficult if the distributions are a bit different from one site to another, if the number of cluster is different, or if the algorithms found very different clusters.

The aim of collaborative clustering is to make different clustering methods collaborate, in order to reach an improvement on each partitioning of some dataset. One of the first collaborative clustering algorithm was introduced in 2002 by Pedrycz [Ped02; Ped05] under the name "*Collaborative Fuzzy Clustering*" (CoFC). This method was designed for the specific case of distributed data where the information cannot be shared between the different sites. This method was based on a modified version of the Fuzzy C-Means algorithm [Ped02]. In collaborative clustering [Cor+18], the group of algorithms solve together learning problems, affecting an individual task to each learner. Interactions are recurrent between each algorithms, responsibility is collective, the action of each algorithm is geared to the performance of the group and vice versa.

The hope is that by collaborating several clustering solutions, each one with its own bias and imperfections, we will achieve a better overall solution. There are three main types of collaboration: horizontal, vertical and hybrid collaboration [Cor+18; GB10]. Horizontal collaborative clustering: all datasets describe the same observations, so all the collaborative datasets have the same number of observations but a different number of variables. Vertical collaborative clustering: all datasets have the same variables. Hybrid collaborative clustering: when we use the two approaches vertical and horizontal collaborative clustering at the same time.

As K-means is not stable, so in our approach we generate L instances of K-means and we make a collaboration between these different instances in the context of quantum learning.

Suppose that we have L instances of K-means algorithm. The general collaborative clustering scheme consists of two phases:

- Phase 1: Generating a local K-means algorithm L times on the data set X. The number of clusters would be the same for all data sets. K-means identifies K cluster centers for each data set. The objective function to minimize on the first phase is:

$$R^{(l)} = \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)}\right)^2 \|x_n^{(l)} - w_k^{(l)}\|^2, \tag{4.1}$$

  where l refers to the data set where the local cluster analysis is performed $l = 1, 2, \ldots, L$.

- Phase 2: After the local phase where each instance gets an initial set of cluster centroids $w_k^{(l)}, l = 1, 2, \ldots, L$. In the second phase, we perform a collaboration between the solutions of each instance algorithm. The goal is that after the collaboration, if an observation of the $X^{(l)}$-th data set is projected onto the l'-th centroid of the l-th instance algorithm, then the same observation of the $X^{(l')}$-th data set is projected on the same l' centroid. The collaborative function to minimize is:

$$C^{(l)} = \sum_{\substack{l'=1, \\ l' \neq l}}^{L} \beta_{l,l'} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)} - g_{nk}^{(l')}\right)^2 \|w_k^{(l)} - w_k^{(l')}\|^2, \tag{4.2}$$

  where $\beta_{l,l'}$ is the weight of the pairwise collaborative $C^{(l)}$ term between the instances l and l', and it's a non-negative coefficient that represents the intensity of collaboration whose value is provided by the user.

Therefore, the global objective function to minimize is given in the following form:

$$R_{Coll} = \sum_{l=1}^{L} \left(R^{(l)} + C^{(l,l')} \right). \tag{4.3}$$

The objective function R_{Coll} consists of two terms. The first term represents the sum of objective functions used by L instances of K-means clustering. The second term reflects the impact of the clustering structures found in other instances. To obtain the update rule for the prototypes let us consider:

$$\begin{aligned} R_{Coll}^{(l)} = & \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)} \right)^2 \| x_n^{(l)} - w_k^{(l)} \|^2 \\ & + \sum_{\substack{l'=1, \\ l' \neq l}}^{L} \beta_{l,l'} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)} - g_{nk}^{(l')} \right)^2 \| w_k^{(l)} - w_k^{(l')} \|^2. \end{aligned} \tag{4.4}$$

In the calculations of the prototypes, the necessary condition for the minimum of the objective function is in the form:

$$\frac{\partial R_{Coll}^{(l)}}{\partial w_{mk}^{(l)}} = 0$$

Recall that, by definition we have:

$$\| \mathbf{x}_n^{(l)} - \mathbf{w}_k^{(l)} \|^2 = \sum_{m=1}^{M} (x_{mn}^{(l)} - w_{mk}^{(l)})^2. \tag{4.5}$$

By using (4.5) and by differentiation with respect to $w_{mk}^{(l)}$ we get:

$$\begin{aligned} \frac{\partial R_{Coll}^{(l)}}{\partial w_{mk}^{(l)}} = & -2 \sum_{n=1}^{N} \left(g_{nk}^{(l)} \right)^2 (x_{mn}^{(l)} - w_{mk}^{(l)}) \\ & - 2 \sum_{\substack{l'=1, \\ l' \neq l}}^{L} \sum_{n=1}^{N} \beta_{l,l'} (w_{mk}^{(l)} - w_{mk}^{(l')}) \left(g_{nk}^{(l)} - g_{nk}^{(l')} \right)^2 \end{aligned} \tag{4.6}$$

Therefore, the straightforward computation gives the following update rule for centroids:

$$w_k^{(l)} = \frac{\sum_{n=1}^{N} \left(g_{nk}^{(l)} \right)^2 x_n^{(l)} + \sum_{\substack{l'=1, \\ l' \neq l}}^{L} \sum_{n=1}^{N} \beta_{l,l'} \left(g_{nk}^{(l)} - g_{nk}^{(l')} \right)^2 x_n^{(l)}}{\sum_{n=1}^{N} \left(g_{nk}^{(l)} \right)^2 + \sum_{\substack{l'=1, \\ l' \neq l}}^{L} \sum_{n=1}^{N} \beta_{l,l'} \left(g_{nk}^{(l)} - g_{nk}^{(l')} \right)^2}, \tag{4.7}$$

where $k = 1, \ldots, K$.

Collaborative K-means clustering algorithm is presented in Algorithm 5.

Algorithm 5: Collaborative Clustering algorithm

Phase 1: Local phase
foreach $X^{(l)}$, $l = 1$ *to* L **do**
Minimize the objective function of K-means:

$$R^{(l)} = \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)}\right)^2 \|x_n^{(l)} - w_k^{(l)}\|^2$$

end
Phase 2: Collaboration phase
foreach $X^{(l)}$, $l = 1$ *to* L **do**
Minimize the objective function of collaborative clustering:

$$R_{coll}^{(l)} = \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)}\right)^2 \|x_n^{(l)} - w_k^{(l)}\|^2 + \sum_{\substack{l'=1, \\ l' \neq l}}^{L} \beta_{l,l'} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(g_{nk}^{(l)} - g_{nk}^{(l')}\right)^2 \|w_k^{(l)} - w_k^{(l')}\|^2$$

Update the centroids by using Equation (4.7).
end

4.1.2 Quantum Collaborative K-means

In this section, we study the collaboration between several clustering results, in particular the collaboration between several models of quantum K-means [Ben+20].

The general quantum collaborative clustering scheme consists of two phases:

Phase 1: Generating $l = 1, 2, \ldots, L$ clusters without collaboration, using a local QK-means algorithm on the dataset as we explained previously. QK-means minimizes the following objective function $R_Q^{(l)}$.

$$R_Q^{(l)} = d^2 r_{max}^4 \, |(G_k^2)^T\rangle \, | \, \langle \phi^{(l)} | \psi^{(l)} \rangle \, |^2 \tag{4.8}$$

where

$$|G_k^2\rangle = \frac{1}{\sqrt{|C_k|}} \sum_{n=1}^{N} g_{nk}^2 \, |n\rangle$$

and

$$g_{nk}^2 = \begin{cases} 1 & \text{if } x_n \in C_k \\ 0 & \text{otherwise.} \end{cases}$$

Phase 2: In the second phase, we perform a quantum collaboration between different clusters. In collaborative QK-means algorithm, when a data instance is presented from the clustering l, the optimization is done to minimize the distance between that instance and the centroids of each local QK-means iteration $l' \neq l$. Therefore the pairwise collaborative $C_Q^{(l)}$ term between the QK-means clustering l and l' is:

$$C_Q^{(l)} = d'^2 r'^4_{max} \times \sum_{\substack{l'=1,\\ l'\neq l}}^{L} \beta_{l,l'} \left| \left((G_k^{(l)} - G_k^{(l')})^2 \right)^T \right\rangle \left| \langle \gamma^{(l)} | \delta^{(l')} \rangle \right|^2 \tag{4.9}$$

Where $|\gamma^{(l)}\rangle$ and $|\delta^{(l')}\rangle$ are the two states that are associated to the vectors $w^{(l)}$ and $w^{(l')}$ .The preparation of the data and the construction of the states $|\gamma^{(l)}\rangle$ and $|\delta^{(l')}\rangle$ follows the same steps as explained previously for the states $|\psi\rangle$ and $|\phi\rangle$.

$$|\gamma^{(l)}\rangle = \frac{1}{\sqrt{d'}} \sum_c |c\rangle \left(\sqrt{1 - \frac{|w_c^{(l)}|^2}{r'^2_{max}}} e^{-i\alpha_c} |0\rangle + \frac{w_c^{(l)}}{r'_{max}} |1\rangle \right) |1\rangle$$

$$|\delta^{(l')}\rangle = \frac{1}{\sqrt{d'}} \sum_c |c\rangle |1\rangle \left(\sqrt{1 - \frac{|w_c^{(l')}|^2}{r'^2_{max}}} e^{-i\beta_c} |0\rangle + \frac{w_c^{(l')}}{r'_{max}} |1\rangle \right)$$

Where $c = \{1, ..., k\}$, and r'_{max} is an upper bound on the maximum value of any feature in the dataset. The input vectors are d'-sparse.

For simplification, let's assume in what follows that $Z = d^2 r^4_{max}$ and $Z' = d'^2 r'^4_{max}$. So, the objective function to minimize is:

$$R_{QColl}^{(l)} = Z \left| (G_k^2)^T \right\rangle \left| \langle \phi^{(l)} | \psi^{(l)} \rangle \right|^2 + Z' \sum_{\substack{l'=1,\\ l'\neq l}}^{L} \beta_{l,l'} \left| \left((G_k^{(l)} - G_k^{(l')})^2 \right)^T \right\rangle \left| \langle \gamma^{(l)} | \delta^{(l')} \rangle \right|^2 \tag{4.10}$$

The objective function presented in Equation (4.10) is composed of two terms, the first one corresponds to the local phase which is the sum of L instances of QK-means and the second one corresponds to the collaborative phase.

The idea of collaboration is to make the algorithms share their results with the goal of getting a better clustering results, and this can be done by adding a collaborative term to constraint the similarity between clustering elements. Where $\beta_{l,l'}$ is a non-negative matrix coefficient that represents the intensity of collaboration whose value is provided by the user.

Finally, the computation of the gradient of $R_{QColl}^{(l)}$ gives the following update rule of the centroid [Reb+19]:

$$|w_k^{(l)}\rangle = \frac{Z \left| (G_k^2)^T X \right\rangle + Z' \sum\limits_{\substack{l'=1,\\ l'\neq l}}^{L} \beta_{l,l'} \left| \left((G_k^{(l)} - G_k^{(l')})^2 \right)^T X \right\rangle}{Z \left| (G_k^2)^T \right\rangle + Z' \sum\limits_{\substack{l'=1,\\ l'\neq l}}^{L} \beta_{l,l'} \left| \left((G_k^{(l)} - G_k^{(l')})^2 \right)^T \right\rangle} \tag{4.11}$$

Quantum collaborative K-means clustering [Ben+20] is presented in Algorithm 6.

Algorithm 6: Quantum Collaborative Clustering algorithm

Phase 1: Local phase
foreach $X^{(l)}$, $l = 1$ *to* L **do**
Minimize the objective function of QK-means:

$$R_Q^{(l)} = Z\,|(G_k^2)^T\rangle\,|\,\langle\phi^{(l)}|\psi^{(l)}\rangle\,|^2$$

end
Phase 2: Collaboration phase
foreach $X^{(l)}$, $l = 1$ *to* L **do**
Minimize the objective function of collaborative clustering:

$$R_{QColl}^{(l)} = Z\,|(G_k^2)^T\rangle\,|\,\langle\phi^{(l)}|\psi^{(l)}\rangle\,|^2 + Z'\sum_{\substack{l'=1,\\ l'\neq l}}^{L}\beta_{l,l'}\,|\left((G_k^{(l)} - G_k^{(l')})^2\right)^T\rangle\,|\,\langle\gamma^{(l)}|\delta^{(l')}\rangle\,|^2$$

Update the centroids using Equation (4.11).
end

4.1.3 Experimental results

The quantum version of collaborative K-means was tested on three real world datasets available for public use in the UCI Machine learning repository [DG17], namely: iris, wine and breast cancer.

Clustering through collaborative quantum K-means

We used three different datasets to compare the quantum Davies-Bouldin (QDB) index for quantum K-means before and after collaboration. These results are represented in Table 4.1. For example in the Iris dataset, we can notice that QDB of the collaboration shows a decreasing behaviour compared to QK-means$_2$ which means that the quality of clustering increases. However, if we compare the collaboration with QK-means$_1$, we see that QDB increases which is normal because the collaboration isn't all the time beneficial for both collaborators it could be good for one as in our case QK-means$_2$ and doesn't improve the quality of the other collaborator QK-means$_1$.

Dataset	QK-means$_1$	QK-means$_2$	QK-means$_{coll}$
Iris	[0.35,0.57]	[0.63, 1.36]	[0.47, 0.84]
wine	[0.49, 0.56]	[0.43, 0.60]	[0.47, 0.59]
Breast Cancer	[0.48, 0.77]	[0.53, 0.97]	[0.52, 0.66]

TABLE 4.1: QK-means & QK-means$_{coll}$ using QDB index

Figure 2 and 3 represent the projection of iris dataset using the principal component analysis. These figures shows the classical collaborative K-means and the quantum collaborative K-means respectively. We can notice that the algorithm of collaborative QK-means has identified the different clusters which are different from each other. Therefore, the quantum collaborative K-means gives a good classification

just like it's classical version, the main difference resides in that with a quantum version the clustering can be done in a time faster than the classical version.

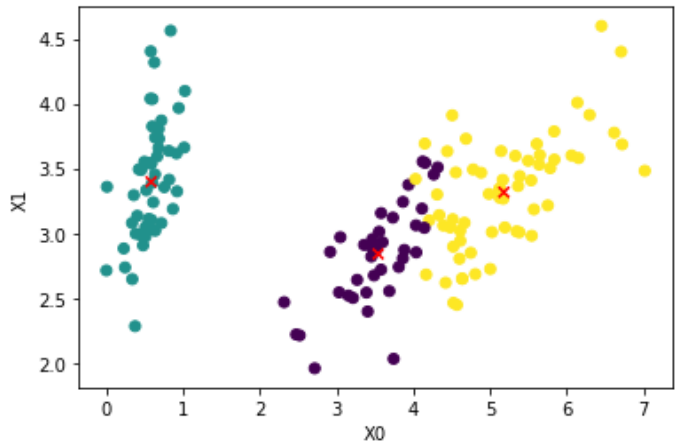


FIGURE 4.4: Collaborative *K*-means clustering

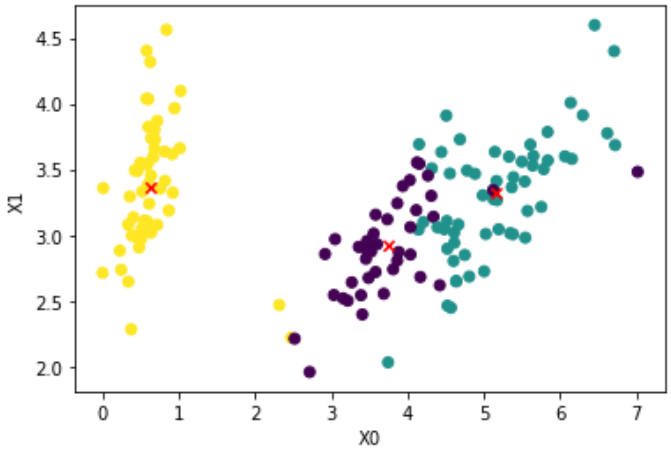


FIGURE 4.5: Collaborative *QK*-means clustering

4.1.4 Computational time complexity of collaborative K-means

As we have seen before, to assign N vectors to K clusters the algorithm of K-means takes time $\mathcal{O}(K \times M \times N^2)$. In the case of collaboration of L instances of K-means algorithm the complexity is therefore $\mathcal{O}(K \times M \times L \times N^2)$.

Let us now analyze the quantum version of the collaborative K-means algorithm; as we have already explained the quantum K-means has a complexity of order $\mathcal{O}(K \times \log(M \times N))$, therefore the collaborative version of the quantum K-means we get at most $\mathcal{O}(K \times L \times \log(M \times N))$.

Compared to the classical version of collaborative K-means, we notice that we have an exponential speed up. Indeed, classical machine learning algorithms take time polynomial in manipulating and classifying large numbers of vectors in high dimensional spaces. On the contrary of quantum computers that can manipulate high dimensional vectors in large tensor product spaces in logarithmic time.

	K-means$_{coll}$	QK-means$_{coll}$
Time computational complexity	$L\mathcal{O}(K \times M \times N^2)$	$L\mathcal{O}(K \times \log(M \times N))$

TABLE 4.2: Time computational complexity of classical & quantum collaborative k-means

4.2 Quantum Collaborative Semi Non-negative Matrix Factorization

4.2.1 Classical Collaborative Semi Non-negative Matrix Factorization

Non-negative matrix factorizations are more and more used as tools for unsupervised classification and visualization of multidimensional datasets, as they allow for the projection of these large data onto small, generally two-dimensional spaces. In this work, we study the collaboration between several clustering results, in particular the collaboration between several models of Non-negative Matrix Factorization (NMF). Each dataset is clustered through an NMF-based clustering approach. We are interested in horizontal collaboration that is very close to multi-view clustering because the data to be collaborated is the same but described by different variables.

4.2.2 General principle

In the context of horizontal collaborative learning, we consider a finite number of L clustering views $\mathcal{A}^{(1)}, ..., \mathcal{A}^{(L)}$ working on different attributes of a data set made of M numerical attributes $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^{M\times 1}$. We note $X^{(l)} = \{\mathbf{x}_1^{(l)}, ..., \mathbf{x}_N^{(l)}\}, \mathbf{x}_n^{(l)} \subseteq \mathbf{x}_n$ the subset of attributes processed by a given view $\mathcal{A}^{(l)}$. We are in the discrete case and each observation $\mathbf{x}_n^{(l)}$ is a vector belonging to a M_l-dimensional euclidean feature space $\mathbb{R}^{M_l\times 1}$. Each clustering view $\mathcal{A}^{(l)}$ uses a NMF factorization. Therefore the corresponding functional for the $\mathcal{A}^{(l)}$ algorithm writes:

$$\mathcal{L}^{(l)}(F^{(l)}, G^{(l)}) = \|X^{(l)} - F^{(l)}(G^{(l)})^T\|^2 \tag{4.12}$$

Here $F^{(l)} = (\mathbf{f}_1^{(l)}, \mathbf{f}_2^{(l)}, \ldots, \mathbf{f}_K^{(l)}) \in \mathbb{R}_+^{M_l\times K}$ represents the matrix of cluster centroids and $G^{(l)} = (\mathbf{g}_1^{(l)}, \mathbf{g}_2^{(l)}, \ldots, \mathbf{g}_K^{(l)}) \in \mathbb{R}_+^{N\times K}$ represents the matrix clusters indicators obtained by the $\mathcal{A}^{(l)}$.

From there, in the collaborative NMF algorithm, when a data unit is presented from the view l, the optimization is done to minimize the distance between that unit and the centroids of each local NMF view $l' \neq l$. We would like that after the collaboration: if an observation of the $X^{(l)}$-th data set is projected onto the k-th centroid of the l-th NMF view, then the same observation viewed as an observation in the $X^{(l')}$-th data set is projected on the same unit k centroid. In order to define the functional in the collaborative part we introduce the matrix $D^{(l)}$ of the euclidean distances of each data in X to the set of centroids $F^{(l)}$ in local instance l-th, more specifically we have $D_{kn}^{(l)} = \|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\|$. Therefore we define the pairwise collaborative $C_{l,l'}$ term between the l-th and l'-th NMF's as follows:

$$C_{l,l'}(F^{(l)}, G^{(l)}) \quad = \quad \|(G^{(l)} - G^{(l')}) \circ D^{(l)}\|_F^2. \tag{4.13}$$

The idea of collaboration is to add a collaborative matching term to constraint the similarity between clustering elements of different databases, has obviously showed

its ability to produce improved clustering solutions. To exchange the clustering information, all local solutions in the collaboration process share common structures see [Ped02]. Therefore the set of centroids $F^{(l)}$ is estimated iteratively by minimizing the objective function:

$$\mathcal{C}(F^{(l)}, G^{(l)}) = \mathcal{L}^{(l)}(F^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot C_{l,l'}(F^{(l)}, G^{(l)}) \tag{4.14}$$

Here $\mathcal{L}^{(l)}$ is the l^{th} local term defined in Equation (4.12), $\beta = (\beta_{l,l'})_{l,l'}$ is the weight of the collaboration fixed by the user satisfying $\sum_{l' \neq l} \beta_{l,l'} = 1$. We could also define a global functional as follows:

$$\mathcal{C}(F, G) = \sum_{l=1}^{L} \left(\mathcal{L}^{(l)}(F^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot C_{l,l'}(F^{(l)}, G^{(l)}) \right) \tag{4.15}$$

Let us first remark that the cost function defined in Equation (4.15) is continuously first differentiable in all variables. Therefore a minimum always exists and could be found by nonlinear programming. The minimization of the global functional should satisfy the constraints: $(G^{(l)})^T G^{(l)} = I$ for all $l \in \{1, \ldots, L\}$ since $G^{(l)}$ is a partition matrix. Therefore a minimum always exists and could be found by nonlinear programming. The minimization of the global functional should satisfy the constraints: $(G^{(l)})^T G^{(l)} = I$ for all $l \in \{1, \ldots, L\}$ since $G^{(l)}$ is a partition matrix. We compute the gradient of the functional in Equation (5.17) see Appendix Section .1.7. Having obtained both gradients for the gradient descent algorithm the update formulates for collaborative NMF writes:

$$\begin{cases} G^{(l)} = G^{(l)} - \eta_{G^{(l)}} & \circ \quad [\nabla_{G^{(l)}} \mathcal{C}(l)], \\ F^{(l)} = F^{(l)} - \eta_{F^{(l)}} & \circ \quad [\nabla_{F^{(l)}} \mathcal{C}(l)]. \end{cases} \tag{4.16}$$

We then obtain the multiplicative rules:

$$F^{(l)} = F^{(l)} \circ \frac{X \left(G^{(l)}\right)^T + \sum_{l' \neq l} \beta_{l,l'} X \left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T}{F^{(l)} G^{(l)} \left(G^{(l)}\right)^T + \sum_{l' \neq l} \beta_{l,l'} \mathbf{1}_{M \times N} \left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \circ F^{(l)}}, \tag{4.17}$$

and

$$G^{(l)} = G^{(l)} \circ \frac{\left(F^{(l)}\right)^T X + \sum_{l' \neq l} \beta_{l,l'} \left[G^{(l')} \circ (D^{(l)})^{\circ 2}\right]}{\left(F^{(l)}\right)^T F^{(l)} G^{(l)} + \sum_{l' \neq l} \beta_{l,l'} G^{(l)} \circ \left(D^{(l)}\right)^{\circ 2}}. \tag{4.18}$$

The division here is component wise matrix division and where $^{\circ 2}$ means element-wise power.

The formulaes in Equation (4.17) and Equation (4.18) correspond to the case of classic NMF. By relaxing the constraints on the positivity of X and $F^{(l)}$ we obtain the semi NMF case. We have in the following multiplicative rule to update $F^{(l)}$ while $G^{(l)}$ is fixed.

$$F^{(l)} = X G^{(l)} ((G^{(l)})^T G^{(l)})^{-1} \tag{4.19}$$

Note $(G^{(l)})^T G^{(l)}$ is a $K \times K$ positive semidefinite matrix. The inversion of this small matrix is trivial. In the case of $G^{(l)}$ the formula is more involved. We first introduce the notations $A^+ = (|A| + A)/2$ and $A^- = (|A| - A)/2$, where all the involved

computations are component wise element operations. We also denote

$$W^{(l)} = \left(F^{(l)}\right)^T X + \sum_{l' \neq l} \beta_{l,l'} \left[G^{(l')} \circ (D^{(l)})^{\circ 2}\right]$$

and

$$H^{(l)} = \left(F^{(l)}\right)^T X + \sum_{l' \neq l} \beta_{l,l'} \left[G^{(l')} \circ (D^{(l)})^{\circ 2}\right] \left(F^{(l)}\right)^T F^{(l)} G^{(l)} + \sum_{l' \neq l} \beta_{l,l'} G^{(l)} \circ \left(D^{(l)}\right)^{\circ 2}.$$

We get the following update rule of $G^{(l)}$ while $F^{(l)}$ is fixed:

$$G^{(l)} = G^{(l)} \circ \sqrt{\frac{(W^{(l)})^+ (H^{(l)})^-}{W^{(l)})^- (H^{(l)})^+}}. \tag{4.20}$$

Here all the database share the same units but described by different attributes. In this case, the number and the size of centroid vectors for all the NMF factorizations will be the same. That's why the objective function of the classical NMF has been modified in order to introduce this constraint on the different factorizations during the collaboration step. The collaborative term is expressed by $C_{l,l'}(F)$ which guaranty that the structure of other databases matches with the current database. Our modified version of the NMF algorithm for horizontal collaboration is shown in Algorithm 7 below.

Algorithm 7: NMF horizontal collaboration Algorithm

Initialization: Initialize all the centroids sets F randomly and β.
Local step:
forall *algorithms* $\mathcal{A}^{(l)}$ **do**
  Minimize the objective function of the NMF.
end
Collaborative step:
forall *algorithms* $\mathcal{A}^{(l)}$ **do**
  Update the partitions matrix of all algorithms (4.19).
  Update the centroids of all algorithms (4.20).
end

4.2.3 Quantum Collaborative Semi Non-negative Matrix Factorization

We have already explained in the last two sections the different steps of quantum semi-non negative matrix factorization, and the classic collaborative semi non-negative matrix factorization. This part will be devoted to the quantum collaborative version of the semi non-negative matrix factorization. As we can notice, the collaborative function is composed of two terms: the first one is the local term and the second one is the collaborative term. The objective function to minimize for the (l) instance is:

$$\mathcal{C}(F^{(l)}, G^{(l)}) = \mathcal{L}^{(l)}(F^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot C_{l,l'}(F^{(l)}, G^{(l)}) \tag{4.21}$$

We have already explained the quantum version of the first term. In what follows, we will show the quantum version of the collaborative term:

$$\mathcal{L}^{(l)}(F^{(l)}, G^{(l)}) = \|X^{(l)} - F^{(l)}(G^{(l)})^T\|_F^2 \tag{4.22}$$

And

$$C_{l,l'}(F^{(l)}, G^{(l)}) = \|(G^{(l)} - G^{(l')}) \circ D^{(l)}\|_F^2. \tag{4.23}$$

In words the $C_{l,l'}$ term is the weighted sum of Euclidean distances between the observed data $x_n^{(l)}$ and all the centroids in $F^{(l)}$, with the weight defined by $G^{(l)} - G^{(l')}$. Note that when $G^{(l)}$ and $G^{(l')}$ agree then the collaborative term is equal to zero and only the local l-th NMF is take into account in the optimization. Note also that by construction $C_{l,l'} \geq 0$ and by definition $D^{(l)} \geq 0$ therefore $C_{l,l'}(F^{(l)}, G^{(l)}) = 0$ if and only if $G^{(l)} - G^{(l')} = 0$ i.e. the data $\mathbf{x}_n^{(l)}$ in the l'-th data set is projected on the same centroid. By computing the Frobenius norm in Equation's (4.22) and (4.23) we have:

$$\mathcal{L}^{(l)}(F^{(l)}, G^{(l)}) = Tr[X^T X - 2X^T F^{(l)}(G^{(l)})^T + G^{(l)}(F^{(l)})^T(F^{(l)})(G^{(l)})^T],$$
$$C_{l,l'}(F^{(l)}, G^{(l)}) = Tr[((G^{(l)} - G^{(l')}) \circ D^{(l)})^T((G^{(l)} - G^{(l')}) \circ D^{(l)})].$$

4.2.4 Problem Statement in Quantum Setting

In this section we formulate the gradient descent procedure which will be thereafter simulated in quantum context. We make uses here of some general linear algebra results and also some well know results about the NMF [LS01]. The gradient descent method unfolds in two following phases:

$$\begin{cases} F^{(l)} = F^{(l)} - \eta_{F^{(l)}} & \circ \quad [\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))], \\ G^{(l)} = G^{(l)} - \eta_{G^{(l)}} & \circ \quad [\nabla_{G^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))]. \end{cases} \tag{4.25}$$

In the collaborative case the role of $F^{(l)}$ and $G^{(l)}$ are no more symmetric role and we will detail both gradients.

Lemma 2 *The gradient of $\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))$ is composed of two terms: the gradient of the homogenous quadratic form with a diagonal matrix and the gradient of an inhomogenous affine form.*

Proof: 2 *Note that the gradient of $\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))$ is the addition of $\nabla_{F^{(l)}}(\|X^{(l)} - F^{(l)}(G^{(l)})^T\|_F^2)$ and $[\nabla_{F^{(l)}} C_{l,l'}(F^{(l)}, G^{(l)})$. We now compute $\nabla_{F^{(l)}}(\|X^{(l)} - F^{(l)}(G^{(l)})^T\|_F^2)$. Recall that the $\nabla_{F^{(l)}}(\|X^{(l)} - F^{(l)}(G^{(l)})^T\|_F^2)$ is obtained in Lemma 1 and is given by*

$$= [vec(F^{(l)})]^T(B_F^{(l)})^T(B_F^{(l)})vec((F^{(l)})) - 2vec(X)^T(B_F^{(l)})vec((F^{(l)})) + tr[X^T X],$$

where $B_F^{(l)} = (G^T \otimes I_M)$ is the $MN \times MK$ matrix the Kronecker product of the matrix $(G^{(l)})^T$ with I_M the M dimensional identity matrix. Also for any $M \times N$ matrix A, $vec(A)$ is the $MN \times 1$ column vector the vectorization of A which obtained by stacking the columns of the matrix A on top of one another.

Note that as in Lemma 1 $(B_F^{(l)})^T(B_F^{(l)}) \in \mathbb{R}_+^{MK \times MK}$ is symmetric matrix by construction. Moreover we have:

$$(B_F^{(l)})^T(B_F^{(l)}) = G^{(l)}(G^{(l)})^T \otimes I_M I_M)$$

where we have used the mixed-product property of the Kronecker product. Since $G^{(l)}(G^{(l)})^T$ is diagonal, it follows that $(B_F^{(l)})^T(B_F^{(l)})$ is diagonal. Again as in Lemma Lemma 1 note that $[vec(F^{(l)})]^T(B_F^{(l)})^T(B_F^{(l)})vec(F^{(l)})$ is a quadratic form with $(B_F^{(l)})^T(B_F^{(l)})$ symmetric matrix and that $vec(X)^T(B_F^{(l)})vec(F^{(l)}) + tr[X^T X]$ is an affine in $F^{(l)}$.

Analogously we compute $\nabla_{G^{(l)}}(\|X^{(l)} - F^{(l)}(G^{(l)})^T\|_F^2)$ *and we get that*

$$= [vec(G^{(l)})]^T(B_G^{(l)})^T(B_G^{(l)})vec((F^{(l)})) - 2vec(X)^T(B_G^{(l)})vec((F^{(l)})) + tr[X^TX],$$

where $B_G^{(l)} = (F^T \otimes I_N)$ *is the* $MN \times KN$ *matrix the Kronecker product of the matrix* $(F^{(l)})^T$ *with* I_N *the* N *dimensional identity matrix.*

As before note that

$$[vec(G^{(l)})]^T(B_G^{(l)})^T(B_F^{(l)})vec(G^{(l)})$$

is a quadratic form with $(B_G^{(l)})^T(B_G^{(l)})$ *symmetric matrix and that* $vec(X)^T(B_G^{(l)})vec(G^{(l)}) + tr[X^TX]$ *is an affine in* $G^{(l)}$.

We now compute the collborative gradients terms, we start with $[\nabla_{F^{(l)}}C_{l,l'}(F^{(l)}, G^{(l)})$. *We compute:*

$$\nabla_{F^{(l)}}Tr[((G^{(l)} - G^{(l')}) \circ D^{(l)})^T((G^{(l)} - G^{(l')}) \circ D^{(l)})]$$

Recall that $D^{(l)}$ *is constructed by using* $F^{(l)}$. *Note that we have by definition:*

$$\begin{aligned} \nabla_{F^{(l)}}C_{l,l'}(F^{(l)}, G^{(l)}) &= -2X\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \\ &+ \mathbf{21}_{M\times N}\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \circ F^{(l)}, \end{aligned} \tag{4.26}$$

where $\mathbf{1}_{M\times N}$ *is is an* $M \times N$ *one's matrix and where* $^{\circ 2}$ *means element-wise power of the matrix. We also have that:*

$$\nabla_{G^{(l)}}C_{l,l'}(F^{(l)}, G^{(l)}) = 2(G^{(l)} - G^{(l')}) \circ (D^{(l)})^{\circ 2}. \tag{4.27}$$

We need now to write these gradients by using the forms $vec(F^{(l)})$, $vec(G^{(l)})$ *respectively. Therefore we have:*

$$\begin{aligned} \nabla_{F^{(l)}}C_{l,l'}(F^{(l)}, G^{(l)}) &= -2vec\left(X\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T\right) \\ &+ 2vec\left(\mathbf{1}_{M\times N}\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T\right) \circ vec(F^{(l)}), \end{aligned} \tag{4.28}$$

We also have that:

$$\nabla_{G^{(l)}}C_{l,l'}(F^{(l)}, G^{(l)}) = 2vec(G^{(l)} - G^{(l')}) \circ vec\left(D^{(l)})^{\circ 2}\right). \tag{4.29}$$

Note that the gradients in the Equations do not contains quadratic terms in $vec(F^{(l)}), vec(F^{(l)})$ *respectively.*

To conclude the proof it remains to precise that the quadratic term for $\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))$ *is given by:*

$$[vec(F^{(l)})]^T(B_F^{(l)})^T(B_F^{(l)})vec(F^{(l)}) \tag{4.30}$$

with $(B_F^{(l)})^T(B_F^{(l)})$ *symmetric matrix, and the affine term is given by:*

$$\begin{aligned} &vec(X)^T(B_F^{(l)})vec(F^{(l)}) \\ &+tr[X^TX] - 2vec\left(X\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T\right) \\ &+2vec\left(\mathbf{1}_{M\times N}\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T\right) \circ vec(F^{(l)}) \end{aligned} \tag{4.31}$$

Analogously for $\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))$ *we have the quadratic term:*

$$[vec(G^{(l)})]^T (B_G^{(l)})^T (B_F^{(l)}) vec(G^{(l)}) \tag{4.32}$$

with $(B_G^{(l)})^T (B_G^{(l)})$ *symmetric matrix, while the affine term is given by:*

$$\begin{array}{l} vec(X)^T (B_G^{(l)}) vec(G^{(l)}) + tr[X^T X] \\ 2vec(G^{(l)} - G^{(l')}) \circ vec\left(D^{(l)})^{\circ 2}\right). \end{array} \tag{4.33}$$

This Lemma shows that both gradients $\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))$ and $\nabla_{F^{(l)}}(\mathcal{C}(F^{(l)}, G^{(l)}))$ respectively are of the form

$$[vec(F^{(l)})]^T A_F vec(F^{(l)}) + C_F vec(F^{(l)}) + D_F$$

and

$$[vec(F^{(l)})]^T A_G vec(F^{(l)}) + C_G vec(F^{(l)}) + D_G$$

respectively. Therefore the descent gradient procedure described in the Paragraph 3.3.2 directly applied. We conclude with some remarks on the complexity of quantum collaborative approach.

4.2.5 Complexity of Collaborative Semi non-negative matrix factorization

As in the Section 3.3.3 recall that the computational complexity for the classic Semi-NMF is of order $T\mathcal{O}(K(MN))$ see [DLJ08], T is the maximum number of iterations. By tacking into account L the number of collaborators we get that the computational complexity for the classic colaborative Semi-NMF is of order $LT\mathcal{O}(K(MN))$. With the same considerations the quantum version of semi NMF is $\mathcal{O}(Kpolylog(MN))$. In the case of collaborative approach with L the number of collaborators it follows that:

	SNMF_{coll}	QSNMF_{coll}
Time computational complexity	$L\mathcal{O}(KMN)$	$L\mathcal{O}(Kpolylog(MN))$

TABLE 4.3: QSNMF_{coll} versus SNMF_{coll}

4.3 Conclusion

In this chapter, we proposed a new approach called Quantum Collaborative K-means which is based on combining several clustering models based on quantum K-means. This collaboration consists of exchanging the information of each algorithm locally in order to find a common underlying structure for clustering.

Comparing the classical version collaborative K-means clustering to its quantum version, we notice that we had a good classification, the only difference is its complexity; we have an exponential speed up: while the classical version takes polynomial time, the quantum version only takes logarithmic time, especially in large data sets. Our proposed approach is illustrated on various databases and the experimental results have shown very promising performance.

We also proposed the quantum version of collaborative Semi Non-negative Matrix Factorization which consists on the collaboration of different clustering methods, in order to reach an agreement on the partitioning of a common dataset. This latter

doesn't have only the advantage of speed up but also by collaborating several clustering solutions, each one with its own bias and imperfections, one will get a better overall solution in terms of classification.

Chapter 5

Other research directions: Collaborative learning to improve the non uniqueness of NMF

Contents

5.1 Introduction

Data clustering is an unsupervised task where the goal is to determine a finite set of categories (groups) to describe a data set according to similarities among its objects. Faced to the difficulty of designing a general purpose of clustering algorithm and choosing a satisfactory or even perfect clustering, the solution is to make several clustering algorithms collaborate. The aim of this collaboration is by combining several clustering solutions, each one with its own bias and imperfection, we expect to get a better overall solution.

The fundamental concept of collaboration is that the clustering algorithms operate locally (on individual subsets) and then collaborate by exchanging information about their structures and results in order to improve their models. Collaborative clustering is a promising field with results that offers several solutions for specific issues (e.g. [Ped05], [FGW10], [FWG07]). The objective of collaborative clustering is to make different clustering methods collaborate, so as to reach an agreement on the partitioning of a common dataset.

There are three main types of collaboration: horizontal, vertical and hybrid collaboration. Horizontal collaborative clustering: all datasets describe the same observations, so all the collaborative datasets have the same number of observations but a different number of variables. This approach can be considered as a multi-view clustering where the treatment is done on multi-represented data. Vertical collaborative clustering: all datasets have the same variables but with different objects. Hybrid collaborative clustering: is the combination of both vertical and horizontal collaborative clustering approaches.

NMF has received a significant amount of attention as an important problem in different areas such as language modeling, feature extraction, computer vision, clustering and text mining [Sha+06] [Cic+09]. NMF [PT94b] is a machine learning technique that is used to decompose large data matrices imposing the non-negativity constraints on the factors. This non-negativity makes the resulting matrices easier to interpret. Frequently, the datasets to be analyzed are non-negative, and sometimes they also have a sparse representation. In machine learning, non-negativity is related to probability distributions while sparseness is related to feature selection.

While applying NMF we get generally a factorization of two matrices, we consider one of the matrices as a matrix containing the centroids of a data set and the other one as a data partition matrix.

NMF solution is not unique: a matrix and its inverse can be used to transform the two factorization matrices by : $X = FG^T = FBB^{-1}G^T$. If the two new matrices FB and $B^{-1}G^T$ non-negative, they form another solution for factorization. The non-negativity of FB and $B^{-1}G^T$ is verified if B is a non-negative matrix. More control on the non-uniqueness of FB and $B^{-1}G^T$ can be obtained with parsimony constraints (sparsity).

In fact, NMF is an unstable algorithm; which means that if we run NMF for the same data we get different factorization between each run. Stability has been widely used in different applications as heuristics for adjusting parameters of clustering algorithms, such as the number of clusters, or the stopping criteria, see [DF02] and [BHEG01].

The first article investigating the uniqueness of NMF is Donoho and Stodden [DS03]. They show that the NMF solution is unique in some situations by using convex duality.

An important result for the NMF uniqueness is due to [Plu02] which shows that in most cases the NMF solutions has a special form.

The article [Lau+08] is another relevant result for the NMF uniqueness study, they obtain necessary and sufficient conditions for the uniqueness. The two conditions are the boundary close and the sufficiently spread property of the column space of the data matrix. They also show that NMF is robust to additive noise under the same conditions.

To tackle the problem of non uniqueness of NMF, we propose in this paper a new approach based on collaborating several Non-negative Matrix Factorization (NMF) models then find a consensus of different collaborations by using Karush-Kuhn-Tucker (KKT) conditions in order to have a solution more stable and solve the problem of non uniqueness of NMF.

The rest of this chapter is organized as follows: in section 2, we describe the Non-negative Matrix Factorization principle and its instability. In section 3, we explain the stability of NMF using collaboration where we describe Karush-Kuhn-Tucker (KKT) optimization problem. The experimental results for stability of NMF are presented in section 4. Finally the chapter ends with a conclusion.

5.2 Non uniqueness of NMF

In this section, we shortly describe the classical Non Negative Matrix Factorisation and its instability (non uniqueness of NMF).

Let $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{R}_+^{M\times N}$, be a non-negative data matrix with M rows and N columns, here $\mathbf{x}_n \in \mathbb{R}_+^{M\times 1}$ represents the n^{th} column of X. In what follows $\|\cdot\|$ stays for the Euclidean norm and $\|\cdot\|_F$ for the Frobenius norm. Let K be a fixed input parameter.

5.2.1 Classical and Convex NMF

The NMF gives a low rank approximation of X by the product of two non-negative matrices FG where the factors are $F = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k) \in \mathbb{R}_+^{M\times K}$, $G = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_K)^T \in \mathbb{R}_+^{K\times N}$ and T denotes the transpose operator. The NMF decomposition could be formulated as a constrained optimization problem by minimizing the following error function:

$$(F, G) = \underset{F,G}{\operatorname{argmin}} \|X - FG\|_F^2. \tag{5.1}$$

The non-negativity constraints problem in the matrix form are $F, G \geq 0$.

The NMF decomposition has been studied early by Golub and Paatero [PT94b],[XHP99]. Several families of algorithms are proposed to solve this matrix approximation problem. The approach proposed by Lee and Seung [LS01] is based on a gradient descent strategy which start with a random initialization of F and after usually unfolds in two phases: at first the functional is minimized with respect to G while in the second phase, the functional is minimized with respect to F. In the Lee and Seung's approach, this algorithm could converge to a stationary point which is not necessarily local minima. Another solution is to use *alternate least square* strategy. There is no guarantee that we can exactly recover the original matrix from F and G, so we will approximate it as best as possible in terms of the approximation error measured in Frobenius norm. The work of Lee and Seung [LS01] revealing that NMF has an inherent clustering property, i.e., it automatically clusters the columns of input matrix X since the matrix columns vector factor $(\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k)$ could be considered as cluster centroids while the the matrix rows vector factor $(\mathbf{g}_1, \mathbf{g}_2, \ldots, g_K)^T$ could be considered as cluster indicators. This fact brought much attention to NMF in machine learning and data mining communities.

The aim of clustering is to cluster the columns of X, so as to optimize the difference between X and the clustered matrix revealing significant block structure. To this end the so-called convex NMF was studied (see [DHS05]). In this context we ask that the columns of the cluster centroids matrix F are the convex combination of the rows of the input matrix X. More precisely we want to find W a positive matrix weight such that $F = XW$. With this additional constraint the functional in Equation (5.1) writes:

$$(W, G) = \underset{W,G}{\operatorname{argmin}}\, \mathcal{L}(X, W, G), \text{ where } \mathcal{L}(X, W, G) = \|X - XWG\|_F^2. \tag{5.2}$$

When G is an orthogonal matrix $G^TG = I$, the resulting non-negative matrix factorization (NMF) is equivalent to relaxed K-means clustering (see [DHS05]). The K-means clustering is one of the most widely used clustering method early developed by Lloyd [LLo82]. To summarize, K-means clustering problem can be formulated as a matrix approximation problem [LS01] where the clustering aim is to minimize the approximation error between the original data X and the reconstructed matrix based on the cluster structures.

5.2.2 Proposed strategies to reduce the instability of NMF

In the literature, the term "stability" is used for various meanings, which are not all equivalent. In our case, the stability of a NMF algorithm refers to its ability to regularly give similar factorization solutions for the same input. Let us consider

$$(W^*, G^*) = \underset{W,G}{\operatorname{argmin}}\, \mathcal{L}(X, W, G), \tag{5.3}$$

with $\mathcal{L}$ defined in Equation (5.2), and $F^* = X^*W^*$ and let $B \in \mathbb{R}_+^{K\times K}$ be an inversible matrix then (F^*B^{-1}, BG^*) is still a solution of the problem in Equation (5.3) with exactly the same residual. Therefore we have non-uniqueness of the solution in the non negative matrix factorization optimization problem defined above by construction. When G is an orthogonal matrix $G^TG = I$ we have the uniqueness of the solution (see [DHS05]).

As mentionned before the optimization procedure to solve the problem (5.3) starts with the random initialization of F therefore by running several NMF on the same data or data drawn from the same source repeatedly, we get different results factorization between each execution due to this initialization. The statistic information, as the expectation and the variance, on the solutions obtained for several NMF's live on a bidimensional manifold. Due to the variability of these statistics this manifold is not well concentrated showing the the instability of NMF algorithm.

That's why that in our work we are intending to deal with the problem of instability of NMF. Our purpose is to propose strategies to obtain a more compact manifold corresponding to the solutions statistics of NMF; which means for the same data if we run NMF several times we will get the almost the same factorization. We obtain by reducing the instability a near stable optimal factorization algorithm.

There are many strategies to reduce the instability of unsupervised algorithms. In the last decade two notables approaches was developed: ensemble clustering [SG02] and collaborative clustering [Ped02]. Their idea was based on the hypothesis that combining large and diverse sets of clustering can produce a more stable and precise solution .

The main difference between ensemble clustering and collaborative clustering [VR11] is that clustering ensemble learning methods aim at finding a single consensus partition, while collaborative clustering does not have this final goal. In short, the field of collaborative clustering is concerned with finding algorithms and functions that allow algorithms to share information and to improve their results based on each other similarities, while the field of ensemble learning is more concerned with finding algorithms and methods to merge the solutions or find a consensus between them.

Since both strategies reduce the instability of unsupervised algorithms, we used ensemble/collaborative clustering techniques to deal with the problem of instability of NMF, based on the hypothesis that combining different clustering solutions can produce a more stable and accurate solution.

The idea behind of ensemble/ collaboration clustering is to minimize an adapted functional defined on the similarity between different clustering results of the same database, which allows to exchange clustering solution informations in order to produce improve clustering solutions. Our strategy to reduce the instability follows the steps: first for the same input we run a finite number of NMF instance, secondly, by using an optimization KKT procedure we combine the different solutions into a single consensus solution clustering, thirdly we use a collaborative approach on the obtained solutions in the first step to improve all these solutions and finally by using

the same KKT procedure we define an optimized clustering solution bases on the collaborative solutions obtained previously.

We will show that the both solutions obtained in the second and fourth step will reduce the instability of NMF. We will also compare these two solutions to show the advantage to use collaborative approach to improve stability in clustering.

5.3 Theoretical background for Exchange Information Strategies

This Section describes the details of the strategies to information exchange between different NMF clustering solutions. We also present the KKT optimization procedure used to define the unique optimal clustering solution associated to several existing solutions. We start by presenting the general principle of our approach.

5.3.1 General principle

Let L be a finite integer and $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ the data matrix. For some fixed run $l \in \{1, \ldots, L\}$ of NMF we obtain the solutions :

$$(W^{(l)}, G^{(l)}) = \underset{W^{(l)}, G^{(l)}}{\operatorname{argmin}} \mathcal{L}(X, W^{(l)}, G^{(l)}). \tag{5.4}$$

Here $F^{(l)} = (\mathbf{f}_1^{(l)}, \mathbf{f}_2^{(l)}, \ldots, \mathbf{f}_k^{(l)}) = X^{(l)}W^{(l)} \in \mathbb{R}_+^{M_l \times K}$ represents the matrix of cluster centroids and $G^{(l)} = (\mathbf{g}_1^{(l)}, \mathbf{g}_2^{(l)}, \ldots, \mathbf{g}_K^{(l)}) \in \mathbb{R}_+^{K \times N}$ represents the matrix clusters indicators obtained by the l-th run NMF. The matrix $G^{(l)}$ as partition matrix is a a matrix of binary indicator such that $g_{kn}^{(l)} = 1$ if $\mathbf{x}_n^{(l)} \in C_k^{(l)}$, and $g_{kn} = 0$ otherwise, the $C_k^{(l)}$ being the k-th cluster. Note that, if the clusters $C_1^{(l)}, \ldots, C_K^{(l)}$ have distinct cluster centroids $\mathbf{f}_1^{(l)}, \mathbf{f}_2^{(l)}, \ldots, \mathbf{f}_k^{(l)}$ each of the N columns of $G^{(l)}$ will contain a single 1 and $K-1$ elements that are 0. Accordingly, the columns of $G^{(l)}$ will sum to one and its row sums will indicate the number elements per cluster. Moreover, since $g_{kn}^{(l)} \in \{0, 1\}$ and each column of $G^{(l)}$ only contains a single 1, the rows of $G^{(l)}$ are pairwise perpendicular which is then to say that the matrix $G^{(l)}(G^{(l)})^T$ is a diagonal matrix.

To minimize the functional in Equation (5.4) we unfold recursively the following steps:

$$\begin{cases} W^{(l)} = \operatorname{argmin}_{W^{(l)}} \mathcal{L}(X, W^{(l)}, G^{(l)}) \\ F^{(l)} = \operatorname{argmin}_{F^{(l)}} \mathcal{L}(X, W^{(l)}, G^{(l)}) \end{cases} \tag{5.5}$$

Following the same gradient descent strategy as Lee and Seung for the optimization problems we first compute the gradients:

$$\nabla_{W^{(l)}}(\|X - XW^{(l)}G^{(l)}\|_F^2) = -2X(G^{(l)})^T + 2XW^{(l)}G^{(l)}(G^{(l)})^T, \tag{5.6}$$

and that

$$\nabla_{G^{(l)}}(\|X - XW^{(l)}G^{(l)}\|_F^2) = -2(XW^{(l)})^T X + 2(XW^{(l)})^T XW^{(l)}G^{(l)}. \tag{5.7}$$

By using adaptative descent rates in the gradient descent strategy, we get the matrix update multiplicative formula:

$$\begin{cases} G^{(l)} & = & G^{(l)} \circ \frac{(XW^{(l)})^T X}{(XW^{(l)})^T XW^{(l)} G^{(l)}} \\ W^{(l)} & = & W^{(l)} \circ \frac{X^T X (G^{(l)})^T}{X^T XW^{(l)} G^{(l)} (G^{(l)})^T}. \end{cases} \tag{5.8}$$

The exchange information between different solutions, consensus or collaborative based will be casted as different optimization problems and will be described next.

5.3.2 Consensus based Exchange Information

In context of consensus clustering or aggregation clustering, we are concerned with the situation in which we have a finite number of different clusterings solutions obtained for a particular dataset and we want to find an unique (consensus) clustering which better fit with respect to some well defined criteria than the existing clusterings. Since NMF has random initialization by using the consensus strategy we test the NMF sensitivity to the initial conditions. This strategy across multiple runs of the NMF algorithm, allows to show the stability of the discovered clusters. Note that as optimization problem, consensus clustering is known to be NP-complete, even for small number of solutions clusterings.

Since we want that the consensus solution represents the median solution between all the different solutions $F^{(l)}, G^{(l)}$ we consider a weighted average between the error of the median solution $\|X - XWG\|_F$ and all the errors $\|X - XW^{(l)}G^{(l)}\|_F, l = 1, \ldots, L$ and we define then the functional:

$$\mathcal{E}(W, G) = \sum_{l=1}^{L} \alpha_l \cdot E_l(W, G), \text{ with } \sum_{l=1}^{L} \alpha_l = 1, \tag{5.9}$$

where $E_l(W, G)$ is the delta between the two errors corresponding to that median/consensus solution and the solution obtained by the run l:

$$E_l(W, G) = (\|X - XWG\|_F - \|X - XW^{(l)}G^{(l)}\|_F)^2. \tag{5.10}$$

Before giving the optimization detail we note that the functional is well defined and due to the convexity of the quadratic function, a minimum always exists. The minimization is realized by using the gradient descent procedure. The gradients computation are given in Appendix Section .1.6. In order to respect the non-negativity contraint on the desired solution we define adaptative rates learning in order to eliminate any subtraction from the update rule as follows:

$$\eta_W = \frac{XW}{(\|X - XWG\|_F)\, XWGG^T + \left(\sum_{l=1}^{L} \alpha_l \cdot \|X - XW^{(l)}G^{(l)}\|_F\right) XWG^T}, \tag{5.11}$$

and

$$\eta_G = \frac{G}{(\|X - XWG\|_F)\, (XW)^T XWG + \left(\sum_{l=1}^{L} \alpha_l \cdot \|X - XW^{(l)}G^{(l)}\|_F\right) (XW)^T X}. \tag{5.12}$$

Once the rates learning defined we write the multiplicative rules:

$$W = W \circ \frac{\left(\|X - XWG\|_F\right) X^T XG^T + \left(\sum_{l=1}^{L} \alpha_l \cdot \|X - XW^{(l)}G^{(l)}\|_F\right) X^T XWGG^T}{\left(\|X - XWG\|_F\right) X^T XWGG^T + \left(\sum_{l=1}^{L} \alpha_l \cdot \|X - XW^{(l)}G^{(l)}\|_F\right) X^T XG^T} \tag{5.13}$$

and

$$G = G \circ \frac{\left(\|X - XWG\|_F\right) (XW)^T X + \left(\sum_{l=1}^{L} \alpha_l \cdot \|X - XW^{(l)}G^{(l)}\|_F\right) (XW)^T XWG}{\left(\|X - XWG\|_F\right) (XW)^T XWG + \left(\sum_{l=1}^{L} \alpha_l \cdot \|X - XW^{(l)}G^{(l)}\|_F\right) (XW)^T X}. \tag{5.14}$$

Let us comment these formula in Equation (5.13) and (5.14). Note that the resulting matrix of centroids F is weighted averages of all $F^{(l)}$ by construction. The resulting partition matrix G in the consensus matrix contains a relative average of the number of times of points were clustered together by the different runs of NMF.

The first algorithm $NMF_{consensus}$ is based on defined for L, NMF models then making a consensus between these runs, see Algorithm 8 below. The algorithm has two steps: *Local* step in which we consider the data matrix X and we learn a NMF for a fixed integer K and for a large number of runs L-th. and the second step *Consensus:* in which we compute a consensus solution from the different L NMF models.

Algorithm 8: NMF consensus algorithm

Initialization: Initialize all the centroids sets randomly and P number of realizations.
For all realizations
Local step:
forall *algorithms* $\mathcal{A}^{(l)}$ **do**
  Minimize the objective function of the NMF (5.4).
end
Consensus step:
Compute the consensus weight α **forall** *algorithms* $\mathcal{A}^{(l)}$ **do**
  Update the partitions matrix of all algorithms (5.14).
  Update the centroids matrix of all algorithms (5.13).
end

5.3.3 Collaborative Approach to Exchange Information

In the collaborative NMF context, when a data unit is presented in the l-th NMF, the optimization is done to minimize the distance between that unit and the centroids of each local NMF $l' \neq l$. We would like that after the collaboration: if an observation of the X-th data set is projected onto the k-th centroid of the l-th NMF, then the same observation viewed as an observation in the X-th data set is projected on the same unit k centroid. To this end we introduce a global functional as follows:

$$\mathcal{C}(W, G) = \sum_{l=1}^{L} \left(\alpha_l \mathcal{L}(W^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot C_{l,l'}(W^{(l)}, G^{(l)}) \right) \tag{5.15}$$

We have used the notations $F = (F^{(l)})_l, G = (G^{(l)})_l, l = 1, \ldots, L$. Here $\mathcal{L}$ is the l^{th} local term defined in Equation (5.4).

In order to define the functional in the collaborative part we introduce the matrix $D^{(l)}$ of the euclidean distances of each data in X to the set of centroids $F^{(l)}$ in local instance l-th, more specifically we have $D^{(l)}_{kn} = \|\mathbf{x}^{(l)}_n - \mathbf{f}^{(l)}_k\|$. Therefore we define the pairwise collaborative $C_{l,l'}$ term between the l-th and l'-th NMF's as follows:

$$C_{l,l'}(W^{(l)}, G^{(l)}) \quad = \quad \|(G^{(l)} - G^{(l')}) \circ D^{(l)}\|^2_F. \tag{5.16}$$

In words the $C_{l,l'}$ term is the weighted sum of Euclidean distances between the observed data $x^{(l)}_n$ and all the centroids in $F^{(l)}$, with the weight defined by $G^{(l)} - G^{(l')}$. Note that when $G^{(l)}$ and $G^{(l')}$ agree then the collaborative term is equal to zero and only the local l-th NMF is take into account in the optimization. Note also that by construction $C_{l,l'} \geq 0$ and by definition $D^{(l)} \geq 0$ therefore $C_{l,l'}(W^{(l)}, G^{(l)}) = 0$ if and only if $G^{(l)} - G^{(l')} = 0$ i.e. the data $\mathbf{x}^{(l)}_n$ in the l'-th data set is projected on the same centroid.

Therefore, the set of centroids $F^{(l)}$ is estimated iteratively by minimizing the objective function:

$$\mathcal{C}(l) = \mathcal{L}(W^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot C_{l,l'}(W^{(l)}, G^{(l)}) \tag{5.17}$$

Here $\beta = (\beta_{l,l'})_{l,l'}$ is the weight of the collaboration fixed by the user satisfying $\sum_{l' \neq l} \beta_{l,l'} = 1$. To get the update formula for $F^{(l)}, G^{(l)}$ we use the gradient descent strategy for the functional defined in Equation (5.15). Let us first remark that the cost function defined in Equation (5.15) is continuously first differentiable in all variables. Therefore a minimum always exists and could be found by nonlinear programming. The minimization of the global functional should satisfy the constraints: $(G^{(l)})^T G^{(l)} = I$ for all $l \in \{1, \ldots, L\}$ since $G^{(l)}$ is a partition matrix. We compute the gradient of the functional in Equation (5.17) see Appendix Section .1.7. Having obtained both gradients for the gradient descent algorithm the update formulates for collaborative NMF writes:

$$\begin{cases} G^{(l)} = G^{(l)} - \eta_{G^{(l)}} & \circ \quad [\nabla_{G^{(l)}} \mathcal{C}(l)], \\ W^{(l)} = W^{(l)} - \eta_{W^{(l)}} & \circ \quad [\nabla_{W^{(l)}} \mathcal{C}(l)]. \end{cases} \tag{5.18}$$

Note that the scalar 2 can be ignored, since we can consider that it is absorbed into the learning rates. In conventional gradient descent, all the elements of the learning rates $\eta_{G^{(l)}}$ and $\eta_{W^{(l)}}$ are positive. Due to the subtraction present in both update rules, this can produce negative elements, violating the non-negativity constraint. To avoid this, Lee and Seung proposed in 2001 [LS01] to use data-adaptive learning rates that would make it impossible to obtain negative elements. The trick is to define the learning rates so that any subtraction disappears from the update rule as follows:

$$\eta_{W^{(l)}} = \frac{XW^{(l)}}{XW^{(l)}G^{(l)}\left((G^{(l)})^T + \sum_{l' \neq l} \beta_{l,l'} \mathbf{1}_{M \times N}\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \circ XW^{(l)}}, \tag{5.19}$$

and

$$\eta_{G^{(l)}} = \frac{G^{(l)}}{\left(XW^{(l)}\right)^T XW^{(l)}G^{(l)} + \sum_{l' \neq l} \beta_{l,l'} G^{(l')} \circ \left(D^{(l)}\right)^{\circ 2}}. \tag{5.20}$$

We then obtain the multiplicative rules:

$$W^{(l)} = W^{(l)} \circ \frac{X^T X \left(G^{(l)}\right)^T + \sum_{l' \neq l} \beta_{l,l'} X^T X \left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T}{X^T X W^{(l)} G^{(l)} \left(G^{(l)}\right)^T + \sum_{l' \neq l} \beta_{l,l'} \mathbf{1}_{M \times N} \left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \circ X^T X W^{(l)}}, \tag{5.21}$$

and

$$G^{(l)} = G^{(l)} \circ \frac{\left(XW^{(l)}\right)^T X + \sum_{l' \neq l} \beta_{l,l'} \left[G^{(l')} \circ (D^{(l)})^{\circ 2}\right]}{\left(XW^{(l)}\right)^T XW^{(l)} G^{(l)} + \sum_{l' \neq l} \beta_{l,l'} G^{(l)} \circ \left(D^{(l)}\right)^{\circ 2}}. \tag{5.22}$$

The division here is component wise matrix division and where $^{\circ 2}$ means element-wise power. Here all the database share the same units but described by different attributes. In this case, the number and the size of centroid vectors for all the NMF factorizations will be the same. That's why the objective function of the classical NMF has been modified in order to introduce this constraint on the different factorizations during the collaboration step. The collaborative term is expressed by $C_{l,l'}$ which guaranty that the structure of other databases matches with the current database.

5.3.4 Optimized weights for the collaborative NMF to improve stability

In this section, we showed that collaborating several NMF improve its stability. In other words, by collaborating several NMF then making a consensus of these collaboration we get a stable NMF. We propose a different collaborative approach in which we modify the objective function using a weighting strategy to reduce the risk of negative collaboration: we study how to optimize the weights of the combination of functions of the type (5.15) can lead to an optimal value of the global function and reduce the risk of negative collaboration by further optimizing weight factors between the algorithms.

Since n $\beta_{l,l'} \geq 0$, let us consider the weight of the collaboration $\beta_{l,l'} = \tau^2_{l,l'}$ in what follows. We are mainly interested in finding the positive weights $\tau^2_{l,l'}$ that will determine the strength of the collaborative term. The minimization of the collaborative functional is based on the dual form of the problem. We do so under the Karush-Kuhn-Tucker conditions (KKT) [KT51] assuming that the weights $\tau_{l,l'}$ respect the condition $\forall l \quad \sum_{l' \neq l}^{L} \tau^2_{l,l'} = 1$. The results of the optimization under the Karush-Kuhn-Tucker conditions are shown below in Equations (5.23). For all $l' \neq l$ we have:

$$\beta_{l,l'} = \frac{|C_{l,l'}|}{\left(\sum_{l''=1}^{L} |C_{l'',l'}|\right)} \tag{5.23}$$

The interpretation of these results is the following: in the context of horizontal collaborative clustering, the global results should be better if each individual algorithm gives higher weights to algorithms that have the most similar solutions compared with the local one (high $\tau^2_{l,l'}$ value for a given NMF run l). Going deeper, we see that the degree to which one algorithm should collaborate with other collaborators that have dissimilar solutions depends on the degree of normalization. From Equation (5.23), each algorithm would mostly collaborate with the algorithms that have the most similar solutions. If several algorithms have the same most similar solution,

they would be given the same weight. The algorithms with the most similar solutions would still be favored to optimize the cost function of the global collaborative framework. But the solutions of algorithms which have a lesser degree of similarity would still be taken into consideration locally. Let us detail the calculus for the Karush-Kuhn-Tucker optimization problem. Given the $C_{l,l'} \geq 0$, we are trying to optimize the matrix $\{\tau_{l,l'}\}_{L\times L}$ as shown in the system below:

$$\begin{cases} \text{argmin}_{\tau} \sum_{l=1}^{L} \left(\mathcal{L}(W^{(l)}, G^{(l)}) + \sum_{l' \neq l} \tau_{l,l'}^2 \cdot C_{l,l'}(W^{(l)}, G^{(l)}) \right) \\ \forall l \quad \sum_{l' \neq l}^{L} (\tau_{l,l'})^2 = 1, \\ \forall (l, l') \quad \tau_{l,l'} \geq 0 \end{cases} \tag{5.24}$$

From the previous system, by using Lagrange multipliers, we get the following KKT conditions:

$$\forall (l, l'), l \neq l' \begin{cases} (1) & \tau_{l,l'} \geq 0 \\ (2) & \sum_{l' \neq l}^{L} (\tau_{l,l'})^2 = 1 \\ (3) & \lambda_{l,l'} \geq 0 \\ (4) & \tau_{l,l'} \cdot \lambda_{l,l'} = 0 \\ (5) & \nu_{l'} \cdot (2 \cdot (\tau_{l,l'})) - C_{l,l'} - \lambda_{l,l'} = 0 \end{cases} \tag{5.25}$$

Let's begin by considering the case where $\lambda_{l,l'} \neq 0$ in (4). Then, we would have $\tau_{l,l'} = 0$ and with (5): $C_{l,l'} = -\lambda_{l,l'} \leq 0$. Since the $C_{l,l'}$ have been defined as non-negative, this case is not possible, therefore we will only consider the case $\tau_{l,l'} \neq 0$ and $\lambda_{l,l'} = 0$. Then, with (5), we have:

$$C_{l,l'} = 2 \cdot \tau_{l,l'}^2 \cdot \nu_{l'} \tag{5.26}$$

From Equation (5.26) and (2) from (5.25), we have:

$$1 = (2 \cdot \nu_{l'})^{-1} \sum_{l'' \neq l'} (C_{l'',l'}) \tag{5.27}$$

By using $\nu_{l'}$ into Equation (5.26) we get:

$$\tau_{l,l'}^2 = \frac{|C_{l,l'}|}{\left(\sum_{l''=1}^{L} |C_{l'',l'}| \right)} \tag{5.28}$$

And thus we have proved the results of Equation (5.23).

The second algorithm $NMF_{coll+consensus}$ (see algorithm 9 below) is based on running L-th collaborative NMF then making collaboration between these L models followed by a consensus between these L collaborative solutions. The second algorithm has three steps the *Local* step the *Collaborative* step in which we consider the L-th models with our proposed beta-optimization by using Eq. 5.15 to get L-th collaborative solutions. and the *Consensus* step in which we compute a collaborative

consensus solution from the l-th collaborative solutions.

Algorithm 9: NMF horizontal collaboration + consensus algorithm

Initialization: Initialize all the centroids sets randomly and P number of realizations.
For all realizations
Local step:
forall *algorithms* $\mathcal{A}^{(l)}$ **do**
  Minimize the objective function of the NMF (5.4).
end
Collaborative step:
Compute the optimized β **with Equation** (5.23)
forall *algorithms* $\mathcal{A}^{(l)}$ **do**
  Update the partitions matrix of all algorithms (5.22).
  Update the centroids matrix of all algorithms (5.21).
end
Consensus step:
Compute the consensus weight α **forall** *algorithms* $\mathcal{A}^{(l)}$ **do**
  Update the partitions matrix of all algorithms (5.14).
  Update the centroids matrix of all algorithms (5.13).
end

5.4 Experimental results

In the previous section, we explained the different steps of consensus and collaboration. The consensus and/or collaboration will be used in order to stabilize NMF. Our purpose in this work is to have a solution more stable which means for the same data matrix X we will get usually the factors F and G having a reduced standard error and stable mean over large number of NMF solution.

To this end, we run NMF several times and we get different results of decomposition (because of its instability), then, we collaborate the outputs of the multiple models of NMF of the same dataset. Afterwards, we make a consensus between these different collaboration in order to address the problem of instability. The experimental protocols are based on the P runs of the two Algorithms (8) and (9).

5.4.1 Validation

Our goal is to show that consensus an/or collaboration allow to reduce the mean, standard error of the reconstruction error solution obtained for P NMF solutions NMF. Note that in the case of consensus solution for L computed NMF the mean and the standard error of the reconstruction error are estimated as simply the average over the reconstruction error vector of length L. Over P models of our algorithms the mean and the standard error of the reconstruction error are estimated as simply the average over the reconstruction error vector of length P.

We then compare the reconstruction error between the consensus solution and the collaborative consensus solution and we show the behaviour of the mean and the standard error of the reconstruction error.

We aim to prove the following proposition that show the behaviour of the standard error for increasing L.

Proposition 1 *Let us assume the errors are bounded and normal distributed over the L models. It follows that the mean standard error over L goes to* 0 *when L go to infinity.*

Proof:
This proposition can be proved in several ways. On the one hand, it can be proved by logical reasoning, and on the other hand by experimentation. We will proceed to its demonstration by both ways.

The process of collaboration between the different NMF models is by design an approach that avoids negative collaboration, see Equations (5.15) and (5.23). That is to say, the proposed collaborative approach can only improve the quality of reconstruction for each model after collaboration with the others. In case where collaboration degrades the performance of a model, it is rejected and the model keeps its initial parameters. Indeed the optimization process in Equation (5.15) has a minimal solution that minimizes the standard error for each model. From the optimization problem in Equation (5.23) we deduce that when the pairwise collaborative term $C_{l,l'}$ is bigger than other collaborative terms then the weight $\beta_{l,l'}$ is bigger than the other different weights. Moreover, with Equation (5.23), for collaborative terms with smaller $\beta_{l,l'}$ -will give a bigger weight to their local term and therefore these models will kept their local parameters.

Therefore, the collaboration process can only improve the performance of each model. Moreover, the greater the number of collaborators, the greater the amount of disponible knowledges. And the more knowledges available, the better the performance of the models after collaboration. This improved performance is reflected in a decrease of the reconstruction error of each model. We can therefore deduce that the more the number of relevant collaborators increases, the more the reconstruction error of each collaborator decreases after collaboration. In other words, when the number of relevant collaborators tends towards infinity, the reconstruction error of the different models tends towards 0. We verify this proposition experimentally afterwards. □

5.4.2 Data sets

To evaluate the interest of the proposed collaborative/consensus approach we used several datasets of different size and complexity: Iris, Wine, Wisconsin Diagnostic Breast Cancer (WDBC) and Waveform. In particular, we will give a more detailed analysis on the WDBC dataset results so that we can better demonstrate the efficiency of our proposed method. The datasets used in our experiments are from the UCI website: *Iris*, *Wine*, *Wisconsin Diagnostic Breast Cancer (WDBC)* and *Waveform data set* .

- *Iris* - Iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.
- *Wine* - Wine is a dataset that is related to a chemical analysis of wines grown in the same region in Italy but derived from different cultivars.
- *Wisconsin Diagnostic Breast Cancer (WDBC)* - This data has 569 instances with 32 variables (ID, diagnosis, 30 real-valued input variables). Each data observation is labeled as benign (357) or malignant (212).
- *Waveform* - Waveform is an artificial dataset containing 5000 samples separated in 3 classes of waves. Each wave is described by 40 features among which 19 are all noise with mean 0 and standard error 1. Each class is generated from a combination of 2 of 3 "base" waves.

As criteria to validate our approach, we test for $P = 10^6$ the mean reconstruction and we also compute the of the reconstruction error over the P sample of reconstruction error. In all the experiments the number of NMF iterations is restricted to 300,

also the number of iterations is fixed to 300. The number of NMF models involved in consensus or collaboration is fixed to $L = 10^2$. In the Figures below we visualize

Dataset	NMF	Mean Rec. Error	Mean Std. Error
Iris	*intial*	3.621	0.513
	consensus	2.192	0.437
	collab + consensus	2.201	0.352
Wine	*intial*	4.766	1.451
	consensus	3.534	1.769
	collab + consensus	3.632	1.405
WDBC	*intial*	9.756	2.382
	consensus	6.963	1.819
	collab + consensus	6.812	1.417
Waveform	*intial*	3.216	1.311
	consensus	2.125	1.118
	collab + consensus	2.024	1.028

TABLE 5.1: Experimental results on different datasets with $P = 10^6$ and $L = 10^2$ NMF models

these results for the considered data bases.

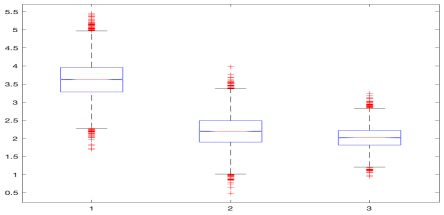


FIGURE 5.1: Iris (1) Initial (2) Consensus and (3) Collaboration+Consensus

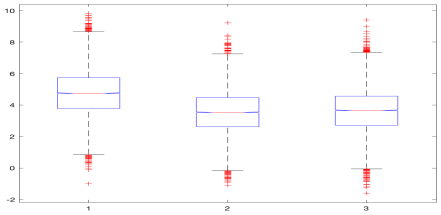


FIGURE 5.2: Wine (1) Initial (2) Consensus and (3) Collaboration+Consensus

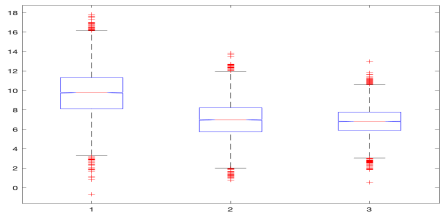


FIGURE 5.3: WDBC (1) Initial (2) Consensus and (3) Collaboration+Consensus

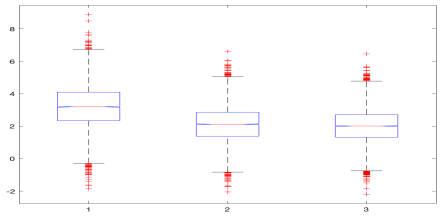


FIGURE 5.4: Wave (1) Initial (2) Consensus and (3) Collaboration+Consensus

In Figures 5.1,5.2,5.3 and 5.4 we show boxplots with $P = 10^6$ and $L = 10^2$ NMF models and different data bases Iris, Wine, WDNC and Wave form respectively. In order to study the dependency on L we do the same experimental protocols for $P = 10^6$ and for different $L = 2 \cdot 10^2$, $L = 5 \cdot 10^2$ and $L = 10 \cdot 10^2$.

Dataset	NMF	Mean Rec. Error	Mean Std. Error
Iris	*intial*	3.421	0.343
	consensus	2.089	0.324
	$collab + consensus$	2.091	0.212
Wine	*intial*	3.676	1.411
	consensus	3.484	1.269
	$collab + consensus$	3.312	1.205
WDBC	*intial*	7.856	2.124
	consensus	5.863	1.219
	$collab + consensus$	4.922	1.117
Waveform	*intial*	3.014	1.023
	consensus	2.025	1.088
	$collab + consensus$	1.924	1.011

TABLE 5.2: Experimental results on different datasets with $P = 10^6$ and $L = 2 \cdot 10^2$ NMF models

Dataset	NMF	Mean Rec. Error	Mean Std. Error
Iris	*intial*	3.152	0.234
	consensus	2.019	0.224
	$collab + consensus$	2.012	0.192
Wine	*intial*	3.476	1.321
	consensus	3.174	1.165
	$collab + consensus$	3.104	1.091
WDBC	*intial*	6.163	2.411
	consensus	4.964	1.139
	$collab + consensus$	4.895	1.080
Waveform	*intial*	2.141	1.013
	consensus	1.995	1.043
	$collab + consensus$	1.897	1.009

TABLE 5.3: Experimental results on different datasets with $P = 10^6$ and $L = 5 \cdot 10^2$ NMF models

Dataset	NMF	Mean Rec. Error	Mean Std. Error
Iris	*intial*	3.054	0.143
	consensus	2.011	0.122
	$collab + consensus$	2.010	0.072
Wine	*intial*	3.268	1.221
	consensus	3.165	1.071
	$collab + consensus$	3.097	1.001
WDBC	*intial*	5.978	2.119
	consensus	4.663	1.091
	$collab + consensus$	4.597	1.010
Waveform	*intial*	2.091	0.093
	consensus	1.897	0.056
	$collab + consensus$	1.821	0.005

TABLE 5.4: Experimental results on different datasets with $P = 10^6$ and $L = 10 \cdot 10^2$ NMF models

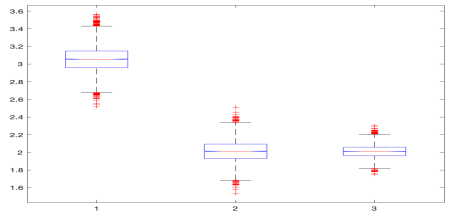


FIGURE 5.5: Iris (1) Initial (2) Consensus and (3) Collaboration+Consensus

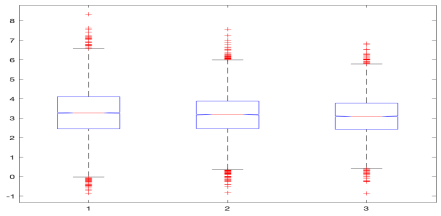


FIGURE 5.6: Wine (1) Initial (2) Consensus and (3) Collaboration+Consensus

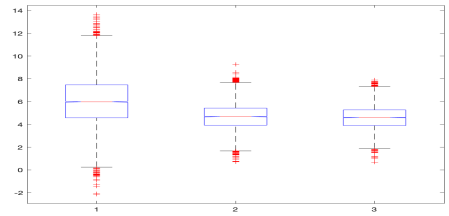


FIGURE 5.7: WDBC (1) Initial (2) Consensus and (3) Collaboration+Consensus

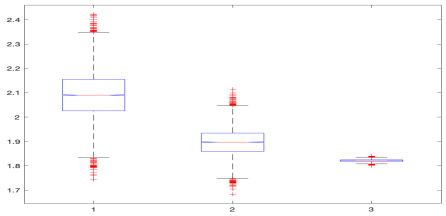


FIGURE 5.8: Wave (1) Initial (2) Consensus and (3) Collaboration+Consensus

In Figures 5.5,5.6,5.7 and 5.8 we show boxplots with $P = 10^6$ and $L = 10 \cdot 10^2$ NMF models and different data bases Iris, Wine, WDNC and Wave form respectively.

Interpretation

From the numerical results we notice that both algorithms allows to mainly reduce the mean average of the reconstruction error compared with the initial mean average of L NMF independent models. We also notice that the standard error of the reconstruction error is reduced by consensus. We also notice that the strategy based on the collaboration followed by a consensus reduce in almost cases the standard error of the final result.

Why these improvements ? The mean average is reduced by the consensus procedure since the consensus procedure is based on a convex combination of the different results. We also notice that mean reconstruction error are very siimilar for the two procedures. Indeed this is due to the fact that the partition matrix in the consensus procedure is a weighted average over the different NMF partition matrix. We notice an improvement the the standard error is reduced. The interest of the collaborative strategy is the reduced standard error in the results. To explain the reduced standard error in the collaborative approach we refer to Equation (5.23).

As explained in Proposition 1 from Equation (5.23), we can get the following: For NMF model l and any other NMF model l', when the pairwise collaborative term $C_{l,l'}$ is big comparatively with the other collaborative terms then the associated collaborative weight $\beta_{l,l'}$ is big comparatively with the other different collaborative weights. This means that any NMF model l should give a stronger collaborative weight to other NMF algorithms having similar results. Moreover, with Equation (5.23), we can interpret in more details the fact that algorithms with relatively weak

collaboration links $\beta_{l,l'}$ -which are not close to any other results- should give a stronger weight to their local term.

If we think about the goal of collaboration, these results make sense: the goal of collaboration is to have the NMF algorithms helping each other. Therefore, when several algorithms find solutions that are similar, it is most likely that they have actually found a structure in the data. As such, collaborating with algorithms that have solutions similar to the local partitioning is a convenient way to avoid the risk of negative collaboration. These results can also be linked to recent works on clustering stability [BDLP06; Lux10]. A clustering is said to be stable if the partitioning remains similar when the data set or the clustering process are perturbed. With this proposed weighting method, the algorithms that will have the strongest collaboration links will be these with the most similar solutions. It matches with the definition of stability: the solutions that feature structures and clusters found through several feature spaces by the different NMF are the most stable.

5.5 Conclusion

In this Chapter we study the non-uniqueness of the NMF models. We propose a novel strategy to control the variability results in the NMF models. This strategy is based on the collaborative learning between several NMF models followed by a consensus between the different s obtained solutions. The weights of the collaboration procedure of several NMF is defined by using Karush-Kuhn-Tucker (KKT), in this way we reduce the standard reconstruction error and also we avoid the negative collaboration between the models. Afterwards, we described in detail our proposed approach which is adapted for collaboration between several NMF and finding a consensus between these different collaborations. Finally, the experimental results showed the effectiveness of our proposed approach which is the reducing of the standard reconstruction error in NMF model. The obtained results show that the standard reconstruction error tends to zero when the number of involved NMF models tends to infinity.

Conclusion

The main purpose of that thesis is to understand why we need quantum computers in the future and how we could transform a classical algorithm to a quantum one, in particular unsupervised algorithms of clustering.

The first chapter presented an overview of quantum computation where we present the principal notions of quantum computation as qubit, superposition, entanglement, and so on. We also described some well known quantum algorithms as Shor's algorithm and Grover's algorithm which are known by their spectacular results in quantum computing as they are faster than any known classical algorithm: the first one by factoring integers in an exponential time and the second one by searching over an unordered data in a quadratic time.

The second chapter is devoted to the state of the art of quantum machine learning algorithms where we described the classical and the quantum version of the most used algorithms in machine learning, namely: K-means, K-medians, principal component analysis, support vector machine and artificial neural networks. Indeed, the steps that follow these algorithms stay the same whether for the classical or the quantum version. The main difference resides in complexity; the quantum version gives an exponential speed-up compared to its classical version.

The third chapter concerns the quantum one-model clustering where we give the different steps that follow a quantum algorithm and we give an analysis of three different methods that estimate the distance for quantum prototypes based clustering algorithms. This analysis is so crucial as it can solve any prototype based clustering algorithm. Through this analysis, we notice that the notion of distance in quantum computing is different than the classical one. Because what counts in the quantum computation is the correlation not the real values of the distance. We also proposed a quantum index to evaluate the quality of clustering in quantum algorithms. We called this index quantum Davies Bouldin as it's the adaptation of the classical Davies Bouldin in the quantum version. This adaptation was so crucial to give a good evaluation of clustering as we can not evaluate a quantum algorithm with classical indexes. We also present the quantum version of K-means and Semi Non-negative Matrix Factorization using quantum gradient descent. We used alternate optimization method for quantum gradient descent for both homogeneous and inhomogeneous part of the objective function. The Quantum Semi Non-negative Matrix Factorization (QSNMF) algorithm leads to an exponential speed-up compared to its classical version: the complexity of the problem becomes $\mathcal{O}(Kpolylog(MN))$ instead of $\mathcal{O}(KMN)$ where M and N are the dimension of the matrix and K is the number of clusters.

The fourth chapter is about the quantum collaborative approach; we implement the quantum version of collaborative K-means algorithm using quantum operations,where we show the different steps of implementing the quantum version of collaborative K-means algorithms; from the preparation of the datasets by transforming it to quantum states and the construction of these states, to the clustering in the quantum version, and also measuring the quality of clustering by adapting a classical criterion to the quantum case. Finally, some experimental results have been done to

compare the quantum algorithms with their classical version. We also present the complexity of each algorithm and we show that quantum algorithms give results in a time much more faster than the classical version.

We also give the quantum collaborative version of semi non-negative matrix factorization where we describe the classical collaborative approach and then we give the quantum version and finally we give the complexity of the quantum collaborative algorithm of semi non-negative matrix factorization.

Other research directions were presented in the fifth chapter where we tried to use the collaborative learning approach to improve the non uniqueness of NMF which means to solve the problem of instability of NMF. The purpose was while running NMF for the same data set, we need to get the same decomposition (factorization). For that purpose, we used a collaboration between several NMF then we made a consensus between these collaborations. The experimental results showed the effectiveness of our approach by giving a reconstruction error near to zero for the same data set. We are intending for future work to test this stability in the quantum version.

To sum up, quantum algorithms give a good classification just like their classical version, the only difference is their complexity: while the classic version takes polynomial time, the quantum version only takes logarithmic time, especially in large data sets. Our focus was on unsupervised methods for pattern clustering tasks, we showed how an improvement in terms of complexity can be gained through quantum computation.

Future work

This thesis offers an opening for future lines of research. Two principal perspectives can be applied; the first one concerns the quantum algorithms namely unsupervised algorithms and neural networks and the second one concerns the collaborative approach in the quantum context.

The quantum version of neural networks need to be explored by making a study between the classical neural networks and their quantum version. In this context, we will make as an application text mining as we have already done this work in classical computers [Ben+18].

We are intending also to explore the instability of NMF in the quantum version that's why we did the sixth chapter which is based on improving the non-uniqueness of NMF. The study of the stability of NMF will be based on the collaborative approach which consists on the collaboration of different clustering methods, in order to reach an agreement on the partitioning of a common dataset [Ben+19a]. This latter doesn't have only the advantage of speed up but also by collaborating several clustering solutions, each one with its own bias and imperfections, one will get a better overall solution in terms of classification and also stability.

Other perspectives are also considered:

- Studying the relationship between quantum K-means and quantum NMF.

  As we have already explained, there is an equivalence between K-means hard clustering and orthogonal NMF [DHS05]. We have already defined the algorithm of both NMF and K-means in the quantum context. As future work, we are intending to explore their equivalence in the quantum context.

- Combine horizontal and vertical approaches to get a new hybrid collaboration approach in the quantum context.

  The hybrid approach is when both collaborative clustering approaches, to know: vertical and horizontal are used at the same time. This could be interesting as we can find a situation where patterns from various sources give rise to common subsets of data as well as being positioned in the same feature space.

- Quantum neural networks

  It is found that none of the proposals for a potential quantum neural network [Tót+96] [AA02] [PM11] model fully exploits both the advantages of quantum physics and computing in neural networks. Schuld et al.[SSP14] provides a critical overview of the different strategies employed to construct a quantum artificial neural network and highlights how most approaches do not meet the requirements of what can be reasonably defined as a quantum artificial neural network. That's why, we think that this field should be more explored by trying to construct a full artificial quantum neural network. The reason for this is that the nonlinear dissipative dynamics of neural computation [Rab+06] are fundamentally different to the linear, unitary dynamics of quantum computing [NC01], which means that the greatest challenge to a quantum artificial neural network is that the quantum mechanics is linear but the classical neural networks require the non linearity. Indeed, two requirements should be taken into consideration to construct a fully quantum neural network, which are:

  1. The bases of neural network theory.
  2. The use and the consistency with quantum theory.

- Parameter optimisation of quantum learning algorithms

  The optimisation of parameters in quantum learning algorithms have not yet been explored and could be seen as a perspective of our future work. Different approaches to quantum computing can be investigated for this purpose. In quantum computing based on unitary quantum gates, the challenge would be to parameterize and gradually adapt the unitary transformations that define the algorithm. Several ideas in that direction have been already investigated [Bis+10] [GM09] [GM13].

  Adiabatic quantum computing could be a solution for the optimisation problem [PL13], or measurement-based quantum computing [Bri+09] could also provide an interesting framework for quantum learning.

To summarize, although there is still a lot of work to be done, quantum machine learning remains a very promising emerging area of research with many potential applications and great theoretical variety.

Appendix

.1 Supplementary Material

In this Appendix we analyze the construction of the collaborative NMF objective function and show the relation with the collaborative $K-$ means objective function introduced by Pedrycz [Ped05].

.1.1 Hard K means and NMF

The K-means clustering is one of the most used clustering methods early developed by Lloyd [LLo82]. The K-means clustering allows to divide the set of columns $\{1,\ldots,N\}$ into K clusters $C=\{C_1,\ldots,C_K\}$ by minimizing the objective function defined by the following sum of squared errors:

$$\mathcal{J}_K=\sum_{k=1}^{K}\sum_{n\in C_k}\|\mathbf{x}_n-\mathbf{f}_k\|^2 \tag{29}$$

Let $G\in\mathbb{R}_+^{K\times N}$ be the binary classification matrix defined by $g_{kn}=1$, if the column $\mathbf{x}_n\in C_k$, and 0 otherwise. Therefore the functional in Equation (29) writes:

$$\mathcal{J}_K=\sum_{k=1}^{K}\sum_{n=1}^{N}g_{kn}\|\mathbf{x}_n-\mathbf{f}_k\|^2 \tag{30}$$

For a fixed partition $C=\{C_1,\ldots,C_K\}$ we consider $n_k=|C_k|$ the number of elements within the cluster k. Then the cluster centroids $F=(\mathbf{f}_1,\mathbf{f}_2,\ldots,\mathbf{f}_k)$ are:

$$\mathbf{f}_k=\frac{1}{n_k}X\cdot\mathbf{g}_k,\quad k=1,\ldots,K \tag{31}$$

and the indicator clustering matrix $(\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_K)^T\in\mathbb{R}_+^{K\times N}$ are given by:

$$\mathbf{g}_k=\frac{1}{\sqrt{n_k}}(0,\ldots,0,\underbrace{1,\ldots,1}_{n_k},0,\ldots,0)^T\in\mathbb{R}_+^{1\times N}$$

Ding et al. [DHS05] showed that the K-means clustering problem can be formulated as a matrix approximation problem where the clustering aim is to minimize the approximation error between the original data X and the reconstructed matrix based on the cluster structures.

Over the past years, several authors have pointed out that k-means clustering can be understood as a constrained matrix factorization problem. However, it appears as if most authors consider this fact self explanatory and hardly discuss it in detail. Since this may confuse our goal here is to rigorously establish the following equalities

for the objective function of hard K-means clustering:

$$\sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\,\|\mathbf{x}_n - \mathbf{f}_k\|^2 = \|X - FG\|_F^2 = \left\|X - XG^T(GG^T)^{-1}G\right\|_F^2, \tag{32}$$

where

$$X \in \mathbb{R}^{M\times N} \text{ is a matrix of vectors } \mathbf{x}_n \in \mathbb{R}^{M\times 1} \tag{33}$$

$$F \in \mathbb{R}^{M\times K} \text{ is a matrix of cluster centroids } \mathbf{f}_k \in \mathbb{R}^{M\times 1} \tag{34}$$

$$G \in \mathbb{R}^{K\times N} \text{is a matrix of binary indicator variables such that :} \tag{35}$$

$$g_{kn} = \begin{cases} 1 & , \quad \text{if } \mathbf{x}_n \in C_k \\ 0 & , \quad \text{otherwise} \end{cases} \tag{36}$$

Throughout, we write $\mathbf{x}_n$ to denote n-th column vector of a matrix X. To refer to the (m,n) element of a matrix X, we either write x_{mn} or $(X)_{|mn}$. The Euclidean norm of a vector will be written as $\|\mathbf{x}\|$ and the Frobenius norm of a matrix as $\|X\|_F$. Regarding the squared Frobenius norm of a matrix, we recall the following properties:

$$\|X\|_F^2 = \sum_{m=1}^{M}\sum_{n=1}^{N} x_{mn}^2 = \sum_{n=1}^{N}\|\mathbf{x}_n\|^2 = \sum_{n=1}^{N}\mathbf{x}_n^T\mathbf{x}_n = \sum_{n=1}^{N}(X^TX)_{nn} = tr(X^TX) \tag{37}$$

Finally, subscripts or summation indices k will be understood to range from 1 to K (the number of clusters), subscripts or summation indices n will range from 1 up to N (the number of data vectors), subscripts or summation indices m will range from 1 up to M (the dimension of data vectors) and subscripts or summation primes indices will be used to expand inner products between vectors or rows and columns of matrices.

To substantiate the claim in Equation (32), we first point out several peculiar properties of the binary indicator matrix G. If the clusters $C_1,\ldots,C_K$ have distinct cluster centroids $\mathbf{f}_1,\ldots,\mathbf{f}_K$, each of the N columns of G will contain a single 1 and $K-1$ elements that are 0. Accordingly, the columns of G will sum to one

$$\sum_{k=1}^{K} g_{kn} = 1, \text{ for all column } n. \tag{38}$$

and its row sums will indicate the number elements per cluster

$$\sum_{n=1}^{N} g_{kn} = n_k, \text{ for all row } k. \tag{39}$$

Moreover, since $g_{kn} \in \{0,1\}$ and each column of G only contains a single 1, the rows of G are pairwise perpendicular because $g_{kn}g_{k'n} = \delta_{kk'}$, δ being the Kronecker symbol, which is then to say that the matrix GG^T is a diagonal matrix where

$$(GG^T)_{kk'} = \sum_{n=1}^{N}(G)_{kn}(G^T)_{nk'} \sum_{n=1}^{N} g_{kn}(g)_{k'n} = n_k\delta_{kk'}.$$

Having familiarized ourselves with these properties of the indicator matrix, we are now positioned to establish the equalities in Equation (32) which we will do in a step by step manner.

We begin our derivation by expanding the conventional K-means objective functional on the left of Equation (32). For this expression, we have

$$\sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\,\|\mathbf{x}_n - \mathbf{f}_k\|^2 = \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}(\mathbf{x}_n^T\mathbf{x}_n - 2\mathbf{x}_n^T\mathbf{f}_k + \mathbf{f}_k^T\mathbf{f}_k) \tag{40a}$$

$$= \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\mathbf{x}_n^T\mathbf{x}_n \tag{40b}$$

$$- 2\sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\mathbf{x}_n^T\mathbf{f}_k \tag{40c}$$

$$+ \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\mathbf{f}_k^T\mathbf{f}_k \tag{40d}$$

First of all, for the term (40b) we find:

$$\sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\mathbf{x}_n^T\mathbf{x}_n = \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\|\mathbf{x}_n\|^2 = \sum_{k=1}^{K}\sum_{n=1}^{N} \|\mathbf{x}_n\|^2 = tr(X^TX),$$

where we have used (38). Second of all, for the term (40c) we observe :

$$\begin{aligned}
\sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\mathbf{x}_n^T\mathbf{f}_k &= \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\sum_{m=1}^{M} x_{mn}f_{mk} \\
&= \sum_{m=1}^{M}\sum_{n=1}^{N} x_{mn}\sum_{k=1}^{K} f_{mk}g_{kn} \\
&= \sum_{n=1}^{N}\sum_{m=1}^{M} (X^T)_{nm}(FG)_{mn} \\
&= \sum_{n=1}^{N} (X^TFG)_{nn} \\
&= tr(X^TFG).
\end{aligned}$$

Third of all, for the term (40d) we note that:

$$T_3 = \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\mathbf{f}_k^T\mathbf{f}_k = \sum_{k=1}^{K}\sum_{n=1}^{N} g_{kn}\|\mathbf{f}_k\| = \sum_{k=1}^{K} \|\mathbf{f}_k\| n_k,$$

where we have used (39).

Next, we look at the second expression in Equation (32). As a squared Frobenius norm of a matrix difference, it can be written as:

$$\|X - FG\|_F^2 = tr[(X - FG)^T(X - FG)] \tag{41a}$$

$$= tr[X^TX] \tag{41b}$$

$$- 2tr[X^TFG] \tag{41c}$$

$$+ tr[(FG)^T(FG)] \tag{41d}$$

Given our earlier results, we immediately recognize that (40b)=(41b) and (40c)=(41c). Regarding the term (41d), we note that, because of the cyclic permutation invariance

of the trace operator, we have

$$\begin{aligned} tr[(FG)^T(FG)] &= tr[G^T F^T FG] = tr[F^T FGG^T] \\ &= \sum_{k=1}^{K} (F^T FGG^T)_{kk} \\ &= \sum_{k=1}^{K} \sum_{n=1}^{N} (F^T F)_{kn} (GG^T)_{nk} \\ &= \sum_{k=1}^{K} (F^T F)_{kk} (GG^T)_{kk} \\ &= \sum_{k=1}^{K} \|\mathbf{f}_k\| n_k, \end{aligned} \tag{42}$$

where we used the fact that GG^T is diagonal. This result, however, shows that (40d)=(41d) and consequently, that (40) and (41) really are equivalent.

Finally, to establish the equality on the right of Equation (32) we ask for the matrix F that, for a given G, would minimize

$$\frac{\partial}{\partial F} \|X - FG\|_F^2 = \frac{\partial}{\partial F} \left(tr[X^T X] - 2tr[X^T FG] + tr[(FG)^T (FG)] \right). \tag{43}$$

Which, upon equation to 0, leads to:

$$F = XG^T (GG^T)^{-1}, \tag{44}$$

which beautifully reflects the fact that each of the $K-$means cluster centroids $\|\mathbf{f}_k\|$ coincides with the mean of the corresponding cluster C_k, namely:

$$\|\mathbf{f}_k\| = \frac{\sum_{n=1}^{N} g_{kn} \mathbf{x}_n}{\sum_{n=1}^{N} g_{kn}} = \frac{1}{n_k} \sum_{\mathbf{x}_n \in C_k} \mathbf{x}_n.$$

Using tedious yet straightforward algebra, we have shown the the problem of hard K-means clustering can be understood as a constrained matrix factorization problem.

For the functional of NMF decomposition defined in Equation (5.1) by definition we have:

$$\begin{aligned} \|X - FG\|_F^2 &= \sum_{k=1}^{K} \left(\sum_{n=1}^{N} g_{nk} \|\mathbf{x}_n - \mathbf{f}_k\| \right)^2 \\ &= \sum_{k=1}^{K} \left(\sum_{n=1}^{N} g_{kn}^2 \|\mathbf{x}_n - \mathbf{f}_k\|^2 + \sum_{n' \neq n} H_{nn'} \right) \\ &= \sum_{k=1}^{K} \left(\sum_{n=1}^{N} g_{kn}^2 \|\mathbf{x}_n - \mathbf{f}_k\|^2 \right), \end{aligned} \tag{45}$$

where we have used:

$$H_{nn'} = g_{kn} g_{kn'} (\mathbf{x}_n - \mathbf{f}_k)^T (\mathbf{x}_{n'} - \mathbf{f}_k) = 0, \tag{46}$$

since G is an orthogonal matrix $GG^T = I$ [DHS05]. Ding et al. [DHS05] showed that (46) are close to zero and therefore $\mathcal{L}(X, FG^T)$ is well approximated by $\mathcal{J}_K$ in the following sense:

$$\left| \mathcal{L}(X, FG^T) - \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbf{g}_{nk}^2 \|\mathbf{x}_n - \mathbf{f}_k\|^2 \right| \leq \mathcal{J}_K. \tag{47}$$

The NMF decomposition has been studied early by Golub and Paatero [PT94b],[XHP99]. Several families of algorithms are proposed to solve this matrix approximation problem. In other words, the factorization FG characterizes the information of X that can be described by the cluster structures contained in F with the indicators contained in G. This formulation of the K-means objective function highlights some good properties of the matrices F and G [RLN13]. These interesting properties are rich in opportunities, in fact, the consideration of these constraints on G allows us to develop different variants of NMF algorithms.

.1.2 The Collaborative approach – Computations

The pairwise collaborative C_l term between the NMF views l and the others views $l' \neq l$ writes:

$$\mathcal{J}(l) = \sum_{n=1}^{N}\sum_{k=1}^{K}\left(g_{kn}^{(l)}\right)^2 \left\|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\right\|^2 + \sum_{l'\neq l}\beta_{l,l'}\mathcal{C}_{l,l'}(F^{(l)}, G^{(l)}) \tag{48}$$

where

$$\mathcal{C}_{l,l'}(F^{(l)}, G^{(l)}) = \sum_{n=1}^{N}\sum_{k=1}^{K}\left(g_{kn}^{(l)} - g_{kn}^{(l')}\right)^2 \left\|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\right\|^2 \tag{49}$$

We consider the functional $\mathcal{C}(F^{(l)}, G^{(l)})$ defined in Equation (48). Note that $G^{(l)} \in U$ where U is a family of all fuzzy partition matrices:

$$U = \left\{ g_{nk}^{(l)} \in \{0,1\}; \sum_{k=1}^{K} g_{kn}^{(l)} = 1,\ \forall n,\ 0 \leq \sum_{n=1}^{N} g_{kn}^{(l)} \leq N, \forall k \right\} \tag{50}$$

We solve the following optimization under contraints problem:

$$\min \mathcal{J}(l) \text{ such that } G^{(l)} \in U \text{ and } \sum_{l'\neq l}\beta_{l,l'} = 1. \tag{51}$$

The corresponding Laplacian writes:

$$\mathcal{L}(F^{(l)}, G^{(l)}, \lambda, \nu) = \mathcal{J}(l) - \lambda\left(\sum_{k=1}^{K} g_{nk}^{(l)} - 1\right) - \nu\left(\sum_{l'\neq l}\beta_{l,l'} - 1\right)$$

where λ and ν are Lagrange multipliers and where: The necessary conditions leading to the local minimum are:

$$\begin{cases} \frac{\partial\mathcal{L}(F^{(l)},G^{(l)},\lambda,\nu)}{\partial\lambda} = 0, & \frac{\partial\mathcal{L}(F^{(l)},G^{(l)},\lambda,\nu)}{\partial\nu} = 0, \\ \frac{\partial\mathcal{L}(F^{(l)},G^{(l)},\lambda,\nu)}{\partial g_{kn}^{(l)}} = 0, & \frac{\partial\mathcal{L}(F^{(l)},G^{(l)},\lambda,\nu)}{\partial f_{mk}^{(l)}} = 0. \end{cases} \tag{52}$$

The first two conditions in Equation (52) return the normalization conditions:

$$\sum_{k=1}^{K} g_{kn}^{(l)} = 1 \text{ and } \sum_{l'\neq l}\beta_{l,l'} = 1. \tag{53}$$

We have for some $s \in \{1,\ldots,K\}$ and some $t \in \{1,\ldots,N\}$ we have

$$\frac{\partial\mathcal{L}(F^{(l)}, G^{(l)}, \lambda, \nu)}{\partial g_{st}^{(l)}} = \frac{\partial\mathcal{J}(l)}{\partial g_{st}^{(l)}} - \lambda$$

Note that

$$\frac{\partial}{\partial g_{st}^{(l)}}\left(g_{kn}^{(l)}\right)^2 = 2g_{kn}^{(l)}\delta_{sk}\delta_{tn},$$

and that

$$\frac{\partial}{\partial g_{st}^{(l)}}\left(g_{kn}^{(l)} - g_{kn}^{(l')}\right)^2 = 2(g_{kn}^{(l)} - g_{kn}^{(l')})\delta_{sk}\delta_{tn}.$$

Therefore

$$\frac{\partial \mathcal{L}(F^{(l)}, G^{(l)}, \lambda, \nu)}{\partial g_{st}^{(l)}} = 2g_{st}^{(l)}\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2 + 2\sum_{l'\neq l}\beta_{l,l'}\left(g_{st}^{(l)} - g_{st}^{(l')}\right)\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2 - \lambda.$$

Solving of $\frac{\partial \mathcal{L}(F^{(l)}, G^{(l)}, \lambda, \nu)}{\partial g_{st}^{(l)}} = 0$ we get:

$$g_{st}^{(l)} = \frac{\lambda + 2\sum_{l'\neq l}\beta_{l,l'}g_{st}^{(l')}\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2}{2(\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2 + \sum_{l'\neq l}\beta_{l,l'}\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2)} \tag{54}$$

$$g_{st}^{(l)} = \frac{\lambda + 2\sum_{l'\neq l}\beta_{l,l'}g_{st}^{(l')}\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2}{4\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2} \tag{55}$$

By using $\sum_{s=1}^{K} g_{st}^{(l)} = 1$ we get:

$$\lambda = 4\left(1 - \frac{1}{2}\left(\sum_{l'\neq l}\beta_{l,l'}\sum_{s=1}^{K}g_{st}^{(l')}\right)\right)\left(\sum_{s=1}^{K}\frac{1}{\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2}\right)^{-1} \tag{56}$$

By plugging the above expression into Equation (54) the following we get Equations (57).

$$g_{st}^{(l)} = \frac{1}{2}\left(\sum_{l'\neq l}^{L}\beta_{l,l'}g_{st}^{(l')}\right) + w_{st}^{(l)}\left(1 - \frac{1}{2}\left(\sum_{l'\neq l}^{L}\beta_{l,l'}\sum_{c=1}^{K}g_{tc}^{(l')}\right)\right) \tag{57}$$

where

$$w_{st}^{(l)} = \left(\sum_{c=1}^{K}\frac{\|\mathbf{x}_t^{(l)} - \mathbf{f}_s^{(l)}\|^2}{\|\mathbf{x}_t^{(l)} - f_c^{(l)}\|^2}\right)^{-1}$$

To compute the update of the prototypes for some $s \in \{1, \ldots, K\}$ and some $t \in \{1, \ldots, M\}$, the necessary condition for the minimum of the objective function is of the form:

$$\frac{\partial \mathcal{L}(F^{(l)}, G^{(l)}, \lambda, \nu)}{\partial f_{ts}^{(l)}} = 0 \tag{58}$$

By using the definition of $\|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\|^2$ in (52) and by differentiation with respect to $f_{ts}^{(l)}$ we get:

$$\frac{\partial}{\partial f_{ts}^{(l)}}\sum_{m=1}^{M}\left(x_{mn}^{(l)} - f_{mk}^{(l)}\right)^2 = -2\left(x_{mn}^{(l)} - f_{mk}^{(l)}\right)\delta_{ks}\delta_{mt}. \tag{59}$$

Therefore we have:

$$\frac{\partial}{\partial f_{ts}^{(l)}} \|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\|^2 = -2 \sum_{n=1}^{N} \left(x_{tn}^{(l)} - f_{ts}^{(l)} \right). \tag{60}$$

and also that

$$\frac{\partial \mathcal{J}(l)}{\partial f_{ts}^{(l)}} = -2 \sum_{l' \neq l}^{L} \sum_{n=1}^{N} \beta_{l,l'} \left(g_{sn}^{(l)} - g_{sn}^{(l')} \right)^2 \left(x_{tn}^{(l)} - f_{ts}^{(l)} \right). \tag{61}$$

For $t = 1, \ldots, M$ and $s = 1, \ldots, K$, solving the Equation (58) we get the following update rule for centroids:

$$f_{ts}^{(l)} = \frac{\sum_{n=1}^{N} \left(g_{sn}^{(l)} \right)^2 x_{tn}^{(l)} + \sum_{l' \neq l}^{L} \sum_{n=1}^{N} \beta_{l,l'} \left(g_{sn}^{(l)} - g_{sn}^{(l')} \right)^2 x_{tn}^{(l)}}{\sum_{n=1}^{N} \left(g_{sn}^{(l)} \right)^2 + \sum_{l' \neq l}^{L} \sum_{n=1}^{N} \beta_{l,l'} \left(g_{sn}^{(l)} - g_{sn}^{(l')} \right)^2}, \tag{62}$$

.1.3 NMF–Computations

The NMF functional is the squared Frobenius norm of a matrix difference, it can be written as:

$$\begin{aligned} \|X - FG\|_F^2 &= tr[(X - FG)^T (X - FG)] && \text{(63a)} \\ &= tr[X^T X] && \text{(63b)} \\ &- tr[X^T FG] && \text{(63c)} \\ &- tr[G^T F^T X] && \text{(63d)} \\ &+ tr[G^T F^T FG] && \text{(63e)} \end{aligned}$$

To derive the update rules for F, G we use the gradient descent method:

$$\begin{cases} G = G - \eta_G \circ [\nabla_G (\|X - FG\|_F^2)], \\ F = F - \eta_F \circ [\nabla_F (\|X - FG\|_F^2)]. \end{cases} \tag{64}$$

We now compute the gradients $[\nabla_G (\|X - FG\|_F^2)]$ and $[\nabla_F (\|X - FG\|_F^2)]$. In order to compute these gradients in matrix form we make use of the following properties of the gradient of traces of product with constant matrix A:

$$\begin{aligned} \nabla_X tr(AX) &= A^T && \text{(65a)} \\ \nabla_X tr(X^T A) &= A && \text{(65b)} \\ \nabla_X tr(X^T AX) &= (A + A^T)X && \text{(65c)} \\ \nabla_X tr(XAX^T) &= X(A + A^T) && \text{(65d)} \end{aligned}$$

For the term in Equation (63b) we have $\nabla_G tr(X^T X) = 0$since the term does not depend on G. For the second term, in Equation (63c), we use Equation (65a) to obtain:

$$\nabla_G tr(X^T FG) = (X^T F)^T = F^T X. \tag{66}$$

For the third term, in Equation (63d), we use Equation (65b) to obtain:

$$\nabla_G tr(G^T F^T X) = F^T X. \tag{67}$$

For the last term, in Equation (63e) we notice that $F^T F$ is constant with respect to G and we use Equation (65c) to write:

$$\nabla_G tr(G^T F^T F G) = [F^T F + (F^T F)^T] G = 2F^T F G. \tag{68}$$

Putting all the terms in Equations (66), (67) and (68) together with the appropriate signs of Equation (63), we finally get:

$$\nabla_G (\|X - FG\|_F^2) = -2F^T X + 2F^T F G. \tag{69}$$

To compute $\nabla_F (\|X - FG\|_F^2)$, we proceed analogously. First, we also have $\nabla_F tr(X^T X) = 0$. For the second term, in Equation (63c), we use the cyclic permutation property of the trace $tr(ABC) = tr(CAB) = tr(BCA)$ and also we use Equation (65a) to obtain:

$$\nabla_F tr(X^T F G) = \nabla_F tr(G X^T F) = (G X^T)^T = X G^T. \tag{70}$$

For the third term, in Equation (63d), we use a permutation in Equation (65b) to obtain:

$$\nabla_F tr(G^T F^T X) = \nabla_F tr(X G^T F^T) = X G^T. \tag{71}$$

For the last term, in Equation (63e) we permute twice and we notice that GG^T is constant with respect to F and we use Equation (65d) to write:

$$\nabla_F tr(G^T F^T F G) = \nabla_F tr(F G G^T F^T) = F[(GG^T)^T + GG^T] = 2FGG^T. \tag{72}$$

And putting the terms in Equations (70), (71) and (72) together we get:

$$\nabla_F (\|X - FG\|_F^2) = -2XG^T + 2FGG^T. \tag{73}$$

We define the learning rates so that any subtraction dissapears from the update rule as follows:

$$\begin{cases} \eta_F = \frac{F}{FGG^T}, \\ \eta_G = \frac{G}{F^T F G}. \end{cases} \tag{74}$$

The additive update rules have thus been transformed into multiplicative update rules, which cannot generate negative elements since all values are positive and only multiplications and divisions are involved at each update:

$$\begin{cases} F = F \circ \frac{XG^T}{FGG^T}, \\ G = G \circ \frac{F^T X}{F^T F G}. \end{cases} \tag{75}$$

.1.4 Remark on NMF collaborative approach

Let us recall the pairwise collaborative $C_{l,l'}$ term between the NMF views l-th and l' given by:

$$\mathcal{C}_{l,l'}(F^{(l)}, G^{(l)}) = \|(G^{(l)} - G^{(l')}) \circ D^{(l)}\|_F^2. \tag{76}$$

Let us write the element wise this functional which is the Frobenius norm of a $K \times N$. matrix. The (k, n) element of this matrix writes:

$$\left((G^{(l)} - G^{(l')}) \circ D^{(l)}\right)_{kn} = \left(g_{kn}^{(l)} - g_{kn}^{(l')}\right) \left\|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\right\|$$

By definition of the Frobenius norm we have:

$$\|(G^{(l)} - G^{(l')}) \circ D^{(l)}\|_F^2 = \sum_{k=1}^{K} \sum_{n=1}^{N} \left(g_{kn}^{(l)} - g_{kn}^{(l')}\right)^2 \left\|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\right\|^2 \tag{77}$$

Note also that the local functional writes

$$\|X^{(l)} - F^{(l)}G^{(l)}\|_F^2 = \sum_{k=1}^{K} \sum_{n=1}^{N} \left(g_{kn}^{(l)}\right)^2 \left\|\mathbf{x}_n^{(l)} - \mathbf{f}_k^{(l)}\right\|^2 \tag{78}$$

By adding the functional in (78) with the functional in (77) we get the collaborative functional proposed by Pedrycz [Ped05] for fuzzy C-means.

.1.5 Remarks on NMF gradients

We first recall the gradients of the local term.

$$\nabla_{F^{(l)}}(\|X^{(l)} - F^{(l)}G^{(l)}\|_F^2) = -2X^{(l)}(G^{(l)})^T + 2F^{(l)}G^{(l)}(G^{(l)})^T, \tag{79}$$

and that

$$\nabla_{G^{(l)}}(\|X^{(l)} - F^{(l)}G^{(l)}\|_F^2) = -2(F^{(l)})^T X^{(l)} + 2(F^{(l)})^T F^{(l)} G^{(l)}. \tag{80}$$

Let us consider $t \in \{1, \ldots, M\}$ and some $s \in \{1, \ldots, K\}$ arbitrary fixed The element wise equivalent for the gradient in Equation (79) is given by:

$$\frac{\partial}{\partial f_{ts}^{(l)}} \|X^{(l)} - F^{(l)}G^{(l)}\|_F^2 = -2 \sum_{n=1}^{N} x_{tn}^{(l)} g_{sn}^{(l)} + 2 \sum_{n=1}^{N} g_{sn}^{(l)} \sum_{k=1}^{K} f_{tk}^{(l)} g_{kn}^{(l)} \tag{81}$$

Let us consider now $t \in \{1, \ldots, N\}$ and some $s \in \{1, \ldots, K\}$ arbitrary fixed The element wise equivalent for the gradient in Equation (80) is given by:

$$\frac{\partial}{\partial g_{st}^{(l)}} \|X^{(l)} - F^{(l)}G^{(l)}\|_F^2 = -2 \sum_{m=1}^{M} f_{ms}^{(l)} x_{mt}^{(l)} + 2 \sum_{m=1}^{M} f_{ms}^{(l)} \sum_{k=1}^{K} f_{mk}^{(l)} g_{kt}^{(l)} \tag{82}$$

Note that the gradient of the collaborative functional $\mathcal{C}_{l,l'}(F^{(l)}, G^{(l)})$ with respect to $F^{(l)}$ writes as follows: Note that we have by definition:

$$\begin{aligned} \nabla_{F^{(l)}} \mathcal{C}_{l,l'}(F^{(l)}, G^{(l)}) &= -2X^{(l)}[(G^{(l)} - G^{(l')})^{\circ 2}]^T \\ &+ \mathbf{21}_{M \times N}[(G^{(l)} - G^{(l')})^{\circ 2}]^T \circ F^{(l)}, \end{aligned} \tag{83}$$

Let us consider $t \in \{1, \ldots, M\}$ and some $s \in \{1, \ldots, K\}$ arbitrary fixed. Let us write this gradient element wise. We first note that:

$$\frac{\partial}{\partial f_{ts}^{(l)}} \|x_n^{(l)} - f_k^{(l)}\|^2 = -2 \sum_{n=1}^{N} \left(x_{tn}^{(l)} - f_{ts}^{(l)}\right) \tag{84}$$

Therefore we get:

$$\frac{\partial \mathcal{C}_{l,l'}(F^{(l)}, G^{(l)})}{\partial f_{ts}^{(l)}} = -2 \sum_{n=1}^{N} \left(g_{sn}^{(l)} - g_{sn}^{(l')}\right)^2 \left(x_{tn}^{(l)} - f_{ts}^{(l)}\right). \tag{85}$$

and

$$\begin{aligned} \nabla_{G^{(l)}} \mathcal{J}(l) &= -2(F^{(l)})^T X^{(l)} + 2(F^{(l)})^T F^{(l)} G^{(l)} \\ &- 2 \sum_{l' \neq l} \beta_{l,l'} [G^{(l)} - G^{(l')} \circ (D^{(l)})^{\circ 2}]. \end{aligned} \tag{86}$$

.1.6 Convex NMF Consensus gradients

We compute the gradients in consensus context. We have:

$$\begin{aligned} \nabla_W \mathcal{E}(W, G) &= \sum_{l=1}^{L} \beta_l \cdot \nabla_W E_l(W, G) \\ &= 2 \sum_{l=1}^{L} \beta_l \cdot (\|X - XWG\|_F - \|X - XW^{(l)}G^{(l)}\|_F) \nabla_W (\|X - XWG\|_F^2) \\ &= 2 \sum_{l=1}^{L} \beta_l \cdot (\|X - XWG\|_F - \|X - XW^{(l)}G^{(l)}\|_F)(-2XG^T + 2XWGG^T) \end{aligned} \tag{87}$$

and

$$\begin{aligned} \nabla_G \mathcal{E}(W, G) &= \sum_{l=1}^{L} \beta_l \cdot \nabla_G E_l(W, G) \\ &= 2 \sum_{l=1}^{L} \beta_l \cdot (\|X - XWG\|_F - \|X - XW^{(l)}G^{(l)}\|_F) \nabla_G (\|X - XWG\|_F^2) \\ &= 2 \sum_{l=1}^{L} \beta_l \cdot (\|X - XWG\|_F - \|X - XW^{(l)}G^{(l)}\|_F)(-2(XW)^T X + 2(XW)^T XWG) \end{aligned} \tag{88}$$

Note that $\nabla_W(\|X - XWG\|_F^2$ and $\nabla_G(\|X - XWG\|_F^2$ are given by Equation (5.6) and Equation (5.7) respectively. It follows that:

$$\begin{aligned} \nabla_W \mathcal{E}(W, G) = &- 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XWG\|_F \right) XG^T \\ &- 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XW^{(l)}G^{(l)}\|_F \right) XWGG^T \\ &+ 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XWG\|_F \right) XWGG^T \\ &+ 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XW^{(l)}G^{(l)}\|_F \right) XG^T \end{aligned} \tag{89}$$

and that:

$$\begin{aligned} \nabla_G \mathcal{E}(W, G) = &- 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XWG\|_F \right) (XW)^T X \\ &- 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XW^{(l)}G^{(l)}\|_F \right) (XW)^T FG \\ &+ 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XWG\|_F \right) (XW)^T XWG \\ &+ 4 \left(\sum_{l=1}^{L} \beta_l \cdot \|X - XW^{(l)}G^{(l)}\|_F \right) (XW)^T X \end{aligned} \tag{90}$$

.1.7 Convex NMF Collaborative gradients

We will now compute the gradient of the functional in Equation (5.17). We have:

$$\begin{cases} \nabla_{W^{(l)}}\mathcal{C}(l) = \nabla_{W^{(l)}}\mathcal{L}(W^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot \nabla_{W^{(l)}} C_{l,l'}(W^{(l)}, G^{(l)}) \\ \nabla_{G^{(l)}}\mathcal{C}(l) = \nabla_{G^{(l)}}\mathcal{L}(W^{(l)}, G^{(l)}) + \sum_{l' \neq l} \beta_{l,l'} \cdot \nabla_{G^{(l)}} C_{l,l'}(W^{(l)}, G^{(l)}) \end{cases} \tag{91}$$

We first notice that $\nabla_{W^{(l)}}(\|X - F^{(l)}G^{(l)}\|_F^2)$ and $\nabla_{G^{(l)}}(\|X - F^{(l)}G^{(l)}\|_F^2)$ are given by by Equation (5.6) and Equation (5.7) respectively. It remains to compute the gradient of $C_{l,l'}(W^{(l)}, G^{(l)})$. Note that we have by definition:

$$\begin{aligned} \nabla_{W^{(l)}} C_{l,l'}(W^{(l)}, G^{(l)}) &= -2X\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \\ &+ \mathbf{21}_{M\times N}\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \circ XW^{(l)}, \end{aligned} \tag{92}$$

where $\mathbf{1}_{M\times N}$ is is an $M \times N$ one's matrix and where $^{\circ 2}$ means element-wise power of the matrix. We also have that:

$$\nabla_{G^{(l)}} C_{l,l'}(W^{(l)}, G^{(l)}) = 2(G^{(l)} - G^{(l')}) \circ (D^{(l)})^{\circ 2}. \tag{93}$$

By adding all these partial gradient computations we get:

$$\begin{aligned} \nabla_{W^{(l)}}\mathcal{C}(l) &= -2X(G^{(l)})^T + 2XW^{(l)}G^{(l)}(G^{(l)})^T \\ &- 2\sum_{l' \neq l} \beta_{l,l'} X\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \\ &+ 2\sum_{l' \neq l} \beta_{l,l'} \mathbf{1}_{M\times N}\left[(G^{(l)} - G^{(l')})^{\circ 2}\right]^T \circ XW^{(l)}. \end{aligned} \tag{94}$$

and

$$\begin{aligned} \nabla_{G^{(l)}}\mathcal{C}(l) &= -2(XW^{(l)})^T X + 2(XW^{(l)})^T XW^{(l)}G^{(l)} \\ &- 2\sum_{l' \neq l} \beta_{l,l'}\left[(G^{(l)} - G^{(l')}) \circ (D^{(l)})^{\circ 2}\right]. \end{aligned} \tag{95}$$

Domain name recommendation based on neural network

.2 Abstract

The number of website is increasing speedily, and clients purchase their website from the enterprise that suggests them the best domain name with a good price. In order to give the relevant domain name, enterprise is always eager to have a good system of suggestion that suits the client request.

Recommender system has been an effective key solution to guide users in a personalized way for discovering the domain name they might be interested in from a large space of possible suggestion. They have become fundamental applications that provides to users the best domain name that meet their needs and preferences.

In this work, we will use a recommender system based on neural network as it is capable to solve many complex tasks and gives better customer satisfaction. We proposed two approaches to recommend domain name, both of these approaches are based on neural network. The first one consists on discovering the similarity between the vocabulary of domain name, while the second one is finding the relevant Top Level Domain corresponding to the context of the domain name. First experiments on GANDI datasets shows the effectiveness of the proposed approaches.

.3 Introduction

Recommender systems can be defined as programs which attempt to recommend the most suitable items, products or services to particular users by predicting a user's interest in an item based on related information about the items, the users and the interactions between items and users. The aim of developing recommender systems is to reduce information overload by retrieving the most relevant information and services from a huge amount of data, thereby providing personalized services. The most important feature of a recommender system is its ability to guess a user's preferences and interests by analyzing the behavior of this user or the behavior of other users to generate personalized recommendations.

Various recommender system techniques have been proposed, and many sorts of recommender system software have been developed recently for a variety of applications. Researchers recognize that recommender systems offer great opportunities and challenges for business, government, education, and other domains, with more recent successful developments of recommender systems for real-world applications becoming apparent.

The past few decades have witnessed the tremendous successes of the use of neural network in many application domains such as speech recognition, image analysis and natural language processing. Applying neural network into recommender system has been gaining momentum thanks to its state-of-the-art performances and high-quality recommendations. In contrast to traditional recommendation models,

neural network provides a better understanding of user's demands, item's characteristics and historical interactions between them.

In this paper, we suggest a system of recommendation of fully qualified domain name based on neural network. When a customer wants to purchase a domain name he rarely knows his qualified domain name, therefore, we will make a system of recommendation to suggest to the customer both the domain name and the extension that suits his recommend by referring to his localization and his research.

The first part is the state of the art where we will present the existing methods concerning text similarity, through three approaches: Probabilistic-based approach, Prediction-based approach and Hybrid approach which is the combination of the two previous approaches.

The second part is the description of the two models for suggesting domain name that are based on neural network: the first method tries to find similarity based on neighbors, while the second one predicts the TLD (Top Level Domain) corresponding to the context of domain name.

The third part concerns the experiment results for the two models based on the real purchase database of GANDI.

.4 State of the art

In this state of the art, we will present the most used methods to find text similarity, based on three approaches: 1) Probabilistic-based approach which is based on probabilistic models, and consists on searching in the whole corpus to find text similarity, 2) Prediction-based approach that more commonly known as embeddings or neural language models, which consists on looking for the neighbors of the text to find text similarity, 3) Hybrid approach which is the combination of the two previous approaches.

.4.1 Probabilistic-based approach

Probabilistic-Based approach is based on probabilistic models, this approach gives a sparse vector that can be easily interpretable by humans. However, the sparse vector can not be very efficient to detect the semantic meaning of the word.

Latent Semantic Indexing

Latent Semantic Indexing (LSI) [Dee+90] is a well-known method that allows to discover hidden concepts in a document. It represents the documents and terms in a way to express the semantic relationship between document/term, document/document or term/term. LSI constructs a term-document matrix from a large piece of text, this matrix describes the occurrences of terms in documents, it is a sparse matrix whose rows correspond to terms and whose columns correspond to documents. As an example of the weighting of the elements of the matrix is tf-idf (term frequency inverse document frequency): the weight of an element of the matrix is proportional to the number of times the terms appear in each document.

After the construction of the occurrence matrix, LSI uses Singular Value Decomposition (SVD) to the term-document matrix in order to reduce the number of rows while preserving the similarity structure among columns. Then, the similarity between words could be compared using cosine similarity.

Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) [BNJ03] is a generative probabilistic model for collections of discrete data such as text corpora. LDA consider that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. In other words, LDA allows to discover topics in text documents, where each topic is characterized by a list of words. The generative process for each document in the corpus is:

1. Choose a distribution over topics $\theta \sim Dir(\alpha)$
2. For each word in the document:
   - Choose a topic from the distribution over topics $z_n \sim Multinomial(\theta)$
   - Choose a word from the vocabulary $w_n \sim Multinomial(\phi_{z_n})$

Each document gives the topic with certain probabilities, each topic in the document has some words with certain probabilities, these words are similar as they share the same context.

Global Vectors for Word Representation (Glove)

Glove [PSM14] is an unsupervised learning algorithm that gives vector representations of words. This algorithm is designed to take the advantages of the two major families of learning word vectors: global matrix factorization methods such as Latent Semantics Analysis [Dee+90] and local context window methods such as Skip gram model [MYZ13]. Glove uses ratios of co-occurrence probabilities, rather than the co-occurrence probabilities themselves, as the ratios between the probabilities of words appearing next to each other carry more information than considering those probabilities individually.

.4.2 Prediction-based approach

Prediction-based approach is more commonly known as embeddings or neural language models, this approach gives a dense vector that gives a good result in the semantic field of the word. However, this dense vector can not be easily interpretable by humans.

word2vec

Word2vec [Mik+13a] is a model for learning vector representations of words, called "word Embeddings".
Word embeddings are a way to capture similarity across words based on the contexts in which they appear [Ron14][Mey16]. The intuition is that words with similar meanings often occur near each other in texts. There are two principal models in word2vec:

- Continuous Bag of Words (CBOW) : Predicting the current word based on the context.
- Skip gram : Predicting the context based on the current word.

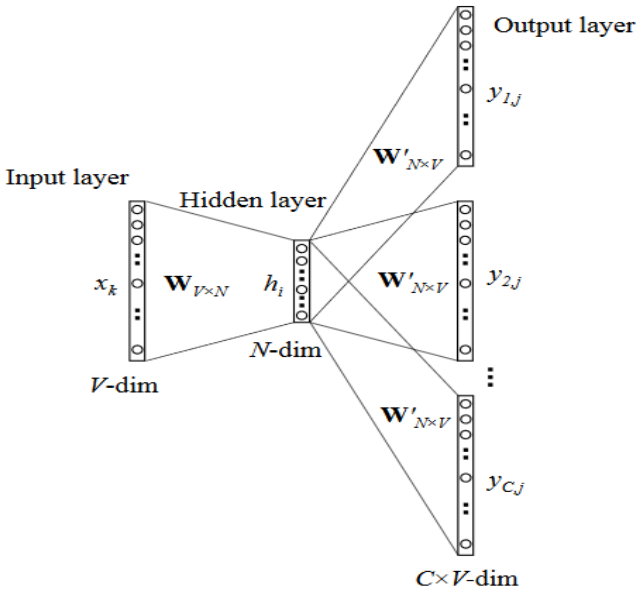


FIGURE 9: The skip gram model

In skip-gram model, we choose a window size C (often this window size is equal to five). Then, we train the neural network to do the following: Given the current word (input word), look at the C words nearby (C words behind and C words ahead). We train the neural network in the vocabulary in order to get the weights of the two matrix between the input/hidden layer (W) and hidden/output layer (W').

Figure 9 shows the schema of the skip gram model. Let's consider that the vocabulary size is V, and the hidden layer size is N. The input is a one-hot encoded vector, which means for a given input word, only one out of V units, $x_1, ..., x_V$, will be 1, and all other units are 0.
The units on adjacent layers are fully connected.

Finally, we will obtain two representations of each word in the vocabulary which are the rows of W and W'. Our goal was to obtain the weights of two matrix W_{V*N} and W'_{N*V} to be able to get the vector representation of words. After getting the vector representation of each word we can measure similarity between words in the vocabulary using cosine similarity.

Paragraph Vector (Doc2vec)

Paragraph Vector [LM14] is an unsupervised algorithm that models variable-length pieces of texts, such as sentences, paragraphs or documents. The algorithm is an adaptation of word2vec, while word2vec can generate word vector, paragraph vector can generate both document vector and word vector depending on a context (topic).

Paragraph Vector has two approaches that are similar to the models of Word2Vec [Mik+13a] (Continuous Bag of Words (CBOW) and Skip-gram).

- Distributed Memory Model of Paragraph Vectors (PV-DM)

  This approach is similar to CBOW, the only difference is that an additional input representing the paragraph id is added. PV-DM attempts to predict a word given its surrounding words and the context (topic) provided by the paragraph id.

PV-DM uses the concept of memory, it remembers what is missing from the current context (topic) of the paragraph, which wasn't part of Word2vec. For example, a text that contains words about mathematics is more likely to use scientific words.

- Distributed Bag of Words version of Paragraph Vector (PV-DBOW)

  This model is similar to Skip-gram, the only difference is that it takes paragraph id as input and it predicts words in the window without any restriction on word order. PV-DBOW attempts to predict words randomly given the paragraph id.

Paragraph Vector has the potential to overcome the weaknesses of many machine learning algorithms as bag-of-words models [Har54]. Its robustness resides in:

- Taking word order into account.
- Taking the semantic of words into consideration.
- Representing the text input as a variable length vector.
- Learning from unlabeled data.

Skip-Thought Vectors

Skip-Thought Vectors [Kir+15] is a model that allows to get sentence vector and it doesn't require labeled data to train (unsupervised sentence vectors). This model inspires from skip-gram model [Mik+13b] and attempts to extend it to sentences. So, instead of having a word to predict its context, Skip-Thought vectors have a sentence to predict the sentences around it. This model follows the encoder-decoder model that is so pervasive in neural machine translation [Cho+14]. Encoder maps words to a sentence vector and decoder is used to generate the surrounding sentences.

Skip-thought model uses an RNN encoder with GRU activations and an RNN decoder with a conditional GRU. Gated Recurrent Unit (GRU) model [Chu+14] is simpler than Long Short Term Memory networks (LSTM) models [HS97], and has been shown to perform as well as LSTM on sequence modeling tasks.

There are one encoder and two decoders, all of them are constructed from recurrent neural networks (RNN). The encoder takes as input a sentence and outputs a vector. The two decoders take the vector as input: The first attempts to predict the previous sentence and the second attempts to predict the next sentence.
While training, Skip-Thought vectors uses two separate models: the first one "uni-skip" is unidirectional encoder (reads the sentence in the forward direction) while the other one "bi-skip" is bidirectional model with forward and backward encoders (reads the sentence forwards and backwards and concatenates the results). Another experiment "combine-skip" was added by using a combined model which consists of the concatenation of the vectors from uni-skip and bi-skip.

Figure 10 illustrates the model of Skip Thought, it shows the input sentence and the two predicted sentences. We are given a sentence tuple (s_{i-1}, s_i, s_{i+1}). with s_i the i-th sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1}. In this example, the input is the sentence triplet :

- A door confronted her.
- She stopped and tried to pull it open.

- It didn't budge.

Unattached arrows are connected to the encoder output. Colors indicate which components share parameters. $< eos >$ is the end of sentence token.

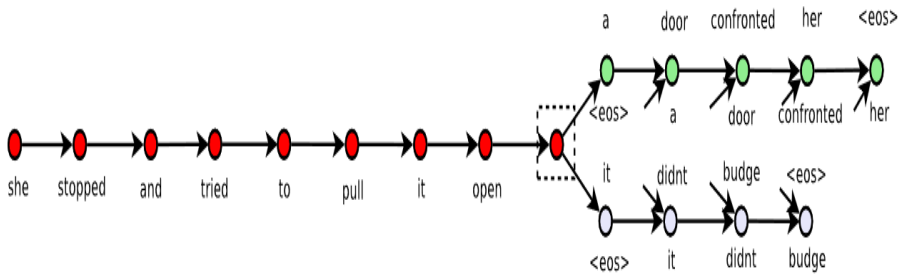


FIGURE 10: The Skip Thought model

Skip-Thought framework proves to be a robust way of modeling the semantic content of sentences through the following tasks:

- Semantic relatedness
- Paraphrase detection
- Image-sentence ranking
- Classification benchmarks

.4.3 Hybrid approach

The question is whether distributional word representations are best learned from Probabilistic-based models or from Prediction-based models?

Probabilistic-based approach tend to form documents as a sparse mixed-membership of topics while Prediction-based approach tend to model documents as dense vectors. The former is simpler to interpret as the vector is sparse while the second one is better in giving the semantic of the word but the vector is uninterpretable.

To take advantages of both approaches, hybrid approach combines the best part of the two previous approaches.

LDA2vec

Lda2vec [Moo16] is a model that mixes two models: the dirichlet topic model (LDA) and word embeddings (word2vec). LDA [BNJ03] is able to create document (and topic) representations that are not so flexible but quite interpretable to humans. While word2vec [Mik+13b] create vector representations of words that can model semantic similarity between words but the vectors are uninterpretable by humans. Thus, Lda2vec aims to solve this by embedding both words and document vectors into the same space and trains both representations simultaneously.

Inspiring from the idea of Tomas Mikolov [Mik+13a] that word vectors can be summed together to form a semantically meaningful combination of both words. We will add word and document vectors to each other to construct a meaningful context vector of them.

For example, taking the word *Germany* and a document about *airlines*. Then the predicted similar words for *Germany* will be *France*, *Spain*, and *Austria*. But if we apply lda2vec, instead of predicting tokens similar to *Germany* alone, the predictions will be similar to both the document and the word, which will give as a result: *Lufthansa*, *Condor Flugdienst*, and *Aero Lloyd*.

Topic2vec

Topic2vec [Niu+15] is an approach that learns topic representations in the same semantic vector space with words. This model is inspired by word2vec, it is also separated in two models: CBOW which predicts both a word and topic vector using the context words, and Skip gram that predicts the context given the current word and its assigned topic by LDA.

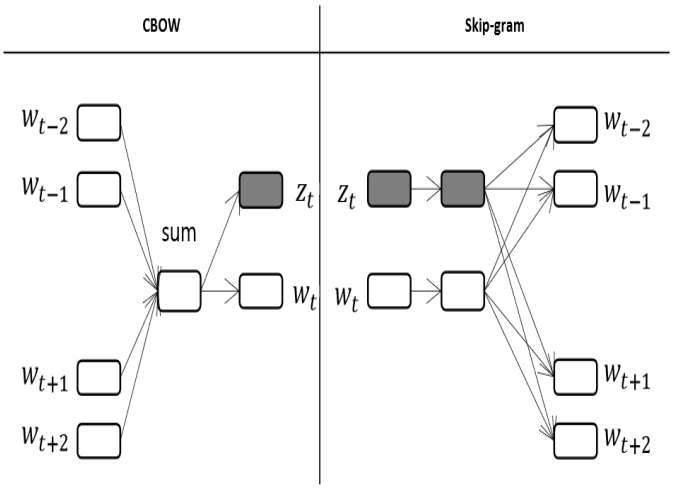


FIGURE 11: The architectures of Topic2Vec where $(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2})$ are context words and w_t is the current word paired with a topic z_t

Topic2Vec aims at learning topic representations along with word representations. When training, given a word-topic sequence of a document $D = w_1 : z_1, ..., w_N : z_N$, where w_i is the word and z_i is the topic inferred from LDA. The learning objective function for CBOW and Skip gram can be defined as follows:

$$F_{CBOW}(D) = \frac{1}{M}\sum_{i=1}^{M}(log\ p(w_i|w_{cxt}) + log\ p(z_i|w_{cxt}))$$

$$F_{Skip-gram}(D) = \frac{1}{M}\sum_{i=1}^{M}\sum_{-k\leqslant c\leqslant k}(log\ p(w_{i+c}|w_i) + log\ p(w_{i+c}|z_i))$$

Thus, topic representations are learned in the same semantic vector space with words.

.5 The proposed approaches

In this section, we present two approaches: the first one aims to find similarity between the vocabulary while the second one is finding the TLD that corresponds to the context of domain name.

.5.1 First approach

This approach is based on training the neural network to do the following: looking for the neighbors for each word in the purchase database in order to find similarity between words of the domain name.

First of all, we will preprocess on the purchase data; we will retrieve the purchase data to tokenize the domain name. Then, we will remove all the noise (stop words). After this, we will build the vocabulary which will be the fully qualified domain name which means the union of the domain name and TLD.

After this, we will train this model on the purchase database which will allow us to get the word vector for each word in the vocabulary.

Finally, we will make a scoring between the domain name and each TLD and we will suggest the TLD with a score above to γ.

Learning phase

We choose a window size C, which means, for each word in the vocabulary (domain name, TLD) we will look for the C words behind and the C words ahead. We will train the neural network to do the following: Given the input word, look at the C words nearby.

The input and output for the first approach is a list of vectors that has the size of the vocabulary, each word is represented as a one hot vector, which means, for a given input word, only one out of the words in the vocabulary will be 1, and all other

words are 0). The vocabulary is composed of domain name and TLD.

Algorithm 10: First approach

Learning phase
Choose a window size C.
Look at the C words nearby.
parameter : Parameter for the algorithm
C: window size
n: neurones
h_i: hidden layer
e_r: error threshold
γ: score threshold
W: weights (initialize randomly)
Input : A list of vectors that has the size of the vocabulary.
Output : The same list of input (vocabulary).
Choose a couple (Input, Output).
Calculate the state of the network by propagation.
Determine the cost.
Apply gradient descent.
Adjust the weights W.
Repeat until obtaining an error equal to e_r.
Prediction phase
Applying cosine similarity between the vector of domain name and each TLD.
Recommend the TLD with the score above of γ.

Prediction phase

After training the neural network, we will get the word vector for each word in the vocabulary (Domain name and TLD).

The prediction of TLD will be done by making a score between the domain name and every TLD. Then, we will recommend the TLD with a score above γ (fixed in the output).

.5.2 Second approach

This approach is based on constructing a neural network that takes as input the domain name and outputs a TLD. We will train the neural network on the purchase data in order to suggest the relevant TLD corresponding to the context of the domain name.

This approach is divided into two parts: the first part is training the neural network to predict the context of the domain name while the second part is based on classifying the TLD using an unsupervised learning method.

First of all, we will preprocess on the purchase data; we will retrieve the purchase data to tokenize the domain name. Then, we will remove all the noise (stop words). After this, we will build the vocabulary.

Learning phase

- `Learn to predict the context of the domain name`

After building the vocabulary, we will train the neural network on that vocabulary, in order to find the context of the domain name. The neural network will be trained to do the following: Taking the domain name as input and outputs the TLD corresponding to the context of the domain name.

The input for the second approach is a list of vectors where each vector is represented as a one hot vector and each vector has the dimension of the vocabulary of the domain name. The output is a list of vectors where each vector is represented as a one hot vector and each vector has the dimension of the vocabulary of the TLD.

Algorithm 11: Second approach

Learning phase
Prediction of the context of domain name
Classification of TLD
parameter : Parameter for the algorithm
n: neurones
h_i: hidden layer
e_r: error threshold
W: weights (initialize randomly)
Input : A list of vectors (Domain Name).
Output : A list of vectors (TLD).

Choose a batch of the couple (Input, Output).
Calculate the state of the network by propagation.
Determine the cost.
Apply gradient descent.
Adjust the weights.
Repeat until obtaining an error equal to e_r.
Prediction phase
Look for the cluster of the predicted TLD.
Apply cosine similarity between the predicted TLD and each TLD in the cluster.
Recommend the top K TLD.

- `Classification of TLD`

To classify TLDs, we will take the word vector for each TLD from the model of the first approach that we have already trained it. After getting the word vector for each TLD, we will calculate the TLD similarity matrix. Then, we will classify TLDs using an unsupervised learning method.

We used an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables.

This transformation is defined in such a way that the first component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set.

K-means is an unsupervised machine learning algorithm that groups a dataset into a specified number (k) of clusters. The algorithm is somewhat naive as it clusters the data into k clusters, even if k is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters.

So, here we will present how to choose the optimal number of clusters using elbow method [KM13], then we will apply k-means clustering.

– Determine optimal number of clusters (K)

The technique to determine the number of clusters(K), is called the elbow method. The idea of the elbow method is that it looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data.
Figure 12 shows an elbow, it is called an elbow method because the curve looks like a human arm and the elbow point gives us the optimum number of clusters. So, we will take the elbow point value as the final number of clusters, which is K=24.

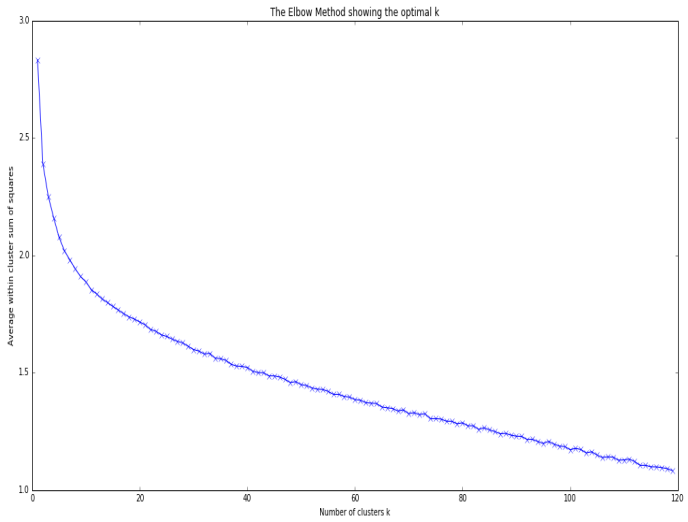


FIGURE 12: The elbow method

– Applying K-means clustering

After choosing the right K (number of clusters), we will apply K-means clustering with K=24.
Figure 13 shows the clusters of TLDs:

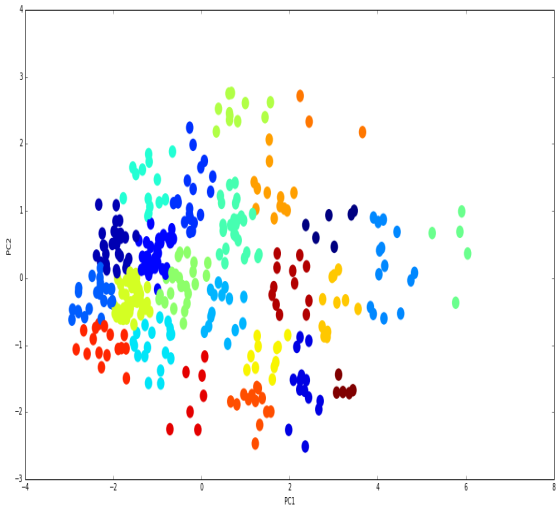


FIGURE 13: Clusters of TLDs

Prediction phase

After training the neural network, we will use it to predict the TLD that corresponds to the context of the domain name. Then, we will search for the cluster that corresponds to the predicted TLD, and we will make cosine similarity between the predicted TLD and each TLD in the cluster, Then we will suggest the top K TLDs, which means the one who has high cosine similarity, and we will save the decreasing order as the client could see the relevant TLD in the beginning.

.6 Experimental results

In this section, we will present the experimental results that we get from the two approaches. In this experiment, the sample was taking from Gandi database, the real purchases that has been done.

1. **First approach**

   In this approach, we took a sample from the purchase database of Gandi that has 10 million purchase, and we test the accuracy. The accuracy was based on : if the predicted TLD coincide with the 10 TLD that has the highest score.

Dataset	Domain Name	TLD	accuracy
10M	6.293.598	386	3,98%

TABLE 5: The evaluation of the first approach

   Indeed, the goal of this approach is to find similarity among the purchase data not the context of the domain name that will give the relevant TLD, that's why we took all the domain name that were purchased not only the one that were bought with GTLD (Generic Top Level Domain). This low accuracy is probably due to the domain name that has been purchased with ccTLD (Country Code Top Level Domain).

2. **Second approach**

   In this approach we took a sample of 10.000 purchase domain name, to be able to make a setting on the neural network in order to choose the best parameters that will give the best results.

Dataset	Domain Name	TLD
10K	7873	45

TABLE 6: The evaluation of the second approach

The activation function is a node that we add to the output end of any neural network. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

To train the neural network, we tested three activation function for the hidden layers : sigmoid function, tanh (hyperbolic tangent), Relu (Rectified Linear Unit).

	Sigmoid $(\frac{1}{1+e^{-x}})$	Relu $(\max(0,x))$	tanh $(\frac{2}{1+e^{-2x}}-1)$
One hot vector	96.9%	96.76%	96.89%
Vector using indexes	81.71%	80.67%	81.08%

TABLE 7: Accuracy by comparing different activation functions on 10K sample

Figure 16 shows the accuracy and the cross entropy using one hot vector with sigmoid function while figure 19 shows the accuracy and cross entropy using a vector with indexes with sigmoid function. It is noticeable that when the cross entropy goes down the accuracy increases until reaching a predefined threshold.

The axis of abscissa represents the number of iteration while the axis of the ordinate represents the accuracy (left) and the cross entropy (right).

Results with one hot vector is much better than using vector with indexes, as the accuracy is much better and also the number of iteration is lower than the one used by the vector with indexes.

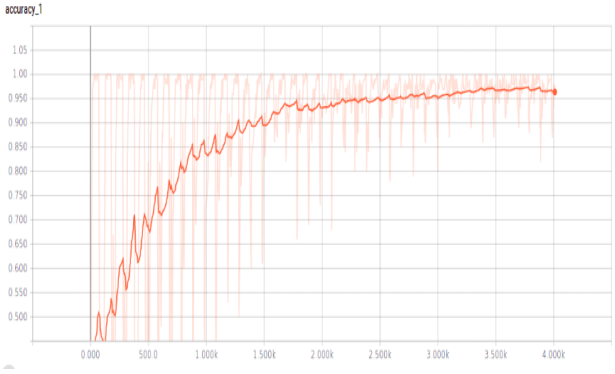


FIGURE 14: Accuracy

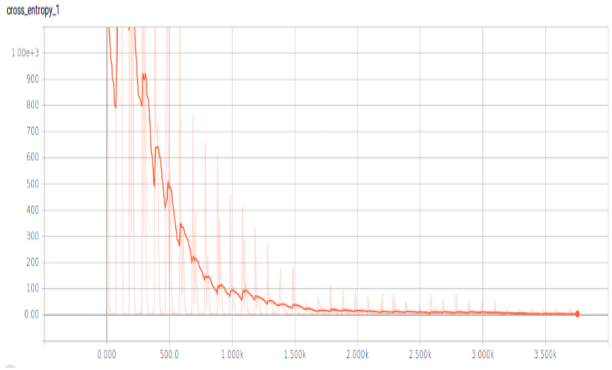


FIGURE 15: Error

FIGURE 16: Results using one hot vector and sigmoid function

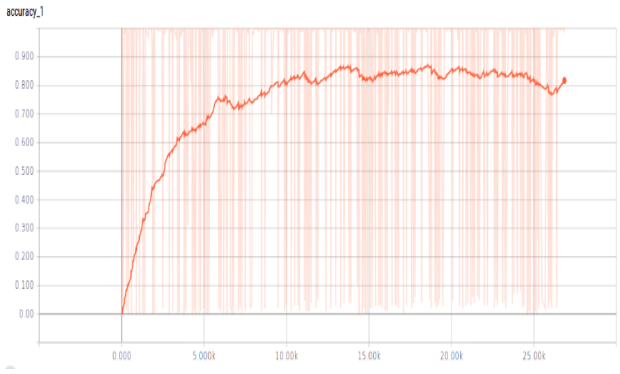


FIGURE 17: Accuracy

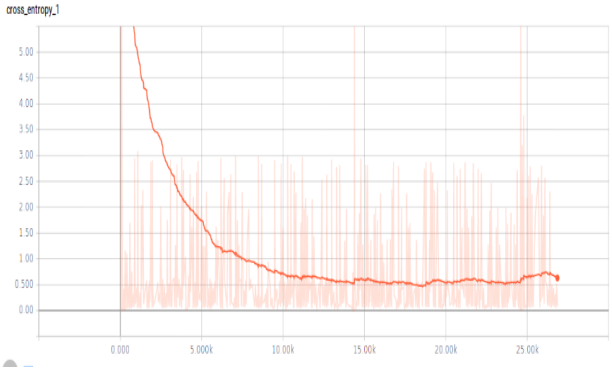


FIGURE 18: Error

FIGURE 19: Results using vector with indexes and sigmoid function

.7 Conclusion

In this work, we have proposed two approaches based on neural network to suggest a relevant domain name. The first approach consists on finding similarity between words of domain name, while the second one is predicting TLD corresponding to the context of domain name. Experiments on GANDI database has shown efficient results for the second approach.

Publications

.8 Publications

- Kaoutar Benlamine, Younès Bennani, Basarab Matei, Nistor Grozavu, "Quantum Semi Non-negative Matrix Factorization", 12th International Conference on Soft Computing and Pattern Recognition (SoCPar), 15th-18th December 2020, ONLINE.
- Kaoutar Benlamine, Younès Bennani, Nistor Grozavu, Basarab Matei, "Quantum Collaborative K-means", IEEE World Congress on Computational Intelligence, IEEE International Joint Conference on Neural Network (IEEE IJCNN), 19-24 July, 2020, Glasgow (UK).
- Kaoutar Benlamine, Younès Bennani, Ahmed Zaiou, Mohamed Hibti, Basarab Matei, Nistor Grozavu, "Distance Estimation for Quantum Prototypes Based Clustering", in Lecture Notes in Computer Science, LNCS Springer, Proc of ICONIP'19 : 26th International Conference on Neural Information Processing, 12th-15th Dec 2019 - Sydney, Australia.
- Kaoutar Benlamine, Nistor Grozavu, Younès Bennani, Basarab Matei, "Collaborative Non-negative Matrix Factorization", IEEE - ICANN'19, 28th International Conference on Artificial Neural Networks, 17-19 September, 2019, Munich, Germany.
- Kaoutar Benlamine, Nistor Grozavu, Younès Bennani, Nicoleta Rogovschi, Ahmed Amamou, Kamel Haddadou, "Domain Name Recommendation Based on Neural Network", The 3rd INNS Conference on Big Data and Deep Learning (INNS BDDL), April 17-19, 2018, Sanur, Bali, Indonesia.

Bibliography

[AA02] M Andrecut and MK Ali. "A quantum neural network model". In: *International Journal of Modern Physics C* 13.01 (2002), pp. 75–88.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABG06] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. "Machine learning in a quantum world". In: *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer. 2006, pp. 431–442.

[ABG07] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. "Quantum clustering algorithms". In: *Proceedings of the 24th international conference on machine learning*. 2007, pp. 1–8.

[Alb83] Peter M Alberti. "A note on the transition probability over C*-algebras". In: *Letters in Mathematical Physics* 7.1 (1983), pp. 25–32.

[Ana19] Sashwat Anagolum. "Quantum machine learning: distance estimation for k-means clustering". In: *https://towardsdatascience.com/quantum-machine-learning-distance-estimation-for-k-means-clustering-26bccfbfcc76* (2019).

[Ang+03] Davide Anguita et al. "Quantum optimization for training support vector machines". In: *Neural Networks* 16.5-6 (2003), pp. 763–770.

[BDLP06] Shai Ben-David, Ulrike von Luxburg, and Dávid Pál. "A Sober Look at Clustering Stability". English. In: *Learning Theory*. Ed. by Gábor Lugosi and HansUlrich Simon. Vol. 4005. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 5–19. ISBN: 978-3-540-35294-5.

[Ben+18] Kaoutar Benlamine et al. "Domain Name Recommendation based on Neural Network". In: *Procedia computer science* 144 (2018), pp. 60–70.

[Ben+19a] Kaoutar Benlamine et al. "Collaborative Non-negative Matrix Factorization". In: *International Conference on Artificial Neural Networks*. Springer. 2019, pp. 655–666.

[Ben+19b] Kaoutar Benlamine et al. "Distance Estimation for Quantum Prototypes Based Clustering". In: *International Conference on Neural Information Processing*. Springer. 2019, pp. 561–572.

[Ben+20] Kaoutar Benlamine et al. "Quantum Collaborative K-means". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[BHEG01] Asa Ben-Hur, Andre Elisseeff, and Isabelle Guyon. "A stability based method for discovering structure in clustered data". In: *Biocomputing 2002*. World Scientific, 2001, pp. 6–17.

[Bis+10] Alessandro Bisio et al. "Optimal quantum learning of a unitary transformation". In: *Physical Review A* 81.3 (2010), p. 032324.

[Blo46] Felix Bloch. "Nuclear induction". In: *Physical review* 70.7-8 (1946), p. 460.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[Bri+09] Hans J Briegel et al. "Measurement-based quantum computation". In: *Nature Physics* 5.1 (2009), pp. 19–26.

[Cai+15] X-D Cai et al. "Entanglement-based machine learning on a quantum computer". In: *Physical review letters* 114.11 (2015), p. 110504.

[Cau89] Maureen Caudill. "Neural networks primer, part VIII". In: *AI Expert* 4.8 (1989), pp. 61–67.

[Cho+14] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[Chu+14] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[Cic+09] Andrzej Cichocki et al. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[Cle+98] Richard Cleve et al. "Quantum algorithms revisited". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339–354.

[Cor+18] Antoine Cornuejols et al. "Collaborative clustering: Why, when, what and how". In: *Information Fusion* 39 (2018), pp. 81–95.

[CP09] Andrzej Cichocki and Anh-Huy Phan. "Fast local algorithms for large scale nonnegative matrix and tensor factorizations". In: *IEICE transactions on fundamentals of electronics, communications and computer sciences* 92.3 (2009), pp. 708–721.

[Cyb89] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[DB79] David L Davies and Donald W Bouldin. "A cluster separation measure". In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), pp. 224–227.

[Dee+90] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), p. 391.

[DF02] Sandrine Dudoit and Jane Fridlyand. "A prediction-based resampling method for estimating the number of clusters in a dataset". In: *Genome biology* 3.7 (2002), research0036–1.

[DG17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[DHS05] Chris Ding, Xiaofeng He, and Horst D Simon. "On the equivalence of nonnegative matrix factorization and spectral clustering". In: *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM. 2005, pp. 606–610.

[Dir39] Paul Adrien Maurice Dirac. "A new notation for quantum mechanics". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 35. 3. Cambridge University Press. 1939, pp. 416–418.

[DJ92] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.

[DLJ08] Chris HQ Ding, Tao Li, and Michael I Jordan. "Convex and semi-nonnegative matrix factorizations". In: *IEEE transactions on pattern analysis and machine intelligence* 32.1 (2008), pp. 45–55.

[DS03] David Donoho and Victoria Stodden. "When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?" In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. NIPS'03. Whistler, British Columbia, Canada: MIT Press, 2003, 1141–1148.

[FF13] Moses Fayngold and Vadim Fayngold. *Quantum mechanics and quantum information: a guide through the quantum world*. John Wiley & Sons, 2013.

[FGW10] Germain Forestier, Pierre Gancarski, and Cédric Wemmert. "Collaborative clustering with background knowledge." In: *Data & Knowledge Engineering* 69.2 (Mar. 9, 2010), pp. 211–228.

[FWG07] G. Forestier, C. Wemmert, and P. Gancarski. "Collaborative Multi-Strategical Classification for Object-Oriented Image Analysis". In: *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications in conjunction with IbPRIA*. June 2007, pp. 80–90.

[GB10] Nistor Grozavu and Younès Bennani. "Topological Collaborative Clustering". In: *Australian Journal of Intelligent Information Processing Systems* 12.3 (2010).

[GECP13] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. "Swap test and Hong-Ou-Mandel effect are equivalent". In: *Physical Review A* 87.5 (2013), p. 052330.

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum random access memory". In: *Physical review letters* 100.16 (2008), p. 160501.

[GM09] Søren Gammelmark and Klaus Mølmer. "Quantum learning by measurement and feedback". In: *New Journal of Physics* 11.3 (2009), p. 033017.

[GM13] Søren Gammelmark and Klaus Mølmer. "Bayesian parameter inference from continuously monitored quantum systems". In: *Physical Review A* 87.3 (2013), p. 032115.

[Gro96] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[Gro97] Lov K Grover. "Quantum mechanics helps in searching for a needle in a haystack". In: *Physical review letters* 79.2 (1997), p. 325.

[Gro98] Lov K Grover. "A framework for fast quantum mechanical algorithms". In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 1998, pp. 53–62.

[GT09] Otfried Gühne and Géza Tóth. "Entanglement detection". In: *Physics Reports* 474.1-6 (2009), pp. 1–75.

[Gu10] Shi-Jian Gu. "Fidelity approach to quantum phase transitions". In: *International Journal of Modern Physics B* 24.23 (2010), pp. 4371–4458.

[GZ01] Sanjay Gupta and RKP Zia. "Quantum neural networks". In: *Journal of Computer and System Sciences* 63.3 (2001), pp. 355–383.

[Har54] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[Hea+98] Marti A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical review letters* 103.15 (2009), p. 150502.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[Joz94] Richard Jozsa. "Fidelity for mixed quantum states". In: *Journal of modern optics* 41.12 (1994), pp. 2315–2323.

[Kir+15] Ryan Kiros et al. "Skip-thought vectors". In: *Advances in neural information processing systems*. 2015, pp. 3294–3302.

[KM13] Trupti M Kodinariya and Prashant R Makwana. "Review on determining number of Cluster in K-Means Clustering". In: *International Journal* 1.6 (2013), pp. 90–95.

[KN98] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse Ising model". In: *Physical Review E* 58.5 (1998), p. 5355.

[KP08] Jingu Kim and Haesun Park. *Sparse nonnegative matrix factorization for clustering*. Tech. rep. Georgia Institute of Technology, 2008.

[KT51] H. W. Kuhn and A. W. Tucker. "Nonlinear programming". In: *Proceedings of 2nd Berkeley Symposium*. Ed. by Berkeley University of California Press. 1951, pp. 481–492.

[Lau+08] Hans Laurberg et al. "Theorems on Positive Data: On the Uniqueness of NMF". In: *Computational Intelligence and Neuroscience* 1 (2008).

[LLo82] S.P. LLoyd. "Least Squares Quantization in PCM". In: *Special Issue on Quantization, IEEE Tr. on Information Theory* 28.2 (1982), pp. 129–137.

[Llo82] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[LM14] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014, pp. 1188–1196.

[LMR13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum algorithms for supervised and unsupervised machine learning". In: *arXiv preprint arXiv:1307.0411* (2013).

[LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (2014), p. 631.

[LS01] Daniel D Lee and H Sebastian Seung. "Algorithms for non-negative matrix factorization". In: *Advances in neural information processing systems*. 2001, pp. 556–562.

[LS99] Daniel D Lee and H Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755 (1999), p. 788.

[Lux10] Ulrike von Luxburg. "Clustering Stability: An Overview". In: *Foundations and Trends in Machine Learning* 2.3 (Mar. 2010), pp. 235–274. ISSN: 1935-8237.

[Mac+67] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[Mey16] David Meyer. "How exactly does word2vec work?" In: (2016).

[Mik+13a] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[Mik+13b] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[Moo16] Christopher E Moody. "Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec". In: *arXiv preprint arXiv:1605.02019* (2016).

[MYZ13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." In: *hlt-Naacl*. Vol. 13. 2013, pp. 746–751.

[NC00] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. 2000.

[NC01] Michael A Nielsen and Isaac L Chuang. "Quantum computation and quantum information". In: *Phys. Today* 54.2 (2001), p. 60.

[Niu+15] Liqiang Niu et al. "Topic2Vec: learning distributed representations of topics". In: *Asian Language Processing (IALP), 2015 International Conference on*. IEEE. 2015, pp. 193–196.

[Ped02] Witold Pedrycz. "Collaborative fuzzy clustering". In: *Pattern Recognition Letters* 23.14 (2002), pp. 1675–1686.

[Ped05] Witold Pedrycz. *Knowledge-based clustering: from data to information granules*. John Wiley & Sons, 2005.

[PK97] Gopathy Purushothaman and Nicolaos B Karayiannis. "Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks". In: *IEEE Transactions on neural networks* 8.3 (1997), pp. 679–693.

[PL13] Kristen L Pudenz and Daniel A Lidar. "Quantum adiabatic machine learning". In: *Quantum information processing* 12.5 (2013), pp. 2027–2070.

[Plu02] M. Plumbley. "Conditions for nonnegative independent component analysis". In: *IEEE Signal Processing Letters* 9.6 (2002), pp. 177–180.

[PM11] Massimo Panella and Giuseppe Martinelli. "Neural networks with quantum architecture and quantum learning". In: *International Journal of Circuit Theory and Applications* 39.1 (2011), pp. 61–77.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation." In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[PT94a] Pentti Paatero and Unto Tapper. "Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values". In: *Environmetrics* 5.2 (1994), pp. 111–126.

[PT94b] Pentti Paatero and Unto Tapper. "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values". In: *Environmetrics* 5.2 (1994), pp. 111–126.

[Rab+06] Mikhail I Rabinovich et al. "Dynamical principles in neuroscience". In: *Reviews of modern physics* 78.4 (2006), p. 1213.

[Reb+16] Patrick Rebentrost et al. "Quantum gradient descent and Newton's method for constrained polynomial optimization". In: *arXiv preprint arXiv:1612.01789* (2016).

[Reb+19] Patrick Rebentrost et al. "Quantum gradient descent and Newton's method for constrained polynomial optimization". In: *New Journal of Physics* 21.7 (2019), p. 073023.

[RK87] Leonard KAUFMAN Peter J RDUSSEEUN and PJ KAUFMAN. "Clustering by means of medoids". In: (1987).

[RLN13] Nicoleta Rogovschi, Lazhar Labiod, and Mohamed Nadif. "A topographical nonnegative matrix factorization algorithm". In: *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*. 2013, pp. 1–6.

[RML13] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big feature and big data classification". In: *arXiv preprint arXiv:1307.0471* (2013).

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". In: *Physical review letters* 113.13 (2014), p. 130503.

[ROAG17] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. "Quantum autoencoders for efficient compression of quantum data". In: *Quantum Science and Technology* 2.4 (2017), p. 045001.

[Ron14] Xin Rong. "word2vec parameter learning explained". In: *arXiv preprint arXiv:1411.2738* (2014).

[Sch35] Erwin Schrödinger. "Discussion of probability relations between separated systems". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 31. 4. Cambridge University Press. 1935, pp. 555–563.

[Sch95] Benjamin Schumacher. "Quantum coding". In: *Physical Review A* 51.4 (1995), p. 2738.

[SG02] Alexander Strehl and Joydeep Ghosh. "Cluster ensembles—a knowledge reuse framework for combining multiple partitions". In: *Journal of machine learning research* 3.Dec (2002), pp. 583–617.

[Sha+06] Farial Shahnaz et al. "Document clustering using nonnegative matrix factorization". In: *Information Processing & Management* 42.2 (2006), pp. 373–386.

[Sho94] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[SL09] Denis Sych and Gerd Leuchs. "A complete basis of generalized Bell states". In: *New Journal of Physics* 11.1 (2009), p. 013006.

[SSP14] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "The quest for a quantum neural network". In: *Quantum Information Processing* 13.11 (2014), pp. 2567–2586.

[Tót+96] Géza Tóth et al. "Quantum cellular neural networks". In: *Superlattices and Microstructures* 20.4 (1996), pp. 473–478.

[Uhl76] Armin Uhlmann. "The "transition probability" in the state space of a-algebra". In: *Reports on Mathematical Physics* 9.2 (1976), pp. 273–279.

[VR11] Sandro Vega-Pons and José Ruiz-Shulcloper. "A Survey of Clustering Ensemble Algorithms". In: *IJPRAI* 25.3 (2011), pp. 337–372.

[Wan+17] Kwok Ho Wan et al. "Quantum generalisation of feedforward neural networks". In: *npj Quantum information* 3.1 (2017), pp. 1–8.

[WEG87] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[Wit14] Peter Wittek. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[WKS18] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. "Quantum nearest-neighbor algorithms for machine learning". In: *Quantum Information and Computation* 15 (2018).

[WZ82] William K Wootters and Wojciech H Zurek. "A single quantum cannot be cloned". In: *Nature* 299.5886 (1982), pp. 802–803.

[XHP99] Y.-L. Xie, P.K. Hopke, and P. Paatero. "Positive matrix factorization applied to a curve resolution problem". In: *Journal of Chemometrics* 12.6 (1999), pp. 357–364.

[YM08] N. Yanofsky and M. Mannucci. *Quantum computing for computer scientists*. Cambridge Press, 2008.