Université de Paris

Ecole doctorale 386 Université de Paris

Institut de Recherche en Informatique Fondamentale

Quantum algorithms for machine learning

Par: Alessandro Luongo

Thèse de doctorat en Informatique

Dirigée par Iordanis Kerenidis
et par Frédéric Magniez

Présentée et soutenue publiquement le 09/11/2020

Devant un jury composé de :

Simon PERDRIX	CR	Université de Lorraine	Rapporteur
Simone SEVERINI	PR	University College London	Rapporteur
Elham KASHEFI	CR	Université Pierre et Marie Curie	Examinatrice
	PR	University of Edinburgh	
Filippo MIATTO	CR	Télécom Paris	President
Iordanis KERENIDIS	DR	Université de Paris	Directeur de thèse
Frédéric MAGNIEZ	DR	Université de Paris	Co-directeur de thèse

Quantum algorithms for machine learning

Alessandro Luongo

November 18, 2020

Résumé

Cette thèse explore la possibilité d'exploiter l'ordinateur et les algorithmes quantiques dans le contexte de l'apprentissage automatique afin de faire de l'analyse de données plus rapidement. Nous savons depuis longtemps que l'ordinateur quantique peut offrir des avantages computationnels par rapport à l'ordinateur classique, car il permet un nouveau paradigme de calcul qui exploite les lois de la mécanique quantique pour offrir une accélération computationnelle.

Dans cette thèse sont proposés des algorithmes quantiques - plus rapides que leur meilleure alternative classique - pour estimer les paramètres des modèles d'apprentissage automatique. En plus d'un ordinateur quantique avec correction d'erreur, nous supposons avoir un accès quantique à la base de données. En d'autres termes, nous faisons l'hypothèse que les données sont stockées dans une mémoire quantique, la contrepartie quantique de la mémoire vive. Dans cette thèse, on étudie des algorithmes pour l'apprentissage supervisé, non supervisé, et pour la statistique. La caractéristique en commun de ces algorithmes est que leur temps d'exécution dépend de manière polylogarithmique du nombre d'éléments du jeu de données, et de manière linéaire du nombre de "features". Le temps d'exécution des algorithmes d'apprentissage automatique quantique présentés ici dépend de caractéristiques de la matrice qui représente les jeux de données étudiés, comme le rang, la norme de Frobenius, la sparsité, et l'erreur que l'on tolère dans l'analyse. On remarque que les meilleures algorithmes classiques sont souvent linéaires dans la dimension du jeu de données.

Pour l'apprentissage non supervisé, on étudie la version quantique de l'algorithme k-means (q-means), un modèle de clustering très connu, et sa généralisation: "modèle de mélange gaussien". Ce travail (QEM) peut être pensé comme la version quantique de l' "espérance-maximisation", un algorithme itératif d'importance considérable en apprentissage automatique classique. Dans le même contexte, on propose aussi des algorithmes quantiques pour estimer la matrice de covariance de données provenant de distributions non gaussiennes, résilients face à la présence de valeurs aberrantes, et un algorithme pour estimer le logarithme du déterminant d'une matrice symétrique définie positive.

Pour l'apprentissage supervisé, nous proposons des algorithmes pour la réduction de la dimensionnalité (Quantum Slow Feature Analysis). La réduction de la dimensionnalité est un processus utilisé sur des bases des données de haute dimensionnalité pour améliorer la précision d'un classificateur. Ici nous proposons aussi un algorithme quantique pour la classification qui est adapté pour être exécuté sur des ordinateurs quantiques avec peu de qubits. Nous proposons également d'autres modèles d'apprentissage automatique qui peuvent être formulés comme un problème à valeur propre généralisé - le même problème qui est sous-jacent à la SFA classique - comme l'Analyse Canonique des Corrélations (CCA), et le "Goulot Gaussien Informationnel".

Puisque chacun de ces algorithmes représente seulement un résultat théorique (i.e. sous forme de preuves de garanties sur le temps d'exécution et de bornes sur l'erreur d'approximation), on doit s'assurer que les performances réelles des algorithmes quantiques offrent des avantages concrets par rapport aux meilleurs algorithmes classiques. Puisque nous n'avons pas accès à des ordinateurs quantiques assez puissants pour exécuter ces algorithmes, nous avons évalué leurs performances par des simulations classiques. Dans ces expériences nous avons évité de construire et simuler des circuits quantiques: nous avons plutôt directement simulé les opérations d'algèbre linéaire bruitée, correspondant à l'exécution des algorithmes quantiques. Les simulations ont été effectuées sur des bases de données standards utilisées dans la communauté de l'apprentissage classique pour qualifier de nouveaux algorithmes et modèles (comme le MNIST). Dans les simulations, nous avons ajouté les mêmes erreurs que l'on s'attendrait à avoir si l'algorithme était exécuté par un ordinateur quantique. Grâce à cela, nous avons étudié la résilience d'une analyse de données aux erreurs de calcul insérées par les algorithmes quantiques, ce qui nous a permis

de comprendre quelles bases de données sont susceptibles d'être analysées efficacement par des algorithmes quantiques. Les expériences ont montré que l'impact du bruit n'affecte pas la qualité de l'analyse. De plus, l'impact des paramètres qui régissent la précision de calcul n'empêche pas des speedup sur les jeux de données massives.

Pour conclure, cette thèse donne l'espoir que l'ordinateur quantique avec accès quantique à une base de données permettra de nouvelles possibilités dans l'apprentissage automatique. Nous pensons que l'apprentissage automatique quantique pourra favoriser de nouvelles avancées technologiques, dès lors que des machine quantiques capables de supporter ces calculs seront prêtes.

Mots clés: Apprentissage automatique quantique, computation quantique, simulation quantique, analyse de données quantique.

Abstract

In this thesis we explore how we can leverage quantum computers and quantum algorithms in the context of machine learning, so as to process and analyze datasets and information faster. It has long been known that quantum computation can offer computational advantages with respect to classical computers.

In this thesis we propose various quantum algorithms that return an estimate of a machine learning model, that are faster than their best classical alternatives. Along with an error-corrected quantum computer, we assume to have quantum access to a dataset. In other words, we assume that the data is stored in a quantum memory: the corresponding quantum version of the classical random-access memory. We study quantum algorithms for supervised and unsupervised learning, dimensionality reduction, and statistics. The common characteristic of these algorithms is that the runtime depends only poly-logarithmically in the number of elements in the dataset, and is usually only linear in the number of features. The runtime of the quantum machine learning algorithm also often depends on characteristics of the matrix that represent the data under analysis, such as its rank, the Frobenius norm, the sparsity, the condition number, and the error we tolerate in the analysis. Note that in the vast majority of the cases, the best classical algorithms are at least linear in the dimension of the data.

For unsupervised learning, we study a quantum version of k-means (q-means), a classical clustering algorithm and its generalization: the Gaussian Mixture Models. This work can be thought of as the quantum version of Expectation-Maximization (QEM): an iterative algorithm of considerable importance in classical machine learning. We also study a quantum algorithm to estimate covariance matrices of data that comes from non-Gaussian distributions, that is resilient to the presence of outliers, and quantum algorithms to estimate the log-determinant of symmetric positive definite matrices.

For supervised learning, we put forward a quantum algorithm for supervised dimensionality reduction (Quantum Slow Feature Analysis). Dimensionality reduction is a preprocessing step that is used in high-dimensional datasets to increase the accuracy of a classifier. Here we propose also a quantum algorithm for classification that is particularly apt for quantum computers with not so many qubits (Quantum Frobenius Distance Classifier). Similarly to QSFA, we enlist some of the classical machine learning algorithms that can be reformulated as a generalized eigenvalue problem - the same computational problem that is underneath the classical Slow Feature Analysis algorithm-, like Canonical Correspondence Analysis, and the Gaussian Information Bottleneck.

Since each of these algorithms represents only a new theoretical result, (i.e we can provide guarantees on its runtime and bounds on the approximation error), we need to make sure that the real performances of the quantum algorithms offers concrete advantages with respect to the effective runtime and the accuracy that is offered by the best classical algorithms. As we don't have access to big-enough quantum computers yet, we assessed the performance of these quantum algorithms via a classical simulation. The experiments bypassed the construction of the quantum circuit and directly performed the noisy linear algebraic operations carried out by the quantum algorithm.

The simulations have been carried out on some datasets that are considered the standard benchmark of new machine learning algorithms, like the MNIST dataset, inserting the same kind of errors that we expect to have in the real execution on the quantum hardware.

In the experiments we studied the robustness of a data analysis to the noise introduced by the quantum algorithm, study the scaling of the runtime algorithm on real data, and thus understand which datasets can be analyzed efficiently by quantum computers. The experiments reveal that the impact of noise in the quantum algorithms does not decrease significantly the accuracy in the data analysis. Furthermore, the impact of the error parameters in the runtime does not prevent quantum speedups for large datasets.

To conclude, this thesis gives hope that quantum computers, with quantum access to

the data, can unlock new capabilities in machine learning. We believe that quantum machine learning can foster further technological advancement, as soon as the hardware that supports this kind of computation will be ready.

Keywords: Quantum machine learning; Quantum algorithms; Quantum Computation; Quantum simulations. Quantum data analysis.

Acknowledgement

Learning is a complex process that we have been failing to fully understand for several decades. One sure things about learning is the importance, for the learner, to have access to samples of the task or the behavior that the agent is supposed to learn. The people I have met during this journey have been my own personal "training set". I want to thank some of the colleagues and friends that have been a (mostly positive :)) example, in either the professional, or human point of view.

I would like to thank Iordanis Kerenidis and Frédéric Magniez for their precious time, insightful discussions, and for making me discover the beauty of algorithms.

I would like to thank Simone Severini and Simon Perdrix for kindly accepting to review this thesis. I am also grateful to Elham Kashefi, and Filippo Miatto for accepting taking part of the jury.

Cyril Allouche, Thomas Ayral, Alex B. Grilo, Alkida Balliu, Severin Bastard, Armando Bellante, Giovanni Bernardi, Timothée G. de Brugiére Amine El Charrat, Electra Eleftheriadou, Rebecca Erbanni, András Gilyén, Yassine Hamoudi, Elham Kashefi, Jane Keys, Isaac Konan, Jonas Landman, Baptiste Louf, Carlo Maragno, Bertrand Marchand, Simon Martiel, Yasmine Masotti, Simon Mauras, Mehdi Mhalla, Filippo Miatto, Valia Mitsou, Ashley Montanaro, Alexandre Nolin, Dennis Olivetti, Michele Orrú, Camilla Penzo, Simon Perdrix, Anupam Prakash, Andrea Rocchetto, Pablo Rotondo, Simone Severini, Changpeng Shao, Yixin Shen, Alex Singh, Sathyawageeswar Subramanian, Anupa Sunny, Daniel Szilagyi, Nunzio Turtullici, Zhouningxin Wang, and many other friends and colleague at Atos and its Quantum Lab, IRIF (specially AlgoComp), the PCQC, and the administrative staff of these organizations.

My hope is that I served you as an example as much as you have been an example for me.

Contents

Chapter 1

Introduction

> We are drowning in information and starving for knowledge
>
> John Naisbitt.

The amount of information humanity is producing is staggering. Eric Schmidt, executive chairman of Google, said in 2010 that as much data was being created every two days, as was created from the beginning of human civilization to the year 2003 [106]. This trend is not expected to cease any time soon: it's expected that in the near future, humans will keep producing an ever-increasing quantity of data. This data ranges among different types (images, text, times series, video, web-pages, structured and unstructured content), and size (from small to big data) with a marked tendency towards unsupervised datasets. To give some representative figures, in October 2001 the traffic volume that transitioned through AMS-IX (the Amsterdam-Internet eXchange: one of the most important nodes on the Internet) exceeded 1 PB for the first time, and it had already doubled by June 2002. 10 years later, October 2010, the traffic volume was 232 PB. As of March 2020 it is 1.90 EB, with an average throughput of more than 4Tbits per second.

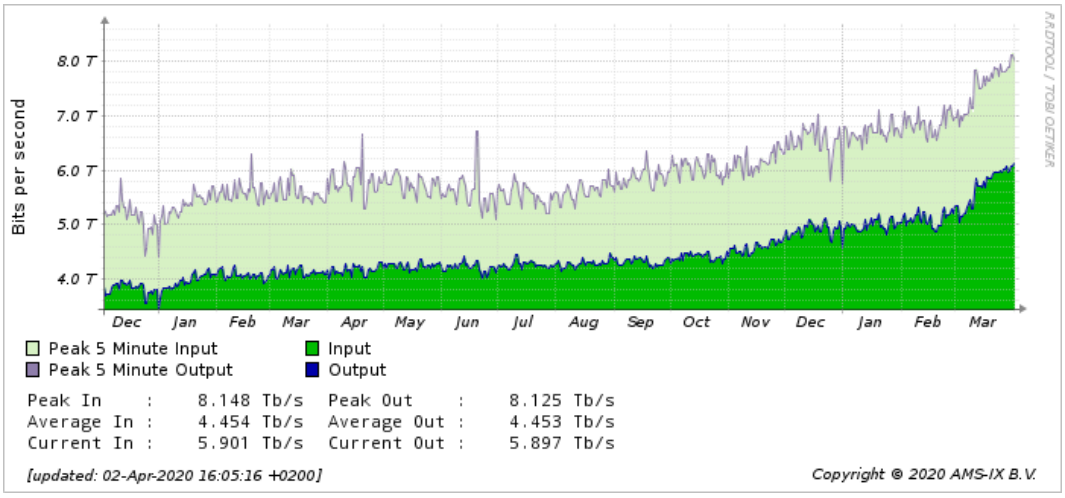


Figure 1.1: Total of yearly bits throughput at AMS-IX during 2019/2020

Each day, 500 million tweets, and 294 billion emails are sent. Portals like Facebook produce 4 petabytes of data daily. Each of the 50 million connected cars we have on the roads today produces 4 terabytes of information. The number of messages that are sent on WhatsApp touches 65 billion. On the biggest search engines of the world, 5 billion web

searches are made. Walmart, already in 2010, had to handle more than 1 million transactions per hour and had a database with 2.5×10^{15} bytes of information [50, 110]. It's estimated that by 2025, we will create daily around 450 exabytes of data. Some of these massive databases became public datasets for machine learning researchers. Among the many, we cite the 3DPEOPLE DATASET: it has around 2 Million images with 40 male and 40 female actors performing 70 actions[118]. For a list of large datasets, we refer to https://www.datasetlist.com/

Processing big datasets require adequate-size processing machines. While it is hard to estimate accurately, it would be safe to say that the available amount of computing power (measured in FLOPS or MIPS) per dollar has increased by a factor of 10 roughly every 4 years over (approximately) the last 25 years. Since the 1940s, the ratio of MIPS/dollar has grown by a factor of 10 roughly every 5 years, and FLOPS/dollar roughly every 7.7 years. Over the past 6-8 years, the rate has been slower: around an order of magnitude every 10-16 years [1]. There is daunting evidence that, after more than half a century, Moore's law is approaching its physical limitations. In fact, it is not possible to decrease the size of the transistors contained in our CPUs anymore. As of today, it is audacious to expect to build transistors much below 14nm (i.e 10^{-9} meter), which is about 70 silicon atoms wide. Below this limit it would be hard not to incur quantum effects that will inevitably result in errors in (classical) computations, thus spoiling the usefulness of modern CPUs. In Figure 1.2 we report the evolution of the density of transistors in CPUs since 1971. As we can see, in the past years, the trend has started to decelerate.

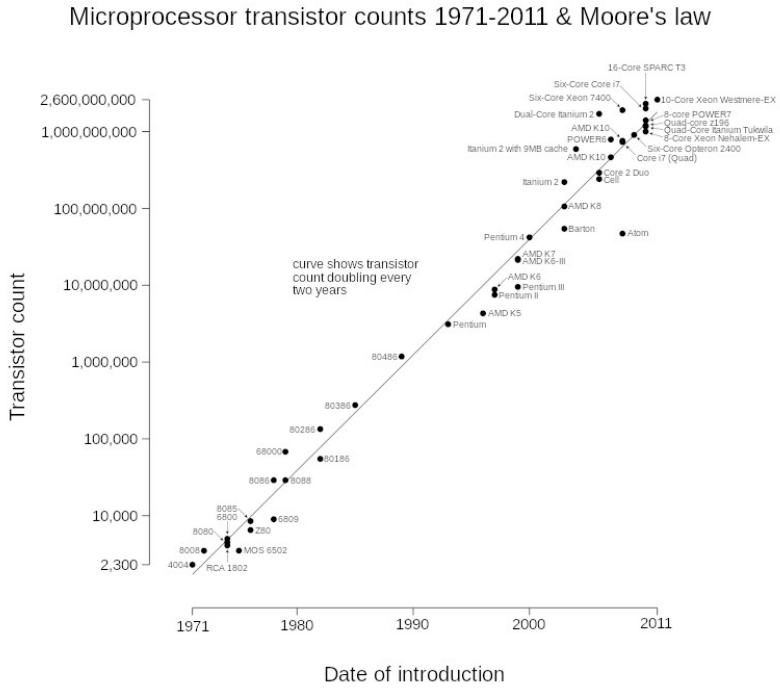


Figure 1.2: By Wgsimon - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=15193542

[1] This data has been measured in single-precision theoretical peak FLOPS. It has been analyzed in [51] and based on Passmark's benchmark scores. Passmark is a popular website that collects data and benchmarks the power of CPUs https://www.cpubenchmark.net/

Thanks to this overflow of information, combined with an increasing amount of computational power, the industry of Machine Learning (ML) has flourished. The ML communitystarted drifting away from the community of Artificial Intelligence, where it was born, acquiring more and more autonomy and space over time. In fact, the ability to learn from data or experiences is just one of the many aspects of human and animal intelligence and was only one among the many facets of the study of AI. Presumably because of the abundance of data and computational power, it has proven to be one of the most successful facets, let alone a very practical one. For reference, important conferences in machine learning, like NeurIPS (f.k.a. NIPS) started in 1987 and they now gather up to 13.000 people. Perhaps due to the effectiveness of ML in discovering structure and relations in data, our businesses, and society as a whole, started relying on these tools to make critical decisions. Nowadays, the ability to extract actionable information is necessary for almost all industries and organizations, even in the public sector and politics. Machine learning has applications in different industrial sectors, including automation, healthcare, Internet of Things, cybersecurity, biomedicine, etc.

The importance of faster ML algorithms can be seen in the following cases:

- Often there is intrinsic value in spending less time in running a computation. There are many cases where the gap between infeasible and feasible computation is between a runtime of weeks versus days. Faster algorithms might also decrease the time-to-market of certain goods. A case in point is the pharmaceutical sector: the time to market of new medicine strongly depends on the time needed to run some physical experiments which aim at testing the properties of molecules and drugs. So far, these experiments are necessary, as the size of the problem surpasses the scale of available HPC and cluster infrastructures. Thus, these experiments cannot be substituted by numerical simulations.

- Faster algorithms directly translate into the ability to process datasets that we currently have, but that we cannot process entirely, or that we process with sub-optimal algorithms and models.

- With faster algorithms, we can improve upon the metrics of interest in analyzing datasets with models that we currently use. In practice, this might translate into using machine learning models with more parameters. This would eventually improve the accuracy or other relevant metrics of choice.

In all these cases, having access to powerful algorithms holds the promise of unlocking commercial and societal value that cannot be underestimated. As the amount of valuable information that we are expected to analyze is increasing, and Moore's law is ending, we probably need to find a new paradigm for rethinking computation.

One decade ago, some important theoretical results - which are discussed in more detail in subsequent sections - opened the door to the possibility of using quantum computers in machine learning. This thesis studies how quantum computers, with quantum access to such data, can be leveraged to get faster-than-classical algorithms for analyzing big datasets and for training faster (or better) machine learning models.

Overall, thanks to quantum computers, business experts estimate gains by end-users (either in form of cost savings or revenue opportunities) to surpass 450 billion dollars annually. Nevertheless, these benefits are expected to be wrapped up in the next two decades. The value for end-users of quantum technologies in 2024 is expected to reach at most 5 billion dollars [96].

1.1 Computing with quantum information

At the beginning of the 20th century, most of physicists believed that the world could be understood in terms of the laws of what we now call classical physics (i.e. Newton's

laws and Maxwell's equations) [129, 18]. Besides few nagging questions, the progress that physicists expected were just minor refinements, to get "an extra decimal place" of accuracy in their formulas. One of these nagging questions was the theoretical explanation of a phenomenon, called "ultraviolet catastrophe". In fact, for a fixed temperature, classical physics explained the intensity of the radiation of a black body as a function of frequency. In contrast, from the experiments, it is possible to see that the spectrum of a electromagnetic radiation follow a characteristic curve. From the theory, the radiation predicted by classical physics was expected to diverge. It was Max Plank, who proposed a way to settle this glaring disagreement between theory and experiments by assuming that the energies associated with the oscillations of electrons must be proportional to integer multiples of their frequency (i.e. the energy comes in discrete quantities). What he once thought to be just a mathematical trick, lead him to obtain the Nobel price in 1918, and laid the foundations of quantum mechanics. The current formulation of quantum mechanics has been standardized only in the 1920s. De Broglie proposed in 1923 the famous wave-particle duality, and Schrödinger (1926) suggested to use partial differential equations to model the evolution of quantum systems like electrons [127].

A few decades later, the contamination between computer science and quantum physics was officially undertaken. In 1961, Landauer observed that the *erasure* of information requires energy, and in principle, any computation that is reversible can be performed without paying any energy cost. [21]. It was only during the 70s, that scientists like Bennett, Fredkin, and Toffoli, developed further the idea of reversible computation. They showed that it is possible to make any classical computation reversible, as long as no information gets lost in the process. For doing this, they proposed a set of reversible logic gates, which later became the set of quantum logic gates for quantum computers [63]. We had to wait until 1982 for a seminal paper that described ideas of using quantum mechanical systems to simulate other quantum physical systems [60]. Feynman conjectured that the evolution of a purely quantum mechanical system cannot be simulated efficiently by a classical computer. Thus, a quantum mechanical system is needed to simulate the behavior of another purely quantum mechanical one. Later on, in 1986, another seminal paper of Richard Feynman [59] formalized a prototypical architecture of a quantum computer (the Feynman quantum computer) [107].

To date, the investigation of how the effects of quantum mechanics can be exploited in computer science has taken many directions. In all these cases, the holy grail is to find a problem with exponential separation between the quantum and classical settings. For instance, we have quantum cryptography, quantum zero-knowledge proofs, quantum communication complexity [17, 9, 120], quantum complexity theory, space complexity in online algorithms [97], query complexity [52, 133], and many other exponential separation results. Within the gate model, it is probably much harder to find computational complexity separations. Except for the well-known result for factoring [131], finding examples of quantum algorithms with "pure" exponential separations is still an open research question. The class of decision problems that can be solved efficiently by a classical computer is called BPP (Bounded-error Probabilistic Polynomial time). Their quantum analog is called BQP (Bounded-error Quantum Polynomial time). While it is simple to show that BPP is contained inside BQP, we do not know if this relation is strict, nor do we fully understand the relation between BQP and NP. On one hand, many researchers do not believe that NP is contained in BQP, namely, we do not believe that quantum computers can solve NP-complete problems. On the other hand, in 2018, Raz and Tal proved the existence of an oracle relative to which BQP $\not\subseteq$ PH [121]. Proving the existence of a problem that lies outside BPP but is inside $BQP \cap NP$ would imply a separation between P and NP. Reaching such a result is non-trivial, thus we don't expect to be easy to find. Nevertheless, the main goal of quantum algorithmic research is to find interesting problems that are in BQP but are not believed to be in BPP.

1.1.1 Quantum Machine Learning

"Quantum machine learning is the most over-hyped and underestimated field in quantum computing"

John Preskill, citing Iordanis Kerenidis.

There are many possible ways to bring together quantum computation and machine learning. In the past ten years, we have witnessed two parallel trends [26]. On one side, researchers tried to leverage machine learning techniques to improve upon our understanding and control of quantum systems, in many different aspects. For instance, machine learning can be used to model the noise that causes decoherence in quantum devices or to improve control-systems that are used in order to direct the evolution of qubits [35]. On the other side - and this is what most people refer to when they use the name QML (Quantum Machine Learning) - it is the use of quantum computers to speed up the calculations of algorithms that are used to build machine learning models. This can either mean using quantum algorithms to estimate (i.e. fit) the parameters of a models or using quantum computers as machines to perform the inference process of a given model. In this case, it is not necessary to return a classical estimate of the model, but the output of the quantum computation will be just inference made by the model. To better contextualize this thesis, we briefly outline the landscape of the research in quantum machine learning. It is mainly divided into three approaches. First, fuelled by recent developments in adiabatic quantum computers, there is a flurry of research aimed at exploiting the adiabatic algorithm for machine learning. Within the context of the circuit model, two more paradigms emerged: the "algorithmic paradigm", and the "parameterized circuit" (also called variational circuits) paradigm. The research in variational circuits is devoted to understanding the capabilities of near-term devices (so-called NISQ, for Noisy Intermediate-Scale Quantum) for machine learning. There, a hybrid quantum-classical optimization procedure uses a classical optimization algorithm to find the angles of a set of parameterized quantum gates. The optimization process finds configurations of the gates such that the quantum processor behaves as desired. One example could be a variational circuit trained on a supervised dataset that learns how to classify correctly new test points. The other part of the research uses the quantum computer as a device to off-load heavy classical computations. This research, stemming from the theoretical computer science community, aims at obtaining machine learning algorithms that can show provable guarantees on the error and the runtime. Despite this attractive aspect, the drawback of theoretical quantum algorithms is that they generally rely on the assumption of being executed on a fault-tolerant (i.e. error corrected) quantum computer.

Adiabatic quantum machine learning Some researchers in QML are exploring the usage of the adiabatic model of computation in order to speed up the training of machine learning models. While the adiabatic paradigm is equivalent to the circuit model [4], in this thesis we will mostly be focusing on the circuit model of quantum computation, which is detailed in reference [112] and recapped in Section 2.1. In that Section, we will briefly report some of the previous major results of applications of quantum computing to machine learning within this framework. It is worth recalling that some of the techniques for quantum linear algebra used in this thesis have been developed originally in the circuit model, can also find efficient implementations in the adiabatic model [139]. We expect that the algorithms presented in this work can be (not without some technical difficulty) translated into algorithms for adiabatic quantum computers.

Variational circuits For sake of completeness, let's briefly discuss the variational approach. As we don't have error-corrected devices yet, this approach focuses on the study of the so-called variational circuits [57, 20]. This approach aims to show that even with small scale devices we can get useful applications for relevant problems. The "raison d'être" of the research in variational quantum circuits is to find useful applications of small quantum computers, where the decoherence and the number of qubits prevents the execution of fully-fledged quantum algorithms. While ML might not be the best target for proving useful applications for near term devices, there are some interesting results in this direction [58]. The idea of using a variational circuit for (supervised) machine learning is the following. In supervised learning the task is to approximate a function $f : \mathcal{X} \to \mathcal{Y}$ given samples from the space $(\mathcal{X} \times \mathcal{Y})$. For variational circuits, the idea is to layout some quantum circuits with parameterized gates. Then, the circuit is run a certain number of times using as initial state some vector x_i from the dataset, for which the output $y_i \in \mathcal{Y}$ is known. Then, the output at the end of the computation is collected. At each iteration, the parameters of the circuit are adjusted (according to a certain rule, which can be a simple gradient-based algorithm) in order to steer the output of the circuit towards the desired one.

Quantum algorithms for machine learning The algorithmic approach - and this is the topic of this thesis - is about writing *algorithms*, where you can prove that a given procedure has a certain probability of failure, you can quantify the error achieved in estimating a certain quantity, and estimate the asymptotic runtime of the computation. While the literature on quantum algorithms is vast, and it represents a useful resource and language for writing quantum machine learning algorithms, in the next chapter we review the main quantum results with particular emphasis on algorithms for ML. An important breakthrough, which unlocked many new possibilities in ML, is the so-called HHL algorithm [80]. This work put together previous results in Hamiltonian simulation, phase estimation, and state preparation, to deliver a quantum algorithm that creates a state that is proportional to the solution of a linear system of equations. Remarkably, the runtime of this algorithm was only poly-logarithmic with respect to the dimension of the linear system. This is in stark contrast with the plethora of classical algorithms we had at the time. It is important to stress that it is not easy to compare the HHL algorithm (or other quantum linear system solvers) with a classical linear system solver, as the output of the two algorithms is considerably different. In the classical case, the output of $A^{-1}b$ consists of the classical description of the vector, while in the quantum case we have only have access to a quantum state that is proportional to $A^{-1}b$. To gain classical knowledge of the quantum state, we need to perform a certain number of measurements on the final state. The number of measurements required to obtain classically the information on the final state depends on the error guarantees required, but it is usually linear in the dimension of the system. More discussion along these lines can be found in [124, 1].

We also stress the fact that the runtime of the quantum algorithm does not take into account the cost needed to set-up the quantum access to the matrix (or the initial known vector). It was recently discovered, with a bit of dismay of much of the quantum machine learning community, that if we were to allow also a classical algorithm some preprocessing time and access to a data structure similar to that described in Section 2, Definition 8, also classical algorithms can achieve an asymptotic scaling which is poly-logarithmic in the problem's size. In 2019 a paper by [144] used techniques from randomized linear algebra in order to propose a classical algorithm for recommendation systems, whose runtime is only poly-logarithmic in the input dimension. While these series of results ruled out possible exponential speedups based on this series of data structures, not all hope for having significant speedups with quantum algorithms for machine learning is lost. Indeed, these classical algorithms have much-worsened dependence on other parameters. Because of this, in practice, these algorithms are not expected to offer any computational advantage.

Some of these algorithms have been implemented, confirming that they are far from being practical [12]. Nevertheless, the theoretical importance of these series of results [145, 39, 69] should not go unnoticed. A nice review of randomized algorithms in linear algebra is [86]. These results highlight even more the limits and the potentials of quantum computation and helps to remove unnecessary and potentially dangerous hype around the subject.

In the wake of the HHL algorithm, and later on a quantum algorithm for recommendation systems, many quantum machine learning algorithms have followed [92, 99, 123, 122, 128, 19]. This thesis explores further how machine learning can benefit from the computational paradigm offered by quantum computers.

1.2 Contribution

Whether quantum processing machines can be used in order to solve efficiently and accurately real-world problems, and lead to benefits for industry and society, remains the billion-dollar question. Of course, in order to capitalize on the power of quantum mechanics, one would need powerful and robust quantum hardware, with quantum access to large datasets. Building such machines is rather hard and not straightforward at all. To justify the efforts of building quantum computers, it is crucial to find practical applications of quantum computing that would have a real economic and societal impact.

The research I conducted during my PhD was addressed towards providing evidence that large-scale quantum computers with quantum access to data will be indeed useful for running quantum algorithms in the context of machine learning. In order to provide evidence that quantum computers can offer advantages over classical computation, my work has been twofold. On one side, I developed new quantum algorithms in the context of machine learning. These algorithms are the quantum analogues of some classical machine learning algorithms: the output of the two computation is the same, up to some numerical error (which we show how to bound). Secondly, I provided evidence through experiments that these quantum algorithms can indeed be useful to solve real problems. We simulated classically the procedures that the quantum algorithm performs including the errors in these procedures. In the experiments, we also estimated the runtime of the quantum algorithms, and compare them with the runtime of the best classical algorithms. In this way, we were able to show that quantum algorithms can provide significant speedups for many cases of interest.

More precisely, we design efficient quantum algorithms for a number of machine learning procedures and applications: dimensionality reduction, iterative algorithms for learning mixture models, clustering, classification, algorithms for estimating the log-determinant (a kind of spectral sum), and for estimation of covariance matrix which are robust to outliers.

- In Chapter 4 we describe an algorithm for solving a dimensionality reduction problem in Machine Learning, namely the Slow Feature Analysis, and show its application to classification, proposing a new "NISQ" (Noisy Intermediate-Scale Quantum) classification algorithm. This work is based on: Kerenidis I. & Luongo A., Quantum classification of the MNIST dataset via slow feature analysis, Physical Review Letter A, 2020. Compared to the original pubblication, this work contains also an application of QSFA to the problem of malware detection: a problem faced by practitioners in cybersecurity.

- In Chapter 5 we describe a quantum algorithm for clustering, which fits the well-known classical k-means model. This results is based on the work: Kerenidis, I. Landman J., Luongo A., & Prakash A., q-means: A quantum algorithm for unsupervised machine learning, in Advances in Neural Information Processing Systems NeurIPS (pp. 4136-4146), 2019.

- In Chapter 6 we describe the quantum version of Expectation-Maximization, a fundamental ML algorithm that is often used to fit ML datasets with hidden variables. This result is based on the work: Kerenidis I., Luongo A., & Prakash A., Quantum Expectation-Maximization for Gaussian mixture models, in International Conference on Machine Learning ICML, 2020.

- In Chapter 7 we describe a quantum algorithm for estimating the log-determinant of symmetric positive-definite matrices, and show some of its applications. This result is based on the work: Luongo A., Shao C., Quantum algorithms for spectral sums and its applications (manuscript under preparation, 2020).

Chapter 2

Introduction to quantum computing

2.1 Preliminaries and notation

For all matrices $A \in \mathbb{R}^{n\times d}$, we recall the definition of singular value decomposition. The matrix A can be written as:

$$A = (U, U_0) \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} (V, V_0)^T.$$

The matrix Σ is a diagonal matrix with $\Sigma_{ii} = \sigma_i$ being the singular values (which we assume to be sorted $\sigma_1 \geq \cdots \geq \sigma_n$). The matrices (U, U_0) and (V, V_0) are orthogonal matrices, which contain a basis for the column and the row space (respectively U and V) and the left null-space and null-space (respectively U_0 and V_0). Oftentimes, it is simpler to define the SVD of a matrix by simply discarding the left and right null spaces, as $A = U\Sigma V^T$, where U, V are orthogonal matrices and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal matrix with real elements, where r is the rank of the matrix. It is well known, but important to remark, that the matrices $AA^T = U\Sigma^2 U^T$ and $A^T A = V\Sigma^2 V^T$ are symmetric. The dimension of the null-space is the number of linearly-dependent columns. For a rank k matrix, the Moore-Penrose pseudo-inverse is defined as $\sum_i^k \frac{1}{\sigma_i} u_i v_i^T$. Another relevant property of SVD is that the nonzero singular values and the corresponding singular vectors are the nonzero eigenvalues and eigenvectors of the matrix $\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$:

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} u_i \\ v_i \end{pmatrix} . = s_i \begin{pmatrix} u_i \\ v_i \end{pmatrix}$$

We will use the following matrix norms:

- $\|A\|_0$ as the number of non-zero elements of the matrix A,
- $\|A\|_1 = \max\limits_{1\leq j\leq n} \sum_{i=0}^n |a_{ij}|$ is the maximum among the sum of the absolute value along the columns of the matrix,
- $\|A\|_2 = \|A\| = \sigma_1$ is the biggest singular value of the matrix,
- $\|A\|_\infty = \max\limits_{1\leq i\leq m} \sum_{j=0}^n |a_{ij}|$ is the maximum among the sum of the absolute values along the rows of the matrix,
- $\|A\|_{\max}$ is the maximal element of the matrix in absolute value.

Note that for symmetric matrices, $\|A\|_\infty = \|A\|_1$. With s we denote the sparsity, that is, the maximum number of non-zero elements of the rows. The dataset that we manipulate in this work are represented by a matrix $V \in \mathbb{R}^{n\times d}$, i.e. each row can be thought as a vector $v_i \in \mathbb{R}^d$ for $i \in [n]$ that represents a single data point. We denote as V_k the optimal rank k approximation of V, that is $V_k = \sum_{i=0}^{k} \sigma_i u_i v_i^T$, where u_i, v_i are the row and column singular vectors respectively and the sum is over the largest k singular values σ_i. We denote as $V_{\geq\tau}$ the matrix $\sum_{i=0}^{\ell} \sigma_i u_i v_i^T$ where σ_ℓ is the smallest singular value which is greater than τ. For a matrix M and a vector x, we define as $M^+_{\leq\theta,\delta} M_{\leq\theta,\delta} x$ the projection of x onto the space spanned by the singular vectors of M whose corresponding singular values are smaller than θ, and some subset of singular vectors whose corresponding singular values are in the interval $[\theta, (1+\delta)\theta]$. We use a standard notation of $\tilde{O}$ for hiding polylogarithmic factors in the big-O notation of the algorithms.

In this manuscript, we will work with various probability distributions, most of which belong to the so-called exponential family, which we recall in the next definition.

Definition 1 (Exponential Family[110])**.** *A probability density function or probability mass function $p(v|\nu)$ for $v = (v_1, \cdots, v_m) \in \mathcal{V}^m$, where $\mathcal{V} \subseteq \mathbb{R}$, $\nu \in \mathbb{R}^p$ is said to be in the exponential family if it can be written as:*

$$p(v|\nu) := h(v) \exp\{o(\nu)^T T(v) - A(\nu)\}$$

where:

- $\nu \in \mathbb{R}^p$ *is called the* canonical or natural *parameter of the family,*
- $o(\nu)$ *is a function of ν (which often is just the identity function),*
- $T(v)$ *is the vector of sufficient statistics: a function that holds all the information the data v holds with respect to the unknown parameters,*
- $A(\nu)$ *is the cumulant generating function, or log-partition function, which acts as a normalization factor,*
- $h(v) > 0$ *is the* base measure *which is a non-informative prior and de-facto is scaling constant.*

Lemma 2 (Hoeffding inequality [83])**.** *Let $X_1, \ldots, X_k$ be independent random variables bounded by the interval $[a, b]$. Define the empirical mean of these variables by $\overline{X} = \frac{1}{k}(X_1 + \cdots + X_k)$, then*

$$Pr(|\overline{X} - \mathbb{E}[X]| \leq \epsilon) \geq 1 - 2\exp\left(-\frac{2k\epsilon^2}{b-a}\right). \tag{2.1}$$

Consequently, if $k \geq (b-a)\epsilon^{-2}\log(2/\eta)$, then $\overline{X}$ provides an ϵ-approximation of $\mathbb{E}[X]$ with probability at least $1-\eta$.

Lemma 3 (Polynomial approximations of $\ln(x)$ [68])**.** *Let $\beta \in (0, 1]$, $\eta \in (0, \frac{1}{2}]$ and $t \geq 1$. There exists a polynomial $\tilde{S}$ such that $\forall x \in [\beta, 1]$, $|\tilde{S}(x) - \frac{\ln(1/x)}{2\ln(2/\beta)}| \leq \eta$, and $\forall x \in [-1, 1]$: $-1 \leq \tilde{S}(x) = \tilde{S}(-x) \leq 1$. Moreover $\deg(\tilde{S}) = O(\frac{1}{\beta}\log(\frac{1}{\eta}))$.*

2.2 Quantum Information

The standard formalism used in Quantum Information is the Dirac's "bra-ket" notation, which we will introduce in this section. We also recall here the postulates of quantum mechanics and take this opportunity to settle the rest of the notation and preliminaries used in this thesis. For the postulates, we follow the standard formulation in [112].

Postulate 1. *Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system. The system is completely described by its state vector, which is a unit vector in the system's state space.*

As quantum states are described by unit vectors, we write $|\psi\rangle$ for a unit vector $\psi \in \mathcal{H}^n$. So for a non-normalized vector $x \in \mathbb{R}^n$, the normalized quantum state is represented as $|x\rangle = \|x\|^{-1} x = \frac{1}{\|x\|} \sum_{i=0}^{n} x_i |i\rangle$. We denote as $\{|i\rangle\}_{i\in[d]}$ the canonical (also called computational) basis for the d dimensional Hilbert space $\mathcal{H}$. The transpose-conjugate of $|x\rangle$ is defined as $\langle x|$. We can think of $|x\rangle$ as a column vector, while $\langle x|$ is a row vector, whose entries have been conjugated. In Dirac's notation, we denote the inner product between two vector as $\langle x|y\rangle$. Their outer product is denoted as $|x\rangle\langle y| = \sum_{i,j\in[d]} x_i y_j |i\rangle\langle j| \in \mathcal{H}^d \otimes \mathcal{H}^d$. The smallest quantum system is called a qubit, and is a 2 dimensional unit vector in $\mathbb{C}^2$. A base for this vector space in quantum notation is denoted as $|0\rangle$ and $|1\rangle$. In this case, the vector $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ for $\alpha, \beta \in \mathbb{C}$ represent a valid quantum state as long as $|\alpha|^2 + |\beta|^2 = 1$.

Postulate 2. *The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2.*

A matrix $U \in \mathbb{C}^{d\times d}$ is said to be unitary if $UU^\dagger = U^\dagger U = I$, that is, if the inverse of U equal to its conjugate transpose. From this fact it follows that unitary matrices are norm-preserving, and thus can be used as suitable mathematical description of a pure quantum evolution. It is a standard exercise to see that the following are all equivalent definition of unitary matrices [49]:

- $\langle Av, Aw\rangle = \langle v, w\rangle$ for all v, w.
- $\|Av\| = \|v\|$ for all v
- $\|Av\| = 1$ if $\|v\| = 1$.
- U is a normal matrix with eigenvalues lying on the unit circle
- $|\det(U)| = 1$
- The columns and the rows of U form an orthonormal basis of $\mathcal{C}^d$
- U can be written as e^{iH} for some Hermitian operator H.

Postulate 3. *Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, then the probability that the result m occurs is given by*

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{2.2}$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}} \tag{2.3}$$

The measurement operators satisfy the completeness equation

$$\sum_m M_m^\dagger M_m = I \tag{2.4}$$

In practice, we will mostly perform projective measurements (also called von Neumann measurements). A projective measurement is described by an *observable*: an Hermitian operator M on the state space of the system being observed. The observable has a spectral decomposition:

$$M = \sum_m mP_m$$

Where P_m is a projector into the eigenspace of M associated with the eigenvalue m. This means that the measurement operator will satisfy the following properties

- P_m is positive definite
- P_m is Hermitian
- $\sum_m P_m = I$
- $(P_m)(P_n) = \delta_{mn}(P_m)$ are orthogonal projections.

Recall that an orthogonal projector P has the properties that $P = P^\dagger$ and $P^2 = P$. Note that the second property derives from the first: all positive definite operators on $\mathbb{C}$ are Hermitian (this is not always the case for positive definite operators on $\mathbb{R}$, as it is simple to find positive definite matrices that are not symmetric). Projective measurements can be understood as a special case of Postulate 3: in addition to satisfying the completeness relation $\sum_m M_m^\dagger M_m = I$ they also are orthogonal projectors. Given a state $|\psi\rangle$, the probability of measuring outcome m is given by:

$$p(m) = \langle\psi| P_m |\psi\rangle .$$

If we were to measure outcome m, then the state of the quantum system after the measurement would be:

$$\frac{P_m |\psi\rangle}{\sqrt{p(m)}}.$$

They have some useful properties. Just to cite one, the average value of a projective measurement in a state $|\psi\rangle$ is define as:

$$E(M) = \sum_m p(m) \tag{2.5}$$

$$= \sum_m m \langle\psi| P_m |\psi\rangle \tag{2.6}$$

$$\langle\psi| (\sum_m mP_m) |\psi\rangle \tag{2.7}$$

$$\langle\psi| M |\psi\rangle \tag{2.8}$$

In practice, our projective operators will be projectors in the computational basis, i.e. $P_m = \sum_{m\in[d]} |m\rangle\langle m|$. From these rules, it is simple to see that the probability that a measurement on a state $|x\rangle = \frac{1}{\|x\|}\sum_i x_i |i\rangle$ gives outcome i is $|x_i|^2 / \|x\|^2$.

Postulate 4. *The state-space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered from 1 through n, and each state is described as $|\psi_i\rangle$, the join state of the total system is $\bigotimes_{j=1}^n |\psi_i\rangle = |\psi_1\rangle |\psi_2\rangle \dots |\psi_n\rangle$.*

A quantum state that cannot be expressed as tensor product of two quantum state is said to be entangled. The mathematical description of two quantum system is achieved using a tensor product. The tensor product between two vectors $|y\rangle \in \mathbb{R}^{d_1}$ and $|y\rangle \in R^{d_2}$ is a vector $|z\rangle \in \mathbb{R}^{d_1 \times d_2}$. We can use the tensor operation to describe the joint evolution

of separate quantum system. Let U_1 be the evolution of a quantum state $|x\rangle$ and U_2 the evolution of a quantum state $|y\rangle$, $U_1 \otimes U_2$ describe the evolution of the quantum system $|x\rangle |y\rangle$. Note that to build a state in $|v\rangle \in \mathcal{H}^n$ we need $\lceil \log n \rceil$ qubits.

There are various models to measure the complexity of a quantum algorithm. We denote with $T(U)$ the time complexity needed to implement U, measured in terms of depth of the circuit, which is roughly the number of time steps (or "clock time") we need, to executed the gates of the quantum circuit U.

2.3 Loading the data in quantum computer: quantum memory models

Along with a fully fledged quantum computer, we also assume to have access to a quantum memory, i.e. a classical data structure that stores classical information, but that can answer queries in quantum superposition. This model is commonly called the *QRAM model*. As we will see in greater detail soon, the task of building the data structure classically requires time that is linear (up to polylogarithmic factors) in the dimension of the data. This observation is better detailed in definition 4. For instance, if we want to have quantum access to a dense matrix $M \in \mathbb{R}^{n \times d}$ the preprocessing runtime will be $O(nd \log(nd))$. To stress more the fact that we are linear in the effective number of elements contained in the matrix (which can often be sparse) can write $\tilde{O}(\|A\|_0)$. The name QRAM is meant to evoke the way classical RAM addresses the data in memory using a tree structure. In the data structure, one can write down the real entries m_{ij} with some precision δ using $\log 1/\delta$ bits.

Note that sometimes, QRAM goes under the name of QROM, to stress the fact that the QRAM is something that can be written during the runtime of the quantum algorithm, but just queried, i.e. read. Furthermore, a QRAM is said to be *efficient* if can be updated by adding, deleting, or modifying an entry in polylogarithmic time w.r.t the size of the data it is storing. Using the following definition 4, we can better define the computational model we are working with.

Definition 4 (QRAM model [91])**.** *An algorithm in the QRAM data structure model that processes a data-set of size m has two steps:*

1. *A pre-processing step with complexity $\widetilde{O}(m)$ that constructs efficient QRAM data structures for storing the data.*

2. *A computational step where the quantum algorithm has access to the QRAM data structures constructed in step 1.*

The complexity of the algorithm in this model is measured by the cost for step 2.

Equipped with this data structure, we are allowed to perform the following operation in a quantum computer.

Definition 5. *[QRAM - Quantum Random Access Memory], [71, 91] A quantum random access memory is a device that stores indexed data (i, x_i) for $i \in [n]$ and $x_i \in \mathbb{R}$ (eventually truncated with some bits of precision). It allows query in the form $|i\rangle |0\rangle \mapsto |i\rangle |x_i\rangle$, and has circuit depth $O(polylog(n))$.*

We say that a dataset is efficiently loaded in the QRAM, if the size of the data structure is linear in the dimension and number of data points and the time to enter/update/delete an element is polylogarithmic in the dimension and number of data points. A direct application of this model is what is commonly assumed in quantum algorithms to access

sparse matrices. This is often useful when dealing with graphs, or simply with sparse Hamiltonians.

Definition 6 (Oracle access in adjacency matrix model)**.** *Let $V \in \mathbb{R}^{n\times d}$, there is an oracle that allows to perform the following mapping $|j,k,z\rangle \mapsto |j,k,z \oplus V_{jk}\rangle$*

Definition 7 (Oracle access in adjacency list model)**.** *Let $V \in \mathbb{R}^{n\times d}$, there is an oracle that allows to perform the: mapping $|i\rangle \mapsto |i\rangle|d(i)\rangle$ where $d(i)$ is the number of entries in row i, and $|j,l\rangle \mapsto |j,\nu(j,l)\rangle$, where $\nu(j,l)$ is the l-th nonzero entry of the j-th column of V.*

In [117], they leveraged definition 5 to allow us to efficiently create superpositions corresponding to the rows of the matrices, i.e. encoding the values of the components of a matrix' row in the amplitudes of a quantum state. Note that this data structure, which sometimes goes under the name KP-trees [122], assumes and extends definition 5. In practice, both are called QRAM, and both rely on two (different) tree data structure for its construction.

Theorem 8 (Quantum access for matrices [92])**.** *Let $V \in \mathbb{R}^{n\times d}$, there is a data structure to store the rows of V such that,*

1. *The time to insert, update or delete a single entry v_{ij} is $O(\log^2(N))$.*
2. *A quantum algorithm with access to the data structure can perform the following unitaries in time $T = O(\log^2 N)$.*
   (a) $|i\rangle|0\rangle \to |i\rangle|v_i\rangle$ *for* $i \in [n]$.
   (b) $|0\rangle \to \sum_{i\in[N]} \|v_i\||i\rangle$.

We report what our QRAM data structure looks like for an input matrix X according to the original definitions in [117, 92]. Each row of the matrix of the dataset is encoded as a tree, where the leaves correspond to the squared values of the matrix elements (along with their sign), while the intermediate nodes store the sum of the sub-tree rooted in each node. Then, in the quantum algorithm, we assume to have quantum access to all the nodes of the tree. In ML is common to pre-process the data, with either a polynomial expansion, normalization, or scaling of the components, for instance in a way such that each feature has unit variance and zero mean. These model suits perfectly these needs, as these operations can be done before storing the data in the quantum accessible data structure.

Recently, the data structure has been extended to allow space for some improvements in the algorithms. In fact, let $A/\mu = P \circ Q$ a decomposition of the matrix A, where the norm of the rows of P and the columns of Q are at most 1, and the symbol $\circ$ is the Hadamard product. In the original formulation, the factorization chosen corresponded to a choice of $\mu = \|A\|_F$. If there is a quantum circuit allowing creation of quantum states proportional to the rows and the columns of P and Q, the runtime of the quantum algorithms based on this improved QRAM data structure can become a function of μ, which can be smaller than the Frobenius norm of A. In [91], they provided such efficient factorization for various choices of μ. In the following we explicitly define a class of functions μ, parameterized by $p \in [0,1]$, that will prove to be very useful in governing the runtime of the quantum algorithms.

Definition 9 (Possible choice of $\mu_p(A)$)**.** *For $s_p(A) = \max_{i\in[n]} \sum_{j\in[d]} A_{ij}^p$, we chose $\mu_p(A)$ to be: $\mu_p(A) = \min_{p\in[0,1]}(\|A\|_F, \sqrt{s_{2p}(A)s_{(1-2p)}(A^T)})$,*

The original definition of QRAM, where $\mu(A) = \|A\|_F$ corresponds to the factorization $A/\|A\|_F = P \circ Q$ where we have $p_{ij} = \frac{a_{ij}}{\|a_i\|}$ and $q_{ij} = \frac{\|a_i\|}{\|A\|_F}$. For the generalized choice

of μ in definition 9, it is necessary to store two quantum accessible data structures, that respectively store the rows and the columns of a function of A. Instead of storing a_{ij} (along with the sign, which is stored separately), we store $sgn(a_{ij})a_{ij}^p$ and a_{ij}^{1-p}. The different terms in the minimum in the definition of $\mu(A)$ correspond to different choices for the data structure for storing A. Note that in the worst case, $\mu(A) \leq \|A\|_F \leq \sqrt{d}$ as we assume that $\|A\| = 1$. Having a small value for $\mu(A)$ is very important, as this value will appear in the runtime of the quantum algorithms. In this thesis we always assume to have quantum access to matrices which are normalized such that $\|A\| \leq 1$.

For details on how to use quantum access to this data structure and proving theorem 8, the reader is referred to [92, Appendix] for the original proof, [91, theorem 4.4] for details on the choices of $\mu(A)$. More explicit proofs for the creation of quantum states with choices of μ different than the Frobenius norm can be found in [37, lemma 25] and [70].

To grasp the importance of this model we discuss the hurdles and bottleneck of doing data analysis on massive datasets. When the data that needs to be processed surpass the size of the available memory, the dataset can only be analyzed with algorithms whose runtime is linear (or at most quadratic) with respect to the size of the dataset. Super-linear computations (like most algorithms based on linear-algebra) are too computationally expensive, as the size of the data is too big to fit in live memory.

Under these conditions, quantum computers can offer significant advantages. The runtime of the whole process of performing data analysis using quantum computers is given by the time of the preprocessing and constructing quantum access, plus the runtime of the quantum procedure. In practice, we want to write algorithms with a total computational complexity of $\tilde{O}(\|A\|_0) + \tilde{O}(\text{poly}(\kappa(A), \mu(A), 1/\epsilon, \log(n), \Gamma))$, where $\kappa(A)$, is the condition number of the matrix, ϵ the error in the approximation of the calculations, and Γ might represent some other parameters that depends on the data, but not on its size. This represents an improvement compared to the runtime of the best classical algorithms in machine learning, which is $O(\text{poly}(\|A\|_0) \times \text{poly}(\kappa(A), 1/\epsilon))$. Note that without resorting to randomized linear algebra (as in the case of the dequantizations), these runtimes are lower bounded by the runtime for matrix multiplication and matrix inversion. As the QRAM preparation is computationally easy to implement, (it requires a single or few passes over the dataset, that we can do when we receive it, and it is can be made in parallel) a quantum data analysis can be considerably faster than the classical one. It is clear that, even if the scaling of the quantum algorithm is sub-linear in the data (it is often, in fact, polylogarithmic in n), if we consider in the runtime of the algorithms also the time to build quantum access we "lose" the exponential gap between the classical and the quantum runtime. Nevertheless, the overall computational cost can still largely favor the quantum procedure, as we expect the final runtime of the quantum algorithm to be comparable to the preprocessing time. Moreover, the preprocessing can be made once, when the data is received. For the choice of the data structure that leads to a value of μ equal to the Frobenius norm of the matrix under consideration, this can be done even on the fly, i.e. while receiving each of the rows of the matrix. For the choice of μ related to a p norm, the construction of the data structure needs only a few passes over the dataset.

2.4 Retrieving the data

To retrieve information from a quantum computer, we are going to use some efficient procedures that allow us to reconstruct classically the information stored in a quantum state. These procedures can be thought of as clever ways of sampling a state $|x\rangle$. The idea for an efficient quantum tomography is that we want to minimize the number of times that the sate $|x\rangle$ is created, and consequently, the number of times we call the process that creates $|x\rangle$.

Much of the current literature in quantum tomography is directed towards reconstructing a classical description of density matrices.

Theorem 10 (Efficient quantum tomography [113]). *An unknown rank-r mixed state $\rho \in \mathbb{C}^{d\times d}$ can be estimated to error ϵ in Frobenius distance using $n = O(d/\epsilon^2)$ copies, or to error ϵ in trace distance using $n = O(rd/\epsilon^2)$ copies.*

The quantum algorithms described in this thesis usually work with pure quantum states. Moreover we assume to have access to the unitary that creates the quantum state that we would like to retrieve and that we have access to the unitary that creates the state (and that we can control it). Under these conditions, the process of performing tomography is greatly simplified. According to the different error guarantees that we require, we can choose between two procedures.

Theorem 11 (Vector state tomography [90] with ℓ_2 guarantees). *Given access to unitary U such that $U|0\rangle = |x\rangle$ and its controlled version in time $T(U)$, there is a tomography algorithm with time complexity $O(T(U)\frac{d\log d}{\epsilon^2})$ that produces unit vector $\widetilde{x} \in \mathbb{R}^d$ such that $\|\widetilde{x} - |x\rangle\|_2 \leq \epsilon$ with probability at least $(1 - 1/poly(d))$.*

Theorem 12 (Vector state tomography [88] with ℓ_∞ guarantees). *Given access to unitary U such that $U|0\rangle = |x\rangle$ and its controlled version in time $T(U)$, there is a tomography algorithm with time complexity $O(T(U)\frac{\log d}{\epsilon^2})$ that produces unit vector $\widetilde{x} \in \mathbb{R}^d$ such that $\|\widetilde{x} - |x\rangle\|_{\ell_\infty} \leq \epsilon$ with probability at least $(1 - 1/poly(d))$.*

Note that in both kinds of tomography, dependence on the error in the denominator is quadratic, and this is because of the Hoeffding inequality, i.e. lemma 2. Another remark on the hypothesis of the algorithms for tomography is that they require a unitary U such that $U|0\rangle \mapsto |x\rangle$ for the $|x\rangle$ in question. Oftentimes, due to the random error in the quantum subroutines used inside the algorithms, this state $|x\rangle$ might slightly change every time. We will see that in this thesis the variance on the state $|x\rangle$ is due to errors in mostly due to the phase estimation procedures. In Section 2.5.1 we discuss ways of making phase estimation algorithms almost deterministic.

More advanced techniques have been recently developed in [157]. There, the authors used the assumption of doing tomography on a state $|x\rangle$ that is in the row space of a rank r matrix A for which we have quantum access. They propose an algorithm to obtain the classical description of the coefficients x_i in the base spanned by the rows $\{A_i\}_{i=0}^r$of A, so that $|x\rangle = \sum_i^r x_i |A_i\rangle$. This requires $\tilde{O}(\text{poly}(r))$ copies of the output states and $\tilde{O}(\text{poly}(r), \kappa^r)$ queries to input oracles. While this procedure has the benefit of not being linear in the output dimension of the final state, the high dependence on the rank might hide the advantages compared to the previous quantum tomography procedures. For completeness, the result is as follows.

Theorem 13 ([157]). *For the state $|v\rangle$ lies in the row space of a matrix $A \in \mathbb{R}^{n\times d}$ with rank r and condition number $\kappa(A)$, the classical form of $|v\rangle$ can be obtained by using $O(r^3\epsilon^2)$ queries to the state $|v\rangle$, $O(r^{11}\kappa^{5r}\epsilon^{-2}\log(1/\delta))$ queries to QRAM oracles of A and $O(r^2)$ additional inner product operations between rows, such that the ℓ_2 norm error is bounded in ϵ with probability at least $1 - \delta$.*

Other strategies to output data from a quantum computer can be sampling, or (if the output of the computation is just a scalar) amplitude estimation. These techniques can be employed when the output of the computation is a scalar.

2.5 Additional quantum results

We will make use of a tool developed in [151]. It is a standard technique in classical computer science to boost the success probability of a randomized algorithm by repeating

it and computing some statistics in the results. For the case of quantum algorithms, by and large, we take multiple copies of the output of the amplitude estimation procedure, compute the median, and reverse the circuit to get rid of the garbage.

Lemma 14 (Median Evaluation [151])**.** *Let $\mathcal{U}$ be a unitary operation that maps*

$$\mathcal{U} : |0^{\otimes n}\rangle \mapsto \sqrt{a}\,|x,1\rangle + \sqrt{1-a}\,|G,0\rangle$$

for some $1/2 < a \leq 1$ in time T. Then there exists a quantum algorithm that, for any $\Delta > 0$ and for any $1/2 < a_0 \leq a$, produces a state $|\Psi\rangle$ such that $\| \, |\Psi\rangle - |0^{\otimes nL}\rangle \, |x\rangle \, \| \leq \sqrt{2\Delta}$ for some integer L, in time

$$2T \left\lceil \frac{\ln(1/\Delta)}{2\left(|a_0| - \frac{1}{2}\right)^2} \right\rceil .$$

We will also use some simple statements from previous literature.

Claim 15. *[From [93]] Let ϵ_b be the error we commit in estimating $|c\rangle$ such that $\||c\rangle - |\overline{c}\rangle\| < \epsilon_b$, and ϵ_a the error we commit in the estimating the norms, $|\,\|c\| - \overline{\|c\|}\,| \leq \epsilon_a \|c\|$. Then $\|\overline{c} - c\| \leq \sqrt{\eta}(\epsilon_a + \epsilon_b)$.*

Claim 16. *[From [91]] Let θ be the angle between vectors x, y, and assume that $\theta < \pi/2$. Then, $\|x - y\| \leq \epsilon$ implies $\||x\rangle - |y\rangle\| \leq \frac{\sqrt{2}\epsilon}{\|x\|}$. Where $|x\rangle$ and $|y\rangle$ are two unit vectors in ℓ_2 norm.*

2.5.1 Phase estimation

Theorem 17. *Phase estimation [94] Let U be a unitary operator, with eigenvectors $|v_j\rangle$ and eigenvalues $e^{\iota\theta_j}$ for $\theta_j \in [-\pi, \pi]$, i.e. we have $U\,|v_j\rangle = e^{\iota\theta_j}\,|v_j\rangle$ for $j \in [n]$. For a precision parameter $\epsilon > 0$, there exists a quantum algorithm that runs in time $O(T(U)logn/\varepsilon)$ and with probability $1-1/poly(n)$ maps a state $|\phi_i\rangle = \sum_{j\in[n]} \alpha_j\,|v_j\rangle$ to the state $\sum_{j\in[n]} \alpha_j\,|v_j\rangle\,|\bar{\theta}_j\rangle$ such that $\bar{\theta}_j \in \theta_j \pm \varepsilon$ for all $j \in [n]$.*

While the standard implementation of phase estimation is based on the quantum Fourier transform (QFT) circuit [112], there have been various improvements [5] which try to soften the dependence on the QFT circuit while retaining the accuracy guarantees offered by the QFT in estimating the angles θ_j.

Remark. Note that the same algorithm described by theorem 17 can be made "consistent", as in the sense of [130]. While in the original formulation of phase estimation two different runs might return different estimates for $\bar{\theta}_j$, with a consistent phase estimation this estimate is fixed, with high probability. It means that the error, between two different runs of phase estimation, is almost deterministic.

2.5.2 Amplitude amplification and amplitude estimation

Amplitude amplification and amplitude estimation are two of the workhorses of quantum algorithms. In this section, we report both the original statement of the theorem, and a simpler version, which is better suited for the context of our algorithms.

Theorem 18 (Amplitude estimation [34])**.** *Given a quantum algorithm*

$$A : |0\rangle \to \sqrt{p}\,|y,1\rangle + \sqrt{1-p}\,|G,0\rangle$$

where $|G\rangle$ is some garbage state, then for any positive integer P, the amplitude estimation algorithm outputs $\tilde{p}$ $(0 \leq \tilde{p} \leq 1)$ such that

$$|\tilde{p} - p| \leq 2\pi \frac{\sqrt{p(1-p)}}{P} + \left(\frac{\pi}{P}\right)^2$$

with probability at least $8/\pi^2$. It uses exactly P iterations of the algorithm A. If $p = 0$ then $\tilde{p} = 0$ with certainty, and if $p = 1$ and P is even, then $\tilde{p} = 1$ with certainty.

In the original amplitude amplification algorithm, we assume to know P, i.e. the correct number of iterations. Later on, a fixed-point version of amplitude amplification has been proposed [74], which has been optimized in [155]. These versions do not require to know P in advance. Recently, various researchers worked on improvements of amplitude estimation by getting rid of the part of the original algorithm that performed the phase estimation (i.e. the Quantum Fourier Transform [112]) [73, 2]. Perhaps a simpler formulation, which hides the complexity of the low-level implementation of the algorithm, and is thus more suitable to be used in quantum algorithms for machine learning is the following.

Lemma 19. *(Amplitude amplification and estimation [34]) If there is unitary operator U such that $U|0\rangle^l = |\phi\rangle = \sin(\theta)|x, 0\rangle + \cos(\theta)|G, 0^\perp\rangle$ then $\sin^2(\theta)$ can be estimated to multiplicative error η in time $O(\frac{T(U)}{\eta \sin(\theta)})$ and $|x\rangle$ can be generated in expected time $O(\frac{T(U)}{\sin(\theta)})$ where $T(U)$ is the time to implement U.*

2.5.3 Quantum linear algebra

In our algorithms, we will also use subroutines for quantum linear algebra. From the first work of [80] that proposed a quantum algorithm for matrix inversion, a lot of progress has been made. In this section, we briefly recall some of the results in quantum linear algebra. We conclude by citing the state-of-the-art techniques for performing not only matrix inversion and matrix multiplication, but also for applying a certain class of functions to the singular values of a matrix. A notable result after HHL, was the ability to perform a quantum version of the singular value decomposition. This idea is detailed in the following theorem.

Theorem 20 (Singular Value Estimation [91])**.** *Let $M \in \mathbb{R}^{n\times d}$ be a matrix with singular value decomposition $M = \sum_i \sigma_i u_i v_i^T$ for which we have quantum access. Let $\varepsilon > 0$ the precision parameter. There is an algorithm with running time $\tilde{O}(\mu(M)/\varepsilon)$ that performs the mapping $\sum_i \alpha_i |v_i\rangle \to \sum_i \alpha_i |v_i\rangle|\tilde{\sigma}_i\rangle$, where $|\tilde{\sigma}_i - \sigma_i| \leq \varepsilon$ for all i with probability at least $1 - 1/poly(n)$.*

Recall that quantum access to a matrix is defined in definition 8, and the parameter μ is defined in definition 9. The relevance of theorem 20 for quantum machine learning is the following: if we can estimate the singular values of a matrix, then we can perform a conditional rotation controlled by these singular values and hence perform a variety of linear algebraic operations, including matrix inversion, matrix multiplication or projection onto a subspace. Based on this result, quantum linear algebra was done using the theorem stated below.

Theorem 21 (Matrix algebra [92, 91])**.** *Let $M := \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{d\times d}$ such that $\|M\|_2 = 1$, and a vector $x \in \mathbb{R}^d$ for which we have quantum access. There exist quantum algorithms that with probability at least $1 - 1/poly(d)$ return*

(i) a state $|z\rangle$ such that $||z\rangle - |Mx\rangle| \leq \epsilon$ in time $\tilde{O}(\kappa^2(M)\mu(M)/\epsilon)$

(ii) a state $|z\rangle$ such that $||z\rangle - |M^{-1}x\rangle| \leq \epsilon$ in time $\tilde{O}(\kappa^2(M)\mu(M)/\epsilon)$

(iii) a state $|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\rangle$ in time $\tilde{O}(\frac{\mu(M)\|x\|}{\delta\theta\left\|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\right\|})$

One can also get estimates of the norms with multiplicative error η by increasing the running time by a factor $1/\eta$.

For a symmetric matrix $M \in \mathbb{R}^{d\times d}$ with spectral norm $\|M\| = 1$ for which we have quantum access, the running time of these algorithms depends on the condition number $\kappa(M)$ of the matrix, that can be replaced by $\kappa_\tau(M)$, a condition threshold where we keep only the singular values bigger than τ, and the parameter $\mu(M)$, a matrix dependent parameter defined in definition 9. The running time also depends logarithmically on the relative error ϵ of the final outcome state. Recall that these linear algebra procedures above can also be applied to any rectangular matrix $V \in \mathbb{R}^{n\times d}$ by considering instead the symmetric matrix $\overline{V} = \begin{pmatrix} 0 & V \\ V^T & 0 \end{pmatrix}$.

Singular value estimation of a product of two matrices

In the course of the work discussed in Chapter 4, we also needed an algorithm to perform Singular Value Estimation of a matrix W that is the product of two matrices P and Q for which we have quantum access. In the final version of the work, this algorithm has been replaced by Singular Value Transformation techniques, which we discuss in the next Section.

Theorem 22 (SVE of product of matrices)**.** *Assume to have quantum access to matrices $P \in \mathbb{R}^{d\times d}$ and $Q \in \mathbb{R}^{d\times d}$. Define $W = PQ = U\Sigma V^T$ and $\epsilon > 0$ an error parameter. There is a quantum algorithm that with probability at least $1 - poly(d)$ performs the mapping $\sum_i \alpha |v_i\rangle \to \sum_i \alpha_i |v_i\rangle |\overline{\sigma_i}\rangle$ where $\overline{\sigma_i}$ is an approximation of the eigenvalues σ_i of W such that $|\sigma_i - \overline{\sigma}_i| \leq \epsilon$, in time $\tilde{O}\left(\frac{(\kappa(P)+\kappa(Q))(\mu(P)+\mu(Q))}{\varepsilon}\right)$.*

Proof. We start by noting that for each singular value σ_i of W there's a corresponding eigenvalue $e^{-i\sigma_i}$ of the unitary matrix e^{-iW}. Also, we note that we know how to multiply by W by applying theorem 21 sequentially with Q and P. This will allow us to approximately apply the unitary $U = e^{-iW}$. The last step will consist of the application of phase estimation to estimate the eigenvalues of U and hence the singular values of W. Note that we need W to be a symmetric matrix because of the Hamiltonian simulation part. In case W is not symmetric, we redefine it as

$$W = \begin{bmatrix} 0 & PQ \\ (PQ)^T & 0 \end{bmatrix}$$

Note we have $W = M_1 M_2$ for the matrices M_1, M_2 stored in QRAM and defined as

$$M_1 = \begin{bmatrix} P & 0 \\ 0 & Q^T \end{bmatrix}, M_2 = \begin{bmatrix} 0 & Q \\ P^T & 0 \end{bmatrix}.$$

We now show how to approximately apply $U = e^{-iW}$ efficiently. Note that for a symmetric matrix W we have $W = V\Sigma V^T$ and using the Taylor expansion of the exponential function we have

$$U = e^{-iW} = \sum_{j=0}^{\infty} \frac{(-iW)^j}{j!} = V \sum_{j=0}^{\infty} \frac{(-i\Sigma)^j}{j!} V^T$$

With $\widetilde{U}$ we denote our first approximation of U, where we truncate the sum after ℓ terms.

$$\widetilde{U} = \sum_{j=0}^{\ell} \frac{(-iW)^j}{j!} = V \sum_{j=0}^{\ell} \frac{(-i\Sigma)^j}{j!} V^T$$

We want to chose ℓ such that $\left\|U - \widetilde{U}\right\| < \epsilon/4$. We have:

$$\left\|U-\widetilde{U}\right\| \leq \left\|\sum_{j=0}^{\infty}\frac{(-iW)^j}{j!}-\sum_{j=0}^{\ell}\frac{(-iW)^j}{j!}\right\| \leq \left\|\sum_{j=\ell+1}^{\infty}\frac{(-iW)^j}{j!}\right\| \leq \tag{2.9}$$

$$\sum_{j=\ell+1}^{\infty}\left\|\frac{(-iW)^j}{j!}\right\| \leq \sum_{j=\ell+1}^{\infty}\frac{1}{j!} \leq \sum_{j=\ell+1}^{\infty}\frac{1}{2^{j-1}} \leq 2^{-\ell+1} \tag{2.10}$$

where we used triangle inequality and that $\left\|W^j\right\| \leq 1$. Choosing $\ell = O(\log 1/\varepsilon)$ makes the error less than $\epsilon/4$.

We cannot apply $\widetilde{U}$ exactly but only approximately, since we need to multiply with the matrices $W^j, j \in [\ell]$ and we do so by using the matrix multiplication algorithm for the matrices M_1 and M_2. For each of these matrices, we use an error of $\frac{\epsilon}{8\ell}$ which gives an error for W of $\frac{\epsilon}{4\ell}$ and an error for W^j of at most $\frac{\epsilon}{4}$. The running time for multiplying with each W^j is at most $O(\ell(\kappa(M_1)\mu(M_1)\log(8\ell/\epsilon)+\kappa(M_2)\mu(M_2)\log(8\ell/\epsilon)))$ by multiplying sequentially. Hence, we will try to apply the unitary U by using the Taylor expansion up to level ℓ and approximating each $W^j, j \in [\ell]$ in the sum through our matrix multiplication procedure that gives error at most $\frac{\epsilon}{4}$.

In order to apply U on a state $|x\rangle = \sum_i \alpha_i |v_i\rangle$, let's assume $\ell+1$ is a power of two and define $N_l = \sum_{j=0}^{l}(\frac{(-i)^j}{j!})^2$. We start with the state

$$\frac{1}{\sqrt{N_l}}\sum_{j=0}^{l}\frac{-i^j}{j!}|j\rangle|x\rangle$$

Controlled on the first register we use our matrix multiplication procedure to multiply with the corresponding power of W and get a state at most $\epsilon/4$ away from the state

$$\frac{1}{\sqrt{N_l}}\sum_{j=0}^{l}\frac{-i^j}{j!}|j\rangle|W^jx\rangle.$$

We then perform a Hadamard on the first register and get a state $\epsilon/4$ away from the state

$$\frac{1}{\sqrt{\ell}}|0\rangle\left(\frac{1}{\sqrt{N'}}\sum_{j=0}^{l}\frac{-i^j}{j!}|W^jx\rangle\right)+|0^{\perp}\rangle|G\rangle$$

where N' just normalizes the state in the parenthesis. Note that after the Hadamard on the first register, the amplitude corresponding to each $|i\rangle$ is the first register is the same. We use this procedure inside an amplitude amplification procedure to increase the amplitude $1/\sqrt{\ell}$ of $|0\rangle$ to be close to 1, by incurring a factor $\sqrt{\ell}$ in the running time. The outcome will be a state $\epsilon/4$ away from the state

$$\left(\frac{1}{\sqrt{N'}}\sum_{j=0}^{l}\frac{-i^j}{j!}|W^jx\rangle\right)=|\tilde{U}x\rangle$$

which is the application of $\widetilde{U}$. Since $\left\|U-\widetilde{U}\right\| \leq \epsilon/4$, we have that the above procedure applies a unitary $\overline{U}$ such that $\left\|U-\overline{U}\right\| \leq \epsilon/2$. Note that the running time of this procedure is given by the amplitude amplification and the time to multiply with W^j, hence we have that the running time is

$$O(\ell^{3/2}(\kappa(M_1)\mu(M_1)\log(8\ell/\epsilon)+\kappa(M_2)\mu(M_2)\log(8\ell/\epsilon))$$

Now that we know how to apply $\overline{U}$, we can perform phase estimation on it with error $\epsilon/2$. This provides an algorithm for estimating the singular values of W with an overall error ϵ. The final running time is

$$O(\frac{\ell^{3/2}}{\epsilon}(\kappa(M_1)\mu(M_1)\log(8\ell/\epsilon) + \kappa(M_2)\mu(M_2)\log(8\ell/\epsilon))$$

We have $\mu(M_1) = \mu(M_2) = \mu(P) + \mu(Q)$ and $\kappa(M_1) = \kappa(M_2) = \frac{max\{\lambda^P_{max},\lambda^Q_{max}\}}{min\{\lambda^P_{min},\lambda^Q_{min}\}} \leq \kappa(P) + \kappa(Q)$, and since $\ell = O(\log 1/\epsilon)$the running time can be simplified to

$$\tilde{O}(\frac{(\kappa(P) + \kappa(Q))(\mu(P) + \mu(Q))}{\epsilon}).$$

□

2.5.4 Linear combination of unitaries, normal forms, and singular value transformations

We continue our journey in quantum linear algebra by discussing the state-of-the-art technique beneath quantum linear algebra, called singular value transformation.

The research of quantum algorithms for machine learning has always used techniques developed in other areas of quantum algorithms. Among the many, we cite quantum algorithms for Hamiltonian simulation and quantum random walks. In fact, using quantum random walks, it is possible to decrease the dependence on the error parameter, from polynomial to polylog$(1/\epsilon)$ [41]. Stemming from the research in Hamiltonian simulation [24, 103, 25, 104, 140], these techniques have been further optimized, pushing them to the limit of almost optimal time and query complexity. A significant progress in the direction of quantum algorithms for linear algebra was the so-called LCU, or linear combination of unitaries [41], which again was developed in the context of the Hamiltonian simulation problem.

Lemma 23 (Linear combination of unitaries [40])**.** *Let $M = \sum_i \alpha_i U_i$ be a linear combination of unitaries U_i with $\alpha_i > 0$ for all i. Let V be any operator that satisfies $V|0^m := \frac{1}{\sqrt{\alpha}}\sum_i \sqrt{\alpha_i}|i$, where $\alpha := \sum_i \alpha_i$. Then $W := V^\dagger UV$ satisfies*

$$W\left|0^{\otimes m}\right\rangle\left|\psi\right\rangle = \frac{1}{\alpha}\left|0^{\otimes m}\right\rangle M\left|\psi\right\rangle + \left|\Psi^\perp\right\rangle \tag{2.11}$$

for all states $|\psi\rangle$, where $U := \sum_i |i\rangle\langle i| \otimes U_i$ and $(|0^{\otimes m}\rangle\langle 0^{\otimes m}| \otimes I)|\Psi^\perp\rangle = 0$.

In recent years a new framework for operating on matrices with a quantum computer was developed, which can be found in works [37, 70]. We now briefly go through the machinery behind these results, as it will be used throughout this thesis.

Definition 24 (Block encoding)**.** *. Let $A \in \mathbb{C}^{2^s \times 2^s}$. We say that a unitary $U \in \mathbb{C}^{(s+a)\times(s+a)}$ is a (α, a, ϵ)-block-encoding of A if:*

$$\|A - \alpha(\langle 0|^a \otimes I)U(|0\rangle^a \otimes I)\| \leq \epsilon$$

We will see that having quantum access to a matrix $A \in \mathbb{C}^{2^w \times 2^w}$, as described in the setting of theorem 8, it is possible to implement a $(\mu(A), w+2, \text{polylog}(\epsilon))$-block-encoding of A [1]. Given matrix U which is a (α, a, δ)-block-encoding of A, and a matrix V which is a (β, b, ϵ)-block-encoding of B, it is simple to obtain a $(\alpha\beta, a+b, \alpha\epsilon + \beta\delta)$-block-encoding

[1]This polylog(ϵ) in the block-encoding is due to approximation error that one commits when creating quantum access to the classical data structures, i.e. is the approximation that derives from truncating a number $n \in \mathbb{R}$ (which represent an entry of the matrix) up to a certain precision ϵ [37, lemma 25].

of AB.

For practical purposes, having a block-encoding of a matrix A, allows one to manipulate its spectra using polynomial approximations of analytic functions. In the following theorem, the notation $P_{\Re}(A)$ means that we apply the polynomial P to the singular values of the matrix A, i.e. $P_{\Re}(A) = \sum_i^r P(\sigma_i) u_i v_i^T$.

Theorem 25 (Polynomial eigenvalue transformation of arbitrary parity [70])**.** *Suppose that U is an (α, a, ϵ)-block-encoding of the Hermitian matrix A. If $\delta \geq 0$ and $P_{\Re} \in \mathbb{R}[x]$ is a degree-d polynomial satisfying that*

- *for all $x \in [-1, 1]$: $|P_{\Re}(x)| \leq \frac{1}{2}$.*

Then there is a quantum circuit $\tilde{U}$, which is an $(1, a+2, 4d\sqrt{\epsilon/\alpha}+\delta)$-encoding of $P_{\mathcal{R}}(A/\alpha)$, and consists of d applications of U and $U^\dagger$ gates, a single application of controlled-U and $O((a+1)d)$ other one- and two-qubit gates. Moreover we can compute a description of such a circuit with a classical computer in time $O(polyd, \log(1/\delta))$.

For instance, matrix inversion can be seen as the problem of implementing the singular value transformation of $x \mapsto 1/x$. For this, one needs to get a polynomial approximation of the function $1/x$. While this might seem a simple task, there are small complications. First, one usually does not consider the whole interval $[-1, 1]$. In practice, one excludes the subset of the domain where the function has singularities (i.e. for $1/x$ is around zero). Then, it is preferable to pick a polynomial of a small degree (and small coefficients), as the depth of the circuit depends linearly on the degree of the polynomial.

Given a (α, a, ϵ)-block-encoding for a matrix A and a quantum state $|b\rangle$, we can obtain a good approximation of $A|b\rangle / \|Ab\|$ by first creating the state $|0^a, b\rangle$ and then applying the block-encoding of A to it. Then, we can amplify the part of the subspace associated to the state $|0\rangle^{\otimes a} A|b\rangle$. Differently, one might use advanced amplification techniques and reach a similar result. This concept is detailed in the following lemma.

Lemma 26 (Applying a block-encoded matrix to a quantum state [37, lemma 24])**.** *Fix any $\varepsilon \in (0, 1/2)$. Let $A \in \mathbb{C}^{N \times N}$ such that $\|A\| \leq 1$ and $|b\rangle$ a normalized vector in $\mathbb{C}^N$, such that $\|A|b\rangle\| \geq \gamma$. Suppose that $|b\rangle$ can be generated in complexity T_b and there is a (α, a, ϵ)-block-encoding of A for some $\alpha \geq 1$, with $\epsilon \leq \varepsilon\gamma/2$, that can be implemented in cost T_A. Then there is a quantum algorithm with complexity*

$$O\left(min(\frac{\alpha(T_A + T_b)}{\gamma}, \frac{\alpha T_A \log(1/\epsilon) + T_B}{\gamma})\right)$$

that terminates with success probability at least $2/3$, and upon success generates the state $A|b\rangle / \|A|b\rangle\|$ to precision ε.

For sake of completeness, we briefly discuss how to prove the first upper bound. Generating $|b\rangle$ and applying the block-encoding of A to it, we create a state that is (ϵ/α)-close to:

$$|0\rangle^{\otimes a} (\frac{1}{\alpha} A|b\rangle) + |0^\perp\rangle$$

From the hypothesis, we know that $\left\|\frac{1}{\alpha} A|b\rangle\right\| \geq \gamma/\alpha$. We can use $O(\alpha/\gamma)$ calls to amplitude amplification on the initial register being $|0\rangle^{\otimes a}$, to get $\frac{\epsilon}{\gamma}$ close to $|0\rangle^{\otimes a} \frac{A|b\rangle}{\|A\||b\rangle}$. The second upper bound is shown by other techniques based on amplitude amplification of singular values of block encoded matrices (i.e. [37, lemma 47], [104, theorem 2, 8]).

Regarding the usage of block-encoding for solving with a quantum computer a linear system of equations (i.e. multiplying a quantum state by the inverse of a matrix, and creating a state $|x\rangle$ proportional to $A^{-1}|b\rangle$) we can proceed analogously. First, we need to create block-encoding access to A^{-1}. Using the following lemma, (where they denoted with κ the condition number of A) we can implement negative powers of Hermitian matrices.

Lemma 27 (Implementing negative powers of Hermitian matrices [37, lemma 9]). *Let $c \in (0,\infty), \kappa \geq 2$, and let A be an Hermitian matrix such that $I/\kappa \leq A \leq I$. Suppose that $\delta = o(\epsilon/(\kappa^{1+c}(1+c)\log^3 \frac{k^{1+c}}{\epsilon}))$ and U is an (α, a, δ)-block-encoding of A that can be implemented using T_U gates. Then, for any ϵ, we can implement a unitary $\tilde{U}$ that is a $(2\kappa^c, a, \epsilon)$-block-encoding of H^{-c} in cost:*

$$O\left(\alpha\kappa(a+T_U)(1+c)\log^2(\frac{k^{1+c}}{\epsilon})\right)$$

Nevertheless, the algorithm that emerges by using the previous lemma has a quadratic dependence on κ. To decrease it to an algorithm linear in κ the authors used variable time amplitude amplifications[7]. Hence, we can restate the theorem 21, with the improved runtimes, as follows.

Theorem 28 (Matrix algebra [37, 70]). *Let $M := \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{d\times d}$ such that $\|M\|_2 = 1$, and a vector $x \in \mathbb{R}^d$ for which we have quantum access in time T_χ. There exist quantum algorithms that with probability at least $1 - 1/poly(d)$ return*

(i) a state $|z\rangle$ such that $||z\rangle - |Mx\rangle| \leq \epsilon$ in time $\tilde{O}(\kappa(M)(\mu(M)+T_\chi)\log(1/\epsilon))$

(ii) a state $|z\rangle$ such that $||z\rangle - |M^{-1}x\rangle| \leq \epsilon$ in time $\tilde{O}(\kappa(M)(\mu(M)+T_\chi)\log(1/\epsilon))$

(iii) a state $|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\rangle$ in time $\tilde{O}(T_\chi \frac{\mu(M)\|x\|}{\delta\theta\left\|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\right\|})$

One can also get estimates of the norms with multiplicative error η by increasing the running time by a factor $1/\eta$.

Another important advantage of the new methods is that it provides easy ways to manipulate sums or products of matrices.

Theorem 29 (Matrix algebra on products of matrices [37, 70]). *Let $M_1, M_2 \in \mathbb{R}^{d\times d}$ such that $\|M_1\|_2 = \|M_2\|_2 = 1$, $M = M_1M_2$, and a vector $x \in \mathbb{R}^d$ for which we have quantum access. There exist quantum algorithms that with probability at least $1 - 1/poly(d)$ return*

(i) a state $|z\rangle$ such that $||z\rangle - |Mx\rangle| \leq \epsilon$ in time $\tilde{O}(\kappa(M)(\mu(M_1)+\mu(M_2))\log(1/\epsilon))$

(ii) a state $|z\rangle$ such that $||z\rangle - |M^{-1}x\rangle| \leq \epsilon$ in time $\tilde{O}(\kappa(M)(\mu(M_1)+\mu(M_2))\log(1/\epsilon))$

(iii) a state $|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\rangle$ in time $\tilde{O}(\frac{(\mu(M_1)+\mu(M_2))\|x\|}{\delta\theta\left\|M^+_{\leq\theta,\delta}M_{\leq\theta,\delta}x\right\|})$

One can also get estimates of the norms with multiplicative error η by increasing the running time by a factor $1/\eta$.

More generally, applying a matrix M which is the product of k matrices, i.e. $M = M_1 \dots M_k$ will result in a runtime of $\kappa(M)(\sum_i^k \mu(M_i))\log(1/\epsilon)$ factors in the runtime.

2.5.5 Distance estimations and quadratic form estimation

In this section, we prove two new lemmas that can be used to estimate the inner products, distances and quadratic forms between vectors. The lemma 30 has been developed in the work [93], while the lemma for estimating the value of quadratic form has been formalized in the work under preparation with Changpeng Shao.

Lemma 30 (Distance / Inner Products Estimation [93]). *Assume for a matrix $V \in \mathbb{R}^{n\times d}$ and a matrix $C \in \mathbb{R}^{k\times d}$ that the following unitaries $|i\rangle|0\rangle \mapsto |i\rangle|v_i\rangle$, and $|j\rangle|0\rangle \mapsto |j\rangle|c_j\rangle$*

can be performed in time T and the norms of the vectors are known. For any $\Delta > 0$ and $\epsilon_1 > 0$, there exists a quantum algorithm that computes

$$\begin{aligned} |i\rangle|j\rangle|0\rangle &\mapsto |i\rangle|j\rangle|\overline{d^2(v_i,c_j)}\rangle, \text{ where } |\overline{d^2(v_i,c_j)} - d^2(v_i,c_j)| \leqslant \epsilon_1 \text{ w.p. } \geq 1-2\Delta, \text{ or} \\ |i\rangle|j\rangle|0\rangle &\mapsto |i\rangle|j\rangle|\overline{(v_i,c_j)}\rangle, \text{ where } |\overline{(v_i,c_j)} - (v_i,c_j)| \leqslant \epsilon_1 \text{ w.p. } \geq 1-2\Delta \end{aligned}$$

in time $\widetilde{O}\left(\frac{\|v_i\|\|c_j\|T\log(1/\Delta)}{\epsilon_1}\right)$.

It is relatively simple to extend the previous algorithm to one that computes an estimate of a quadratic form. We will consider the case where we have quantum access to a matrix A and compute the quadratic forms v^TAv and $v^TA^{-1}v$. The extension to the case when we have two different vectors, i.e. v^TAu and $v^TA^{-1}u$ is trivial.

Lemma 31 (Estimation of quadratic forms)**.** *Assume to have quantum access to a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ such that $\|A\| \leq 1$, and to a matrix $V \in \mathbb{R}^{n\times d}$. For $\epsilon > 0$, there is a quantum algorithm that performs the mapping $|i\rangle|0\rangle \mapsto |i\rangle|\overline{s_i}\rangle$, for $|s_i - \overline{s_i}| \leq \epsilon$, where s_i is either:*

- $(|v_i\rangle, A|v_i\rangle)$ *in time $O(\frac{\mu(A)}{\epsilon})$*
- $(|v_i\rangle, A^{-1}|v_i\rangle)$ *in time $O(\frac{\mu(A)\kappa(A)}{\epsilon})$*

The algorithm can return an estimate of $\overline{(v_i, Av_i)}$ such that $\overline{(v_i,Av_i)} - (v_i,Av_i) \leq \epsilon$ using quantum access to the norm of the rows of V by increasing the runtime by a factor of $\|v_i\|^2$.

Proof. Let's analyze first the case where we want to compute the quadratic form with A, and after the case for A^{-1}. Recall that the matrix A can be decomposed in an orthonormal basis $|u_i\rangle$. We can use theorem 28 to perform the following mapping:

$$|i\rangle|v_i\rangle|0\rangle = |i\rangle\frac{1}{N_i}\sum_j^n \alpha_{ij}|u_j\rangle|0\rangle \mapsto |i\rangle\frac{1}{N_i}\sum_i^n\left(\lambda_i\alpha_{ij}|u_i,0\rangle + \sqrt{1-\gamma^2}|G,1\rangle\right) = \quad (2.12)$$

$$|i\rangle\left(\|Av_i\||Av_i,0\rangle + \sqrt{1-\gamma^2}|G,1\rangle\right) = |i\rangle|\psi_i\rangle, \quad (2.13)$$

where $N_i = \sqrt{\sum_j^n \alpha_{ij}^2}$. We define $|\phi_i\rangle = |v_i,0\rangle$. Using controlled operations, we can then create the state:

$$\frac{1}{2}|i\rangle(|0\rangle(|\phi_i\rangle + |\psi_i\rangle) + |1\rangle(|\phi_i\rangle - |\psi_i\rangle)) \quad (2.14)$$

It is simple to check that, for a given register $|i\rangle$, the probability of measuring 0 is:

$$p_i(0) = \frac{1 + \|Av_i\|\langle Av_i|v_i\rangle}{2}$$

We analyze the case where we want to compute the quadratic form for A^{-1}. For a $C = O(1/\kappa(A))$, we create instead the state:

$$|i\rangle\frac{1}{\sqrt{\sum_i^n \alpha_i^2}}\sum_i^n\left(\frac{C}{\lambda_i}\alpha_i|v_i,0\rangle + \sqrt{1-\gamma^2}|G,1\rangle\right) = |i\rangle|\psi_i\rangle \quad (2.15)$$

In this case, the probability of measuring 0 in state of Equation 2.14 is

$$p_i(0) = \frac{1 + C\left\|A^{-1}v_i\right\|\langle A^{-1}v_i|v_i\rangle}{2}$$

For both cases, we are left with the task of coherently estimating the measurement probability in a quantum register and boost the success probability of this procedure. The unitaries that create the states in Equation 2.14 and 2.15 (i.e before a measurement on the ancilla qubit) describe a mapping: $U_1 : |i\rangle|0\rangle \mapsto \frac{1}{2}|i\rangle\left(\sqrt{p_i(0)}|y_i,0\rangle + \sqrt{1-p_i(0)}|G_i,1\rangle\right)$. As in [151], the idea is to use amplitude estimation, i.e. theorem 18, along with median evaluation lemma 14. We can apply amplitude estimation to obtain a unitary

$$U_2|i\rangle|0\rangle \mapsto \frac{1}{2}|i\rangle\left(\sqrt{\alpha}|p_i(0),y_i,0\rangle + \sqrt{1-\alpha}|G_i',1\rangle\right) \tag{2.16}$$

and estimate $p_i(0)$ such that $|p_i(0) - \overline{p_i(0)}| < \epsilon$ for the case of $v_i^T A v_i$ and we choose a precision ϵ/C for the case of $v_i^T A^{-1} v_i$ to get the same accuracy. Amplitude estimation theorem, i.e. theorem 18 fails with probability $\leq \frac{8}{\pi^2}$. The runtime of this procedure is given by combining the runtime of creating the state $|\psi_i\rangle$, amplitude estimation, and the median lemma. Since the error in the matrix multiplication step is negligible, and assuming quantum access to the vectors is polylogarithmic, the final runtime is $O(\log(1/\delta)\mu(A)\log(1/\epsilon_2)/\epsilon)$, with an additional factor $\kappa(A)$ for the case of the quadratic form of A^{-1}.

Note that if we want to estimate a quadratic form of two unnormalized vectors, we can just multiply this result by their norms. Note also that the absolute error ϵ now becomes relative w.r.t the norms, i.e. $\epsilon\|v_i\|^2$. If we want to obtain an absolute error ϵ', as in the case with normalized unit vectors, we have to run amplitude estimation with precision $\epsilon' = O(\epsilon/\|v_i\|^2)$. To conclude, this subroutine succeeds with probability $1-\gamma$ and requires time $O(\frac{\mu(A)\log(1/\gamma)\log(1/\epsilon_2)}{\epsilon_1})$, with an additional factor of $\kappa(A)$ if we were to consider the quadratic form for A^{-1}, and an additional factor of $\|v_i\|^2$ if we were to consider the non-normalized vectors v_i. This concludes the proof of the lemma. □

Note that this algorithm can be extended by using another index register to query for other vectors from another matrix W, for which we have quantum access. This extends the capabilities to estimating inner products in the form $|i\rangle|j\rangle|w_i^T A v_i\rangle$.

Chapter 3

Classical machine learning

In this chapter, we review and introduce the part of classical machine learning that has been studied in the course of this thesis. Special emphasis is put on formalizing the connection between the machine learning problems and their linear-algebraic formulation.

3.1 Supervised learning

Supervised (or predictive) machine learning is the part of machine learning that deals with supervised datasets, i.e. data where each sample x_i comes along with supervised information, i.e. a piece of data y_i. It helps the intuition thinking that the supervised information comes from a stochastic process that maps vectors x_i to vectors y_i. The goal is to model the mapping on the whole input space $\mathcal{X}$ to the output space $\mathcal{Y}$ given a set of input-output pairs $D = \{(x_i, y_i)\}_{i=0}^{n}$. Usually, the input space is a subset of $\mathbb{R}^d$, and the output space is usually either $\mathbb{R}$ or a finite set K of small cardinality. It is practical, for the sake of exposition to consider the training set organized into a matrix $X \in \mathbb{R}^{n \times d}$ and the matrix $Y \in R^n$ or $Y \in [K]^n$. The components of a vector x_i, i.e. a row of X are called *features, attributes, or covariates*. The matrix X is called *design matrix*, or simply the dataset. The vector y_i is called the *response variable*. If the response variable is categorical (or nominal), the problem is known as classification, or pattern recognition. If the response variable is real-valued we interpret this problem as learning a function $f : \mathbb{R}^d \mapsto \mathbb{R}$ and we call this problem regression. Different assumptions on the structure of f lead to different machine learning models. Each model can be trained (or fitted) with different algorithms.

3.2 Unsupervised learning

Unsupervised learning [110], which sometimes goes under the name of knowledge discovery, is the part of machine learning that deals with understanding unlabeled data. In the dataset $D = \{x_i\}_{i=0}^{n}$ we don't have anymore any supervised information. In this case, it is common to understand the structure of the process generating the samples by formalizing a *density estimation* problem: we want to learn the parameters θ of a function $p(x_i|\theta)$ that models the distribution of the process that has generated the samples. The importance of unsupervised learning lies in the stunning similarity with human and animal learning. Furthermore, most of the dataset that we have are unsupervised, as it is costly to provide supervised information from experts or humans. The most common example of unsupervised learning is clustering, where we want to partition into groups a given dataset. As an example, imagine having a set comprising of images of cats and dogs, without knowing which image is a cat or which is a dog. An unsupervised learning algorithm is supposed to learn how

to split the dataset correctly, by understanding the characteristics and features that allows discriminating between images of different kinds. Just to name a few of the more concrete examples, in astronomy, clustering is often used to discover new kinds of stars, in biology, it is used to find new kinds of cells, in cybersecurity, to perform anomaly detection, and so on.

We refer to the number of clusters in the dataset with a letter K. The first goal in clustering is to understand the right number of different groups in the data (which might not be known a-priori). The second goal is to estimate which cluster each point x_i belongs to. We define z_i for $z_i \in [K]$ as the cluster to which point x_i is assigned to. The value of z_i is often called *hidden* or *latent variable*. Unsupervised learning can be seen as the task of guessing the value of the hidden variable, by computing $z_i = \arg\max_k p(z_i = k|x_i, \theta)$. For this, an unsupervised learning algorithm has to model (implicitly or explicitly) the joint probability distribution $p(x, y)$.

While latent variables have extensive applications, in this thesis we will focus on the case where latent variables are used to represent a discrete latent state (as in clustering).

3.3 Generative and discriminative machine learning

There is another insightful way of organizing machine learning models. They can either be *generative* or *discriminative*. A discriminative model learns just the mapping $p(y|x)$, and provides a way to classify points (i.e. infer the value of y_i), without actually knowing "how" the point x_i has been generated. Examples of such models are: k-nearest neighbors, Logistic regression, Support Vector Machines, Decision Trees, Random Forest, Neural Networks, and so on. Examples of such models in this thesis are the QFDC, QSFA in chapter 4. On the other way, generative models output a model for the joint probability distribution $p(x, y)$. This is similar to the unsupervised learning case, but in this cases the dependence on y (which can be a hidden variable) is made explicit. In general, discriminative models make fewer assumptions, as generative models often need to do some assumption on the structure of $p(x)$. Such generative models are one of the most promising approaches to unsupervised problems. The goal of a generative model is to learn a probability distribution that is most likely to have generated the data collected in a training set. Fitting the model consists of learning the parameters of a probability distribution p in a certain parameterized family, that best describes our vectors x_i, y_i. In case the data is unsupervised, generative models learn the probability distribution $p(x_i)$ assuming the existence of some hidden variables z_i. Examples of such models in this thesis are q-means and GMM, i.e. chapter 5 and 6. A possible way to fit a generative model is to formulate the problem of finding the parameters of the family of distribution as an optimization problem. This is often done using the so-called maximum likelihood estimation (MLE). One can think of the *likelihood* as the function that we use to measure how good a model is for explaining a given dataset. For a given machine learning model with parameters γ, the likelihood of our data set X is the probability that the data have been generated by the model with parameters γ, assuming each point is independent and identically distributed. We think of the likelihood as a function of γ, holding the dataset X fixed. For $p(x_i|\gamma)$ the probability that a point x_i comes from model γ, the likelihood is defined as:

$$L(\gamma; X) := \prod_{i=1}^{n} p(x_i|\gamma) \tag{3.1}$$

From this formula, we can see that in order to find the best parameters γ^* of our model we need to solve an optimization problem. For numerical and analytical reasons, instead of maximizing the likelihood L, it is common practice to find the best model by maximizing the *log-likelihood* function $\ell(\gamma; X) = \log L(\gamma; X) = \sum_{i=1}^{n} \log p(x_i|\gamma)$. In this context, we

want to find the model that maximizes the log-likelihood:

$$\gamma^*_{ML} := \arg\max_{\gamma} \sum_{i=1}^{n} \log p(x_i|\gamma). \tag{3.2}$$

The procedure to calculate the log-likelihood depends on the specific algorithm used to model the data. A possible solution would be to use a gradient-based optimization algorithm on ℓ. It is often the case that, due to the indented landscape of the likelihood function, gradient-based techniques often do not perform well. Therefore, it is common to find other strategies to find do maximum likelihood estimation. One of which is the Expectation-Maximization (EM) algorithm, detailed in chapter 6.

3.4 Dimensionality Reduction

Dimensionality reduction (DR), a technique used both in supervised and unsupervised learning, refers to the procedure by which the dimension of the input data is reduced while retaining most of the meaningful information contained therein. It is often a necessary step when trying to solve practical problems in machine learning and there are many techniques for performing it. For instance, it is used to decrease the variance of a model, since it can lead to models with a fewer number of parameters, and it might just reduce the noise in the data. It is also necessary when the runtime of the algorithm has polynomial dependence on the number of features, as it is often the case for nowadays datasets. In the context of big data analysis, by removing features that carry low information (like features that are strictly proportional to other features, or features for which the data contains too little information), it is possible to optimize the storage space. It can be also used for data visualization. Most importantly, supervised algorithms often suffer from the *curse of dimensionality*: by allowing large dimensionality of data, the informative power of the data points in the training set decreases, thus leading to a degradation in classification performances. One solution to improve the accuracy would be to increase the number of elements in the training set, but this is not always possible nor desirable, so the common route is to decrease the dimension of the data. Mathematically, the idea of the dimensionality reduction algorithms is to map vectors from a high dimensional space $\mathcal{X}$ to a low dimensional space $\mathcal{Y}$, such that the most meaningful information (according to some criteria) is preserved. Of course, understanding which criterion to use is far from trivial.

The choice of the right DR algorithm depends on the nature of the data as well as on the type of algorithm that will be applied after the dimensionality reduction. A very well-known DR algorithm is the Principal Component Analysis (PCA), which projects the data points onto the subspace spanned by the eigenvectors associated to the k largest eigenvalues of the covariance matrix of the data. In this way, the projection holds "most of the information" of the dataset. It is possible to show [110] that for a subspace of dimension k, this choice of eigenvectors minimizes the reconstruction error, i.e. the distance between the original and the projected vectors. However, PCA is not always the best choice of dimensionality reduction. PCA projects the data into the subspace along which the data has more variance. This does not take into consideration the information that different points might belong to different classes, and there are cases in which PCA can worsen the performance of the classifier. Other methods, like Fisher Linear Discriminant and Slow Feature Analysis take into account the variance of every single class of points. Indeed, FLD projects the data in a subspace trying to maximize the distance between points belonging to different clusters and minimizing the distance between points belonging to the same cluster, thus preserving or increasing the accuracy.

3.5 The Generalized Eigenvalue Problems in Machine Learning

Here we review the connection between the so-called Generalized Eigenvalue Problem (GEP) and some models in machine learning. In classical literature, this is a well-known subject [67, 48, 31].

Definition 32 (Generalized Eigenvalue Problem)**.** *Let $A, B \in \mathbb{R}^{d\times d}$ be two SPD matrices. The GEP is defined as:*

$$AW = BW\Lambda \tag{3.3}$$

The columns $w_i \in \mathbb{R}^d$ of W and the values $\lambda_i = \Lambda_{ii} \in \mathbb{R}$ of the diagonal matrix Λ are the so-called generalized eigenvector and eigenvalues.

The generalized eigenvalue problem is denoted by (A, B) (note that the order in the pair matters). As is evident, the canonical eigenvalue problem is a special case of the GEP where $B = I$. In this work, we will often consider the case when matrices A and B consist of expectation values from stochastic processes, that is, these are covariance matrices of some sort. Furthermore, while A can be symmetric semi-positive definite, we require B to be invertible, and thus symmetric positive definite. The GEP is related to the so-called *Raylight quotient*: a variational extremum problem related to the ratio of two quadratic forms involving matrix A and B:

$$\rho(w) := \frac{w^T Aw}{w^T Bw} \tag{3.4}$$

There many different optimization problems that can be reduced to a GEP, which we report here for completeness [48, 67]. One can see that the norm of w does not change the value of the optimization problem. Therefore, we can impose an additional constraint on w. In this way, we can reformulate the problem as a constrained optimization problem, without losing any solution. This constraint is $w^T Bw = 1$. We now describe the relation between Equation 3.4 and Equation in definition 32.

Optimization Form 1 (Vector form) For $\phi \in \mathbb{R}^d$

$$\max_{\phi} \quad w^T Aw \tag{3.5a}$$

$$\text{subject to} \quad w^T Bw = 1 \tag{3.5b}$$

We can reduce the optimization problem 3.5 to the GEP using Lagrangian multipliers. The Lagrangian equation associated to this optimization problem is $\mathcal{L} = w^T Aw - \lambda(w^T Bw - 1)$. Equating the derivative of the Lagrangian to 0 gives:

$$\frac{\partial \mathcal{L}}{\partial w} = 2Aw - 2\lambda Bw = 0 \mapsto 2Aw = 2\lambda Bw \mapsto Aw = \lambda Bw. \tag{3.6}$$

This can be generalized to multiple vectors w_i, and expressed in Matrix form, as follows:

Optimization Form 1 (Matrix form) For $W \in \mathbb{R}^{d\times d}$.

$$\max_{w} \quad Tr[W^T AW] \tag{3.7a}$$

$$\text{subject to} \quad W^T BW = I \tag{3.7b}$$

There is a dual formulation to this problem, that consists in minimizing the Frobenius norm of a matrix. This formulation occurs often as the dual formulation of the previous optimization problems. For instance, instead of maximizing correlation, one would like to minimize some error.

Optimization Form 2 (Vector Form) For $w \in \mathbb{R}^d$, and a matrix $X \in \mathbb{R}^{d\times n}$.

$$\min_{w} \quad \|X - ww^T X\|_F^2 = Tr[X^T X - XX^T ww^T] \tag{3.8a}$$
$$\text{subject to} \quad w^T Bw = 1 \tag{3.8b}$$

This is the GEP (XX^T, B).

Optimization Form 2 (Matrix Form) For $w \in \mathbb{R}^d$, and a matrix $X \in \mathbb{R}^{d\times n}$.

$$\min_{W} \quad \|X - WWTX\|_F^2 = Tr[X^T X - XX^T WW^T] \tag{3.9a}$$
$$\text{subject to} \quad W^T BW = 1 \tag{3.9b}$$

This is the GEP (XX^T, B).

Classical algorithms Computationally, fast algorithms for solving the GEP have been proposed through the years [65, 141, 31]. In general, a simple-but-dirty solution would be to left multiply both sides of Equation 3.3 by B^{-1}, and re-conduct this to the canonical eigenvalue problem. Unfortunately, this solution suffers often from problems of numerical instability and is rarely the preferred solution. It is well-known that symmetric eigenvalue problems are more stable to numerical errors. For this, a slightly more intricate solution, but computationally more expensive, consist of making the matrix A symmetric: more stable to perturbations and numerical errors [150]. Other classical randomized algorithms for solving the GEP exist. Among the best classical algorithms, we cite [65], where they propose an algorithm to extract the top k solutions of a GEP in time $O(\frac{kz\sqrt{\kappa}}{\rho}\log(1/\epsilon)\log(k\kappa/\rho)))$ where z is $\|A\|_0 + \|B\|_0$, κ is the biggest condition number between $\kappa(A)$ and $\kappa(B)$, and ρ is the relative eigenvalue gap, i.e. $1 - \frac{|\lambda_k+1|}{|\lambda_k|}$. These algorithms might not give the best possible runtime in all the machine learning models described below. In fact, according to the definition of the matrices A and B, other algorithms for extracting the generalized eigenvectors might be used instead. We will see that quantum computers might lead to better runtimes, by removing the dependence on the size of the two matrices (it becomes polylogarithmic), albeit with a worsening in the other parameters. For this kind of runtimes, the price that we have to pay is a preprocessing step, detailed in the previous chapter. We conclude this section with a useful observation from classical literature.

Lemma 33 ([65])**.** *Let (w_i, σ_i), be an eigenpairs of the symmetric matrix $B^{-1/2}AB^{-1/2}$. Then $B^{-1/2}w_i$ is an eigenvector of $B^{-1}A$ with eigenvalue σ_i.*

Proof. The proof follows from noting that:

$$B^{-1}A(B^{-1/2}w_i) = B^{-1/2}(B^{-1/2}AB^{-1/2})w_i = \sigma_i B^{-1/2}w_i$$

□

3.5.1 Slow Feature Analysis - SFA

SFA is a dimensionality reduction algorithm, better detailed in chapter 4. Here, for consistency, we report the definition of the optimization problem, as stated in [23]. For a dataset consisting in a set of n different d-dimensional vectors represented by the matrix $X^{n\times d}$, along with their labels $l_i \in [k]$, stored in a matrix $L \in [K]^n$. Given a training set X, L of k different classes, we expect to learn $k-1$ functions $g_j(x_i)$ with $j \in [k-1]$ whose output $y_i = [g_1(x(i)) \cdots g_{k-1}(x_i)]$ is constant and similar for the training samples of the same class. More formally, for a a the normalization factor defined as: $\sum_{k=1}^{K} \binom{T_k}{2}$. we want to minimize for all j:

$$\Delta_j = \Delta(y_j) = \frac{1}{a} \sum_{k=1}^{K} \sum_{\substack{s,t\in T_k \\ s<t}} (g_j(\boldsymbol{x}^{(k)}(s)) - g_j(\boldsymbol{x}^{(k)}(t)))^2 \tag{3.10}$$

with the following constraints:

1. $\frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{T_k} g_j(\boldsymbol{x}^{(k)}(i)) = 0 \quad \forall j \in [K-1]$
2. $\frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{T_k} g_j(\boldsymbol{x}^{(k)}(i))^2 = 1 \quad \forall j \in [K-1]$
3. $\frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{T_k} g_j(\boldsymbol{x}(i)^{(k)}) g_v(\boldsymbol{x}(i)^{(k)}) = 0 \quad \forall v < j \in [K-1]$

Define $B = X^T X$ as the covariance matrix and $A = \dot{X}^T \dot{X}$ is the derivative covariance matrix. This matrix $\dot{X} \in \mathbb{R}^{m\times d}$ is obtained by taking the pairwise difference between vectors that belongs to *the same* class. This is a kind of "derivative" between random pairs of elements in the dataset. The solution consist in computing the $K-1$ eigenvector of the Generalized Eigenvalue Problem (GEP):

$$AW = BW\Lambda \tag{3.11}$$

The objective function that we want to minimize in this case is given by:

$$\min J_{SFA}(w) := \frac{w^T A w}{w^T B w} \tag{3.12}$$

Solving this problem consists in making the matrix B the identity, i.e. whitening the data. A dataset is said to be whitened when the covariance matrix of the data is the identity. In practice we multiplied by $X^{-1/2}$ and redefined A accordingly, by computing the covariance matrices of the whitened derivatives vectors.

While the original formulation of SFA is about finding vectors that represent the directions of slowly varying features, in the context of supervised learning these vectors represent the subspace that maximally separates different clusters. It has been shown that solving the optimization problem upon which SFA is based is equivalent to other dimensionality reduction algorithms, like Laplacian Eigenmaps [136] and Fisher Linear Discriminant [95], thus a quantum algorithm for SFA also provides algorithms for Laplacian Eigenmaps and Fisher Linear Discriminant.

3.5.2 Linear Discriminant Analysis - LDA

Linear Discriminant Analysis is an unsupervised dimensionality reduction algorithm. LDA is known to be optimal in the sense that it projects the data in the subspace that minimize the distance between points of the same cluster and maximize the distance between points that belongs to different clusters. Sometimes the name LDA is used as a synonym for Fisher Discriminant Analysis, but we reserve the name FDA for the case when there are only two classes.

For n labeled points (x_i, y_i) for $x_i \in \mathbb{R}^d$ and $y_i \in [K]$ we defined $\mu_j = \frac{1}{|S_j|}\sum_{i \in S_j} x_i$ and $\mu = \frac{1}{n}\sum_{i \in X} x_i$ The matrix S_W is the so called within class covariance matrix, and the matrix S_B is the between-class covariance matrix. They are defined as:

$$S_W := \sum_{j \in [K]} \sum_{i \in [|S_j|]} (x - \mu_c)(x - \mu_c)^T \tag{3.13}$$

$$S_B := \sum_{j \in [K]} |S_j|(\mu_c - \mu)(\mu_c - \mu)^T \tag{3.14}$$

The optimization problem that we are trying to solve with LDA is:

$$\max J_{FLD}(w) := \frac{w^T S_B w}{w^T S_W w} \tag{3.15}$$

Note that for very small α, the contribution of S_B to A can be neglected, making $A \approx \frac{2}{N} S_W$. The GEP formulation of LDA is therefore

$$S_b w = \lambda S_w w$$

but instead of taking the smallest eigenvalues, we select the eigenvectors corresponding to the biggest eigenvalues.

3.5.3 Canonical Correlation Analysis - CCA

Canonical Correlation Analysis (CCA) is an algorithm for finding correlations between two different representations of the same data. Geometrically, the problem of finding correlations can be reduced to the problem of finding basis vectors for two sets of variables such that the correlation between the projections of the variables into the new basis vectors is mutually maximized [79]. A tutorial on CCA can be found in [149]. The input of the CCA problem is a dataset that consists of tuples of vectors $\{(x_i, y_i)\}_{i=0}^n$ where $x_i \in \mathbb{R}^{d_1}$ and $y_i \in \mathbb{R}^{d_2}$. Assuming the data is normalized in order to have zero mean and unit variance on each component, the sample covariance matrices of the data are $\Sigma_X = \frac{1}{n} X^T X$ and $\Sigma_Y = \frac{1}{n} Y^T Y$. We define the cross-covariance matrix Σ_{XY} as $X^T Y$, and analogously Σ_{YX} as $Y^T X$ (sometimes these matrices are called in literature cross-scatter matrix).

It is simpler to introduce the problem solved by CCA as the problem of finding the features w_x, w_y such that the directions Xw_x and Yw_y in the column space of X and Y have a minimal angle between each other:

$$w_x, w_y = \arg\max_{w_x, w_y} cos((Xw_x, Yw_y))$$

$$= \arg\max_{w_x, w_y} \frac{(Xw_x)^T (Yw_y)}{\sqrt{(Xw_x)^T (Xw_x)}\sqrt{(Yw_y)(Yw_y)}}$$

$$= \arg\max_{w_x, w_y} \frac{w_x^T \Sigma_{XY} w_y}{\sqrt{w_x^T S_{XX} w_x}\sqrt{w_y^T S_{YY} w_y}}$$

Again, we can see that the solution is independent on the norm of the weight vectors w_x, w_y. Thus, we can solve the optimization problem with a constraint on the value of the norms of the weight vectors, i.e that $w_x^T \Sigma_{XX} w_x = 1$ and $w_y^T \Sigma_{YY} w_y = 1$. We thus have formulated a constraint optimization problem:

$$w_x, w_y = \arg\max_{w_x, w_y} \|Xw_w, Yw_y\|^2 \text{ such that } \|Xw_x\|^2 = \|Yw_y\|^2 = 1 \tag{3.16}$$

We reformulate this optimiation problem using Lagrangian multipliers:

$$\mathcal{L}(w_x, w_y, \lambda_x, \lambda_y) = w_x^T \Sigma_{XY} w_y - \lambda_x w_x^T \Sigma_{XX} w_x - \lambda_y w_y^T \Sigma_{YY} w_y$$

Taking derivatives with respect to w_x and w_y, and setting the equations to 0 gives the following systems of equations:

$$\begin{cases} \Sigma_{XY} w_y = \lambda_x \Sigma_{XX} w_x \\ \Sigma_{YX} w_x = \lambda_y \Sigma_{YY} w_y \end{cases}$$

As $\lambda_x w_x^T \Sigma_{XX} w_x = w_x^T \Sigma_{XY} w_y = w_y^T \Sigma_{YX} w_x = \lambda_y w_y^T \Sigma_{YY} w_y$, and as $w_x^T \Sigma_{XX} w_x = w_y^T \Sigma_{YY} w_y = 1$ we can observe that the two eigenvalues λ_x and λ_y are identical, and we denote them just with λ.

$$\begin{cases} \Sigma_{XY} w_y = \lambda \Sigma_{XX} w_x \\ \Sigma_{YX} w_x = \lambda \Sigma_{YY} w_y \end{cases} \tag{3.17}$$

These system of equation represent the following GEP [48, 79], in a form that makes explicit both w_x and w_y in a single equation:

$$\begin{pmatrix} 0 & \Sigma_{XY} \\ \Sigma_{YX} & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \lambda \begin{pmatrix} \Sigma_{XX} & 0 \\ 0 & \Sigma_{YY} \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} \tag{3.18}$$

As we know, this GEP can be restated as a canonical eigenvalue problem:

$$\begin{pmatrix} 0 & \Sigma_{XX}^{-1} \Sigma_{XY} \\ \Sigma_{YY}^{-1} \Sigma_{YX} & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \lambda \begin{pmatrix} w_x \\ w_y \end{pmatrix} \tag{3.19}$$

This can be made symmetric by setting the vectors $v_x = \Sigma_{XX}^{1/2} w_x$ and $v_y = \Sigma_{YY}^{1/2} w_y$

$$\begin{pmatrix} 0 & \Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{1/2} \\ \Sigma_{YY}^{-1/2} \Sigma_{YX} \Sigma_{XX}^{-1/2} & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \lambda \begin{pmatrix} v_x \\ v_y \end{pmatrix} \tag{3.20}$$

There is an alternative formulation of the GEP for CCA. Recalling that we assume the matrix Σ_{YY}^{-1} is invertible, we can express w_y in the first equation of the system of equations 3.17 as

$$w_y = \frac{\Sigma_{YY}^{-1} \Sigma_{YX} w_x}{\lambda} \tag{3.21}$$

Via substitution in the linear system, i.e. in Equation 3.17, we can express the singular vectors w_x as solutions of the following GEP:

$$\Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX} w_x = \lambda^2 \Sigma_{XX} w_x \tag{3.22}$$

Then, we are able to use Equation 3.21 to find the corresponding w_y. Again, using the fact that Σ_{XX} is invertible, we can reconduct this GEP to a symmetric eigenvalue problem by recalling that we can define Σ_{XX} as $\Sigma_{XX} = \Sigma_{XX}^{1/2} \Sigma_{XX}^{1/2}$. This is the approach preferred in [79, 16]. The solution of the CCA can be also expressed using SVD, as discovered Halay in 1957 [82], and suggested again Ewerbring and others in 1990 [56]. First, we consider the SVD of

$$\Sigma_{XX} \Sigma_{XY}^{-1} \Sigma_{YY} = U \Sigma V^T \tag{3.23}$$

The singular values of Σ correspond to the canonical correlation. The matrices of canonical correlation W_x and W_y are obtained as:

$$W_x = \Sigma_{XX}^{-1/2} U \quad \text{and} \quad W_y = \Sigma_{YY}^{-1/2} V$$

.

3.5.4 Gaussian Information Bottleneck Method - GIBM

The Information Bottleneck Method (IBM) is a general and powerful model of learning, that has its foundation in information theory. The IBM has been originally proposed as a way to extract the relevant information from a random variable $\mathcal{X}$, with respect to another random variable $\mathcal{Y}$, while compressing $\mathcal{X}$ as much as possible. With such a broad formulation, the IBM can be applied to both supervised and unsupervised problems (as the $\mathcal{Y}$ can be just thought as being a hidden variable for the unsupervised case). The compressed representation that we extract from $\mathcal{X}$ can be thought as of another random variable, that we can call T. It can be shown that T is an approximation of a minimal sufficient statistic of $\mathcal{X}$ w.r.t. $\mathcal{Y}$.

Definition 34 (Sufficient Statistics [61, 77])**.** . *Let $Y \in \mathcal{Y}$ be an unknown parameter and $X \in \mathcal{X}$ be a random variable with conditional probability distribution function $p(x|y)$. Given a function $f : \mathcal{X} \to \mathcal{S}$, the random variable $S = f(X)$ is called a sufficient statistic for Y if:*

$$\begin{aligned} &\forall x \in \mathcal{X}, y \in \mathcal{Y} : \\ &\quad P(X = x|Y = y, S = s) = P(X = x|S = s). \end{aligned} \tag{3.24}$$

The meaning of definition 34 is that S captures all the information about Y that is available in X. As a consequence, we can state the following theorem:

Theorem 35. *Let S be a probabilistic function of X. Then S is a sufficient statistic for Y if and only if:*

$$I(S;Y) = I(X;Y)$$

Here, $I(X;Y)$ denotes the mutual information between the random variable X and the random variable Y. It is simple to note that, according to this definition, the identity function makes the random variable X a sufficient statistic for Y. To make something useful out of this definition, we should introduce a way of compressing information contained in X. For this, comes to our aid the definition of minimal sufficient statistic.

Definition 36. *(Minimal Sufficient Statistic) A sufficient statistic S is said to be minimal if it is a function of all other sufficient statistics, i.e.:*

$$\forall T;\ T \textit{ is sufficient statistic } \Rightarrow \exists\, g; S = g(T). \tag{3.25}$$

The relation between minimal sufficient statistics and information theory is stated in the following theorem:

Theorem 37 ([77])**.** *Let X be a sample drawn from a distribution function that is determined by the random variable Y. The statistic S is an MSS for Y if and only if it is a solution of the optimization process*

$$\min_{T:\textit{is sufficient statistic}} I(X;T). \tag{3.26}$$

Using theorem 35 we can restate the previous optimization problem as

$$\min_{T:I(T;Y)=I(X;Y)} I(X;T). \tag{3.27}$$

The goal of the IBM is to find a compressed representation of X that preserves as much information as possible about the random variable Y. In this sense, T is a minimal sufficient statistic for random variable X and Y. The trade-off is obtained using a Lagrangian multiplier which we call β. Note that we are interested in the stochastic maps from X to T and T to Y. Using theorem 35, we can restate the previous theorem as:

Definition 38 (Information Bottleneck)**.**

$$\min_{p(\tilde{x}|x)} I(\tilde{X};X) - \beta I(\tilde{X};Y) \tag{3.28}$$

The first term is meant to measure the compression, and the second term is meant to measure the relevance of the sufficient statistic T with respect to Y. In the IBM, T is meant to be a representation of X in a "meaningful semantic space". The Lagrangian multiplier $\beta \in [0, \infty)$ decides the trade-off representation and compression. Surprisingly enough, the parameter β allows one to train models that are explicit with respect to the bias/variance trade-off. This optimization problem is closely related to the Rate-Distortion theorem of Shannon, which described the trade-off between compression and resilience to errors in a channel. For further information, we refer to [146, 134]. Far from being just a theoretical tool to analyze learning, the IBM has been used to solve a stack of problems in machine learning [84, 135]. According to the problem under consideration, the IB can be computed via a plethora of different algorithms [134]. There is one case where the solution of the IB can be computed analytically: under the assumption that the random variables X, Y are jointly multivariate Gaussian distributions. Let the cross-covariance matrices be defined as usual, i.e. $\Sigma_{XY} := X^T Y$ and $\Sigma_{YX} := Y^T X$, where we assume that the matrices X and Y are scaled to have features with zero means and unit variance. We also define the conditional covariance, or canonical correlation matrix as: $\Sigma_{X|Y} = \Sigma_X - \Sigma_{XY}\Sigma_Y^{-1}\Sigma_{YX}$. In the Gaussian Information Bottleneck, we need to access the left singular vectors of:

$$A = \Sigma_{X|Y}\Sigma_X^{-1}$$

The number of eigenvectors to extract is a function specified by a trade-off parameter β.

It is straightforward to see that GIB and CCA are two ways of looking at the same problem. Recall the GEP problem of CCA in Equation 3.22 is:

$$\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}w_x = \lambda^2\Sigma_{XX}w_x \tag{3.29}$$

The matrix A is defined as $I - \Sigma_{XY}\Sigma_Y^{-1}\Sigma_{YX}\Sigma_{XX}^{-1}$. Adding or removing the identity matrix from another matrix will only shift its spectrum, so we can consider the left singular vectors of $\Sigma_{XY}\Sigma_Y^{-1}\Sigma_{YX}\Sigma_{XX}^{-1}$. Taking the transpose of this matrix, we see that this is exactly the same matrix of the GEP in Equation 3.29.

Part I

Quantum Algorithms for Machine Learning

Chapter 4

Quantum slow feature analysis and classification

4.1 Introduction to Slow Feature Analysis

Slow Feature Analysis (SFA) is a dimensionality reduction technique proposed in the context of computational neurosciences as a way to model part of the visual cortex of humans. In the last decades, it has been applied in various areas of machine learning. In this chapter we propose a quantum algorithm for slow feature analysis, and detail its application for performing dimensionality reduction on a real dataset. We also simulate the random error that the quantum algorithms might incur. We show that, despite the error caused by the algorithm, the estimate of the model that we obtain is good enough to reach high accuracy on a standard dataset widely used as benchmark in machine learning. Before providing more details on this result, we give a brief description of dimensionality reduction and introduce the model of slow feature analysis in this context.

SFA has been shown to model a kind of neuron (called complex cell) situated in the cortical layer in the primary visual cortex (called V1) [22]. SFA can be used in machine learning as a DR algorithm, and it has been successfully applied to enhance the performance of classifiers [156, 23]. SFA was originally proposed as an *online, nonlinear, and unsupervised algorithm* [152]. Its task was to learn slowly varying features from generic input signals that vary rapidly over time [23, 152]. SFA has been motivated by the *temporal slowness principle*, that postulates that while the primary sensory receptors (like the retinal receptors in an animal's eye) are sensitive to very small changes in the environment and thus vary on a very fast time scale, the internal representation of the environment in the brain varies on a much slower time scale. The slowness principle is a hypothesis for the functional organization of the visual cortex and possibly other sensory areas of the brain [153] and it has been introduced as a way to model the transformation invariance in natural image sequences [156]. SFA is an algorithm that formalizes the slowness principle as a nonlinear optimization problem. In [28, 138], SFA has been used to do nonlinear blind source separation. Although SFA has been developed in the context of computational neurosciences, there have been many applications of the algorithm to solve ML related tasks. A prominent advantage of SFA compared to other algorithms is that it is almost hyperparameter-free. The only parameters to chose are in the preprocessing of the data, e.g. the initial PCA dimension and the nonlinear expansion that consists of a choice of a polynomial of (usually low) degree p. Another advantage is that it is guaranteed to find the optimal solution within the considered function space [55]. For a detailed description of the algorithm, we suggest [137]. With appropriate preprocessing, SFA can be used in conjunction to a supervised algorithm to acquire classification capabilities. For instance

it has been used for pattern recognition to classify images of digits in the famous MNIST database [23]. SFA can be adapted to be used to solve complex tasks in supervised learning, like face and human action recognition [75, 156, 142].

We can use SFA for classification in the following way. One can think of the training set a set of vectors $x_i \in \mathbb{R}^d, i \in n$. Each x_i belongs to one of K different classes. A class T_k has $|T_k|$ vectors in it. The goal is to learn $K-1$ functions $g_j(x_i), j \in [K-1]$ such that the output $y_i = [g_1(x_i), \cdots, g_{K-1}(x_i)]$ is very similar for the training samples of the same class and largely different for samples of different classes. Once these functions are learned, they are used to map the training set in a low dimensional vector space. When a new data point arrive, it is mapped to the same vector space, where classification can be done with higher accuracy. SFA projects the data points onto the subspace spanned by the eigenvectors associated to the k smallest eigenvalues of the derivative covariance matrix of the data, which we define in the next Section.

4.2 The computational problem and the SFA Algorithm

Now we introduce the optimization problem in its most general form as it is commonly stated for classification [23]. Let $a = \sum_{k=1}^{K} \binom{|T_k|}{2}$. For all $j \in [K-1]$, minimize:

$$\Delta(y_j) = \frac{1}{a} \sum_{k=1}^{K} \sum_{\substack{s,t \in T_k \\ s<t}} (g_j(x_s) - g_j(x_t))^2$$

with the following constraints:

1. $\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in T_k} g_j(x_i) = 0$
2. $\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in T_k} g_j(x_i)^2 = 1$
3. $\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in T_k} g_j(x_i) g_v(x_i) = 0 \quad \forall v < j$

The minimization of the delta values $\Delta(y_j)$ encodes the requirement on the output signal to vary "as slow as possible", and thus the delta values are our measure of slowness. They are the average of the square of the first order derivative (over time) of the j-th component of the output signal y_t. The first requirement states that the average over time of each component of the signal should be zero, and it is stated just for convenience, such that the other two requirements take a simple form. The second requirement asks for the variance over time of each component of the signal to be 1. It is equivalent to saying that each signal should carry some information and avoid the trivial solution $g_j(\cdot) = 0$. The third requirement is to say that we want the signals to be decorrelated with each other. This also introduces an order, such that the first signal is the slowest, the second signal is the second slowest and so on. The first and the second constraint also avoid the trivial solution $y_i = 0$. Intuitively, the decorrelation constraint forces different functions g_j to encode different "aspects" of the input, maximizing the information carried by the output signal.

In order for the minimization problem to be computationally feasible at scale, the g_j's are restricted to be linear functions w_j such that the output signal becomes $y_i = [w_1^T x_i, \cdots w_{K-1}^T x_i]^T$ or else $Y = XW$, where $X \in \mathbb{R}^{n \times d}$ is the matrix with rows the input samples and $W \in \mathbb{R}^{d \times (K-1)}$ the matrix that maps the input matrix X into a lower dimensional output $Y \in \mathbb{R}^{n \times (K-1)}$. In case it is needed to capture nonlinear relations in the dataset, one performs a standard nonlinear polynomial expansion on the input data as

preprocessing. Usually, a polynomial expansion of degree 2 or 3 is chosen. For example we can take:

$$x = [x_1, x_2, x_3] \to \left[x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2, x_1, x_2, x_3\right].$$

The choice of the nonlinear expansion is important for using SFA in machine learning contexts. If it is a low dimensional expansion, it might not solve the task with high accuracy, while if the dimension is too high, it might overfit the training data, and therefore not generalize properly to test data. This technique also goes under the name of polynomial kernel.

We also need to satisfy the constraint on the average of the signal being zero and have unit variance. This is not a strong requirement, since it can be enforced beforehand, by preprocessing the dataset. This requires only linear time with respect to the dimensions of the data, and in practice consist in removing the mean and scale the components by their variance. Namely, we assume that the j-th component of the i-th vector in the dataset satisfies the condition:

$$(x_i)_j := \frac{(\tilde{x}_i)_j - E[(\tilde{x}_i)_j]}{\sqrt{E[((\tilde{x}_i)_j - E[(\tilde{x}_i)_j])^2]}},$$

where with $\tilde{x}(i)$ we define a raw signal with arbitrary mean and variance, $E[\tilde{x}_j(i)]$ the expected value of a single component of the vectors. This allows us to rewrite the minimization problem including the constraints of zero mean and unit variance. We can restate the definition of the delta function as the the GEP in Optimization Form 3.5:

$$\Delta(y_j) = \frac{w_j^T A w_j}{w_j^T B w_j}, \tag{4.1}$$

where the matrix B is called the sample covariance matrix and defined as:

$$B := \frac{1}{n}\sum_{i\in[n]} x_i x_i^T = X^T X \tag{4.2}$$

and the matrix A is called the sample derivative covariance matrix and defined as:

$$A := \frac{1}{a}\sum_{k=1}^{K}\sum_{\substack{i,i'\in T_k \\ i<i'}} (x_i - x_{i'})(x_i - x_{i'})^T = \frac{1}{a}\sum_{k=1}^{K} \dot{X_k}^T \dot{X_k} := \dot{X}^T\dot{X}. \tag{4.3}$$

Note also, that we can approximate the matrix A by subsampling from all possible pairs $(x_i, x_{i'})$ from each class and this is indeed what happens in practice.

4.2.1 Slowly varying signals

We formalize the concept of slowly varying signals. While this won't have any utility in the classical algorithm, it will allow us to bound nicely the runtime of the quantum algorithm, in the case when the data has "structure" that can be extracted by the SFA algorithm. In general, a slow signal is a signal that change slowly over time. This concept is formalized in the context of SFA by requiring that the whitened signal can be reconstructed without too much error from the projection on a few slow feature vectors. Formally, for a given K, and a set of slow feature vectors $w_1 \dots w_{K-1}$, we define a slowly varying signal as follows.

Definition 39 (Slowly varying signal)**.** *Let $X \in \mathbb{R}^{n\times d}$ and $Y \in [K]^n$ be a dataset and its labels. Let the rows x_i of X have whitened representation z_i. For the $K-1$ slow feature vectors $w_j, j \in [K]$, let y_i be the slow feature representation of x_i. We say that X is γ_K-slow if:*

$$\frac{\sum_{i=0}^{n} \|z_i\|}{\sum_{i=0}^{n} \|y_i\|} \leq \gamma_K$$

If we use SFA for doing dimensionality reduction in the context of supervised learning, a dataset is slowly varying if all the elements in the same class are similar to each other. In this way, by projecting the original images in the subspace spanned by a small number of slow feature vectors, we can reconstruct most of the information of the original dataset. We stress that this assumption is not needed for the quantum algorithm to work, but instead it will only be useful to give guarantees on the runtime of the algorithm.

4.2.2 The SFA algorithm

The SFA algorithm basically provides a solution to the generalized eigenvalue problem $AW = \Lambda BW$ and outputs the eigenvectors corresponding to the smallest eigenvalues. As we said we assume that the data has been normalized and polynomially expanded.

The first step of the algorithm is to whiten the data. This will reduce the problem into a normal eigenvalue problem; the second step is to perform PCA in order to find the eigenvalues and eigenvectors. We refer to [55] for a more complete description.

Step 1: Whitening the data

Recall that $X \in \mathbb{R}^{n\times d}, A, B \in \mathbb{R}^{d\times d}$. We now show how to whiten the data by right multiplying with the matrix $B^{-1/2} = [(X^T X)]^{-1/2}$. Then, the input matrix becomes $Z = XB^{-1/2}$ and the covariance matrix of the whitened data $Z^T Z$ is thus the identity.

Claim 40. *Let $Z := XB^{-1/2}$ be the matrix of whitened data. Then $Z^T Z = I$.*

Proof. Let $X = U\Sigma V^T$. We defined $B = V\Sigma^2 V^T$. As $Z = U\Sigma V^T (V\Sigma^{-1} V^T) = UIV$ It follows that $Z^T Z = I$. □

Step 2: Projection in slow feature space

The second step of SFA consists of outputting the $K-1$ "slowest" eigenvectors of the derivative covariance matrix of the whitened data $A := \dot{Z}^T \dot{Z}$, where $\dot{Z}$ is defined similar to $\dot{X}$ by using the whitened samples instead of the original ones. Note that taking the derivatives of the whitened data is equal to whitening the derivatives.

Claim 41. *Let $A = \dot{Z}^T \dot{Z}$. Then $A = (B^{-1/2})^T \dot{X}^T \dot{X} B^{-1/2}$.*

Proof.

$$\begin{aligned} A &= \dot{Z}^T \dot{Z} = \frac{1}{a} \sum_{k=1}^{K} \sum_{\substack{i,i' \in T_k \\ i<i'}} (z_i - z_{i'})(z_i - z_{i'})^T \\ &= (B^{-1/2})^T \frac{1}{a} \sum_{k=1}^{K} \sum_{\substack{i,i' \in T_k \\ i<i'}} (x_i - x_{i'})(x_i - x_{i'})^T B^{-1/2} \\ &= (B^{-1/2})^T \dot{X}^T \dot{X} B^{-1/2} \end{aligned}$$

□

This observation allow us to whiten the data with a quantum procedure. Recall that the matrix A is usually approximated with a small fraction of all the possible derivatives, roughly linear (and not quadratic) on the number of data points. In our case we take the number of rows of the derivative matrix to be just double the number of data points, and in the experiment we show that this does not compromise the accuracy.

SFA - Algorithm 1 (Classical) Slow Feature Analysis

Require:
Input $X \in \mathbb{R}^{n \times d}$ (normalized and polynomially expanded), and $K < d \in \mathbb{N}$

Ensure:
$Y = ZW$, where $Z = XB^{-1/2}$ is the whitened input signal, and $W \in \mathbb{R}^{d \times (K-1)}$ are the $K-1$ eigenvectors of the matrix $A = \dot{Z}^T \dot{Z}$ corresponding to the smallest eigenvalues

1: Whiten the signal: $Z := XB^{-1/2}$, and create $\dot{Z}$ from Z.
2: Perform PCA on the derivative covariance matrix $A = \dot{Z}^T \dot{Z}$ of the whitened data.
3: Return $Y = ZW$, the projection of whitened data onto W, the $K-1$ slowest eigenvectors of A

4.3 Quantum Slow Feature Analysis

We are finally ready to put forward a quantum procedure for SFA. Specifically, we want to map the input matrix X to the output matrix $Y = XW$, and then we will see how to estimate classically the slow feature vectors $W \in \mathbb{R}^{(K-1)\times d}$. For this, we assume to have quantum access to the matrices X and $\dot{X}$, as in definition 5. We start by describing a circuit that approximately performs the unitary $U_{QSFA} : |X\rangle \to |Y\rangle$ where $|X\rangle$ is the quantum state we obtain by having quantum access to X, the dataset, and $|Y\rangle := \frac{1}{\|Y\|_F} \sum_{i=0}^{n} \|y_i\| \, |i\rangle \, |y_i\rangle$. As the classical algorithm, QSFA is divided in two parts. In the first step we whiten the data, i.e. we map the state $|X\rangle$ to the state $|Z\rangle = |XB^{-1/2}\rangle$, and in the second step we approximately project $|Z\rangle$ onto the subspace spanned by the smallest eigenvectors of the whitened derivative covariance matrix $A = \dot{Z}^T \dot{Z}$.

Step 1: Whitening the data Recall that $X = \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{n \times d}$, and $A, B \in \mathbb{R}^{d \times d}$. We now show how to whiten the data having quantum access to the matrix X. As $B^{-1/2}$ is a symmetric matrix with eigenvectors the column singular vectors of X and eigenvalues equal to $1/|\sigma_i|$. Using quantum linear algebra procedure, i.e. theorem 28, we can multiply with $B^{-1/2}$ our state $|X\rangle$. Thus, we have the following corollary.

Corollary 42 (Whitening algorithm)**.** *Assume to have quantum access to $X = \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{n \times d}$, as in theorem 8. Let $Z = XB^{-1/2}$ the matrix of whitened data. There exists a quantum algorithm that produces as output a state $|\bar{Z}\rangle$ such that $\| |\bar{Z}\rangle - |Z\rangle \| \leq \varepsilon$ in expected time $\tilde{O}(\kappa(X)\mu(X) \log 1/\varepsilon))$.*

Step 2: Projection in slow feature space The previous Corollary gives a way to build quantum access to the rows of the whitened matrix Z, up to some error ϵ. Now we want to project this state onto the subspace spanned by the eigenvectors associated to the $K-1$ "slowest" eigenvectors of the whitened derivative covariance matrix $A := \dot{Z}^T \dot{Z}$, where $\dot{Z}$ is the whitened derivative matrix $\dot{Z} = \dot{X}B^{-1/2}$. Let θ a threshold value and δ a precision parameter, that governs the error we tolerate in the projection threshold. Recall that $A_{\leq \theta, \delta}$ we denote a projection of the matrix A onto the vector subspace spanned by the union of the singular vectors associated to singular values that are smaller than θ and some subset of singular vectors whose corresponding singular values are in the interval $[\theta, (1+\delta)\theta]$.

To perform the projection, we will need a threshold for the eigenvalues that will give us the subspace of the $K-1$ slowest eigenvectors. A priori, we don't know the appropriate threshold value, and thus it must be found experimentally through binary search since it depends on the distribution of singular values of the matrix representing the dataset. We can now describe and analyse the entire QSFA algorithm.

As in the previous section, we note that the eigenvalues of $A_{\dot{Z}}$ are the squares of the singular values of $\dot{Z}$, and the two matrices share the same column space: $\dot{Z} = U\Sigma V^T$, and $A_{\dot{Z}} = V\Sigma^2 V^T$. Claim 41 tells us that whitening the derivatives of the signal is equal to taking the derivatives of the whitened data. theorem 29 provides exactly the procedure for accessing the rows of $\dot{Z}$, since we know how to multiply with $\dot{X}$ and with $B^{-1/2}$.

QSFA 2 Quantum Slow Feature Analysis

Require:
Quantum access to matrices $X \in \mathbb{R}^{n\times d}$ and $\dot{X} \in \mathbb{R}^{n\times d}$, parameters $\epsilon, \theta, \delta, \eta > 0$.
Ensure:
A state $|\overline{Y}\rangle$ such that $\| |Y\rangle - |\overline{Y}\rangle \| \leq \epsilon$, with $Y = A^+_{\leq\theta,\delta} A_{\leq\theta,\delta} Z$

1: Create the state $|X\rangle := \frac{1}{\|X\|_F} \sum_{i=1}^n \|x_i\| \, |i\rangle \, |x_i\rangle$
2: (Whitening algorithm) Map $|X\rangle$ to $|\overline{Z}\rangle$ with $\| |\overline{Z}\rangle - |Z\rangle \| \leq \epsilon$ and $Z = XB^{-1/2}$.
3: (Projection in slow feature space) Use theorem 28 project $|\overline{Z}\rangle$ onto the slow eigenspace of A using threshold θ and precision δ (i.e. $A^+_{\leq\theta,\delta} A_{\leq\theta,\delta} \overline{Z}$)
4: Perform amplitude amplification and estimation on the register $|0\rangle$ with the unitary U implementing steps 1 to 3, to obtain $|\overline{Y}\rangle$ with $\| |\overline{Y}\rangle - |Y\rangle \| \leq \epsilon$ and an estimator $\overline{\|Y\|}$ with multiplicative error η.

We conclude this section by stating the main dimensionality reduction theorem of this paper.

Theorem 43 (QSFA algorithm)**.** *Assume to have quantum access to $X = \sum_i \sigma_i u_i v_i^T \in \mathbb{R}^{n\times d}$ and its derivative matrix $\dot{X} \in \mathbb{R}^{n \log n \times d}$. Let $\epsilon, \theta, \delta, \eta > 0$. There exists a quantum algorithm that produces as output a state $|\overline{Y}\rangle$ with $\| |\overline{Y}\rangle - |A^+_{\leq\theta,\delta} A_{\leq\theta,\delta} Z\rangle \| \leq \epsilon$ in time*

$$\tilde{O}\left(\frac{(\kappa(X) + \kappa(\dot{X}))(\mu(X) + \mu(\dot{X}))}{\delta\theta} \gamma_{K-1} \right)$$

and an estimator $\overline{\|Y\|}$ with $|\overline{\|Y\|} - \|Y\| \,| \leq \eta \|Y\|$ with an additional $1/\eta$ factor.

Proof. QSFA consists of two steps. The first step is the whitening, which can be performed in time $\tilde{O}(\kappa(X)\mu(X)\log(1/\epsilon))$ and provide the state $|\overline{Z}\rangle$ using Corollary 42. It is simple to verify that creating a state $|Z\rangle$ of whitened data such that $Z^T Z = I$ can be done using quantum access just to the matrix X, as $Z = XB^{-1/2}$. The second step is the projection of whitened data in the slow feature space, which is spanned by the eigenvectors of $A = \dot{Z}^T \dot{Z}$. This matrix shares the same right eigenvectors of $\dot{X}B^{-1/2}$, which is simple to check that we can efficiently access using the QRAM constructions of X and $\dot{X}$. Using the algorithm for quantum linear algebra, i.e. theorem 29, we know that the projection (without the amplitude amplification) takes time equal to the ratio $\mu(X) + \mu(\dot{X})$ over the threshold parameter, in other words it takes time $\tilde{O}(\frac{(\mu(X)+\mu(\dot{X})}{\delta\theta})$. Finally, the amplitude amplification and estimation depends on the size of the projection of $|\overline{Z}\rangle$ onto the slow eigenspace of A, more precisely it corresponds to the factor $O(\frac{\|\overline{Z}\|}{\left\|A^+_{\leq\theta,\kappa} A_{\leq\theta,\kappa} \overline{Z}\right\|})$, which is roughly the same if we look at Z instead of $\overline{Z}$. Note also that Z is the whitened data, which means that each whitened vector should look roughly the same on each direction. This implies that the ratio should be proportional to the ratio of the dimension of the whitened data over the dimension of the output signal. The final runtime of the algorithm is:

$$\tilde{O}\left(\left(\kappa(X)\mu(X)\log(1/\varepsilon) + \frac{(\kappa(X) + \kappa(\dot{X}))(\mu(X) + \mu(\dot{X}))}{\delta\theta} \right) \frac{\|Z\|}{\left\|A^+_{\leq\theta,\delta} A_{\leq\theta,\delta} Z\right\|} \right)$$

Note that the last ratio in this runtime was defined as γ_{K-1} in definition 39. From this, the runtime in the statement of the theorem follows.

□

We will see in the following sections that in the case of the MNIST dataset the projection ratio γ is small enough.

There are cases where we want to use the quantum computer to estimate the model fitted by SFA. For this, we would like the quantum computer to create a quantum state proportional to the slow feature vectors, and then retrieve them using quantum tomography. Assuming to use a tomography with ℓ_2 guarantees, we have the following corollary:

Corollary 44 (Fitting SFA model)**.** *Assume to have quantum access to dataset $X \in \mathbb{R}^{n\times d}$ and its derivative matrix $\dot{X} \in \mathbb{R}^{n\log n\times d}$, and let $\epsilon > 0$. There exists a quantum algorithm that estimates the SFA model $\{w_j\}_{j=0}^{K}$ such that $|w_j - \overline{w_j}| \leq \epsilon$ with high probability, in time:*

$$O(d^{1.5}\sqrt{K}\frac{\kappa(X)\kappa(\dot{X}X)(\mu(X)+\mu(\dot{X})}{\epsilon^2})$$

Proof. Recall that the model of slow feature analysis is obtained as the eigenvectors associated to the smallest eigenvalues of the symmetric matrix $A = B^{-1/2}\dot{X}^T\dot{X}B^{-1/2}$. The matrix of eigenvectors is applied on the whitened data after the conditions of zero mean and unit variance have been satisfied. Thus, in order to extract the SFA model, we rely on lemma 33, that is, first we extract the eigenvectors of A associated to its smallest eigenvalues, and then we multiply these resulting vectors by $B^{-1/2}$ for the whitening part. More precisely, a vector in the slow-feature space is defined as $y_i = Mz_i$, where M is the $(K-1\times d)$ matrix of eigenvectors of the whitened derivative covariance matrix A, and z_i is the whitened vector $B^{-1/2}x_i$. The model is thus obtained by the matrix of eigenvectors $M \in \mathbb{R}^{(K-1)\times d}$, right multiplied by $B^{-1/2}$. Further observe that the matrices $X, \dot{X}, B, A$ share the same row space.

We start by using quantum access to X and quantum linear algebra to create a state proportional to the uniform superposition of its left and right singular vectors, analogously to the first whitening step of QSFA. More formally, we use quantum linear algebra to multiply the second register by $X^\dagger$, and perform this mapping:

$$|X\rangle = \frac{1}{\|X\|_F}\sum_i^n \|x_i\||i\rangle|x_i\rangle = \frac{1}{\|X\|_F}\sum_i^d \sigma_i(X)|v_i\rangle|u_i\rangle \mapsto \frac{1}{\sqrt{n}}\sum_i^d |v_i\rangle|u_i\rangle$$

Then, we perform singular value estimation with error ϵ_1 on the product of two matrices $\dot{X}B^{-1/2}$, which can be done either using our singular value estimation of product of matrices, i.e. lemma 22, or [38, lemma 27], which does not depend on the condition number of the matrices, and thus is faster. As noted before in this thesis, these singular values of this matrix are the square root of the singular values of the whitened derivative matrix A, and here we denote them $\sigma_i(\sqrt{A})$. Thus, we obtain the following state:

$$\frac{1}{\sqrt{d}}\sum_i^d |u_i\rangle|v_i\rangle|\sigma_i(\sqrt{A})\rangle$$

We project the previous state on the subspace spanned by the singular vectors that correspond to singular values smaller than a certain threshold θ, (which can be found by binary search), such that $|\{\sigma_i|\sigma_i \leq \theta\}| \approx K-1$. Using amplitude amplification this operation can be performed occurring in a multiplicative cost of $\frac{1}{\sqrt{K/d}}$. Let ϵ_1 be the

precision in singular value estimation, then the total runtime is of

$$\tilde{O}((\kappa(X)\mu(X)) + \sqrt{\frac{d}{K}}\frac{\mu(X)+\mu(\dot{X})}{\epsilon_1})$$

The resulting state is

$$\frac{1}{\sqrt{K-1}}\sum_{i}^{K-1}|u_i\rangle|v_i\rangle|\sigma_i(\sqrt{A})\rangle \tag{4.4}$$

The runtime of this procedure is:

$$O\left(\kappa(X)\mu(X)\log(1/\epsilon_{mult}) + \sqrt{\frac{d}{K}}\frac{(\mu(X)+\mu(\dot{X}))}{\epsilon_1}(a+T_U)(\log(1/\Delta)\right) \tag{4.5}$$

The addend on the left is the runtime of the matrix multiplication step, needed to create the uniform superposition of singular vectors, and the second addend comes from the singular value estimation procedure of [38, lemma 27]. Here a is the number of ancilla qubit used to obtain the block encoding of the matrix, Δ is the failure probability, and T_U is the time needed to implement the block encoding of the matrix $\dot{X}X$, which consist in using sequentially the quantum access to the two matrices. Using quantum access to the matrices, we consider T_U to be polylogarithmic.

To conclude the algorithm, we recall that thanks to lemma 33, we can multiply the central register by $B^{-1/2}$, thus obtaining a state which stores the representation of the model $w_i{}_i^K$.

$$\frac{1}{\sqrt{K-1}}\sum_{i}^{K-1}|u_i\rangle|w_i\rangle|\sigma_i(\sqrt{A})\rangle$$

As we know from quantum linear algebra, i.e. theorem 28 the runtime for obtaining the previous state is $\kappa(X)(\mu(X)+T_\chi)$ where T_χ is the runtime needed to produce the state in Equation 4.4, so this runtimes introduces a dependence of $\kappa(X)$ in the runtime in equation 4.5. To retrieve the model, we can now sample the singular values σ_i from the last register, and obtain one of the K vectors of the model w_i with uniform probability. Each of these vectors can be used in a ℓ_2 tomography, with precision ϵ. Note that each of the samples of the third register will give a valid state to perform tomography on, so we do not incur any polynomial overhead in this sampling. Note the initial step of the algorithm for obtaining the uniform superposition of the left and right singular vectors is negligible compared to the rest of the algorithm. We pick ϵ_1 small enough to be able to distinguish the smallest singular values of $\dot{X}B^{-1/2}$, i.e. we pick $\epsilon_1 \leq 1/2\kappa(A)$, thus introducing dependence in the runtime of $\kappa(\dot{X}X)$.

The cost of the ℓ_2 tomography, which we repeat on the K states that represent the vectors of the model introduce a dependence of $O\left(K\frac{d}{\epsilon^2}T\right)$, where T is the computational cost of producing the state.

$$O\left(K\frac{d}{\epsilon^2}\left(\kappa^2(X)\mu(X)\log(1/\epsilon_{mult}) + \sqrt{\frac{d}{K}}\kappa(X)\kappa(\dot{X}X)(\mu(X)+\mu(\dot{X}))\log(1/\Delta)\right)\right)$$

The final runtime of this algorithm is thus the one in the statement of the theorem.

□

4.4 Quantum Frobenius Distance Classifier

In this Section we define the quantum Frobenius Distance (QFD) classification, a novel quantum method that classifies a new data point based on the average square ℓ_2-distance between the new data point and each labeled cluster, i.e. the average of the square distances between the new data point and each point in the cluster. The running time is logarithmic in the number and dimension of the data points and proceeds by creating a weighted superposition of all points in a cluster and the new data point, and then estimating the distance of each point with the new data point in superposition, hence allowing to estimate the average square distance. Quantum Frobenius Distance Classifier has running time that is logarithmic in the dimension and number of data points. It assumes quantum access to the dataset, or that the data comes directly from some quantum process. The classification algorithm assigns a test point $x(0)$ to the cluster k whose points have minimum normalized average squared ℓ_2 distance to $x(0)$.

Let X_k be defined as the matrix whose rows are the vectors corresponding to the k-th cluster, and $|T_k|$ is the number of elements in the cluster. For the test point $x(0)$, define the matrix $X(0) \in \mathbb{R}^{|T_k| \times d}$ which just repeats the row $x(0)$ $|T_k|$ times. Then, we define

$$F_k(x(0)) = \frac{\|X_k - X(0)\|_F^2}{2(\|X_k\|_F^2 + \|X(0)\|_F^2)},$$

which corresponds to the average normalized squared distance between $x(0)$ and the cluster k. Let $h : \mathcal{X} \to [K]$ our classification function. We assign to $x(0)$ a label according to the following rule:

$$h(x(0)) := \arg\min_{k \in [K]} F_k(x(0)) \tag{4.6}$$

We will estimate $F_k(x(0))$ efficiently using the algorithm below. Using quantum access to the data points we can create a superposition of all vectors in the cluster as quantum states, have access to their norms and to the total number of points, and norm of the cluster. We define a normalizing factor $N_k = \|X_k\|_F^2 + \|X(0)\|_F^2 = \|X_k\|_F^2 + |T_k| \|x(0)\|^2$. First we explain how to compute a single Frobenius Distance $F_k(x(0))$ in Algorithm 3, and then the classification procedure follows trivially, and is described in Algorithm 4.

For the analysis, just note that the probability of measuring $|1\rangle$ in Equation 4.7 is:

$$\frac{1}{2N_k}\left(|T_k| \|x(0)\|^2 + \sum_{i \in T_k} \|x(i)\|^2 - 2 \sum_{i \in T_k} \langle x(0), x(i) \rangle\right) = F_k(x(0)).$$

By Hoeffding bounds, to estimate $F_k(x(0))$ with error η we would need $O(\frac{1}{\eta^2})$ samples. For the running time, we assume all unitaries are efficient either because the quantum states can be prepared directly by some quantum procedure or given that the classical vectors are stored in the QRAM as described in Appendix 8, hence the algorithm runs in time $\tilde{O}(\frac{1}{\eta^2})$. We can of course use amplitude estimation and save a factor of η. Depending on the application one may prefer to keep the quantum part of the classifier as simple as possible or optimize the running time by performing amplitude estimation.

Given this estimator we can now define the QFD classifier.

QFDC 4 Quantum Frobenius Distance classifier

Require:
Quantum access to K matrices X_k of elements of different classes. A test vector $x(0)$. Error parameter $\eta > 0$.

Ensure:
A label for $x(0)$.

1: **for** $k \in [K]$ **do**
2: Use the QFE algorithm to compute $F_k(x(0))$ on X_k and $x(0)$ with precision η.
3: **end for**
4: Output $h(x(0)) = \arg\min_{k\in[K]} F_k(x(0))$.

The running time of the classifier can be made $\tilde{O}(\frac{K}{\eta})$ when using amplitude amplification. We will see that in fact η does not have to be very small in order to classify correctly the dataset where the clusters are nicely separated (as we will see from the experiments). Specifically, we will see that for the MNIST dataset, after the dimensionality reduction we can take $\eta = 1/10$, since the clusters are pretty well separated.

4.5 Experiments

The MNIST dataset, is a commonly used benchmark to test the validity of newly proposed classifiers. Classical classification techniques can achieve around 98-99% accuracy, with neural network solutions exceeding 99% (the MNIST dataset is quite simple and this is why it is often used as a first benchmark). As previously introduced in the main text, classical SFA has also been applied on the same dataset with accuracy 98.5%, with initial PCA of dimension 35 and polynomial expansion of degree 3, and we will closely follow that classification procedure. Our goal is to study a quantum classifier with two properties: very good accuracy and efficiency. We detail our quantum classifier by going through the three parts in any classical classifier: preprocessing, training, and testing. The error in the whitening procedure has been simulated by adding noise from a truncated Gaussian distribution centered on each singular value with unit variance. For the error in the projection part, this comes only from potentially projecting on a different space than the one wanted. By taking the right θ (around 0.3 or 0.05 depending on the polynomial expansion) and a small enough δ (around 1/20) for the error, we guarantee in practice that the projection is indeed on the smallest $K-1$ eigenvectors.

Data preprocessing. The MNIST dataset is composed of $n = 60000$ images in the training set and 10000 images in the test set, where each sample is a black and white image of a handwritten digit of 28×28 pixels. The methodology is as follows: first, the dimension of the images is reduced with a PCA to something like 35 (or around 90, depending on the polynomial expansion degree we use in the following step). Fortunately, efficient incremental algorithms for PCA exist, where it is not required to fully diagonalize a covariance matrix, and the running time depends on the number of dimensions required as output. Second, a polynomial expansion of degree 2 or 3 is applied, hence making the dimension up to 10^4. Third, the data is normalized so as to satisfy the SFA requirements of zero mean and unit variance. Overall, the preprocessing stage creates around $n = 10^5$ vectors $x(i), i \in [n]$ of size roughly $d = 10^4$ and the running time of the preprocessing is of the order of $\tilde{O}(nd)$, with $nd \approx 10^9$. With a real quantum computer we would add a further step, which is to load the preprocessed data in the QRAM. This take only one pass over the data, and creates a data structure (i.e. a circuit) which is linear in the size of the data. Hence, the overall preprocessing takes time of $\tilde{O}(nd)$.

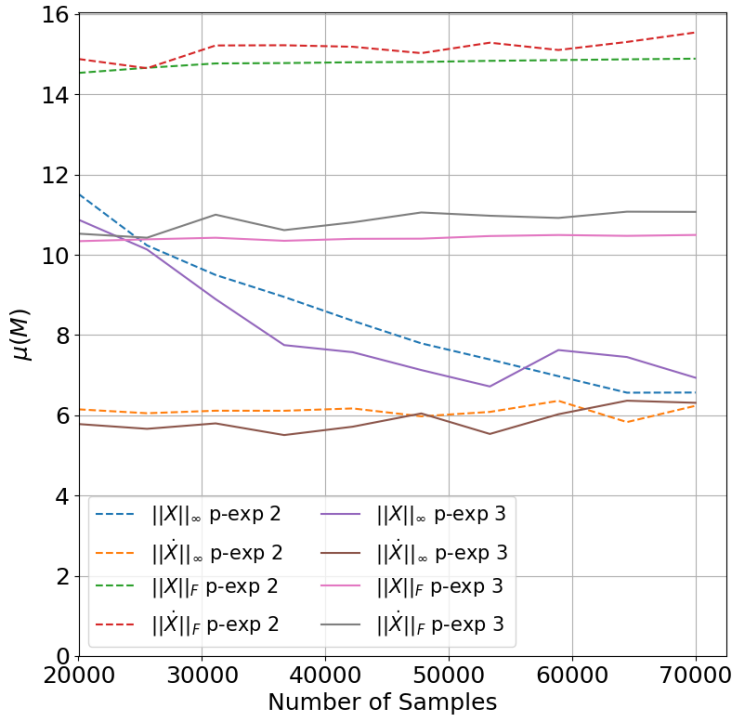


Figure 4.1: Sensitivity analysis of the parameter μ for the matrices X and $\dot{X}$ while increasing n. The graph show the value of the Frobenius norm and the max ℓ_1-norm as two options for μ. The two upper dotted lines represent the Frobenius norm of X and $\dot{X}$ for polynomial expansion of degree 2. The thick horizontal lines above 10 shows the trend of the Frobenius norm for the polynomial expansion of degree 3. For the MNIST dataset, we see that both the Frobenius norm and the maximum ℓ_1 norm are practically constant when we increase the number of data points in the training set.

Training. The classical SFA procedure outputs a small number $(K-1)$ of "slow" eigenvectors of the derivative covariance matrix, where K is the number of different classes, and here $K = 10$. This is in fact the bottleneck for classical algorithms and this is why the dimension was kept below $d = 10^4$ with polynomial expansion, which still requires intensive HPC calculations. Generically, the running time is between quadratic and cubic, and hence of the order 10^{13}. Once these eigenvectors are found, each data point is projected onto this subspace to provide n vectors of $(K-1)$ dimensions which are stored in memory. As the points are labelled, we can find the centroid of each cluster. Note that at the end, the quantum procedure does not output a classical description of the eigenvectors, neither does it compute all vectors $y(i)$, as the classical counterpart could do. Nevertheless, given a quantum state $|x(i)\rangle$ it can produce the quantum state $|y(i)\rangle$ with high probability and accuracy.

Testing. For the testing stage, the classifier trained with the errors, is used to classify the 10000 images in the test set. Classically, the testing works as following: one projects the test data point $x(0)$ onto the slow feature space, to get a $(K-1)$-dimensional vector $y(0)$. Then, a classification algorithm is performed, for example kNN (i.e. one finds the k closest neighbours of $y(0)$ and assigns the label that appears the majority of times). The

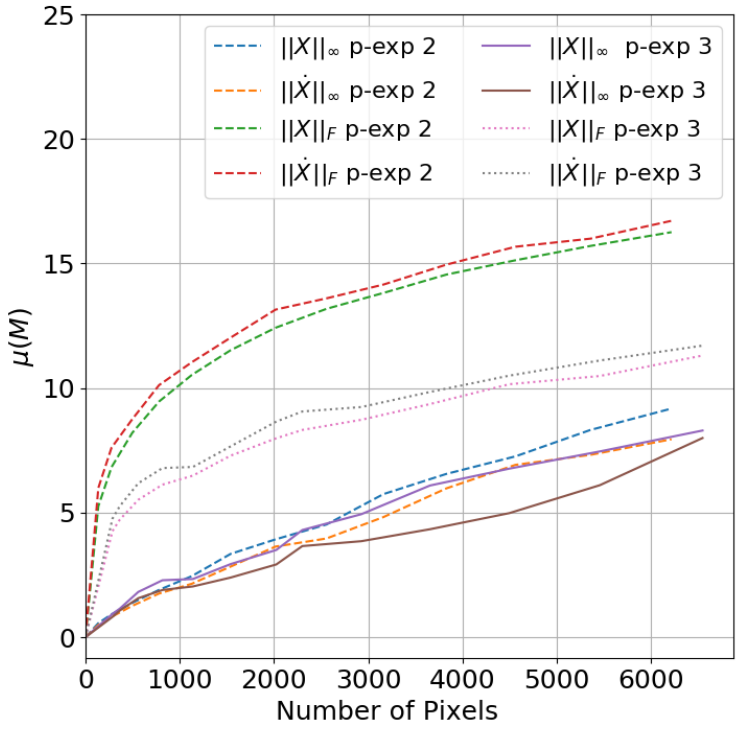


Figure 4.2: Sensitivity analysis of the parameter μ for the matrices X and $\dot{X}$ while increasing d. The graph show the value of the Frobenius norm and the max ℓ_1-norm (i.e. $\|\cdot\|_\infty$) as two options for μ. When we increase the dimension of the data points by changing the PCA dimension, before the polynomial expansion. In general, the maximum ℓ_1 norms are always upper bounded by the Frobenius norms. The dashed lines are for the polynomial expansion of degree 2. The bottom thick lines represent the maximum ℓ_1 norm for the polynomial expansion of degree 3, and the dotted central line for the Frobenius norm of the same polynomial expansion.

complexity of this step is $O(Kd)$ for the projection of the test vector, plus the time of the classification in the slow feature space. The kNN algorithm, for example, is linear in the number of data points times the dimension of the points ($\tilde{O}(nd)$). In Nearest Centroid algorithm (a supervised classification algorithm where the label of a new point is assigned to the cluster with closest barycenter), if in the training stage we have found the centroids, then classification can be done in time $\tilde{O}(Kd)$. In our final algorithm we perform the training and testing together, i.e. using QSFA and QFD together.

Parameters of experiment. We now estimate a number of parameters appearing in the running time of the quantum classifier.

Number and dimension of data points. For the MNIST we have that nd is of the order of 10^9 (including data points and derivative points).
The parameter μ for the matrices X and $\dot{X}$. We analyze the parameter $\mu(x)$, $\mu(\dot{X})$ as the number of data points in the training set and the dimension of the input vectors increases (PCA dimension + polynomial expansion). We know that μ is bounded by the Frobenius norm of the matrix. We also look at the case where μ is defined as the maximum l_1 norm

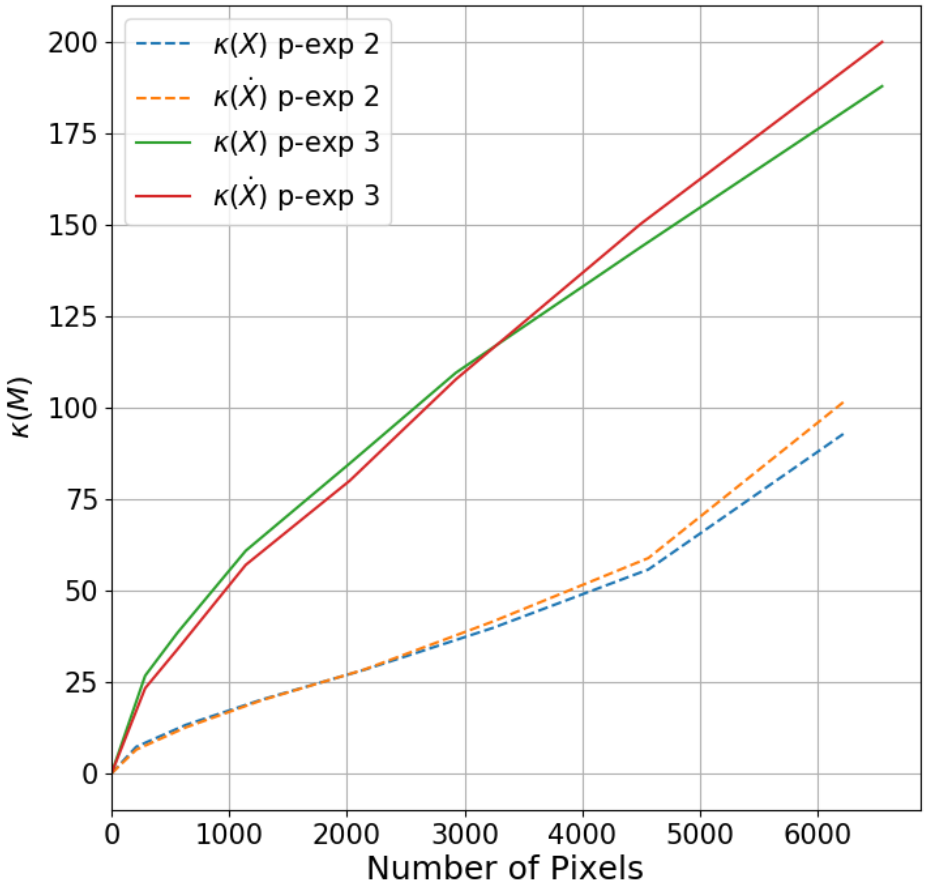


Figure 4.3: Sensitivity analysis of the condition number of X and $\dot{X}$ while increasing d, the number of pixels, and discarding the 0 singular values. We plot the condition number of a polynomial expansion of degree 2 (two dashed lines in the bottom) and degree 3 (upper lines).

of the rows of the matrices, plotted in Figure 4.1. Matrices are normalized to have spectral norm 1. The good choice of μ is practically constant as we increase the number of points in the dataset. All the l_1 norms in the experiment were less than 11. We also plot the Frobenius norm and the maximum l_1 norm as the dimension of the vectors in the dataset increases. While the Frobenius norm somewhat increases with the dimension, the maximum l_1 norm remains stable. This could be expected since in the preprocessing a PCA is done, making the input matrices in fact quite low rank. Indeed, after the polynomial expansion the Frobenius norm does not increase much since we only add higher order terms, and note that all entries of the matrices are smaller than 1, since $\|X\|_{max} \leq \|X\|_2 \leq 1$. On the other hand, the scaling and normalization of X helps keeping the l_1 norm even lower. It is important to state here that one gains a factor 10^3 just by taking the correct quantum algorithm for performing linear algebra and not an off-the-shelf one. Such decisions will be crucial in reaching the real potential of quantum computing for machine learning applications.

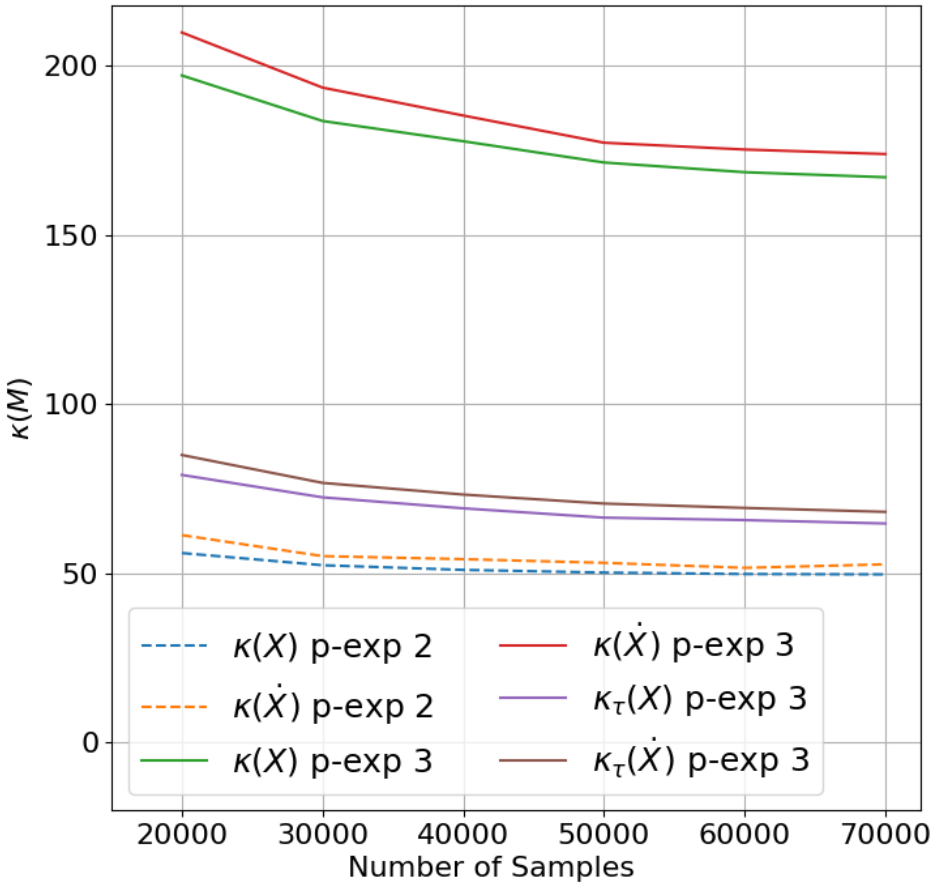


Figure 4.4: Sensitivity analysis of the condition number of X and $\dot{X}$ while increasing the size of the training set. We plot the condition number after discarding the 0 singular values. The two central lines represent the condition number for a polynomial expansion of degree 3 while discarding the bottom 1% of the smallest singular values.

Condition number for the matrix X. Figure 4.3 and 4.4 tells us that condition number is rather stable, in fact decreasing. As explained, we do not need to have the real condition number in the running time but a threshold under which we ignore the smaller eigenvalues. In fact, retaining just 99.5% of the singular values does not considerably penalize the accuracy and achieves a behavior of growing much more slowly as we increase the dimension, with a value around 10^2.

Error parameters. There are four error parameters, ε for the matrix multiplication procedure, δ and θ for the projection procedure, and η for the estimate of the norms in the classification. For ε, it appears only within a logarithm in the running time, so we can take it to be rather small. For the projection, we took $\delta \approx 1/20$ and from the simulations we have that $\theta \approx 0.3$ for polynomial expansion of degree 2 and $\theta \approx 0.05$ for polynomial expansion of degree 3. Last, it is enough to take $\eta = 1/10$. Note also, that these parame-

Table 4.1: Accuracy of relevant experiments for various combination of classifiers and polynomial expansion. Here we have chosen $\epsilon/\kappa(X) = 10^{-7}, \delta = 0.054, \eta = 1/10$ and 10.000 derivatives per class.

$QSFA_2$											
d (PCA)	$\|X\|$	$\|\dot{X}\|$	$\|x(i)\|_1$	$\|\dot{x}(i)\|_1$	$\kappa(X)$	$\kappa(\dot{X})$	$\kappa_t(X)$	$\kappa_t(\dot{X})$	θ	$\%_t$	%
860 (40)	22	102	0.9	1.4	41	22	32	15	0.38	96.4	96.4
3220 (80)	41	197	2.7	3.8	119	61	65	30	0.32	97.3	97.4
4185 (90)	46	215	3.1	4.4	143	74	72	35	0.31	97.4	97.4
$QSFA_3$											
d (PCA)	$\|X\|$	$\|\dot{X}\|$	$\|x(i)\|_1$	$\|\dot{x}(i)\|_1$	$\kappa(X)$	$\kappa(\dot{X})$	$\kappa_t(X)$	$\kappa_t(\dot{X})$	θ	$\%_t$	%
5455 (30)	73	81	5.9	4.3	276	278	149	149	0.06	98.2	98.3
8435 (35)	96	102	7.8	5.7	369	389	146	156	0.05	97.5	98.4
9138 (36)	101	108	8.0	5.3	388	412	149	159	0.04	97.7	98.5

ters are pretty stable when increasing the dimension, as they only depend on whether we perform a polynomial expansion of 2 or 3.

Projection ratio. The ratio between the norm of the vectors in the whitened space over the projected vector in the slow feature space is well bounded. For an initial PCA dimension of 40 and polynomial expansion of degree 2, this ratio is 10 with variance 0.0022, while for a polynomial expansion of degree 3 and a PCA dimension of 30 is 20 with 0.0007 variance.

Table 4.1 provides the exact values of all parameters in the experiment.

4.6 Detection of DGA domains with QSFA

In this section we describe how a quantum machine learning algorithm can be used to increase the accuracy of a classifier for anomaly detection, in a machine learning problem which appears in cybersecurity.

The totality of the malware that exists nowadays is shipped with a DGA (Domain Generating Algorithm: an algorithm that generates random domain names). The reason for these algorithms to exists is connected to the way an infected machine communicates with its Command & Control server (C&C). A C&C is a machine to which all infected computers connect to, and sends commands to be executed on the infected machines. If a malware is shipped with a predefined list of IP addresses or domains name to contact, this information can be easily extracted by reverse engineering the malware, and thus blocked promptly at internet-wide level by the Internet Service Providers. Therefore, the DNS name of the C&C server needs to be hidden inside the malware with great care by the authors of the malware. One possible strategy is to use a randomized algorithm that generates a list of possible candidates DNS names based on a pre-shared seed of randomness between the infected machines and the C&C server. Each day the C&C server will use the DGA to create a new domain, and then it will register it, i.e. allowing a correct name resolution from the DGA domain to the IP address of the C&C server. That day, the infected machines, will query some randomly generated domain names, until they find the domain registered by the C&C server. The choice of the domain leverages randomness from the scurrent date and the pre-shared randomness. While most of the DNS queries won't be successful (the DNS server will issue a NXDomain response from the server), one query will resolve to the machine owned by the attacker, thus allowing infected machines to connect to the C&C. The malware can now receive new commands to execute on the infected machines [11]. Being able to detect DGA domains by analyzing DNS traffic is desirable for network administrators because the presence of a DGA domain name in the DNS queries is a strong indicator of network compromise. The problem of discriminating

between human generated domains and DGA has been vastly studied in literature using machine learning. [45, 147].

We remark that in the following experiment we only assume that the information available to us is the domain name presented in the queries. More realistic models should encompass more features, like information related on the timing of the requests, and historical information on the sender of the query (like the IP address of computer issuing the query), similarity between queries received by the DNS server in a certain amount of time, and so on. As done in literature, we limit our analysis in favour of a simpler model, as our aim is just to show how to use QSFA to improve the performances of a classifier in anomaly detection. We believe that the using more features, this will lead to further improvements in the performances of an anomaly detection system.

4.6.1 The experiment

A common way to work with DGA names for machine learning analysis comes from the Natural Language Processing domain. In NLP is common to map strings of text in a vector space, in a way that distances between vectors can be used as a proxy for similarity between strings. A technique used to map text documents in a vector space is called vectorization: a procedure that converts a collection of documents (or strings) to a matrix of token counts. In our case, the documents are represented by domain names, and the tokens counts are the presence of the n-grams in a domain. An n-gram is just a sequence of n characters from the domain name. Therefore, each document is mapped to a binary (sparse) vector v, whose v_i component is 1 if the domain contain the i-th n-gram. The length of the vector is given by the number of the n-grams extracted from the domain names in the training set. To create the list of n-grams, we use a list of valid (i.e. human generated) domain names. Once we have the list of n-grams in the dataset, we are ready to create our feature matrix, used in QSFA algorithm. To create the input matrices for QSFA we put together a list of human generated domain names, along with a list of DGA domains. Then, we perform vectorization on all these domains, using the n-grams extracted from the dataset of human generated domains. This represents the matrix X in the SFA algorithm. The supervised information consists in knowing if a domain is a DGA or not. From X, we take pairwise differences of domains that belong to the same class, and we form the matrix $\dot{X}$. Using classical SFA we extract a single slow-feature vector (remember that the number of slow feature vectors is $K-1$ where K is the number of classes, which is 2 for binary classification).

This approach represents a novel way of performing anomaly detection using a token matrix, that previously was not possible using classical computers. As the size of this matrix can considerably exceed the size that allows us to perform linear algebraic operations (i.e. algorithms whose runtime is polynomial in the size of the matrix), we are restricted to performing only operations that are linear in the number of columns and rows. The usual way to use the token matrix in this case, is to compute, for each domain, a score value. This score consists simply in the logarithm of the count of 1 entries in the vectorized domain. For example, if a domain contains k n-grams that have been extracted from the list of human generated domains, its score is $\log(k)$. Ideally, DGA domains should not have a high score, while all the n-grams in a human generated domain will result in a "1" entry in its vectorized form. This score, is then used in other ML algorithms as an additional feature.

In this experiment, the training set used to generate the matrix of tokens counts consist in the famous Alexa list of top 1 million web sites. The list of DGA domains has been obtained from [3], which has a repository of the source code of the domain generation algorithms used in real implementation of famous malware. From the Alexa list, we extracted all the n-grams for $n \in \{3, 4, 5\}$. Thus, for a dataset of n domains, this tokenization produces a matrix $M \in \{0, 1\}^{n \times d}$, where d is the number of distinct n-grams present in the training set. As the original dataset could not fit the constraint of the hardware (

an HPC architecture consisting in 384 cores and 7 Tb of memory), in order to make the problem tractable we reduced the dataset by picking domains containing only the top 15 most frequent letters in the list of benign domains. In this way, we were able to reduce the tokenization matrix to have size $n = 60000$ and $d = 30000$, where n is the number of selected domains and d the number of n-grams.

Using the feature extracted using the slow feature vector, along with entropy and length, we were able to improve the accuracy of 3 different classifiers, obtaining the following results:

- Logistic Regression classifier from 89% to 90.5% (+1.5%)
- Naive Bayes classifier from 89.3% to 92.3% (+3%)
- Decision Trees classifier from 91.4% to 94.0% (+2.6%)

In conclusion, this is an example of how quantum computation is not only capable of solving faster (or better) previous problem already tackled by classical computer, but is also capable of unlocking new ways of processing data that previously were unavailable (such as performing linear algebraic operations on the token matrix). This experiment highlights the importance of quantum computers in machine learning. Using a fault tolerant quantum computer, along with quantum access to the data, it is possible to extract the slow feature vectors of models whose dimension is currently not tractable with classical computers. In fact, just to run this experiment on a dataset of reduced dimensionality (i.e. taking only domain names containing the top 15 most frequent letters in the dataset) we used more than 2 hours of HPC calculations. I expect that quantum computer will be able, not only to speed up machine learning task that we are already able to solve, but also offer new and better ways of solving problems that we now solve in a sub-optimal way.

4.7 Conclusions

In this chapter we analyzed some applications of quantum linear algebra to propose the quantum version of a fundamental algorithm for dimensionality reduction in supervised machine learning and statistics. We carefully analyzed its performance on standard datasets and proposed a non-trivial application of the SFA model in a context where linear algebra based algorithms are too computationally expensive to be applied.

QFDE 3 Quantum Frobenius Distance Estimator

Require:
Quantum access to the matrix X_k of cluster k and to a test vector $x(0)$. Error parameter $\eta > 0$.

Ensure:
An estimate $\overline{F_k(x(0))}$ such that $|F_k(x(0)) - \overline{F_k(x(0))}| < \eta$.

1: s:=0
2: **for** $r = O(1/\eta^2)$ **do**
3: Create the state

$$\frac{1}{\sqrt{N_k}}\Big(\sqrt{|T_k|}\,\|x(0)\|\,|0\rangle + \|X_k\|_F\,|1\rangle\Big)\,|0\rangle\,|0\rangle$$

4: Apply the unitary that maps:

$$|0\rangle\,|0\rangle \mapsto |0\rangle\,\frac{1}{\sqrt{|T_k|}}\sum_{i\in T_k}|i\rangle \quad \text{and} \quad |1\rangle\,|0\rangle \mapsto |1\rangle\,\frac{1}{\|X_k\|_F}\sum_{i\in T_k}\|x_i\|\,|i\rangle$$

to the first two registers to get

$$\frac{1}{\sqrt{N_k}}\Big(\,|0\rangle\sum_{i\in T_k}\|x(0)\|\,|i\rangle + |1\rangle\sum_{i\in T_k}\|x(i)\|\,|i\rangle\Big)\,|0\rangle$$

5: Apply the unitary that maps

$$|0\rangle\,|i\rangle\,|0\rangle \mapsto |0\rangle\,|i\rangle\,|x(0)\rangle \quad \text{and} \quad |1\rangle\,|i\rangle\,|0\rangle \mapsto |1\rangle\,|i\rangle\,|x_i\rangle$$

to get the state

$$\frac{1}{\sqrt{N_k}}\Big(\,|0\rangle\sum_{i\in T_k}\|x(0)\|\,|i\rangle\,|x(0)\rangle + |1\rangle\sum_{i\in T_k}\|x(i)\|\,|i\rangle\,|x(i)\rangle\Big) \tag{4.7}$$

6: Apply a Hadamard to the first register to get

$$\begin{aligned}&\frac{1}{\sqrt{2N_k}}\,|0\rangle\sum_{i\in T_k}\Big(\,\|x(0)\|\,|i\rangle\,|x(0)\rangle + \|x(i)\|\,|i\rangle\,|x(i)\rangle\Big) + \\ &\frac{1}{\sqrt{2N_k}}\,|1\rangle\sum_{i\in T_k}\Big(\,\|x(0)\|\,|i\rangle\,|x(0)\rangle - \|x(i)\|\,|i\rangle\,|x(i)\rangle\Big)\end{aligned} \tag{4.8}$$

7: Measure the first register and if the outcome is $|1\rangle$ then s:=s+1
8: **end for**
9: Output $\frac{s}{r}$.

Chapter 5

Q-means

In this section we detail a quantum algorithm for unsupervised learning, which can be seen as the quantum version of the well known k-means algorithm. This algorithm is one of the simplest, yet most commonly used clustering algorithms. We first introduce the classical algorithm, then propose a definition of the k-mean model that makes is robust to error in the model. Then, we explain how to derive a quantum version of the k-means algorithm and show its performance on experimental data.

5.1 The k-means algorithm

The k-means algorithm was introduced in [102], and is extensively used for unsupervised problems. The inputs to k-means algorithm are vectors $x_i \in \mathbb{R}^d$ for $i \in [n]$. These points must be partitioned in k subsets according to a similarity measure, which in k-means is the Euclidean distance between points. The output of the k-means algorithm is a list of k cluster centers, which are called *centroids*. The algorithm starts by selecting k initial centroids randomly or using efficient heuristics like the k-means++ [14]. It then alternates between two steps: (i) Each data point is assigned the label of the closest centroid. (ii) Each centroid is updated to be the average of the data points assigned to the corresponding cluster. These two steps are repeated until convergence, that is, until the change in the centroids during one iteration is sufficiently small.

More precisely, we are given a dataset X of vectors $x_i \in \mathbb{R}^d$ for $i \in [n]$. At step t, we denote the k clusters by the sets C_j^t for $j \in [k]$, and each corresponding centroid by the vector c_j^t. At each iteration, the data points x_i are assigned to a cluster C_j^t such that $C_1^t \cup C_2^t \cdots \cup C_K^t = V$ and $C_i^t \cap C_l^t = \emptyset$ for $i \neq l$. Let $d(x_i, c_j^t)$ be the Euclidean distance between vectors x_i and c_j^t. The first step of the algorithm assigns each x_i a label $\ell(x_i)^t$ corresponding to the closest centroid, that is

$$\ell(x_i)^t = \text{argmin}_{j\in[k]}(d(x_i, c_j^t)).$$

The centroids are then updated, $c_j^{t+1} = \frac{1}{|C_j^t|}\sum_{i\in C_j^t} x_i$, so that the new centroid is the average of all points that have been assigned to the cluster in this iteration. We say that we have converged if for a small threshold τ (which might be data dependent) we have:

$$\frac{1}{k}\sum_{j=1}^{k} d(c_j^t, c_j^{t-1}) \leqslant \tau.$$

The loss function that this algorithm aims to minimize is the RSS (residual sums of squares), the sum of the squared distances between points and the centroid of their cluster.

$$\text{RSS} := \sum_{j\in[k]}\sum_{i\in C_j} d(c_j, x_i)^2$$

As the RSS decrease at each iteration of the k-means algorithm, the algorithm therefore converges to a local minimum for the RSS. The number of iterations T for convergence depends on the data and the number of clusters. A single iteration has complexity of $O(knd)$ since the n vectors of dimension d have to be compared to each of the k centroids. The centroids obtained at time t are stored in the matrix $C^t \in \mathbb{R}^{k\times d}$, such that the j^{th} row c_j^t for $j \in [k]$ represents the centroid of the cluster $\mathcal{C}_j^t$.

From a computational complexity point of view, we recall that it is NP-hard to find a clustering that achieves the global minimum for the RSS. There are classical clustering algorithms based on optimizing different loss functions, however the k-means algorithm uses the RSS as the objective function. The algorithm can be super-polynomial in the worst case (the number of iterations is $2^{\omega(\sqrt{n})}$ [13]), but the number of iterations is usually small in practice. The k-means algorithm with a suitable heuristic like k-means++ (described later on) to initialize the centroids finds a clustering such that the value for the RSS objective function is within a multiplicative $O(\log n)$ factor of the minimum value [14].

5.1.1 $\delta-k-$means

We now consider a δ-robust version of the k-means in which we introduce some noise. The noise affects the algorithms in both of the steps of k-means: label assignment and centroid estimation.

- Let c_i^* be the closest centroid to the data point x_i. Then, the set of possible labels $L_\delta(x_i)$ for x_i is defined as follows:

$$L_\delta(x_i) = \{c_p : |d^2(c_i^*, x_i) - d^2(c_p, x_i)| \leq \delta\}$$

  The assignment rule selects arbitrarily a cluster label from the set $L_\delta(x_i)$.

- We add $\delta/2$ noise during the calculation of the centroid. Let $\mathcal{C}_j^{t+1}$ be the set of points which have been labeled by j in the previous step. For δ-k-means we pick a centroid c_j^{t+1} with the property that:

$$\left\| c_j^{t+1} - \frac{1}{|\mathcal{C}_j^{t+1}|} \sum_{x_i \in \mathcal{C}_j^{t+1}} x_i \right\| < \frac{\delta}{2}.$$

One way to see this is to perturb the centroid with some noise.

Let us add two remarks on the δ-k-means. First, for a dataset that is expected to have clusterable data, and for a small δ, the number of vectors on the boundary that risk to be misclassified in each step, that is the vectors for which $|L_\delta(x_i)| > 1$ is typically much smaller compared to the vectors that are close to a unique centroid. Second, we also increase by $\delta/2$ the convergence threshold from the k-means algorithm. All in all, δ-k-means is able to find a clustering that is robust when the data points and the centroids are perturbed with some noise of magnitude $O(\delta)$. As we will see in this work, q-means is the quantum equivalent of δ-k-means.

5.2 The q-means algorithm

The q-means algorithm is given as Algorithm 5. At a high level, it follows the same steps as the classical k-means algorithm (and the EM algorithm for GMM), where we now use quantum subroutines for distance estimation, finding the minimum value among a set of elements, matrix multiplication for obtaining the new centroids as quantum states, and efficient tomography. First, we pick some random initial points, using the quantum version of a classical techniques (like the k-means++ idea [14], for which we give a quantum

algorithm later). Then, in Steps 1 and 2 all data points are assigned to a cluster. In Steps 3 and 4 we update the centroids of the clusters and retrieve the information classically. The process is repeated until convergence.

5.2.1 Step 1: Centroid distance estimation

The first step of the algorithm estimates the square distance between data points and clusters using a quantum procedure.

Theorem 45 (Centroid Distance estimation)**.** *Let a data matrix $V \in \mathbb{R}^{n\times d}$ and a centroid matrix $C \in \mathbb{R}^{k\times d}$ be stored in QRAM, such that the following unitaries $|i\rangle|0\rangle \mapsto |i\rangle|x_i\rangle$, and $|j\rangle|0\rangle \mapsto |j\rangle|c_j\rangle$ can be performed in time $O(\log(Nd))$ and the norms of the vectors are known. For any $\Delta > 0$ and $\epsilon_1 > 0$, there exists a quantum algorithm that performs the mapping*

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{n}|i\rangle \otimes_{j\in[k]} (|j\rangle|0\rangle) \mapsto \frac{1}{\sqrt{N}}\sum_{i=1}^{n}|i\rangle \otimes_{j\in[k]} (|j\rangle|\overline{d^2(x_i,c_j)}\rangle),$$

where $|\overline{d^2(x_i,c_j)} - d^2(x_i,c_j)| \leqslant \epsilon_1$ with probability at least $1-2\Delta$, in time $\widetilde{O}\left(k\frac{\eta}{\epsilon_1}\log(1/\Delta)\right)$ where $\eta = \max_i(\|x_i\|^2)$.

The proof of the theorem follows rather straightforwardly from lemma 30. In fact one just needs to apply the distance estimation procedure k times. Note also that the norms of the centroids are always smaller than the maximum norm of a data point which gives us the factor η.

5.2.2 Step 2: Cluster assignment

At the end of Step 1, we have coherently estimated the square distance between each point in the dataset and the k centroids in separate registers. We can now select the index j that corresponds to the centroid closest to the given data point, written as $\ell(x_i) = \text{argmin}_{j\in[k]}(d(x_i,c_j))$. As taking the square of a number is a monotone function, we do not need to compute the square root of the distance in order to find $\ell(x_i)$.

Lemma 46 (Circuit for finding the minimum)**.** *Given k different $\log p$-bit registers $\otimes_{j\in[k]}|a_j\rangle$, there is a quantum circuit U_{min} that maps $(\otimes_{j\in[p]}|a_j\rangle)|0\rangle \to (\otimes_{j\in[k]}|a_j\rangle)|\text{argmin}(a_j)\rangle$ in time $O(k\log p)$.*

Proof. We append an additional register for the result that is initialized to $|1\rangle$. We then repeat the following operation for $2 \leq j \leq k$, we compare registers 1 and j, if the value in register j is smaller we swap registers 1 and j and update the result register to j. The cost of the procedure is $O(k\log p)$. □

The cost of finding the minimum is $\widetilde{O}(k)$ in step 2 of the q-means algorithm, while we also need to uncompute the distances by repeating Step 1. Once we apply the minimum finding lemma 46 and undo the computation we obtain the state

$$|\psi^t\rangle := \frac{1}{\sqrt{N}}\sum_{i=1}^{n}|i\rangle|\ell^t(x_i)\rangle\,. \tag{5.3}$$

5.2.3 Step 3: Centroid state creation

The previous step gave us the state $|\psi^t\rangle = \frac{1}{\sqrt{N}}\sum_{i=1}^{n}|i\rangle|\ell^t(x_i)\rangle$. The first register of this state stores the index of the data points while the second register stores the label for the

Algorithm 5 q-means.

Require: Data matrix $X \in \mathbb{R}^{n\times d}$ stored in QRAM data structure. Precision parameters δ for k-means, error parameters ϵ_1 for distance estimation, ϵ_2 and ϵ_3 for matrix multiplication and ϵ_4 for tomography.

Ensure: Outputs vectors $c_1, c_2, \cdots, c_k \in \mathbb{R}^d$ that correspond to the centroids at the final step of the δ-k-means algorithm.

Select k initial centroids $c_1^0, \cdots, c_k^0$ and store them in QRAM data structure.

t=0

repeat

Step 1: Centroid Distance Estimation

Perform the mapping (theorem 45)

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{n}|i\rangle \otimes_{j\in[k]}|j\rangle|0\rangle \mapsto \frac{1}{\sqrt{N}}\sum_{i=1}^{n}|i\rangle \otimes_{j\in[k]}|j\rangle|\overline{d^2(x_i,c_j^t)}\rangle \tag{5.1}$$

where $|\overline{d^2(x_i,c_j^t)} - d^2(x_i,c_j^t)| \leq \epsilon_1$.

Step 2: Cluster Assignment

Find the minimum distance among $\{d^2(x_i,c_j^t)\}_{j\in[k]}$ (lemma 46), then uncompute Step 1 to create the superposition of all points and their labels

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle \otimes_{j\in[k]}|j\rangle|\overline{d^2(x_i,c_j^t)}\rangle \mapsto \frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle|\ell^t(x_i)\rangle \tag{5.2}$$

Step 3: Centroid states creation

3.1 Measure the label register to obtain a state $|\chi_j^t\rangle = \frac{1}{\sqrt{|\mathcal{C}_j^t|}}\sum_{i\in\mathcal{C}_j^t}|i\rangle$, with prob. $\frac{|\mathcal{C}_j^t|}{N}$

3.2 Perform matrix multiplication with matrix V^T and vector $|\chi_j^t\rangle$ to obtain the state $|c_j^{t+1}\rangle$ with error ϵ_2, along with an estimation of $\left\|c_j^{t+1}\right\|$ with relative error ϵ_3 (theorem 28).

Step 4: Centroid Update

4.1 Perform tomography for the states $|c_j^{t+1}\rangle$ with precision ϵ_4 using the operation from Steps 1-3 (theorem 11) and get a classical estimate $\overline{c}_j^{t+1}$ for the new centroids such that $|c_j^{t+1} - \overline{c}_j^{t+1}| \leq \sqrt{\eta}(\epsilon_3+\epsilon_4) = \epsilon_{centroids}$

4.2 Update the QRAM data structure for the centroids with the new vectors $\overline{c}_0^{t+1}\cdots\overline{c}_k^{t+1}$.

t=t+1

until convergence condition is satisfied.

data point in the current iteration. Given these states, we need to find the new centroids $|c_j^{t+1}\rangle$, which are the average of the data points having the same label.

Let $\chi_j^t \in \mathbb{R}^N$ be the characteristic vector for cluster $j \in [k]$ at iteration t scaled to unit ℓ_1 norm, that is $(\chi_j^t)_i = \frac{1}{|C_j^t|}$ if $i \in \mathcal{C}_j$ and 0 if $i \notin \mathcal{C}_j$. The creation of the quantum states corresponding to the centroids is based on the following simple claim.

Claim 47. *Let $\chi_j^t \in \mathbb{R}^N$ be the scaled characteristic vector for $\mathcal{C}_j$ at iteration t and $X \in \mathbb{R}^{n\times d}$ be the data matrix, then $c_j^{t+1} = X^T\chi_j^t$.*

Proof. The k-means update rule for the centroids is given by $c_j^{t+1} = \frac{1}{|C_j^t|}\sum_{i\in C_j} x_i$. As the columns of X^T are the vectors x_i, this can be rewritten as $c_j^{t+1} = X^T\chi_j^t$. □

The above claim allows us to compute the updated centroids c_j^{t+1} using quantum linear algebra operations. In fact, the state $|\psi^t\rangle$ can be written as a weighted superposition of the characteristic vectors of the clusters.

$$|\psi^t\rangle = \sum_{j=1}^{k}\sqrt{\frac{|C_j|}{N}}\left(\frac{1}{\sqrt{|C_j|}}\sum_{i\in\mathcal{C}_j}|i\rangle\right)|j\rangle = \sum_{j=1}^{k}\sqrt{\frac{|C_j|}{N}}\,|\chi_j^t\rangle\,|j\rangle$$

By measuring the last register, we can sample from the states $|\chi_j^t\rangle$ for $j \in [k]$, with probability proportional to the size of the cluster. We assume here that all k clusters are non-vanishing, in other words they have size $\Omega(N/k)$. Given the ability to create the states $|\chi_j^t\rangle$ and given that the matrix V is stored in QRAM, we can now perform quantum matrix multiplication by X^T to recover an approximation of the state $|X^T\chi_j\rangle = |c_j^{t+1}\rangle$ with error ϵ_2, as stated in theorem 28. Note that the error ϵ_2 only appears inside a logarithm. The same theorem allows us to get an estimate of the norm $\left\|X^T\chi_j^t\right\| = \left\|c_j^{t+1}\right\|$ with relative error ϵ_3. For this, we also need an estimate of the size of each cluster, namely the norms $\|\chi_j\|$. We already have this, since the measurements of the last register give us this estimate, and since the number of measurements made is large compared to k (they depend on d), the error from this source is negligible compared to other errors.

The running time of this step is derived from theorem 28 where the time to prepare the state $|\chi_j^t\rangle$ is the time of Steps 1 and 2. Note that we do not have to add an extra k factor due to the sampling, since we can run the matrix multiplication procedures in parallel for all j so that every time we measure a random $|\chi_j^t\rangle$ we perform one more step of the corresponding matrix multiplication. Assuming that all clusters have size $\Omega(N/k)$ we will have an extra factor of $O(\log k)$ in the running time by a standard coupon collector argument. We set the error on the matrix multiplication to be $\epsilon_2 \ll \frac{\epsilon_4^2}{d\log d}$ as we need to call the unitary that builds c_j^{t+1} for $O(\frac{d\log d}{\epsilon_4^2})$ times. We will see that this does not increase the runtime of the algorithm, as the dependence of the runtime for matrix multiplication is logarithmic in the error.

5.2.4 Step 4: Centroid update

In Step 4, we need to go from quantum states corresponding to the centroids, to a classical description of the centroids in order to perform the update step. For this, we will apply the ℓ_2 vector state tomography algorithm, i.e theorem 11, on the states $|c_j^{t+1}\rangle$ that we create in Step 3. Note that for each $j \in [k]$ we will need to invoke the unitary that creates the states $|c_j^{t+1}\rangle$ a total of $O(\frac{d\log d}{\epsilon_4^2})$ times for achieving $\||c_j\rangle - |\overline{c_j}\rangle\| < \epsilon_4$. Hence, for performing the tomography of all clusters, we will invoke the unitary $O(\frac{k(\log k)d(\log d)}{\epsilon_4^2})$ times where the $O(k\log k)$ term is the time to get a copy of each centroid state.

The vector state tomography gives us a classical estimate of the unit norm centroids within error ϵ_4, that is $\||c_j\rangle - |\overline{c_j}\rangle\| < \epsilon_4$. Using the approximation of the norms $\|c_j\|$ with

relative error ϵ_3 from Step 3, we can combine these estimates to recover the centroids as vectors. The analysis is described in the following claim:

Claim 48. *Let ϵ_4 be the error we commit in estimating $|c_j\rangle$ such that $\||c_j\rangle - |\overline{c_j}\rangle\| < \epsilon_4$, and ϵ_3 the error we commit in the estimating the norms, $|\,\|c_j\| - \overline{\|c_j\|}\,| \leq \epsilon_3 \|c_j\|$. Then $\|\overline{c_j} - c_j\| \leq \sqrt{\eta}(\epsilon_3 + \epsilon_4) = \epsilon_{centroid}$.*

Proof. We can rewrite $\|c_j - \overline{c_j}\|$ as $\left\|\|c_j\|\,|c_j\rangle - \overline{\|c_j\|}\,|\overline{c_j}\rangle\right\|$. It follows from triangle inequality that:

$$\left\|\overline{\|c_j\|}\,|\overline{c_j}\rangle - \|c_j\|\,|c_j\rangle\right\| \leq \left\|\overline{\|c_j\|}\,|\overline{c_j}\rangle - \|c_j\|\,|\overline{c_j}\rangle\right\| + \|\|c_j\|\,|\overline{c_j}\rangle - \|c_j\|\,|c_j\rangle\|$$

We have the upper bound $\|c_j\| \leq \sqrt{\eta}$. Using the bounds for the error we have from tomography and norm estimation, we can upper bound the first term by $\sqrt{\eta}\epsilon_3$ and the second term by $\sqrt{\eta}\epsilon_4$. The claim follows. □

Tomography on approximately pure states

Let us make a remark about the ability to use theorem 11 to perform tomography in our case. The updated centroids will be recovered in Step 4 using the vector state tomography algorithm in theorem 11 on the composition of the unitary that prepares $|\psi^t\rangle$ and the unitary that multiplies the first register of $|\psi^t\rangle$ by the matrix V^T. The input of the tomography algorithm requires a unitary U such that $U|0\rangle = |x\rangle$ for a fixed quantum state $|x\rangle$. However, the labels $\ell(x_i)$ are not deterministic due to errors in distance estimation, hence the composed unitary U as defined above therefore does not produce a fixed pure state $|x\rangle$. Nevertheless, two different runs of the same unitary returns a quantum state which we can think of a mixed state that is a good approximation of a pure state.

We therefore need a procedure that finds labels $\ell(x_i)$ that are a deterministic function of x_i and the centroids c_j for $j \in [k]$. One solution is to change the update rule of the δ-k-means algorithm to the following: Let $\ell(x_i) = j$ if $\overline{d(x_i, c_j)} < \overline{d(x_i, c_{j'})} - 2\delta$ for $j' \neq j$ where we discard the points to which no label can be assigned. This assignment rule ensures that if the second register is measured and found to be in state $|j\rangle$, then the first register contains a uniform superposition of points from cluster j that are δ far from the cluster boundary (and possibly a few points that are δ close to the cluster boundary). Note that this simulates exactly the δ-k-means update rule while discarding some of the data points close to the cluster boundary. The k-means centroids are robust under such perturbations, so we expect this assignment rule to produce good results in practice.

A better solution is to use consistent phase estimation instead of the usual phase estimation for the distance estimation step , which can be found in [130, 8]. The distance estimates are generated by the phase estimation algorithm applied to a certain unitary in the amplitude estimation step. The usual phase estimation algorithm does not produce a deterministic answer and instead for each eigenvalue λ outputs with high probability one of two possible estimates $\overline{\lambda}$ such that $|\lambda - \overline{\lambda}| \leq \epsilon$. Instead, here as in some other applications we need the consistent phase estimation algorithm that with high probability outputs a deterministic estimate such that $|\lambda - \overline{\lambda}| \leq \epsilon$.

For what follows, we assume that indeed the state in Equation 5.3 is almost a pure state, meaning that when we repeat the procedure we get the same state with very high probability.

5.3 Initialization: q-means++

Before running k-means, one usually chooses the first k centroids by using the k-means++ technique from [14]. A first centroid is chosen uniformly at random and we compute its

distance to all points of the dataset. Then we sample one point with a weighted probability distribution corresponding to their squared distance to the centroid. We repeat the previous step with this new point as centroid, until k centroids have been chosen.

A quantum analogue, q-means++, can be implemented efficiently using the square distance subroutine described for the q-means algorithm form theorem 45. Starting with a random index i we compute the following state in time $\tilde{O}(\frac{\eta}{\epsilon_1})$:

$$|i\rangle \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j\rangle |d(x_i, x_j)\rangle$$

Where x_i is the initial centroid. We can then convert the distance register as amplitudes using a controlled rotation after a simple arithmetic circuit.

$$|i\rangle \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j\rangle |d(x_i, x_j)\rangle \left(\frac{d(x_i, x_j)}{2\eta} |0\rangle + \beta |1\rangle \right)$$

Each distance has been normalized by $2\eta \geq max_{i,j}(d(x_i, x_j))$ to be a valid amplitude. After undoing the distance computation subroutine in the second register, we perform an amplitude amplification on $|0\rangle$. This creates the state

$$|{+}{+}\rangle := \frac{1}{Z} \sum_{j=0}^{n-1} d(x_i, x_j) |j\rangle$$

where Z is the normalization factor $\sqrt{\sum_{j=0}^{n-1} d^2(x_i, x_j)}$. We can sample a value j that will represent the next centroid chosen for iteration $t = 0$. To create the state $|{+}{+}\rangle$ we need to perform amplitude amplification, and repeat $O(1/\sqrt{P(0)})$ times the distance estimation procedure, with $P(0)$ being the probability of measuring $|0\rangle$. Since $P(0) = \frac{1}{n} \left(\sum \frac{d(x_i, x_j)}{2\eta} \right)^2$

$$\frac{1}{\sqrt{P(0)}} = \frac{2\eta}{\sqrt{\frac{1}{N} \left(\sum d^2(x_i, x_j) \right)^2}} \leq \frac{2\eta}{\sqrt{\frac{1}{N} \sum d^2(x_i, x_j)}}$$

In the end we repeat $k-1$ times this circuit, for a total time of $\tilde{O}(k \frac{4\eta^2}{\epsilon_1 \sqrt{\mathbb{E}(d^2(x_i, x_j))}})$. In order to be adapt this initialization subroutine with δ-k-means algorithm, it suffice to pick $\epsilon_1 < \delta/2$.

5.4 Analysis

We provide the theorem of the running time and accuracy of the q-means algorithm.

Theorem 49 (q-means)**.** *For a data matrix $X \in \mathbb{R}^{n \times d}$ for which we have quantum access, and parameter $\delta > 0$, the q-means algorithm with high probability outputs centroids consistent with the classical δ-k-means algorithm, in time*

$$\widetilde{O} \left(kd \frac{\eta}{\delta^2} \kappa(X)(\mu(X) + k\frac{\eta}{\delta}) + k^2 \frac{\eta^{1.5}}{\delta^2} \kappa(V)\mu(V) \right)$$

per iteration.

We prove the theorem in Sections 5.4.1 and 5.4.2.

5.4.1 Error analysis

In this section we determine the error parameters in the different steps of the quantum algorithm so that the quantum algorithm behaves the same as the classical δ-k-means. More precisely, we will determine the values of the errors $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ in terms of δ. In this way, the cluster assignment of all data points made by the q-means algorithm is consistent with a classical run of the δ-k-means algorithm, and also that the centroids computed by the q-means after each iteration are again consistent with centroids that can be returned by the δ-k-means algorithm.

The cluster assignment in q-means happens in two steps. The first step estimates the square distances between all points and all centroids. The error in this procedure is of the form

$$|\overline{d^2(c_j, x_i)} - d^2(c_j, x_i)| < \epsilon_1.$$

for a point x_i and a centroid c_j.

The second step finds the minimum of these distances without adding any error. For the q-means to output a cluster assignment consistent with the δ-k-means algorithm, we require that:

$$\forall j \in [k], \quad |\overline{d^2(c_j, x_i)} - d^2(c_j, x_i)| \leq \frac{\delta}{2}$$

which implies that no centroid with distance more than δ above the minimum distance can be chosen by the q-means algorithm as the label. Thus we need to take $\epsilon_1 < \delta/2$.

After the cluster assignment of the q-means (which happens in superposition), we update the clusters, by first performing a matrix multiplication to create the centroid states and estimate their norms, and then a tomography to get a classical description of the centroids. The error in this part is $\epsilon_{centroids}$, as defined in Claim 48, namely:

$$\|\overline{c}_j - c_j\| \leq \epsilon_{centroid} = \sqrt{\eta}(\epsilon_3 + \epsilon_4).$$

Again, for ensuring that the q-means is consistent with the classical δ-k-means algorithm we take $\epsilon_3 < \frac{\delta}{4\sqrt{\eta}}$ and $\epsilon_4 < \frac{\delta}{4\sqrt{\eta}}$. Note also that we have ignored the error ϵ_2 that we can easily deal with, since it only appears in a logarithmic factor in the runtime.

5.4.2 Runtime analysis

As the classical algorithm, the runtime of q-means depends linearly on the number of iterations, so here we analyze the cost of a single step. The cost of tomography for the k centroid vectors is $O(\frac{kd \log k \log d}{{\epsilon_4}^2})$ times the cost of preparation of a single centroid state $|c_j^t\rangle$. A single copy of $|c_j^t\rangle$ is prepared applying the matrix multiplication by V^T procedure on the state $|\chi_j^t\rangle$ obtained using square distance estimation. The time required for preparing a single copy of $|c_j^t\rangle$ is $O(\kappa(V)(\mu(V) + T_\chi)\log(1/\epsilon_2))$ by theorem 28 where T_χ is the time for preparing $|\chi_j^t\rangle$. The time T_χ is $\widetilde{O}\left(\frac{k\eta \log(\Delta^{-1}) \log(nd)}{\epsilon_1}\right) = \widetilde{O}(\frac{k\eta}{\epsilon_1})$ by theorem 30. The cost of norm estimation for k different centroids is independent of the tomography cost and is $\widetilde{O}(\frac{kT_\chi \kappa(V)\mu(V)}{\epsilon_3})$. Combining together all these costs and suppressing all the logarithmic factors we have a total running time of:

$$\widetilde{O}\left(kd\frac{1}{\epsilon_4^2}\kappa(X)(\mu(X) + k\frac{\eta}{\epsilon_1}) + k^2\frac{\eta}{\epsilon_3\epsilon_1}\kappa(X)\mu(X)\right) \tag{5.4}$$

The analysis in section 5.4.1 shows that we can take $\epsilon_1 = \delta/2$, $\epsilon_3 = \frac{\delta}{4\sqrt{\eta}}$ and $\epsilon_4 = \frac{\delta}{4\sqrt{\eta}}$. Substituting these values in the above running time, it follows that the running time of the q-means algorithm is:

$$\widetilde{O}\left(kd\frac{\eta}{\delta^2}\kappa(V)\left(\mu(V) + k\frac{\eta}{\delta}\right) + k^2\frac{\eta^{1.5}}{\delta^2}\kappa(V)\mu(V)\right).$$

This completes the proof of theorem 49.

A few concluding remarks regarding the running time of q-means. For dataset where the number of points is much bigger compared to the other parameters, the running time for the q-means algorithm is an improvement compared to the classical k-means algorithm. For instance, for most problems in data analysis, k is eventually small (< 100). The number of features $d \leq n$ in most situations, and it can eventually be reduced by applying a quantum dimensionality reduction algorithm first (which have running time polylogarithmic in d). To sum up, q-means has the same output as the classical δ-k-means algorithm (which approximates k-means), it conserves the same number of iterations, but has a running time only polylogarithmic in n, giving an exponential speedup with respect to the size of the dataset.

5.5 Simulations on real data

In order to assess the capability of the quantum algorithm to provide accurate classification results, we used some simulations. However, since neither quantum simulators nor quantum computers large enough to test q-means are available currently, we tested the equivalent classical implementation of δ-k-means. For implementing the δ-k-means, we changed the assignment step of the k-means algorithm to select a random centroid among those that are δ-close to the closest centroid and added $\delta/2$ error to the updated clusters.

We benchmarked our q-means algorithm the well-known MNIST dataset of handwritten digits. To measure and compare the accuracy of our clustering algorithm, we ran the k-means and the δ-k-means algorithms for different values of δ on a training dataset and then we compared the accuracy of the classification on a test set, containing data points on which the algorithms have not been trained, using a number of widely-used performance measures.

The MNIST dataset is composed of 60.000 handwritten digits as images of 28x28 pixels (784 dimensions). From this raw data we first performed some dimensionality reduction processing, then we normalized the data such that the minimum norm is one. Note that, if we were doing q-means with a quantum computer, we could use efficient quantum procedures for dimensionality reduction, like the one described in Chapter 4, or other quantum dimensionality reduction algorithms like [100, 43]. As we are testing an unsupervised learning algorithm, we discard the labels from the simulation. For preprocessing of the data, we first performed a Principal Component Analysis (PCA), retaining data projected in a subspace of dimension 40. After normalization, the value of η was 8.25 (maximum norm of 2.87), and the condition number was 4.53. Figure 5.1 represents the evolution of the accuracy during the k-means and δ-k-means for 4 different values of δ. In this numerical experiment, we can see that for values of the parameter η/δ of order 20, both k-means and δ-k-means reached a similar, yet low accuracy in the classification in the same number of steps. It is important to notice that the MNIST dataset, without other preprocessing than dimensionality reduction, is hard to cluster with high accuracy, due to the curse of dimensionality.

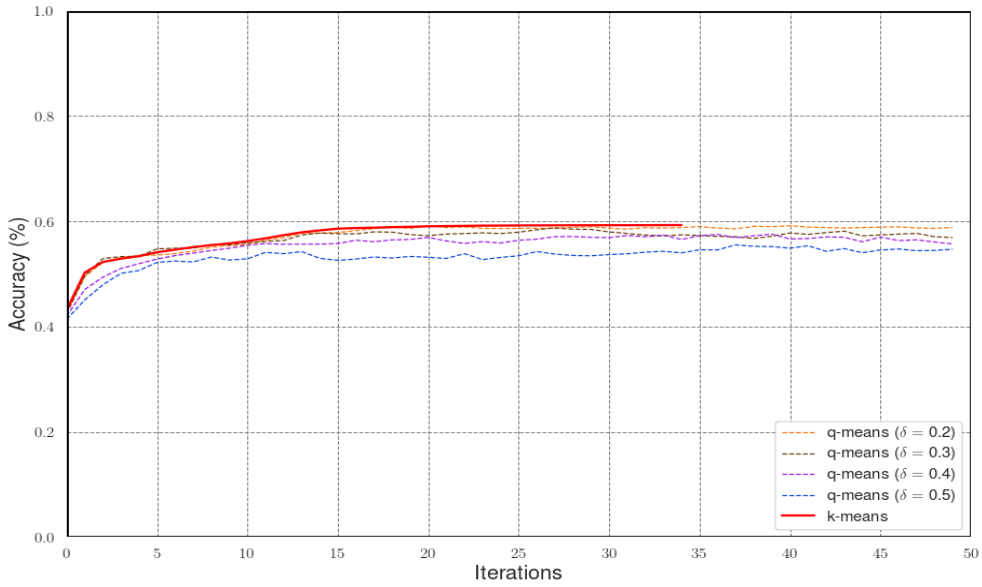


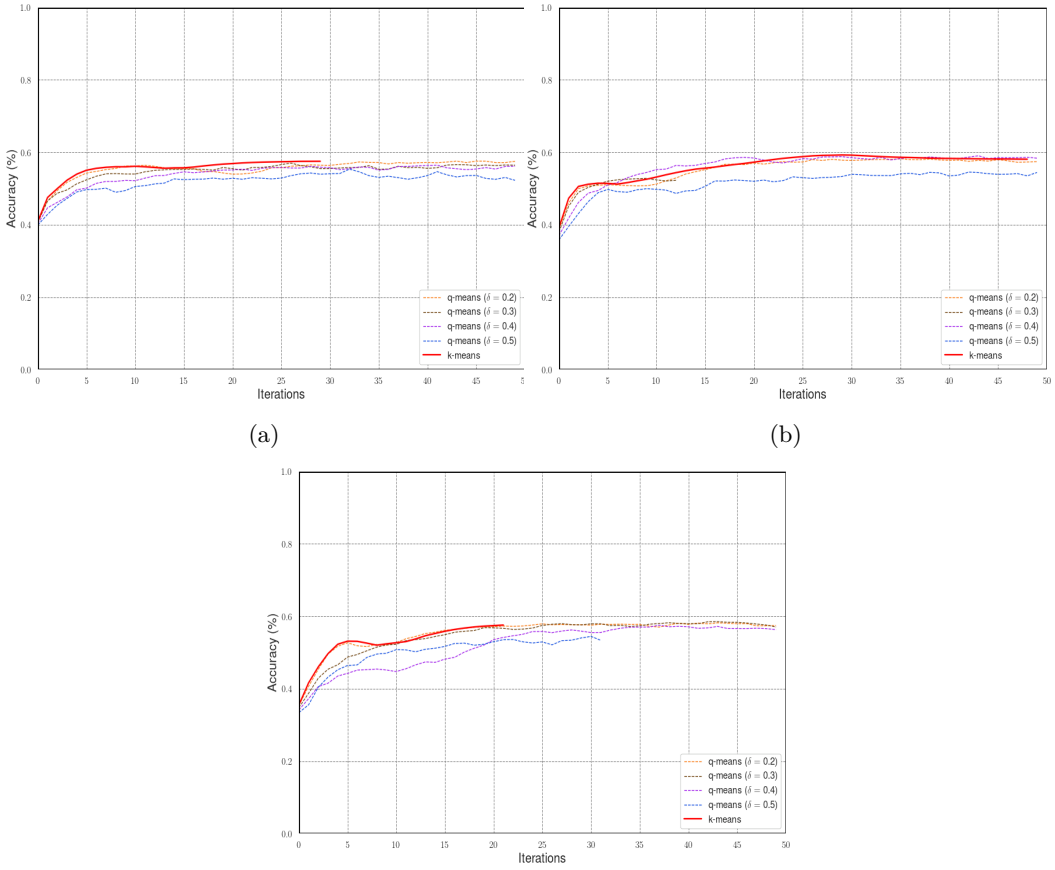
Figure 5.1: Accuracy evolution on the MNIST dataset under k-means and δ-k-means for 4 different values of δ. Data has been preprocessed by a PCA to 40 dimensions. All versions converge in the same number of steps, with a drop in the accuracy while δ increases. The apparent difference in the number of steps until convergence is just due to the stopping condition for k-means and δ-k-means.

On top of the accuracy measure (ACC), we also evaluated the performance of q-means against many other metrics, reported in Table 5.2. More detailed information about these metrics can be found in [54, 81]. We introduce a specific measure of error, the Root Mean Square Error of Centroids (RMSEC), which a direct comparison between the centroids predicted by the k-means algorithm and the ones predicted by the δ-k-means. It is a way to know how far the centroids are predicted. Note that this metric can only be applied to the training set. For all these measures, except RMSEC, a bigger value is better. Our simulations show that δ-k-means, and thus the q-means, even for values of δ (between $0.2 - 0.5$) achieves similar performance to k-means, and in most cases the difference is on the third decimal point.

Algo	Dataset	ACC	HOM	COMP	V-M	AMI	ARI	RMSEC
k-means	Train	0.582	0.488	0.523	0.505	0.389	0.488	0
	Test	0.592	0.500	0.535	0.517	0.404	0.499	-
δ-k-means, $\delta = 0.2$	Train	0.580	0.488	0.523	0.505	0.387	0.488	0.009
	Test	0.591	0.499	0.535	0.516	0.404	0.498	-
δ-k-means, $\delta = 0.3$	Train	0.577	0.481	0.517	0.498	0.379	0.481	0.019
	Test	0.589	0.494	0.530	0.511	0.396	0.493	-
δ-k-means, $\delta = 0.4$	Train	0.573	0.464	0.526	0.493	0.377	0.464	0.020
	Test	0.585	0.492	0.527	0.509	0.394	0.491	-
δ-k-means, $\delta = 0.5$	Train	0.573	0.459	0.522	0.488	0.371	0.459	0.034
	Test	0.584	0.487	0.523	0.505	0.389	0.487	-

Table 5.1: A sample of results collected from the same experiments as in Figure 5.1. Different metrics are presented for the train set and the test set. ACC: accuracy. HOM: homogeneity. COMP: completeness. V-M: v-measure. AMI: adjusted mutual information. ARI: adjusted rand index. RMSEC: Root Mean Square Error of Centroids.

These experiments have been repeated several times and each of them presented a similar behavior despite the random initialization of the centroids.



(a) (b)

(c)

Figure 5.2: Three accuracy evolutions on the MNIST dataset under k-means and δ-k-means for 4 different values of δ. Each different behavior is due to the random initialization of the centroids

Finally, we present a last experiment with the MNIST dataset with a different data preprocessing. In order to reach higher accuracy in the clustering, we replace the previous dimensionality reduction by a Linear Discriminant Analysis (LDA). Note that a LDA is a supervised process that uses the labels (here, the digits) to project points in a well chosen lower dimensional subspace. Thus this preprocessing cannot be applied in practice in unsupervised machine learning. However, for the sake of benchmarking, by doing so k-means is able to reach a 87% accuracy, therefore it allows us to compare k-means and δ-k-means on a real and almost well-clusterable dataset. In the following, the MNIST dataset is reduced to 9 dimensions. The results in Figure 5.3 show that δ-k-means converges to the same accuracy than k-means even for values of η/δ down to 16. In some other cases, δ-k-means shows a faster convergence, due to random fluctuations that can help escape faster from a temporary equilibrium of the clusters.

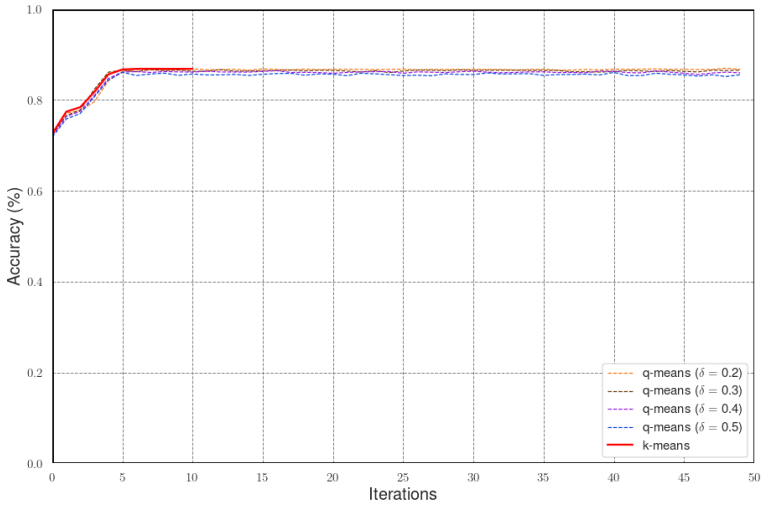


Figure 5.3: Accuracy evolution on the MNIST dataset under k-means and δ-k-means for 4 different values of δ. Data has been preprocessed to 9 dimensions with a LDA reduction. All versions of δ-k-means converge to the same accuracy than k-means in the same number of steps.

Algo	Dataset	ACC	HOM	COMP	V-M	AMI	ARI	RMSEC
k-means	Train	0.868	0.736	0.737	0.737	0.735	0.736	0
	Test	0.891	0.772	0.773	0.773	0.776	0.771	-
q-means, $\delta = 0.2$	Train	0.868	0.737	0.738	0.738	0.736	0.737	0.031
	Test	0.891	0.774	0.775	0.775	0.777	0.774	-
q-means, $\delta = 0.3$	Train	0.869	0.737	0.739	0.738	0.736	0.737	0.049
	Test	0.890	0.772	0.774	0.773	0.775	0.772	-
q-means, $\delta = 0.4$	Train	0.865	0.733	0.735	0.734	0.730	0.733	0.064
	Test	0.889	0.770	0.771	0.770	0.773	0.769	-
q-means, $\delta = 0.5$	Train	0.866	0.733	0.735	0.734	0.731	0.733	0.079
	Test	0.884	0.764	0.766	0.765	0.764	0.764	-

Table 5.2: A sample of results collected from the same experiments as in Figure 5.3. Different metrics are presented for the train set and the test set. ACC: accuracy. HOM: homogeneity. COMP: completeness. V-M: v-measure. AMI: adjusted mutual information. ARI: adjusted rand index. RMSEC: Root Mean Square Error of Centroids.

Let us remark, that the values of η/δ in our experiment remained between 3 and 20. Moreover, the parameter η, which is the maximum square norm of the points, provides a worst case guarantee for the algorithm, while one can expect that the running time in practice will scale with the average square norm of the points. For the MNIST dataset after PCA, this value is 2.65 whereas $\eta = 8.3$.

In conclusion, our simulations show that the convergence of δ-k-means is almost the same as the regular k-means algorithms for values of δ which are sufficiently large. This provides evidence that the q-means algorithm will have as good performance as the classical k-means, and its running time will be significantly lower for large datasets.

5.6 Conclusions

In this chapter we developed q-means, the quantum version of the k-means algorithm, theoretically analyzed its asymptotic running time and tested the convergence of this iterative

algorithm on the MNIST dataset. The q-means is a clustering algorithm with provable guarantees on its runtime, and opens the way for quantum applications in unsupervised learning. In this work we ameliorated and better formalized subroutines to estimate distances and inner products between vectors. Moreover, this work made it clear that often when writing a quantum machine learning algorithm it is necessary to slightly modify the original definition of the classical model under consideration, so as to take the error of the quantum procedures into account. As we will see in the next chapter, this will also be the case for Gaussian mixture model, a generalization of k-means.

Chapter 6

Quantum Expectation-Maximization

In this section we put forward the quantum version of Expectation-Maximization (EM). EM is a versatile iterative algorithm that have been broadly used (and discovered) in many part of machine learning and statistics. As is common in machine learning literature, we introduce the Expectation-Maximization algorithm by using it to fit Gaussian mixture models (GMM). As the name hints, a Gaussian mixture model is a way of describing a probability density function as a combination of different Gaussian distributions. GMM, and in general all the mixture models, are a popular generative model in machine learning.

6.1 Expectation-Maximization and Gaussian mixture models

The intuition behind mixture models is to model complicated distributions by using a group of simpler (usually uni-modal) distributions. In this setting, the purpose of the learning algorithm is to model the data by fitting the joint probability distribution which most likely have generated our samples. Recall that, given a sufficiently large number of mixture components, it is possible to approximate any density defined in $\mathbb{R}^d$ [110]. In this section we describe formally GMM, which is a popular mixture model used to solve unsupervised classification problems. Then we propose the first quantum algorithm to fit a GMM with a quantum computer. While there are many classical algorithms that can be used to fit a mixture of Gaussians (which we detail later), this quantum algorithm resemble as much as possible Expectation-Maximization: a iterative method to find (local) maxima of maximum likelihood and maximum a posteriori optimization problems, especially used in unsupervised settings.

Recall that in the unsupervised case, we are given a training set of unlabeled vectors $v_1 \cdots v_n \in \mathbb{R}^d$ which we represent as rows of a matrix $V \in \mathbb{R}^{n \times d}$. Let $y_i \in [k]$ one of the k possible labels for a point v_i. We posit that for a GMM the joint probability distribution of the data $p(v_i, y_i) = p(v_i|y_i)p(y_i)$, is defined as follow: $y_i \sim \text{Multinomial}(\theta)$ for $\theta \in \mathbb{R}^k$, and $p(v_i|y_i = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$. The θ_j are the *mixing weights*, i.e. the probabilities that $y_i = j$, and $\mathcal{N}(\mu_j, \Sigma_j)$ is the Gaussian distribution centered in $\mu_j \in \mathbb{R}^d$ with covariance matrix $\Sigma_j \in \mathbb{R}^{d \times d}$. Note that the variables y_i are unobserved, and thus are called *latent* variables. There is a simple interpretation for this model. We assume the data is created by first selecting an index $j \in [k]$ by sampling according to Multinomial(θ), and then a vector v_i is sampled from $\mathcal{N}(\mu_j, \Sigma_j)$. Fitting a GMM to a dataset reduces to finding an assignment for the parameters:

$$\gamma = (\theta, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (\theta, \mu_1, \cdots, \mu_k, \Sigma_1, \cdots, \Sigma_k) \tag{6.1}$$

that best maximize the log-likelihood (defined in Section 3) for a given dataset. We will now see how the log-likelihood is defined for a GMM. We use the letter ϕ to represent our *base distribution*, which in this case is the probability density function of a Gaussian $\mathcal{N}(\mu, \Sigma)$:

$$\phi(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{6.2}$$

With this formulation, a GMM is expressed as:

$$p(v) = \sum_{j=1}^{k} \theta_j \phi(v; \mu_j, \Sigma_j) \tag{6.3}$$

where θ_j are the *mixing weights* of the multinomial distribution such that $\sum_{j=1}^{k} \theta_j = 1$. The probability for an observation v_i to be assigned to the component j is given by:

$$r_{ij} = p(y_i = j|v_i; \theta, \mu, \Sigma) = \frac{\theta_j \phi(v_i; \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \theta_l \phi(v_i; \mu_l, \Sigma_l)}. \tag{6.4}$$

This value is called *responsibility*, and corresponds to the posterior probability of the sample i being assigned label j by the current model. More generally, for any base distribution in the mixture, the responsibility of the i-th vector in cluster j can be written as [110]:

$$r_{ij} = \frac{p(y_i = j; \gamma)p(v_i|y_i = j; \gamma)}{\sum_{j'=1}^{k} p(y_i = j'; \gamma)p(v_i|y_i = j'; \gamma)} \tag{6.5}$$

As anticipated, to find the best parameters of our generative model, we maximize the log-likelihood of the data. To conclude, for GMM, the likelihood is given by the following formula [111]:

$$\ell(\gamma; V) = \ell(\theta, \boldsymbol{\mu}, \boldsymbol{\Sigma}; V) = \sum_{i=1}^{n} \log p(v_i\ ;\ \theta, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{n} \log \sum_{y_i=1}^{k} p(v_i|y_i\ ;\ \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(y_i; \theta) \tag{6.6}$$

Alas, it is seldom possible to solve maximum likelihood estimation analytically (i.e. by finding the zeroes of the derivatives of Equation (6.6)), and this is one of those cases. Expectation-Maximization is an iterative algorithm that solves numerically the optimization problem of ML estimation. To complicate things, the likelihood function for GMM is not convex, and thus we might find some local minima [81]. Note that the algorithm used to fit GMM can return a local minimum which might be different than γ^*: the model that represents the global optimum of the likelihood function.

6.1.1 Expectation-Maximization for GMM

The intuition behind EM is simple. If we were to know the latent variable y_i, then the log-likelihood for GMM would be:

$$\ell(\gamma; V) = \sum_{i=1}^{n} \log p(v_i \mid y_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(y_i; \theta) \tag{6.7}$$

This formula can be easily maximized with respect to the parameters $\theta, \boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$. In the Expectation step we calculate the missing variables y_i, given a guess of the parameters $(\theta, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of the model. Then, in the Maximization step, we use the estimate of the latent variables obtained in the Expectation step to update the estimate of the parameters. While in the Expectation step we calculate a lower bound on the likelihood, in the Maximization

step we maximize it. Since at each iteration the likelihood can only increase, the algorithm is guaranteed to converge, albeit possibly to a local optimum (see [81] for the proof). During the Expectation step all the responsibilities are calculated, while in the Maximization step we update our estimate on the parameters $\gamma^{t+1} = (\theta^{t+1}, \boldsymbol{\mu}^{t+1}, \boldsymbol{\Sigma}^{t+1})$.

The stopping criterion for GMM is usually a threshold on the increment of the log-likelihood: if the log-likelihood changes less than a threshold between two iterations, then the algorithm stops. Notice that, since the value of the log-likelihood significantly depends on the amount of data points in the training sets, it is often preferable to adopt a scale-free stopping criterion, which does not depend on the number of samples. For instance, in the toolkit scikit-learn [116] the stopping criterion is given by a tolerance on the average increment of the log-probability, which is chosen to be smaller than a certain ϵ_τ, say 10^{-3}. More precisely, the stopping criterion is $|\mathbb{E}[\log p(v_i;\gamma^t)] - \mathbb{E}[\log p(v_i;\gamma^{t+1})]| < \epsilon_\tau$ which we can estimate as $|\frac{1}{n}\sum_{i=1}^n \log p(v_i;\gamma^t) - \frac{1}{n}\sum_{i=1}^n \log p(v_i;\gamma^{t+1})| < \epsilon_\tau$.

Algorithm 6 Expectation-Maximization for GMM

Require: Dataset V, tolerance $\tau > 0$.

Ensure: A GMM $\gamma^t = (\theta^t, \boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$ that maximizes locally the likelihood $\ell(\gamma; V)$ up to tolerance τ.

Select $\gamma^0 = (\theta^0, \boldsymbol{\mu}^0, \boldsymbol{\Sigma}^0)$ using classical initialization strategies described in Subsection 6.1.2.

$t = 0$

repeat

Expectation

$\forall i, j$, calculate the responsibilities as:

$$r_{ij}^t = \frac{\theta_j^t \phi(v_i; \mu_j^t, \Sigma_j^t)}{\sum_{l=1}^k \theta_l^t \phi(v_i; \mu_l^t, \Sigma_l^t)} \tag{6.8}$$

Maximization

Update the parameters of the model as:

$$\theta_j^{t+1} \leftarrow \frac{1}{n}\sum_{i=1}^n r_{ij}^t \tag{6.9}$$

$$\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t v_i}{\sum_{i=1}^n r_{ij}^t} \tag{6.10}$$

$$\Sigma_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t (v_i - \mu_j^{t+1})(v_i - \mu_j^{t+1})^T}{\sum_{i=1}^n r_{ij}^t} \tag{6.11}$$

t=t+1

until

$$|\ell(\gamma^{t-1}; V) - \ell(\gamma^t; V)| < \tau \tag{6.12}$$

Return $\gamma^t = (\theta^t, \boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$

6.1.2 Initialization strategies for EM

Unlike k-means clustering, choosing a good set of initial parameters for a mixture of Gaussian is by no means trivial, and in multivariate context is known that the solution is problem-dependent. There are plenty of proposed techniques, and here we describe a few

of them. Fortunately, these initialization strategies can be directly translated into quantum subroutines without impacting the overall running time of the quantum algorithm.

The simplest technique is called *random EM*, and consists in selecting initial points at random from the dataset as centroids, and sample the dataset to estimate the covariance matrix of the data. Then these estimates are used as the starting configuration of the model, and we may repeat the random sampling until we get satisfactory results.

A more standard technique borrows directly the initialization strategy of *k-means++* proposed in [14], and extends it to make an initial guess for the covariance matrices and the mixing weights. The initial guess for the centroids is selected by sampling from a suitable, easy to calculate distribution. This heuristic works as following: Let c_0 be a randomly selected point of the dataset, as first centroid. The other $k-1$ centroids are selected by selecting a vector v_i with probability proportional to $d^2(v_i, \mu_{l(v_i)})$, where $\mu_{l(v_i)}$ is the previously selected centroid that is the closest to v_i in ℓ_2 distance. These centroids are then used as initial centroids for a round of k-means algorithm to obtain $\mu_1^0 \cdots \mu_j^0$. Then, the covariance matrices can be initialized as $\Sigma_j^0 := \frac{1}{|\mathcal{C}_j|}\sum_{i\in\mathcal{C}_j}(v_i-\mu_j)(v_i-\mu_j)^T$, where $\mathcal{C}_j$ is the set of samples in the training set that have been assigned to the cluster j in the previous round of k-means. The mixing weights are estimated as $\mathcal{C}_j/n$. Eventually Σ_j^0 is regularized to be a PSD matrix.

There are other possible choices for parameter initialization in EM, for instance, based on *Hierarchical Agglomerative Clustering (HAC)* and the *CEM* algorithm. In CEM we run one step of EM, but with a so-called classification step between E and M. The classification step consists in a hard-clustering after computing the initial conditional probabilities (in the E step). The M step then calculates the initial guess of the parameters [36]. In the *small EM* initialization method we run EM with a different choice of initial parameters using some of the previous strategies. The difference here is that we repeat the EM algorithm for a few numbers of iterations, and we keep iterating from the choice of parameters that returned the best partial results. For an overview and comparison of different initialization techniques, we refer to [30, 27].

Special cases of GMM. What we presented in the previous section is the most general model of GMM. For simple datasets, it is common to assume some restrictions on the covariance matrices of the mixtures. The translation into a quantum version of the model should be straightforward. We distinguish between these cases:

1. **Soft k-means**. This algorithm is often presented as a generalization of k-means, but it can actually be seen as special case of EM for GMM - albeit with a different assignment rule. In soft k-means, the assignment function is replaced by a softmax function with *stiffness* parameter β. This β represents the covariance of the clusters. It is assumed to be equal for all the clusters, and for all dimensions of the feature space. Gaussian Mixtures with constant covariance matrix (i.e. $\Sigma_j = \beta I$ for $\beta \in \mathbb{R}$) can be interpreted as a kind of soft or fuzzy version of k-means clustering. The probability of a point in the feature space being assigned to a certain cluster j is:

$$r_{ij} = \frac{e^{-\beta\|x_i-\mu_i\|^2}}{\sum_{l=1}^{k} e^{-\beta\|x_i-\mu_l\|^2}}$$

   where $\beta > 0$ is the stiffness parameter.

2. **Spherical**. In this model, each component has its own covariance matrix, but the variance is uniform in all the directions, thus reducing the covariance matrix to a multiple of the identity matrix (i.e. $\Sigma_j = \sigma_j^2 I$ for $\sigma_j \in \mathbb{R}$).

3. **Diagonal**. As the name suggests, in this special case the covariance matrix of the distributions is a diagonal matrix, but different Gaussians might have different diagonal covariance matrices.

4. **Tied**. In this model, the Gaussians share the same covariance matrix, without having further restriction on the Gaussian.

5. **Full**. This is the most general case, where each of the components of the mixture have a different, SDP, covariance matrix.

6.1.3 Dataset assumptions in GMM

We make explicit an assumption on the dataset, namely that all elements of the mixture contribute proportionally to the total responsibility:

$$\frac{\sum_{i=1}^{n} r_{ij}}{\sum_{i=1}^{n} r_{il}} = \Theta(1) \quad \forall j, l \in [k] \tag{6.13}$$

This is equivalent to assuming that $\theta_j/\theta_l = \Theta(1) \quad \forall j, l \in [k]$. The algorithm we propose can of course be used even in cases where this assumption does not hold. In this case, the running time will include a factor as in Eq. 6.13 which for simplicity we have taken as constant in what follows. Note that classical algorithms would also find it difficult to fit the data in certain cases, for example when some of the clusters are very small. In fact, it is known (and not surprising) that if the statistical distance between the probability density function of two different Gaussian distributions is smaller than $1/2$, then we can not tell for a point v from which Gaussian distribution it belongs to, even if we knew the parameters [109]. Only for convenience in the analysis, we also assume the dataset as being normalized such that the shortest vector has norm 1 and define $\eta := max_i \|v_i\|^2$ to be the maximum norm squared of a vector in the dataset.

6.1.4 EM and other mixture models

Expectation-Maximization is widely used for fitting mixture models in machine learning [110]. Most mixture models use a base distribution in the exponential family: Poisson [42] (when the observations are a mixture of random counts with a fixed rate of occurrences), Binomial and Multinomial (when the observations have 2 or multiple possible outcomes, like answers in a survey or a vote) and log-normal [53], exponential (when samples have a latent variable that represents a failure of a certain kind, which is often modeled by the exponential distribution) [66], Dirichlet multinomial [154], and others.

Besides fitting mixture models based on the exponential family, the EM algorithm has several other applications. It has been used to fit mixtures of experts, mixtures of the student T distribution (which does not belong to the exponential family, and can be fitted with EM using [98]) and for factor analysis, probit regression, and learning Hidden Markov Models [110].

Theorem 50 (Multivariate Mean Value theorem [126])**.** *Let U be an open set of $\mathbb{R}^d$. For a differentiable functions $f : U \mapsto \mathbb{R}$ it holds that $\forall x, y \in U$, $\exists c$ such that $f(x) - f(y) = \nabla f(c) \cdot (x - y)$.*

Proof. Define $h : [0, 1] \mapsto U$ where $h(t) = x + t(y - x)$. We can define a new function $g(t) := f \circ h = f(x + t(y - x))$. Note that both functions are differentiable, and so its their composition. Therefore, we can compute the derivative of g using the chain rule: $g' = (f \circ h)' = (f' \circ h)h'$. This gives:

$$g'(t) = (\nabla f(h(t)), h'(t)) = (f(x + t(y - a)), y - x)$$

Because of the one-dimensional Mean Value theorem applied to $g'(t)$, we have that $\exists t_0$ such that $g(1) - g(0) = f(y) - f(x) = g'(t_0) = (f(x + t_0(y - x)), y - x)$. Setting $c = x + t_0(y - x)$, we have that $f(y) - f(x) = \nabla f(c) \cdot (y - x)$. The second statement follows directly by Cauchy-Schwarz. □

Theorem 51 (Componentwise *Softmax* function $\sigma_j(v)$ is Lipschitz continuous)**.** *For $d > 2$, let $\sigma_j : \mathbb{R}^d \mapsto (0,1)$ be the* softmax *function defined as $\sigma_j(v) = \frac{e^{v_j}}{\sum_{l=1}^{d} e^{v_l}}$ Then σ_j is Lipschitz continuous, with $K \leq \sqrt{2}$.*

Proof. We need to find the K such that for all $x, y \in \mathbb{R}^d$, we have that $\|\sigma_j(y) - \sigma_j(x)\| \leq K \|y - x\|$. Observing that σ_j is differentiable and that if we apply Cauchy-Schwarz to the statement of the Mean-Value-theorem we derive that $\forall x, y \in U$, $\exists c$ such that $\|f(x) - f(y)\| \leq \|\nabla f(c)\|_F \|x - y\|$. So to show Lipschitz continuity it is enough to select $K \leq \|\nabla \sigma_j\|_F^* = \max_{c \in \mathbb{R}^d} \|\nabla \sigma_j(c)\|$.

The partial derivatives $\frac{d\sigma_j(v)}{dv_i}$ are $\sigma_j(v)(1 - \sigma_j(v))$ if $i = j$ and $-\sigma_i(v)\sigma_j(v)$ otherwise. So $\|\nabla \sigma_j\|_F^2 = \sum_{i=1}^{d-1}(-\sigma(v)_i \sigma_j(v))^2 + \sigma_j(v)^2(1 - \sigma_j(v))^2 \leq \sum_{i=1}^{d-1} \sigma(v)_i \sigma_j(v) + \sigma_j(v)(1 - \sigma_j(v)) \leq \sigma_j(v) \sum_{i=0}^{d-1} \sigma_i(v) + 1 - \sigma_j(v) \leq 2\sigma_j(v) \leq 2$. In our case we can deduce that: $\|\sigma_j(y) - \sigma_j(x)\| \leq \sqrt{2} \|y - x\|$ so $K \leq \sqrt{2}$.

□

6.2 Quantum Expectation-Maximization for GMM

In this section, we present a quantum Expectation-Maximization algorithm to fit a GMM. The algorithm can also be adapted to fit other mixtures models where the probability distributions belong to the exponential family. As the GMM is both intuitive and one of the most widely used mixture models, our results are presented for the GMM case.

An approximate version of GMM Here we define an approximate version of GMM, that we fit with QEM algorithm. The difference between this formalization and the original GMM is simple. Here we make explicit in the model the *approximation error* introduced during the iterations of the training algorithm.

Definition 52 (Approximate GMM)**.** *Let $\gamma^t = (\theta^t, \boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t) = (\theta^t, \mu_1^t \cdots \mu_k^t, \Sigma_1^t \cdots \Sigma_k^t)$ a model fitted by the standard EM algorithm from γ^0 an initial guess of the parameters, i.e. γ^t is the error-free model that standard EM would have returned after t iterations. Starting from the same choice of initial parameters γ^0, fitting a GMM with the QEM algorithm with $\Delta = (\delta_\theta, \delta_\mu)$ means returning a model $\overline{\gamma}^t = (\overline{\theta}^t, \overline{\boldsymbol{\mu}}^t, \overline{\boldsymbol{\Sigma}}^t)$ such that:*

- $\left\| \overline{\theta}^t - \theta^t \right\| < \delta_\theta$,
- $\left\| \overline{\mu_j}^t - \mu_j^t \right\| < \delta_\mu$ *for all $j \in [k]$,*
- $\left\| \overline{\Sigma_j}^t - \Sigma_j^t \right\| \leq \delta_\mu \sqrt{\eta}$ *for all $j \in [k]$.*

Quantum access to the mixture model Here we explain how to load into a quantum computer a GMM and a dataset represented by a matrix V. This is needed for a quantum computer to be able to work with a machine learning model. The definition of quantum access to other kind of models is analogous. For ease of exposure, we define what does it means to have quantum access to a GMM and its dataset. This definition is basically an extension of theorem 8.

Definition 53 (Quantum access to a GMM)**.** *We say that we have quantum access to a GMM of a dataset $V \in \mathbb{R}^{n \times d}$ and model parameters $\theta_j \in \mathbb{R}$, $\mu_j \in \mathbb{R}^d, \Sigma_j \in \mathbb{R}^{d \times d}$ for all $j \in [k]$ if we can perform in time $O(polylog(d))$ the following mappings:*

- $|j\rangle |0\rangle \mapsto |j\rangle |\mu_j\rangle$,
- $|j\rangle |i\rangle |0\rangle \mapsto |j\rangle |i\rangle |\sigma_i^j\rangle$ *for $i \in [d]$ where σ_i^j is the i-th rows of $\Sigma_j \in \mathbb{R}^{d \times d}$,*

- $|i\rangle|0\rangle \mapsto |i\rangle|v_i\rangle$ *for all* $i \in [n]$,
- $|i\rangle|0\rangle|0\rangle \mapsto |i\rangle|vec[v_i v_i^T]\rangle = |i\rangle|v_i\rangle|v_i\rangle$ *for* $i \in [n]$,
- $|j\rangle|0\rangle \mapsto |j\rangle|\theta_j\rangle$.

Algorithm 7 QEM for GMM

Require: Quantum access to a GMM model, precision parameters $\delta_\theta, \delta_\mu$, and threshold ϵ_τ.

Ensure: A GMM $\overline{\gamma}^t$ that maximizes locally the likelihood $\ell(\gamma; V)$, up to tolerance ϵ_τ.

1: Use a heuristic (like lemma 5.3 in Chapter 5) to determine the initial guess $\gamma^0 = (\theta^0, \boldsymbol{\mu}^0, \boldsymbol{\Sigma}^0)$, and build quantum access as in definition 53 those parameters.
2: Use lemma 66 to estimate the log determinant of the matrices $\{\Sigma_j^0\}_{j=1}^k$.
3: t=0
4: **repeat**
5: **Step 1:** Get an estimate of θ^{t+1} using lemma 57 such that $\left\|\overline{\theta}^{t+1} - \theta^{t+1}\right\| \leq \delta_\theta$.
6: **Step 2:** Get an estimate $\{\overline{\mu_j}^{t+1}\}_{j=1}^k$ by using lemma 59 to estimate each $\left\|\mu_j^{t+1}\right\|$ and $|\mu_j^{t+1}\rangle$ such that $\left\|\mu_j^{t+1} - \overline{\mu_j}^{t+1}\right\| \leq \delta_\mu$.
7: **Step 3:** Get an estimate $\{\overline{\Sigma_j}^{t+1}\}_{j=1}^k$ by using lemma 60 to estimate $\left\|\Sigma_j^{t+1}\right\|_F$ and $|\Sigma_j^{t+1}\rangle$ such that $\left\|\Sigma_j^{t+1} - \overline{\Sigma_j}^{t+1}\right\| \leq \delta_\mu\sqrt{\eta}$.
8: **Step 4:** Estimate $\mathbb{E}[\overline{p(v_i;\gamma^{t+1})}]$ up to error $\epsilon_\tau/2$ using theorem 61.
9: **Step 5:** Build quantum access to γ^{t+1}, and use lemma 66 to estimate the determinants $\{\overline{\log det(\Sigma_j^{t+1})}\}_{j=0}^k$.
10: $t = t + 1$
11: **until**

$$|\mathbb{E}[\overline{p(v_i;\gamma^t)}] - \mathbb{E}[\overline{p(v_i;\gamma^{t-1})}]| < \epsilon_\tau$$

12: Return $\overline{\gamma}^t = (\overline{\theta}^t, \overline{\boldsymbol{\mu}}^t, \overline{\boldsymbol{\Sigma}}^t)$

Quantum initialization strategies For the initialization of γ^0 in the quantum algorithm we can use the same initialization strategies as in classical machine learning. For instance, we can use the classical *random EM* initialization strategy for QEM.

A quantum initialization strategy can also be given using the *k-means++* initializion strategy, which we discuss in Chapter 5. It returns k initial guesses for the centroids $c_1^0 \cdots c_k^0$ consistent with the classical algorithm in time $\left(k^2 \frac{2\eta^{1.5}}{\epsilon\sqrt{\mathbb{E}(d^2(v_i,v_j))}}\right)$, where $\mathbb{E}(d^2(v_i,v_j))$ is the average squared distance between two points of the dataset, and ϵ is the tolerance in the distance estimation. From there, we can perform a full round of q-means algorithm and get an estimate for $\mu_1^0 \cdots \mu_k^0$. With q-means and the new centroids store in the QRAM we can create the state

$$|\psi^0\rangle := \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle|l(v_i)\rangle . \tag{6.14}$$

Where $l(v_i)$ is the label of the closest centroid to the i-th point. By sampling $S \in O(d)$ points from this state we get two things. First, from the frequency f_j of the second register we can have an guess of $\theta_j^0 \leftarrow |\mathcal{C}_j|/n \sim f_j/S$. Then, from the first register we can estimate $\Sigma_j^0 \leftarrow \sum_{i\in S}(v_i - \mu_j^0)(v_i - \mu_j^0)^T$. Sampling $O(d)$ points and creating the state in Equation (6.14) takes time $\widetilde{O}(dk\eta)$ by theorem 30 and the minimum finding procedure, i.e. lemma 46.

Techniques illustrated in [108] can also be used to quantize the CEM algorithm which needs a hard-clustering step. Among the different possible approaches, the *random* and the *small EM* greatly benefit from a faster algorithm, as we can spend more time exploring the space of the parameters by starting from different initial seeds, and thus avoid local minima of the likelihood.

6.2.1 Expectation

In this step of the quantum algorithm we are just showing how to compute efficiently the responsibilities as a quantum state. First, we compute the responsibilities in a quantum register, and then we show how to put them as amplitudes of a quantum state. At each iteration of Quantum Expectation-Maximization (specifically, in the Expectation step), we assume to have quantum access to the determinant of the covariance matrices. In Chapter 7 we will detail quantum algorithms for the problem of computing the log-determinant. From the error analysis we will see that the cost of comping the log-determinant of the covariance matrices (even with classical algorithms) is smaller than the cost of the ohter quantum step, we can discard the cost of computing the log-determinant in the analysis of the quatnum algorithms. Thus, we do not explicitly write the time to compute the determinant from now on in the algorithm and when we say that we update Σ we include an update on the estimate of $\log(\det(\Sigma))$ as well. Classical algorithms often depend linearly on $nnz(\Sigma)|\log(\det(\Sigma))|$, which can be upper bounded by $\widetilde{O}(d^2)$, where d is the dimension of the covariance matrix. Note that it is often the case that GMM is run with diagonal covariance matrix, thus making the estimation of the determiant trivial.

Lemma 54 (Quantum Gaussian Evaluation)**.** *Suppose we have stored in the QRAM a matrix $V \in \mathbb{R}^{n\times d}$, the centroid $\mu \in \mathbb{R}^d$ and a SPD covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ of a multivariate Gaussian distribution $\phi(v|\mu,\Sigma)$, such that $\|\Sigma\| \leq 1$. Also assume to have an absolute $\epsilon_1/2$ estimate for $\log(\det(\Sigma))$. Then for $\epsilon_1 > 0$, there exists a quantum algorithm that with probability $1-\gamma$ performs the mapping $U_{G,\epsilon_1} : |i\rangle|0\rangle \to |i\rangle|\overline{s_i}\rangle$ such that $|s_i - \overline{s_i}| < \epsilon_1$, where $s_i = -\frac{1}{2}((v_i-\mu)^T\Sigma^{-1}(v_i-\mu) + d\log 2\pi + \log(det(\Sigma)))$ is the exponent for the Gaussian probability density function in Equation* (6.2)*. The running time of the algorithm is,*

$$T_{G,\epsilon_1} = O\left(\frac{\kappa(\Sigma)\mu(\Sigma)\log(1/\gamma)}{\epsilon_1}\eta\right). \tag{6.15}$$

Proof. We use quantum linear algebra and inner product estimation to estimate the quadratic form $(v_i-\mu)^T\Sigma^{-1}(v_i-\mu)$ to error ϵ_1. We decompose the quadratic form as $v_i^T\Sigma^{-1}v_i - 2v_i^T\Sigma^{-1}\mu + \mu^T\Sigma^{-1}\mu$ and separately approximate each term in the sum to error $\epsilon_1/8$ using lemma 31. The runtime for this operation is $O(\frac{\mu(\Sigma)\kappa(\Sigma)\eta}{\epsilon_1})$. With this, we obtain an estimate for $\frac{1}{2}((v_i-\mu)^T\Sigma^{-1}(v_i-\mu)$ within error ϵ_1. Recall that (through the algorithm in lemma 66 we also have an estimate of the log-determinant to error $\epsilon_1/2$. With these factors, we obtain an approximation for $-\frac{1}{2}((v_i-\mu)^T\Sigma^{-1}(v_i-\mu) + d\log 2\pi + \log(\det(\Sigma)))$ within error ϵ_1. □

Using controlled operations it is simple to extend the previous theorem to work with multiple Gaussians distributions (μ_j,Σ_j). That is, we can control on a register $|j\rangle$ to do $|j\rangle|i\rangle|0\rangle \mapsto |j\rangle|i\rangle|\phi(v_i|\mu_j,\Sigma_j)\rangle$. In the next lemma we will see how to obtain the responsibilities r_{ij} using the previous theorem and standard quantum circuits for doing arithmetic, controlled rotations, and amplitude amplification. The lemma is stated in a general way, to be used with any probability distributions that belong to an exponential family.

Lemma 55 (Error in the responsibilities of the exponential family)**.** *Let $v_i \in \mathbb{R}^n$ be a vector, and let $\{p(v_i|\nu_j)\}_{j=1}^k$ be a set of k probability distributions in the exponential family, defined as $p(v_i|\nu_j) := h_j(v_i)exp\{o_j(\nu_j)^T T_j(v_i) - A_j(\nu_j)\}$. Then, if we have estimates for*

each exponent with error ϵ, then we can compute each r_{ij} such that $|\overline{r_{ij}} - r_{ij}| \leq \sqrt{2k}\epsilon$ for $j \in [k]$.

Proof. The proof follows from rewriting the responsibility of Equation (6.8) and (6.5) as:

$$r_{ij} := \frac{h_j(v_i)\exp\{o_j(\nu_j)^T T(v_i) - A_j(\nu_j) \ + \log\theta_j\}}{\sum_{l=1}^{k} h_l(v_i)\exp\{o_l(\nu_l)^T T(v_i) - A_l(\nu_l) \ + \log\theta_l\}} \tag{6.16}$$

In this form, it is clear that the responsibilities can be seen a *softmax* function, and we can use theorem 51 to bound the error in computing this value.

Let $T_i \in \mathbb{R}^k$ be the vector of the exponent, that is $t_{ij} = o_j(\nu_j)^T T(v_i) - A_j(\nu_j) + \log\theta_j$. In an analogous way we define $\overline{T_i}$ the vector where each component is the estimate with error ϵ. The error in the responsibility is defined as $|r_{ij} - \overline{r_{ij}}| = |\sigma_j(T_i) - \sigma_j(\overline{T_i})|$. Because the function σ_j is Lipschitz continuous, as we proved in theorem 51 with a Lipschitz constant $K \leq \sqrt{2}$, we have that, $|\sigma_j(T_i) - \sigma_j(\overline{T_i})| \leq \sqrt{2}\left\|T_i - \overline{T_i}\right\|$. The result follows as $\left\|T_i - \overline{T_i}\right\| < \sqrt{k}\epsilon$. □

The next lemma provides a quantum algorithm for calculating the responsibilities for the particular case of a Gaussian mixture model.

Lemma 56 (Calculating responsibilities)**.** *Suppose we have quantum access to a GMM with parameters $\gamma^t = (\theta^t, \boldsymbol{\mu}^t, \boldsymbol{\Sigma}^t)$. There are quantum algorithms that can:*

1. *Perform the mapping $|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|\overline{r_{ij}}\rangle$ such that $|\overline{r_{ij}} - r_{ij}| \leq \epsilon_1$ with probability $1-\gamma$ in time:*
$$T_{R_1,\epsilon_1} = \widetilde{O}(k^{1.5} \times T_{G,\epsilon_1})$$
2. *For a given $j \in [k]$, construct state $|\overline{R_j}\rangle$ such that $\left\||\overline{R_j}\rangle - \frac{1}{\sqrt{Z_j}}\sum_{i=0}^{n} r_{ij}|i\rangle\right\| < \epsilon_1$ where $Z_j = \sum_{i=0}^{n} r_{ij}^2$ with high probability in time:*
$$T_{R_2,\epsilon_1} = \widetilde{O}(k^2 \times T_{R_1,\epsilon_1})$$

Proof. For the first statement, let's recall the definition of responsibility: $r_{ij} = \frac{\theta_j \phi(v_i;\mu_j,\Sigma_j)}{\sum_{l=1}^{k}\theta_l\phi(v_i;\mu_l,\Sigma_l)}$. With the aid of U_{G,ϵ_1} of lemma 54 we can estimate $\log(\phi(v_i|\mu_j,\Sigma_j))$ for all j up to additive error ϵ_1, and then using the current estimate of θ^t, we can calculate the responsibilities create the state,

$$\frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i\rangle\left(\bigotimes_{j=1}^{k}|j\rangle|\overline{\log(\phi(v_i|\mu_j,\Sigma_j))}\rangle\right)\otimes|\overline{r_{ij}}\rangle\,.$$

The estimate $\overline{r_{ij}}$ is computed by evaluating a weighted softmax function with arguments $\overline{\log(\phi(v_i|\mu_j,\Sigma_j)}$ for $j \in [k]$. The estimates $\overline{\log(\phi(v_i|\mu_j,\Sigma_j)}$ are then uncomputed. The runtime of the procedure is given by calling k times lemma 54 for Gaussian estimation (the arithmetic operations to calculate the responsibilities are absorbed).

Let us analyze the error in the estimation of r_{ij}. The responsibility r_{ij} is a softmax function with arguments $\log(\phi(v_i|\mu_j,\Sigma_j))$ that are computed up to error ϵ_1 using lemma 54. As the softmax function has a Lipschitz constant $K \leq \sqrt{2}$ by lemma 55, we choose precision for lemma 54 to be $\epsilon_1/\sqrt{2k}$ to get the guarantee $|\overline{r_{ij}} - r_{ij}| \leq \epsilon_1$. Thus, the total cost of this step is $T_{R_1,\epsilon_1} = k^{1.5}T_{G,\epsilon_1}$.

Now let's see how to encode this information in the amplitudes, as stated in the second claim of the lemma. We estimate the responsibilities r_{ij} to some precision ϵ and perform a controlled rotation on an ancillary qubit to obtain,

$$\frac{1}{\sqrt{n}}|j\rangle\sum_{i=0}^{n}|i\rangle|\overline{r_{ij}}\rangle\left(\overline{r_{ij}}|0\rangle + \sqrt{1-\overline{r_{ij}}^2}|1\rangle\right). \tag{6.17}$$

We then undo the circuit on the second register and perform amplitude amplification on the rightmost auxiliary qubit being $|0\rangle$ to get $|\overline{R_j}\rangle := \frac{1}{\|\overline{R_j}\|}\sum_{i=0}^{n}\overline{r_{ij}}\,|i\rangle$. The runtime for amplitude amplification on this task is $O(T_{R_1,\epsilon}\cdot\frac{\sqrt{n}}{\|\overline{R_j}\|})$.

Let us analyze the precision ϵ required to prepare $|\overline{R_j}\rangle$ such that $\left\||R_j\rangle - |\overline{R_j}\rangle\right\| \leq \epsilon_1$. As we have estimates $|r_{ij} - \overline{r_{ij}}| < \epsilon$ for all i, j, the ℓ_2-norm error $\left\|R_j - \overline{R_j}\right\| = \sqrt{\sum_{i=0}^{n}|r_{ij} - \overline{r_{ij}}|^2} < \sqrt{n}\epsilon$. Applying Claim 16, the error for the normalized vector $|R_j\rangle$ can be bounded as $\left\||R_j\rangle - |\overline{R_j}\rangle\right\| < \frac{\sqrt{2n\epsilon}}{\|R_j\|}$. By the Cauchy-Schwarz inequality we have that $\|R_j\| \geq \frac{\sum_i^n r_{ij}}{\sqrt{n}}$. We can use this to obtain a bound $\frac{\sqrt{n}}{\|R_j\|} < \frac{\sqrt{n}}{\sum_i r_{ij}}\sqrt{n} = O(k)$, using the dataset assumptions in section 6.1.3. If we choose ϵ such that $\frac{\sqrt{2n\epsilon}}{\|R_j\|} < \epsilon_1$, that is $\epsilon \leq \epsilon_1/k$ then our runtime becomes $T_{R_2,\epsilon_1} := \widetilde{O}(k^2 \times T_{R_1,\epsilon_1})$.

□

6.2.2 Maximization

Now we need to get a new estimate for the parameters of our model. This is the idea: at each iteration we recover the new parameters from the quantum algorithms as quantum states, and then by performing tomography we can update the QRAM that gives us quantum access to the GMM for the next iteration. In these sections we will show how.

Updating mixing weights θ

Lemma 57 (Computing θ^{t+1})**.** *We assume quantum access to a GMM with parameters γ^t and let $\delta_\theta > 0$ be a precision parameter. There exists an algorithm that estimates $\overline{\theta}^{t+1} \in \mathbb{R}^k$ such that $\left\|\overline{\theta}^{t+1} - \theta^{t+1}\right\| \leq \delta_\theta$ in time*

$$T_\theta = O\left(k^{3.5}\eta^{1.5}\frac{\kappa(\Sigma)\mu(\Sigma)}{\delta_\theta^2}\right)$$

Proof. An estimate of θ_j^{t+1} can be recovered from the following operations. First, we use lemma 56 (part 1) to compute the responsibilities to error ϵ_1, and then perform the following mapping, which consists of a controlled rotation on an auxiliary qubit:

$$\frac{1}{\sqrt{nk}}\sum_{\substack{i=1\\j=1}}^{n,k}|i\rangle\,|j\rangle\,|\overline{r_{ij}}^t\rangle \mapsto \frac{1}{\sqrt{nk}}\sum_{\substack{i=1\\j=1}}^{n,k}|i\rangle\,|j\rangle\left(\sqrt{\overline{r_{ij}}^t}\,|0\rangle + \sqrt{1-\overline{r_{ij}}^t}\,|1\rangle\right)$$

The previous operation has a cost of T_{R_1,ϵ_1}, and the probability of getting $|0\rangle$ is $p(0) = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k}r_{ij}^t = \frac{1}{k}$. Now observe that, by definition, $\theta_j^{t+1} = \frac{1}{n}\sum_{i=1}^{n}r_{ij}^t$. Let $Z_j = \sum_{i=1}^{n}\overline{r_{ij}}^t$ and define state $|\sqrt{R_j}\rangle = \left(\frac{1}{\sqrt{Z_j}}\sum_{i=1}^{n}\sqrt{\overline{r_{ij}}^t}\,|i\rangle\right)|j\rangle$. After amplitude amplification on $|0\rangle$ we have the state,

$$\begin{aligned}
|\sqrt{R}\rangle &:= \frac{1}{\sqrt{n}}\sum_{\substack{i=1\\j=1}}^{n,k}\sqrt{\overline{r_{ij}}^t}\,|i\rangle\,|j\rangle \\
&= \sum_{j=1}^{k}\sqrt{\frac{Z_j}{n}}\left(\frac{1}{\sqrt{Z_j}}\sum_{i=1}^{n}\sqrt{\overline{r_{ij}}^t}\,|i\rangle\right)|j\rangle \\
&= \sum_{j=1}^{k}\sqrt{\overline{\theta_j}^{t+1}}\,|\sqrt{R_j}\rangle\,|j\rangle\,. \qquad (6.18)
\end{aligned}$$

The probability of obtaining outcome $|j\rangle$ if the second register is measured in the standard basis is $p(j) = \overline{\theta_j}^{t+1}$. An estimate for θ_j^{t+1} with precision ϵ can be obtained by either sampling the last register, or by performing amplitude estimation. In this case, we can estimate each of the values θ_j^{t+1} for $j \in [k]$. Sampling requires $O(\epsilon^{-2})$ samples to get epsilon accuracy on θ (by the Chernoff bounds), but does not incur any dependence on k. As the number of cluster k is relatively small compared to $1/\epsilon$, we chose to do amplitude estimation to estimate all θ_j^{t+1} for $j \in [k]$ to error $\epsilon/\sqrt{k}$ in time,

$$T_\theta := O\left(k \cdot \frac{\sqrt{k} T_{R_1,\epsilon_1}}{\epsilon}\right). \tag{6.19}$$

We analyze the error in this procedure. The error introduced by the estimation of responsibility in lemma 56 is $|\overline{\theta_j}^{t+1} - \theta_j^{t+1}| = \frac{1}{n}\sum_i |\overline{r_{ij}}^t - r_{ij}^t| \leq \epsilon_1$ for all $j \in [k]$, pushing the error on the vector $\theta^{t+1} \in \mathbb{R}^k$ up to $\left\|\overline{\theta}^{t+1} - \theta^{t+1}\right\| \leq \sqrt{k}\epsilon_1$. The total error in ℓ_2 norm due to amplitude estimation is at most ϵ as it estimates each coordinate of $\overline{\theta_j}^{t+1}$ to error $\epsilon/\sqrt{k}$. As the errors sums up additively, we can use the triangle inequality to bound them. The total error is at most $\epsilon + \sqrt{k}\epsilon_1$. As we require the final error to be upper bounded by $\left\|\overline{\theta}^{t+1} - \theta^{t+1}\right\| < \delta_\theta$, we choose parameters $\sqrt{k}\epsilon_1 < \delta_\theta/2 \Rightarrow \epsilon_1 < \frac{\delta_\theta}{2\sqrt{k}}$ and $\epsilon < \delta_\theta/2$. With these parameters, the overall running time of the quantum procedure is $T_\theta = O(k^{1.5}\frac{T_{R_1,\epsilon_1}}{\epsilon}) = O\left(k^{3.5}\frac{\eta^{1.5}\cdot\kappa^2(\Sigma)\mu(\Sigma)}{\delta_\theta^2}\right)$.

□

Updating the centroids μ_j

We use quantum linear algebra to transform the uniform superposition of responsibilities of the j-th mixture into the new centroid of the j-th Gaussian. Let $R_j^t \in \mathbb{R}^n$ be the vector of responsibilities for a Gaussian j at iteration t. The following claim relates the vectors R_j^t to the centroids μ_j^{t+1}.

Claim 58. *Let $R_j^t \in \mathbb{R}^n$ be the vector of responsibilities of the points for the Gaussian j at time t, i.e. $(R_j^t)_i = r_{ij}^t$. Then $\mu_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}^t v_i}{\sum_{i=1}^n r_{ij}^t} = \frac{V^T R_j^t}{n\theta_j}$.*

The proof is straightforward.

Lemma 59 (Computing $\boldsymbol{\mu_j^{t+1}}$)**.** *We assume we have quantum access to a GMM with parameters γ^t. For a precision parameter $\delta_\mu > 0$, there is a quantum algorithm that calculates $\{\overline{\mu_j}^{t+1}\}_{j=1}^k$ such that for all $j \in [k]$ $\left\|\overline{\mu_j}^{t+1} - \mu_j^{t+1}\right\| \leq \delta_\mu$ in time*

$$T_\mu = \widetilde{O}\left(\frac{kd\eta\kappa(V)(\mu(V) + k^{3.5}\eta^{1.5}\kappa(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\right)$$

Proof. A new centroid μ_j^{t+1} is estimated by first creating an approximation of the state $|R_j^t\rangle$ up to error ϵ_1 in the ℓ_2-norm using part 2 of lemma 56. Then, we use the quantum linear algebra algorithms in theorem 28 to multiply R_j by V^T, and obtain a state $|\overline{\mu_j}^{t+1}\rangle$ along with an estimate for the norm $\left\|V^T R_j^t\right\| = \left\|\overline{\mu_j}^{t+1}\right\|$ with error ϵ_3. The last step of the algorithm consists in estimating the unit vector $|\overline{\mu_j}^{t+1}\rangle$ with precision ϵ_4, using ℓ_2 tomography. The tomography depends linearly on d, which we expect to be bigger than the precision required by the norm estimation. Thus, we assume that the runtime of the norm estimation is absorbed by the runtime of tomography. We obtain a final runtime of $\widetilde{O}\left(k\frac{d}{\epsilon_4^2} \cdot \kappa(V)\left(\mu(V) + T_{R_2,\epsilon_1}\right)\right)$.

Let's now analyze the total error in the estimation of the new centroids. In order to satisfy the condition of the robust GMM of definition 52, we want the error on the

centroids to be bounded by δ_μ. For this, Claim 15 help us choose the parameters such that $\sqrt{\eta}(\epsilon_{tom} + \epsilon_{norm}) = \delta_\mu$. Since the error ϵ_2 for quantum linear algebra appears as a logarithmic factor in the running time, we can choose $\epsilon_2 \ll \epsilon_4$ without affecting the runtime.

Let $\overline{\mu}$ be the classical unit vector obtained after quantum tomography, and $|\widehat{\mu}\rangle$ be the state produced by the quantum linear algebra procedure starting with an approximation of $|R_j^t\rangle$. Using the triangle inequality we have $\||\mu\rangle - \overline{\mu}\| < \left\|\overline{\mu} - |\widehat{\mu}\rangle\right\| + \left\||\widehat{\mu}\rangle - |\mu\rangle\right\| < \epsilon_4 + \epsilon_1 < \epsilon_{tom} < \delta_\mu/2\sqrt{\eta}$. The errors for the norm estimation procedure can be bounded similarly as $|\,\|\mu\| - \overline{\|\mu\|}| < |\,\|\mu\| - \widehat{\|\mu\|}| + |\overline{\|\mu\|} - \widehat{\|\mu\|}| < \epsilon_3 + \epsilon_1 < \epsilon_{norm} \leq \delta_\mu/2\sqrt{\eta}$. We therefore choose parameters $\epsilon_4 = \epsilon_1 = \epsilon_3 \leq \delta_\mu/4\sqrt{\eta}$. Again, as the amplitude estimation step we use for estimating the norms does not depends on d, which is expected to dominate the other parameters, we omit the cost for the amplitude estimation step. We have the more concise expression for the running time of:

$$\widetilde{O}\left(\frac{kd\eta\kappa(V)(\mu(V) + k^{3.5}\eta^{1.5}\kappa(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\right) \tag{6.20}$$

□

Lemma 60 (Computing Σ_j^{t+1})**.** *We assume we have quantum access to a GMM with parameters γ^t. We also have computed estimates $\overline{\mu_j}^{t+1}$ of all centroids such that $\left\|\overline{\mu_j}^{t+1} - \mu_j^{t+1}\right\| \leq \delta_\mu$ for precision parameter $\delta_\mu > 0$. Then, there exists a quantum algorithm that outputs estimates for the new covariance matrices $\{\overline{\Sigma}_j^{t+1}\}_{j=1}^k$ such that $\left\|\Sigma_j^{t+1} - \overline{\Sigma}_j^{t+1}\right\|_F \leq \delta_\mu\sqrt{\eta}$ with high probability, in time,*

$$T_\Sigma := \widetilde{O}\Big(\frac{kd^2\eta\kappa(V)(\mu(V') + \eta^{2.5}k^{3.5}\kappa(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\Big)$$

Proof. It is simple to check, that the update rule of the covariance matrix during the maximization step can be reduced to [110, Exercise 11.2]:

$$\Sigma_j^{t+1} \leftarrow \frac{\sum_{i=1}^n r_{ij}(v_i - \mu_j^{t+1})(v_i - \mu_j^{t+1})^T}{\sum_{i=1}^n r_{ij}} = \frac{\sum_{i=1}^n r_{ij}v_iv_i^T}{n\theta_j} - \mu_j^{t+1}(\mu_j^{t+1})^T \tag{6.21}$$

$$= \Sigma_j' - \mu_j^{t+1}(\mu_j^{t+1})^T \tag{6.22}$$

First, note that we can use the previously obtained estimates of the centroids to compute the outer product $\mu_j^{t+1}(\mu_j^{t+1})^T$ with error $\delta_\mu \|\mu\| \leq \delta_\mu\sqrt{\eta}$. The error in the estimates of the centroids is $\overline{\mu} = \mu + e$ where e is a vector of norm δ_μ. Therefore $\left\|\mu\mu^T - \overline{\mu}\,\overline{\mu}^T\right\| < 2\sqrt{\eta}\delta_\mu + \delta_\mu^2 \leq 3\sqrt{\eta}\delta_\mu$. Because of this, we allow an error of $\sqrt{\eta}\delta_\mu$ also for the term Σ_j'. Now we discuss the procedure for estimating Σ_j'. We estimate $|\mathrm{vec}[\Sigma_j']\rangle$ and $\left\|\mathrm{vec}[\Sigma_j']\right\|$. To do it, we start by using quantum access to the norms and part 1 of lemma 6.8. With them, for a cluster j, we start by creating the state $|j\rangle\frac{1}{\sqrt{n}}\sum_i|i\rangle|r_{ij}\rangle$, Then, we use quantum access to the norms to store them into another register $|j\rangle\frac{1}{\sqrt{n}}\sum_i|i\rangle|r_{ij}\rangle|\|v_i\|\rangle$. Using an ancilla qubit we can obtain perform a rotation, controlled on the responsibilities and the norm, and obtain the following state:

$$|j\rangle\frac{1}{\sqrt{n}}\sum_i^n|i\rangle|r_{ij}\rangle|\|v_i\|\rangle\left(\frac{r_{ij}\|v_i\|}{\sqrt{\eta}}|0\rangle + \gamma|1\rangle\right)$$

We undo the unitary that created the responsibilities in the second register and the query on the norm on the third register, and we perform amplitude amplification on the ancilla qubit being zero. The resulting state can be obtained in time $O(R_{R_1,\epsilon_1}\frac{\sqrt{n\eta}}{\|V_R\|})$, where $\|V_R\|$

is $\sqrt{\sum_i r_{ij}^2 \|v_i\|^2}$. Successively, we query the QRAM for the vectors v_i and we obtain the following state:

$$\frac{1}{V_R}\sum_i r_{ij}\|v_i\||i\rangle|v_i\rangle \tag{6.23}$$

On which we can apply quantum linear algebra subroutine, multiplying the first register with the matrix V^T. This will lead us to the desired state $|\Sigma_j'\rangle$, along with an estimate of its norm.

As the runtime for the norm estimation $\frac{\kappa(V)(\mu(V)+T_{R_2,\epsilon_1}))\log(1/\epsilon_{mult})}{\epsilon_{norms}}$ does not depend on d, we consider it smaller than the runtime for performing tomography. Thus, the runtime for this operation is:

$$O(\frac{d^2\log d}{\epsilon_{tom}^2}\kappa(V)(\mu(V)+T_{R_2,\epsilon_1}))\log(1/\epsilon_{mult})).$$

Let's analyze the error of this procedure. We want a matrix $\overline{\Sigma_j'}$ that is $\sqrt{\eta}\delta_\mu$-close to the correct one: $\left\|\overline{\Sigma_j'}-\Sigma_j'\right\|_F = \left\|\text{vec}[\overline{\Sigma_j'}]-\text{vec}[\Sigma_j']\right\|_2 < \sqrt{\eta}\delta_\mu$. Again, the error due to matrix multiplication can be taken as small as necessary, since is inside a logarithm. From Claim 15, we just need to fix the error of tomography and norm estimation such that $\eta(\epsilon_{unit}+\epsilon_{norms}) < \sqrt{\eta}\delta_\mu$ where we have used η as an upper bound on $\|\Sigma_j\|_F$. For the unit vectors, we require $\left\||\Sigma_j'\rangle-\overline{|\Sigma_j'\rangle}\right\| \leq \left\|\overline{|\Sigma_j'\rangle}-\widehat{|\Sigma_j'\rangle}\right\| + \left\|\widehat{|\Sigma_j'\rangle}-|\Sigma_j'\rangle\right\| < \epsilon_4+\epsilon_1 \leq \eta\epsilon_{unit} \leq \frac{\delta_\mu\sqrt{\eta}}{2}$, where $\overline{|\Sigma_j'\rangle}$ is the error due to tomography and $\widehat{|\Sigma_j'\rangle}$ is the error due to the responsibilities in lemma 56. For this inequality to be true, we choose $\epsilon_4=\epsilon_1<\frac{\delta_\mu/\sqrt{\eta}}{4}$.

The same argument applies to estimating the norm $\left\|\Sigma_j'\right\|$ with relative error : $|\left\|\Sigma_j'\right\| - \overline{\left\|\Sigma_j'\right\|}| \leq |\overline{\left\|\Sigma_j'\right\|}-\widehat{\left\|\Sigma_j'\right\|}| + |\widehat{\left\|\Sigma_j'\right\|}-\left\|\Sigma_j'\right\||<\epsilon+\epsilon_1 \leq \delta_\mu/2\sqrt{\eta}$ (where here ϵ is the error of the amplitude estimation step used in theorem 28 and ϵ_1 is the error in calling lemma 56. Again, we choose $\epsilon=\epsilon_1\leq\frac{\delta_\mu/\sqrt{\eta}}{4}$.

Since the tomography is more costly than the amplitude estimation step, we can disregard the runtime for the norm estimation step. As this operation is repeated k times for the k different covariance matrices, the total runtime of the whole algorithm is given by $\widetilde{O}\Big(\frac{kd^2\eta\kappa(V)(\mu(V)+\eta^2k^{3.5}\kappa(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\Big)$. Let us also recall that for each of new computed covariance matrices, we use lemma 66 to compute an estimate for their log-determinant and this time can be absorbed in the time T_Σ. □

6.2.3 Quantum estimation of log-likelihood

Now we are going to show how it is possible to get an estimate of the log-likelihood using a quantum procedure. A good estimate of the log-likelihood is crucial, as it is used as stopping criteria for the quantum algorithm. Recall that the log-likelihood is defined as:

$$\ell(\gamma;V)=\sum_{i=1}^n\log\sum_{j\in[k]}\theta_j\phi(v_i;\mu_j,\Sigma_j)=\sum_{i=1}^n\log p(v_i;\gamma)$$

Classically, we stop to iterate the EM algorithm when $|\ell(\gamma^t;V)-\ell(\gamma^{t+1};V)|<n\epsilon$, or equivalently, we can set a tolerance on the average increase of the log of the probability: $|\mathbb{E}[\log p(v_i;\gamma^t)]-\mathbb{E}[\log p(v_i;\gamma^{t+1})]|<\epsilon$. In the quantum algorithm it is more practical to estimate $\mathbb{E}[p(v_i;\gamma^t)]=\frac{1}{n}\sum_{i=1}^n p(v_i;\gamma)$. From this we can estimate an upper bound on the log-likelihood (with the help of the the Jensen inequality) as:

$$n\log\mathbb{E}[p(v_i)]=\sum_{i=1}^n\log\mathbb{E}[p(v_i)]\geq\sum_{i=1}^n\log p(v_i)=\ell(\gamma;V)$$

.

Lemma 61 (Quantum estimation of likelihood)**.** *We assume we have quantum access to a GMM with parameters γ^t. For $\epsilon_\tau > 0$, there exists a quantum algorithm that estimates $\mathbb{E}[p(v_i;\gamma^t)]$ with absolute error ϵ_τ in time*

$$T_\ell = \widetilde{O}\left(k^{1.5}\eta^{1.5}\frac{\kappa(\Sigma)\mu(\Sigma)}{\epsilon_\tau^2}\right)$$

Proof. We obtain the likelihood from the ability to compute the value of a Gaussian distribution and quantum arithmetic. Using the mapping of lemma 54 with precision ϵ_1, we can compute $\phi(v_i|\mu_j,\Sigma_j)$ for all the Gaussians. We can build the state $|i\rangle\bigotimes_{j=0}^{k-1}|j\rangle|\overline{p(v_i|j;\gamma_j)}\rangle$. Then, by knowing θ, and by using quantum arithmetic we can compute in a register the probability of a point belonging to the mixture of Gaussian's: $p(v_i;\gamma)=\sum_{j\in[k]}\theta_j p(v_i|j;\gamma)$ (note that this operation require undoing the previous steps). for simplicity, we now drop the notation for the model γ and write $p(v_i)$ instead of $p(v_i;\gamma)$. Doing the previous calculations in a quantum computer, leads to the creation of the state $|i\rangle|p(v_i)\rangle$. To get an estimate of $\mathbb{E}[p(v_i)]$, we perform the mapping $|i\rangle|p(v_i)\rangle\mapsto|i\rangle\left(\sqrt{p(v_i)}|0\rangle+\sqrt{1-p(v_i)}|1\rangle\right)$ and estimate $p(|0\rangle)\simeq\mathbb{E}[p(v_i)]$ with amplitude estimation on the ancilla qubit being zero.

To get a ϵ_τ-estimate of $p(0)$ we need to decide the precision parameter we use for estimating $\overline{p(v_i|j;\gamma)}$ and the precision required by amplitude estimation. Let $\overline{p(0)}$ be the ϵ_1-error introduced by using lemma 54 and $\widehat{p(0)}$ the error introduced by amplitude estimation. Using triangle inequality we set $\left\|p(0)-\widehat{p(0)}\right\|<\left\|\widehat{p(0)}-\overline{p(0)}\right\|+\left\|\overline{p(0)}-p(0)\right\|<\epsilon_\tau$.

To have $|p(0)-\widehat{p(0)}|<\epsilon_\tau$, we should set ϵ_1 such that $|\overline{p(0)}-p(0)|<\epsilon_\tau/4$, and we set the error in amplitude estimation and in the estimation of the probabilities to be $\epsilon_\tau/2$. The runtime of this procedure is therefore:

$$\widetilde{O}\left(k\cdot T_{G,\epsilon_\tau}\cdot\frac{1}{\epsilon_\tau\sqrt{p(0)}}\right)=\widetilde{O}\left(k^{1.5}\eta^{1.5}\cdot\frac{\kappa(\Sigma)\mu(\Sigma)}{\epsilon_\tau^2}\right)$$

□

6.2.4 Quantum MAP estimate of GMM

Maximum likelihood is not the only way to estimate the parameters of a model, and in certain cases might not even be the best one. For instance, in high-dimensional spaces, it is standard practice for ML estimates to overfit. Moreover, it is often the case that we have prior information on the distribution of the parameters, and we would like our models to take this information into account. These issues are often addressed using a Bayesian approach, i.e. by using a so-called Maximum a posteriori estimate (MAP) of a model [110, Section 14.4.2.8]. MAP estimates work by assuming the existence of a *prior* distribution over the parameters γ. The posterior distribution we use as objective function to maximize comes from the Bayes' rule applied on the likelihood, which gives the posterior as a product of the likelihood and the prior, normalized by the evidence. More simply, we use the Bayes' rule on the likelihood function, as $p(\gamma;V)=\frac{p(V;\gamma)p(\gamma)}{p(V)}$. This allows us to treat the model γ as a random variable, and derive from the ML estimate a MAP estimate:

$$\gamma^*_{MAP}=\arg\max_\gamma\sum_{i=1}^n\log p(\gamma|v_i) \tag{6.24}$$

Among the advantages of a MAP estimate over ML is that it avoids overfitting by having a kind of regularization effect on the model [110, Section 6.5]. Another feature consists in injecting into a maximum likelihood model some external information, perhaps from domain experts. This advantage comes at the cost of requiring "good" prior information on the problem, which might be non-trivial. In terms of labelling, a MAP estimates

correspond to a *hard clustering*, where the label of the point v_i is decided according to the following rule:

$$y_i = \arg\max_j r_{ij} = \arg\max_j \log p(v_i|y_i = j;\gamma) + \log p(y_i = j;\gamma) \tag{6.25}$$

Deriving the previous expression is straightforward using the Bayes' rule, and by noting that the softmax is rank-preserving, and we can discard the denominator of r_{ij} - since it does not depend on γ - and it is shared among all the other responsibilities of the points v_i. Thus, from Equation 6.24 we can conveniently derive Equation 6.25 as a proxy for the label. Fitting a model with MAP estimate is commonly done via the EM algorithm as well. The Expectation step of EM remains unchanged, but the update rules of the Maximization step are slightly different. In this work we only discuss the GMM case, for the other cases the interested reader is encouraged to see the relevant literature. For GMM, the prior on the mixing weight is often modeled using the Dirichlet distribution, that is $\theta_j \sim \text{Dir}(\boldsymbol{\alpha})$. For the rest of parameters, we assume that the conjugate prior is of the form $p(\mu_j, \Sigma_j) = NIW(\mu_j, \Sigma_j|\boldsymbol{m}_0, \iota_0, \nu_0, \boldsymbol{S}_0)$, where $\text{NIW}(\mu_j, \Sigma_j)$ is the Normal-inverse-Wishart distribution. The probability density function of the NIW is the product between a multivariate normal $\phi(\mu|m_0, \frac{1}{\iota}\Sigma)$ and a inverse Wishart distribution $\mathcal{W}^{-1}(\Sigma|\boldsymbol{S}_0, \nu_0)$. NIW has as support vectors μ with mean μ_0 and covariance matrices $\frac{1}{\iota}\Sigma$ where Σ is a random variable with inverse Wishart distribution over positive definite matrices. NIW is often the distribution of choice in these cases, as is the conjugate prior of a multivariate normal distribution with unknown mean and covariance matrix. A shorthand notation, let's define $r_j = n\theta_j = \sum_{i=1}^n r_{ij}$. As in [110], we also denote with $\overline{x_j}^{t+1}$ and $\overline{S_j}^{t+1}$ the maximum likelihood estimate of the parameters $(\mu_j^{t+1})_{ML}$ and $(\Sigma_j^{t+1})_{ML}$. For MAP, the update rules are the following:

$$\theta_j^{t+1} \leftarrow \frac{r_j + \alpha_j - 1}{n + \sum_j \alpha_j - k} \tag{6.26}$$

$$\mu_j^{t+1} \leftarrow \frac{r_j \overline{x_j}^{t+1} + \iota_0 \boldsymbol{m}_0}{r_j + \iota_0} \tag{6.27}$$

$$\Sigma_j^{t+1} \leftarrow \frac{\boldsymbol{S}_0 + \overline{S_j}^{t+1} + \frac{\iota_0 r_j}{\iota_0 + r_j}(\overline{x_j}^{t+1} - \boldsymbol{m}_0)(\overline{x_j}^{t+1} - \boldsymbol{m}_0)^T}{\nu_0 + r_k + d + 2} \tag{6.28}$$

Where the matrix $\boldsymbol{S}_0$ is defined as:

$$\boldsymbol{S}_0 := \frac{1}{k^{1/d}} Diag(s_1^2, \cdots, s_d^2), \tag{6.29}$$

where each value s_j is computed as $s_j := \frac{1}{n}\sum_{i=1}^n (x_{ij} - \sum_{i=1}^n x_{ij}))^2$ which is the pooled variance for each of the dimension j. For more information on the advantages, disadvantages, and common choice of parameters of a MAP estimate, we refer the interested reader to [110]. Using the QEM algorithm to fit a MAP estimate is straightforward, since once the ML estimate of the parameter is recovered from the quantum procedures, the update rules can be computed classically.

Corollary 62 (QEM for MAP estimates of GMM)**.** *We assume we have quantum access to a GMM with parameters γ^t. For parameters $\delta_\theta, \delta_\mu, \epsilon_\tau > 0$, the running time of one iteration of the quantum maximum a posteriori (QMAP) algorithm is*

$$O(T_\theta + T_\mu + T_\Sigma + T_\ell),$$

for

$$\begin{aligned} T_\theta &= \widetilde{O}\left(k^{3.5}\eta^{1.5}\frac{\kappa(\Sigma)\mu(\Sigma)}{\delta_\theta^2}\right) \\ T_\mu &= \widetilde{O}\left(\frac{kd\eta\kappa(V)(\mu(V)+k^{3.5}\eta^{1.5}\kappa(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\right) \\ T_\Sigma &= \widetilde{O}\Big(\frac{kd^2\eta\kappa^2(V)(\mu(V')+\eta^{2.5}k^{3.5}\kappa(\Sigma)\mu(\Sigma))}{\delta_\mu^3}\Big) \\ T_\ell &= \widetilde{O}\left(k^{1.5}\eta^{1.5}\frac{\kappa(\Sigma)\mu(\Sigma)}{\epsilon_{tau}^2}\right) \end{aligned}$$

For the range of parameters of interest, the running time is dominated by T_Σ.

6.3 Experiments

In this section, we present the results of some experiments on real datasets to estimate the runtime of the algorithm, and bound the value of the parameters that governs the runtime, like $\kappa(\Sigma)$, $\kappa(V)$, $\mu(\Sigma)$, $\mu(V)$, δ_θ, and δ_μ, and we give heuristic for dealing with the condition number. We can put a threshold on the condition number of the matrices Σ_j, by discarding singular values which are smaller than a certain threshold. This might decrease the runtime of the algorithm without impacting its performances. This is indeed done often in classical machine learning models, since discarding the eigenvalues smaller than a certain threshold might even improve upon the metric under consideration (i.e. often the accuracy), by acting as a form of regularization [110, Section 6.5]. This is equivalent to limiting the eccentricity of the Gaussians. We can do similar considerations for putting a threshold on the condition number of the dataset $\kappa(V)$. Recall that the value of the condition number of the matrix V is approximately $1/\min(\{\theta_1,\cdots,\theta_k\}\cup\{d_{st}(\mathcal{N}(\mu_i,\Sigma_i),\mathcal{N}(\mu_j,\Sigma_j))|i\neq j\in[k]\})$, where d_{st} is the statistical distance between two Gaussian distributions [85]. We have some choice in picking the definition for μ: in previous experiments it has been found that choosing the maximum ℓ_1 norm of the rows of V lead to values of $\mu(V)$ around 10 for the MNIST dataset [89, 93]. Because of the way μ is defined, its value will not increase significantly as we add vectors to the training set. In case the matrix V can be clustered with high-enough accuracy by distance-based algorithms like k-means, it has been showed that the Frobenius norm of the matrix is proportional to $\sqrt{k}$, that is, the rank of the matrix depends on the number of different classes contained in the data. Given that EM is just a more powerful extension of k-means, we can rely on similar observations too. Usually, the number of features d is much more than the number of components in the mixture, i.e. $d \gg k$, so we expect d^2 to dominate the $k^{3.5}$ term in the cost needed to estimate the mixing weights, thus making T_Σ the leading term in the runtime. We expect this cost to be be mitigated by using ℓ_∞ form of tomography but we defer further experiment for future research.

As we said, the quantum running time saves the factor that depends on the number of samples and introduces a number of other parameters. Using our experimental results we can see that when the number of samples is large enough one can expect the quantum running time to be faster than the classical one. One may also save some more factors from the quantum running time with a more careful analysis.

To estimate the runtime of the algorithm, we need to gauge the value of the parameters δ_μ and δ_θ, such that they are small enough so that the likelihood is perturbed less than ϵ_τ, but big enough to have a fast algorithm. We have reasons to believe that on well-clusterable data, the value of these parameters will be large enough, such as not to impact dramatically the runtime. A quantum version of k-means algorithm has already been simulated on the MNIST dataset under similar assumptions [93]. The experiment concluded that, for datasets that are expected to be clustered nicely by this kind of clustering algorithms, the

value of the parameters δ_μ did not decrease by increasing the number of samples nor the number of features. There, the value of δ_μ (which in their case was called just δ) has been kept between 0.2 and 0.5, while retaining a classification accuracy comparable to the classical k-means algorithm. We expect similar behaviour in the GMM case, namely that for large datasets the impact on the runtime of the errors $(\delta_\mu, \delta_\theta)$ does not cancel out the exponential gain in the dependence on the number of samples, and we discuss more about this in the next paragraph. The value of ϵ_τ is usually (for instance in scikit-learn [116]) chosen to be 10^{-3}. We will see that the value of η has always been 10 on average, with a maximum of 105 in the experiments.

	MAP		ML	
	avg	max	avg	max
$\|\Sigma\|_2$	0.22	2.45	1.31	3.44
$\|\log\det(\Sigma)\|$	58.56	70.08	14.56	92.3
$\kappa^*(\Sigma)$	4.21	50	15.57	50
$\mu(\Sigma)$	3.82	4.35	2.54	3.67
$\mu(V)$	2.14	2.79	2.14	2.79
$\kappa(V)$	23.82	40.38	23.82	40.38

Table 6.1: We estimate some of the parameters of the VoxForge [148] dataset. Each model is the result of the best of 3 different initializations of the EM algorithm. The first and the second rows are the maximum singular values of all the covariance matrices, and the absolute value of the log-determinant. The column $\kappa^*(\Sigma)$ shows the condition number after the smallest singular values have been discarded.

6.4 Experiments

We analyzed a dataset which can be fitted well with the EM algorithm [125, 10, 148]. Specifically, we used EM to do speaker recognition: the task of recognizing a speaker from a voice sample, having access to a training set of recorded voices of all the possible speakers. The training set consist in 5 speech utterances for 38 speakers (i.e. clips of a few seconds of voice speech). For each speaker, we extract the mel-frequency cepstral coefficients (MFCC) of the utterances [125], resulting in circa 5000 vectors of 40 dimensions. This represent the training set for each speaker. A speaker is then modeled with a mixture of 16 different diagonal Gaussians. The test set consists of other 5 or more unseen utterances for each of the same speakers. To label an utterance with a speaker, we compute the log-likelihood of the utterance for each trained model. The label consist in the speaker with highest log-likelihood. The experiment has been carried in form of classical simulation on a laptop computer. We repeated the experiment using a perturbed model, where we added some noise to the GMM at each iteration of the training, as in definition 52. Then we measured the accuracy of the speaker recognition task. At last, we measured condition number, the absolute value of the log-determinant, and the value of $\mu(V)$ and $\mu(\Sigma)$. In this way we can test the stability and accuracy of the approximate GMM model introduced in Section 6.2, under the effect of noise. For values of $\delta_\theta = 0.038$, $\delta_\mu = 0.5$, we correctly classified 98.7% utterances. The baseline for ML estimate of the GMM is of 97.1%. We attribute the improved accuracy to the regularizing effect of the threshold and the noise, as the standard ML estimate is likely to overfit the data. We report the results of the measurement in Table 6.1.

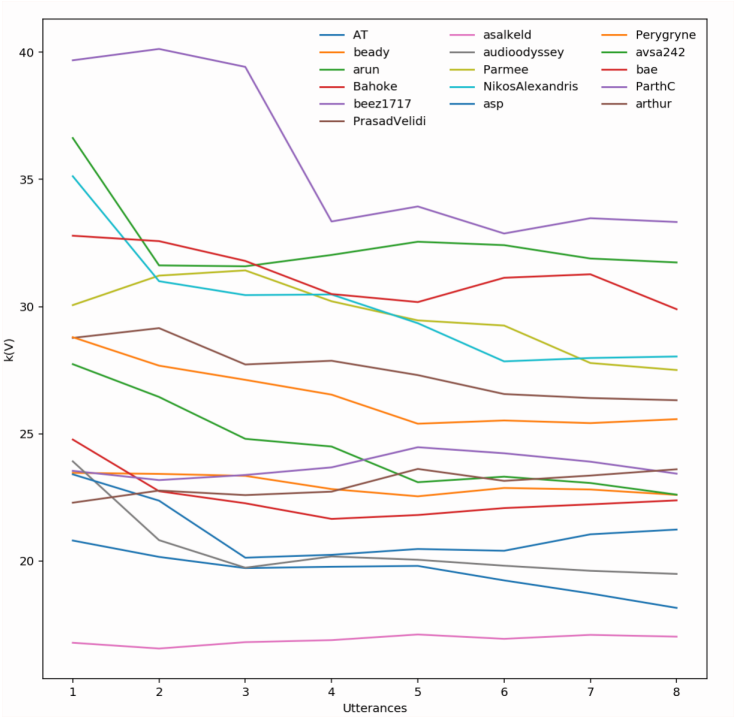
We used a subset of the voices that can be found on VoxForge [148]. The training set consist in at 5 speech utterances from 38 speakers. An utterance is a wav audio clips of a few seconds of voice speech. In order to proceed with speech recognition from raw wav audio files, we need to proceed with classical feature extraction procedures. In the speech

recognition community is common to extract from audio the Mel Frequency Cepstrum Coefficients (MFCCs) features [125], and we followed the same approach. We selected $d = 40$ features for each speaker. This classical procedure, takes as input an audio file, and return a matrix where each row represent a point in $\mathbb{R}^{40}$, and each row represent a small window of audio file of a few milliseconds. Due to the differences in the speakers' audio data, the different dataset $V_1 \dots V_{38}$ are made of a variable number of points which ranges from $n = 2000$ to 4000. Then, each speaker is modeled with a mixture of 16 Gaussians with diagonal covariance matrix. The test set consists of other 5 (or more) unseen utterances of the same 38 speakers. The task is to correctly label the unseen utterances with the name of the correct speaker. This is done by testing each of the GMM fitted during the training against the new test sample. The selected model is the one with the highest likelihood. In the experiments, we compared the performances of classical and quantum algorithm, and measured the relevant parameters that govern the runtime of the quantum algorithm. We used scikit-learn [116] to run all the experiments.

We also simulated the impact of noise during the training of the the GMM fitted with ML estimate, so to assure the convergence of the quantum algorithm. For almost all GMM fitted using 16 diagonal covariance matrices, there is at least a Σ_j with bad condition number (i.e up to 2500 circa). As in [93, 89] we took a threshold on the matrix by discarding singular values smaller than a certain value. Practically, we discarded any singular value smaller than 0.07. In the experiment, thresholding the covariance matrices not only did not made the accuracy worse, but had also a positive impact on the accuracy, perhaps because it has a regularizing effect on the model. For each of the GMM γ^t estimated with ML estimate, we perturbed γ at each iteration. Then, we measured the accuracy on the test set. For each model, the perturbation consists of three things. First we add to each of the components of θ some noise from the truncated Gaussian distribution centered in θ_i in the interval $(\theta_i - \delta_\theta/\sqrt{k}, \theta_i + \delta_\theta/\sqrt{k})$ with unit variance. This can guarantee that overall, the error in the vector of the mixing weights is smaller than δ_θ. Then we perturb each of the components of the centroids μ_j with Gaussian noise centered in $(\mu_j)_i$ on the interval $((\mu_j)_i - \frac{\delta_\mu}{\sqrt{d}}), (\mu_j)_i + \frac{\delta_\mu}{\sqrt{d}})$. Similarly, we perturbed also the diagonal matrices Σ_j with a vector of norm smaller than $\delta_\mu\sqrt{\eta}$, where $\eta = 10$. As we are using a diagonal GMM, this reduces to perturbing each singular value with Gaussian noise from a truncated Gaussian centered Σ on the interval $((\Sigma_j)_{ii} - \delta_\mu\sqrt{\eta}/\sqrt{d}, (\Sigma_j)_{ii} + \delta_\mu\sqrt{\eta}/\sqrt{d})$. Then, we made sure that each of the singular values stays positive, as covariance matrices are SPD. Last, the matrices are thresholded, i.e. the eigenvalues smaller than a certain threshold σ_τ are set to σ_τ. This is done in order to make sure that the effective condition number $\kappa(\Sigma)$ is no bigger than a threshold κ_τ. With a $\epsilon_\tau = 7 \times 10^{-3}$ and 70 iterations per initialization, all runs of the 7 different initialization of classical and quantum EM converged. Once the training has terminated, we measured all the values of $\kappa(\Sigma), \kappa(V), \mu(V), \mu(\Sigma), \log\det(\Sigma)$ for both ML and for MAP estimate. The results are in the Table 6.1. Notably, thresholding the Σ_j help to mitigate the errors of noise as it regularized the model. In fact, using classical EM with ML estimation, we reached an accuracy of 97.1%. With parameters of $\delta_\mu = 0.5$, $\delta_\theta = 0.038$, and a threshold on the condition number of the covariance matrices of Σ_j of 0.07, we reached an accuracy of 98.7%.

We further analyzed experimentally the evolution of the condition number $\kappa(V_i)$ while adding vectors from all the utterances of the speakers to the training set V_i. As we can see from Figure 6.1, all the condition numbers are pretty stable and do not increase by adding new vectors to the various training sets $V_1, \dots V_n$.

Figure 6.1: Evolution of $\kappa(V_i)$ where V_i is the data matrix obtained by all the utterances available from the i-th speaker to the training set. For all the different speaker, the condition number of the matrix V_i is stable, and does not increase while adding vectors to the training set.



6.5 Conclusions

Given the tremendous importance of the classical Expectation-Maximization algorithm, we believe it is very important to consider quantum versions of EM. Here we proposed a quantum Expectation-Maximization algorithm, and showed how to use it to fit a GMM. We analyzed theoretically the asymptotic runtime of QEM, and estimated it on real-world datasets, in order to better understand cases where quantum computers can offer a computational advantage. This work enlarges the possible applications of quantum computing in the area of unsupervised learning. We leave for future work the task of testing the algorithm with further experiments (i.e. bigger and different types of datasets), and further optimizations, like procedures for hyperparameter tuning.

Chapter 7

Algorithms for log-determinant and its applications

In this chapter we better analyze previous quantum and classical algorithms for computing the log-determinant of SPD matrices, propose new and faster quantum algorithms that cover a broader class of matrices, and explore the application thereof. In this section, we give quantum algorithms for estimating the following quantity.

Definition 63 (Log-determinant of A)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a real symmetric positive definite (SPD) matrix. Let $\sigma_1, \ldots, \sigma_n$ be the singular values of A. Then the log-determinant of A is defined by:*

$$\log\det(A) := \sum_{j=1}^{n} \log \sigma_j. \tag{7.1}$$

Observe that, while the determinant of a SPD matrix is always positive (because the eigenvalues are all positive), the log-determinant can be either positive or negative. Under the assumption that the singular values of the matrix lies in the interval $(0, 1]$, the log-determinant is always a negative quantity. In case $\|A'\| \geq 1$, we define a matrix $A = A'/\alpha$ where $\alpha \geq \sigma_1(A')$. Then, we can recover the log-determinant of A' as [32, lemma 5]:

$$\log\det(A') = n\log(\alpha) + \log\det(A) \tag{7.2}$$

7.1 Classical algorithms for the log-determinant

The problem of approximating the log-determinant has been widely studied in classical linear algebra and machine learning. Here we report some results which have been used in the first formulation of the quantum Expectation-Maximization for Gaussian mixture models. The idea of exploiting the dualism between Taylor series approximation and matrix powers of approximation has been exploited in [33], where the authors were able to prove the following theorem.

Theorem 64 (Determinant estimation - Boutsidis [33])**.** *Let $M \in \mathbb{R}^{n\times n}$ be a positive definite matrix with eigenvalues in $(\sigma_{min}, 1)$. Then, for all $\delta \in (0, 1)$ and $\epsilon > 0$ there is a classical algorithms that outputs $\overline{\log\det(M))}$ such that $|\overline{\log\det(M))} - \log\det(M)| \leq 2\epsilon|\log\det(M)|$ with probability at least $1-\delta$ in time: $T_{Det,\epsilon} := O(\frac{\log(1/\epsilon)\log(1/\delta)}{\epsilon^2\sigma_{min}}\|M\|_0)$*

Subsequently, [78] managed to save a further factor of $\sqrt{\kappa(\Sigma)}$ by using different algorithmic techniques and polynomial approximation. The authors replaced the Taylor approximation with Chebyshev approximation, leading to one of the fastest classical algorithms for the log-determinant.

Theorem 65 (Determinant estimation - Han [78])**.** *Let $M \in \mathbb{R}^{n\times n}$ be a positive definite matrix with eigenvalues in $(\sigma_{min}, 1)$. Then, for all $\delta \in (0,1)$ and $\epsilon > 0$ there is a classical algorithms that outputs $\overline{\log\det(M)}$ such that $|\overline{\log\det(M)} - \log\det(M)| \leq 2\epsilon|\log\det(M)|$ with probability at least $1-\delta$ in time: $T_{Det,\epsilon} := O(\frac{\log(1/\epsilon)\log(1/\delta)}{\epsilon^2\sigma_{min}}\|M\|_0)$*

Given access to an algorithm that estimates the log-determinant of a normalized matrix (i.e. when $\|A\| \leq 1$) with relative error, we can estimate the log-determinant of *non*-normalized matrices with *absolute* error.

Lemma 66 (Log-determinant evaluation with absolute error)**.** *Let A be an algorithm that runs in time T and returns an estimate of the log-determinant of a SPD matrix A such that $\|A\| \leq 1$ with multiplicative error. Then, there is an algorithm that, given as input a SPD matrix Σ, $\epsilon > 0$, and $0 < \delta < 1$, outputs an estimate $\overline{\log\det(\Sigma)}$ such that $|\overline{\log(det(\Sigma))} - \log(det(\Sigma))| \leq \epsilon$ with probability $1-\delta$ in time:*

$$O\left(T\log\det(\Sigma)|\right)$$

Proof. In order to apply Algorithm A, we need to be sure that all the eigenvalues lie in $(\sigma_{min}, 1)$. In order to satisfy this condition, we can scale the matrix by a constant factor c, such that $\Sigma' = \Sigma/c$. In this way, $\log det(\Sigma') = \log\prod_i^d(\sigma_i/c)$. Therefore, $\log(det(\Sigma')) = \log(det(\Sigma)) - \log(c^d)$. This will allow us to recover the value of $\log det(\Sigma)$ by Algorithm A. We run the Algorithm with precision $\epsilon = 1/4$ to get an estimate γ such that $|\log\det(\Sigma') - \overline{\log\det(\Sigma')}| \leq 2\epsilon|\log\det(\Sigma')|$. Note that this means also that $\frac{1}{2} \leq \frac{\gamma}{\log(\det(\Sigma))} \leq \frac{3}{2}$. Then, to have an estimate with absolute error ϵ, we apply again Algorithm A with precision $\epsilon' = \frac{\epsilon}{4\gamma}$. This concludes the proof by noting that we have an estimate for $\log(\det(\Sigma))$ with error $2\epsilon'\log(\det(\Sigma)) \leq \epsilon$ in time:

$$O\left(T|\log\det(\Sigma))|\right).$$

Note that the estimate of $\|\Sigma\|$ can be estimated with a quantum algorithm with scaling $O(\frac{\|\Sigma\|_2}{\|\Sigma\|_F\epsilon}))$ which is bounded by the runtime of the algorithm for the log-determinant estimation. [91, Algorithm 4.3] □

We can use theorem 65 with lemma 66 together, and obtain the following classical algorithm for the log-determinant:

Corollary 67 (Classical log-determinant estimation)**.** *There is an algorithm that, given as input a SPD matrix Σ, $\epsilon > 0$, and $0 < \delta < 1$, outputs an estimate $\overline{\log\det(\Sigma)}$ such that $|\overline{\log det(\Sigma)} - \log\det(\Sigma))| \leq \epsilon$ with probability $1-\delta$ in time:*

$$\widetilde{O}\left(\epsilon^{-2}\sqrt{\kappa}(\Sigma)\log(1/\delta)nnz(\Sigma)|\log\det(A)\right)$$

In the following, we show how to use a quantum computer to obtain faster algorithms for estimating the log-determinant. For this, we reduce the problem of estimating the log-determinant to estimating the trace of a particularly singular value transformed matrix. Thus, we split the task of estimating the trace of a matrix and the task of computing the logarithm of the singular values of the matrix via singular value transformation.

7.2 Trace estimation

The algorithm for trace estimation is based on the observation that $Tr[A] = \frac{1}{n}\sum_{i=1}^n e_i^T A e_i$. Classical algorithms approximate this value by using certain probe vectors which can be either Rademacher vectors (i.e. each entry is $+1$ or -1 with equal probability) or Gaussian distributed. Along with the classical Rademacher and Gaussian estimator, there is another kind of sample vector that has been studied in classical literature, and is more relevant for quantum computers.

In this section we briefly recall the definition of stochastic trace estimator, and recall classical results on unit vector estimation techniques. We will see that even using the unit vector estimator, it is not possible do improve upon the additive error approximation of $n\epsilon$.

Definition 68 ((ϵ, δ)-approximator of trace)**.** *Let A be a symmetric positive semi-definite matrix. A randomized trace estimator T is an (ϵ, δ)-approximator of* $\mathrm{Tr}[A]$ *if*

$$\Pr[|T - \mathrm{Tr}[A]| \leq \epsilon \mathrm{Tr}[A]] \geq 1 - \delta. \tag{7.3}$$

Definition 69 (Unit vector estimator [15])**.** *A unit vector estimator for a SPD matrix $A \in \mathbb{R}^{n \times n}$ is*

$$U_M = \frac{n}{M} \sum_{i=1}^{M} z_i^T A z_i \tag{7.4}$$

where the z_i's are M independent uniform random samples from the standard basis $\{e_1, \dots e_n\}$ of $\mathbb{R}^n$.

It is evident that this estimator does not depend on the off-diagonal elements of A, so the quality of the estimator is affected by the distribution of the elements A_{ii}. If the diagonal elements are uniform, then only one sample will suffice, but if the distribution is skewed, we need many more samples. It is easy to address this difficulty by mixing the matrix so to make the diagonal element more uniform. Instead of computing the trace of A, we compute the trace of $\mathcal{F}A\mathcal{F}^T$, for a unitary matrix $\mathcal{F}$ defined as follows.

Definition 70 (Random mixing matrix)**.** *A random mixing matrix is a unitary matrix $\mathcal{F} = FD \in \mathbb{R}^{n \times n}$ where F is an $n \times n$ fixed unitary matrix called* seed *and D is an $n \times n$ matrix where the diagonal is a Rademacher vector, i.e. D_{ii} is ± 1 with equal probability.*

In this way, we obtained that $\mathcal{F}$ flattens the distribution of all the diagonal of A. We are ready to define a new estimator.

Definition 71 (Mixed unit vector estimator)**.** *A mixed unit-vector estimator for a SPD matrix $A \in \mathbb{R}^{n \times n}$ is*

$$T_M = \frac{n}{M} \sum_{i=1}^{M} z_i^T \mathcal{F} A \mathcal{F}^T z_i \tag{7.5}$$

where the z_i's are M independent uniform random samples from $\{e_1, \dots e_n\}$

The quality of $\mathcal{F}$ is evaluated on the parameter $\eta = \|F\|_{\max}^2$: the smaller η the better the mixing capacity [15]. It is possible to see that this value is reached in case the matrix F is a Discrete Fourier Transform, where this value is $1/n$. There are other possible choices for the seed matrix, but we focus on this one, as performing a QFT in a fault tolerant quantum computer is relatively simple. For reference on the various possible choices of seed matrices, we recommend [15]. In [15, theorem 8.4, remark 8.5] they prove the following result:

Theorem 72 (Mixed Unit Vector estimator with DFT matrix [15])**.** *The mixed unit vector estimator T_M with the DFT matrix as the seed matrix F is an (ϵ, δ)-approximator of* $\mathrm{Tr}[A]$ *for $M \geq 2\epsilon^{-2} \log^2(4n^2/\delta) \ln(4/\delta)$.*

7.2.1 Quantum trace estimation

The advantage of using the mixed stochastic trace estimator with a quantum computer is the following. Using other stochastic trace estimators (like the Rademacher vectors or the Gaussian vectors), we would incur a penalty in the runtime given by the norm of the probe vectors. Using the mixed unit trace estimator, we can remove this dependence. In the following, albeit we use the block-encoding formalism, we drop the variable of the number

of qubits. That is, for a (α, a, ϵ) block-encoding we will just write (α, ϵ). In the following, we put forward a simple algorithm for trace estimation, based on Definition 69, and then compare its performances with the quantum algorithm based on Definition 71.

Lemma 73 (Quantum trace estimation)**.** *Assume to have a $(\mu(A), a, 0)$ block encoding access to a SPD matrix $A \in \mathbb{R}^{n\times n}$. For an $\epsilon > 0$, there is a quantum algorithm that estimates $\overline{\mathrm{Tr}[A]}$ with high probability, such that:*

- $|\mathrm{Tr}[A] - \overline{\mathrm{Tr}[A]}| \leq \epsilon Tr[A]$ *in time:* $\widetilde{O}(\frac{\mu(A)n}{Tr[A]\epsilon}) \leq \widetilde{O}(\frac{\mu(A)\kappa}{\epsilon})$
- $|\mathrm{Tr}[A] - \overline{\mathrm{Tr}[A]}| \leq \epsilon n$ *in time:* $\widetilde{O}(\frac{\mu(A)}{\epsilon})$

Proof. The algorithm is as follows. We apply the block-encoding of A to $\frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i\rangle$, and we obtain $|\psi\rangle = \frac{1}{\sqrt{n}}\sum_{i}^{n}(\frac{1}{\mu(A)}\||a_i\||a_i, 0\rangle + \sqrt{1-\gamma^2}|G, 1\rangle)$, where a_i is the i-th column of A. Then, again, we use quantum subroutine to estimate the inner product between $|\psi\rangle$ and $|\phi\rangle = \frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i, 0\rangle$. It is simple to check that:

$$\langle\psi|\phi\rangle = \frac{\sum_i^n \||a_i\|\langle i|a_i\rangle}{n\mu(A)} = \frac{1}{n}\sum_i^n e_i^T\frac{A}{\mu(A)}e_i = \frac{Tr[A/\mu(A)]}{n}.$$

The error in matrix multiplication can be taken as small as possible as it will affect the runtime only polylogarithmically. In order to estimate $\langle\psi|\phi\rangle$, we use amplitude estimation with relative error ϵ_1 on the Hadamard test. This procedure creates a state $\sqrt{p(0)}|G, 0\rangle + \sqrt{1-p(0)}|B, 1\rangle$, where the probability of measuring $|0\rangle$ on the leftmost qubit is $p(0) = \frac{1+\langle\psi|\phi\rangle}{2}$, and we use the hypothesis that the matrix is SPD, and thus $\langle\psi|\phi\rangle > 0$. Amplitude estimation returns a relative error on $p(0)$, and an estimate of $\langle\psi|\phi\rangle$ with additive error ϵ_1 which we set equal to $\epsilon/\mu(A)$, i.e. $|\overline{\langle\psi|\phi\rangle} - \langle\psi|\phi\rangle| \leq \epsilon_1$, and we to obtain $|\frac{Tr[A/\mu(A)]}{n} - \mu(A)\overline{\langle\psi|\phi\rangle}| \leq \epsilon$ in time $\widetilde{O}(\frac{\mu(A)}{\epsilon})$ (note that the factor $\sqrt{p(0)}$ that should appear at the denominator of the runtime of amplitude estimation in the case of a Hadamard test of a SPD matrix is bounded by 1/2). Multiplying this estimate by n, we obtain the absolute error bound. If we want to obtain a relative error ϵ, we can use amplitude estimation with precision $\epsilon_1 = \frac{\epsilon}{\mu(A)}\frac{Tr[A]}{n}$, which raises the runtime to $\widetilde{O}(\frac{\mu(A)n}{Tr[A]\epsilon}) \leq \widetilde{O}(\frac{\mu(A)\kappa}{\epsilon})$, as $\frac{n}{Tr[A]} \leq \frac{n}{n\sigma_{min}} \leq \kappa$, under the assumption that $\|A\|_2 \leq 1$.

□

Alas, this algorithm has a dependence on n, the dimension of the space. We could hope to amend this by changing the vector we use in the quadratic form. Along with the matrix A, we now assume to have quantum access to a Rademacher matrix. A Rademacher matrix is a diagonal matrix whose entries are ± 1, each with probability 1/2. For a Rademacher matrix D of size $n \times n$, $\tilde{O}(n)$ operations suffice to create quantum access to D. In the following algorithm we also assume to have quantum access to a Rademacher diagonal matrix D. For us, this Rademacher matrix will fulfil the role of the matrix D in the Definition 71 of mixed unit vector estimator.

Lemma 74 (Quantum Mixed unit vector trace estimation)**.** *Assume to have quantum access to a SPD matrix $A \in \mathbb{R}^{n\times n}$ such that $\|A\| \leq 1$, and a diagonal Rademacher matrix $D \in \mathbb{R}^{n\times n}$. There is a quantum algorithm that estimates $\overline{\mathrm{Tr}[A]}$ such that $|\mathrm{Tr}[A] - \overline{\mathrm{Tr}[A]}| \leq \epsilon$ in time:* $\widetilde{O} = (n\frac{\mu(A)}{\epsilon^3})$

Proof. We start by combining quantum access to the matrix D and A. Using just one ancilla qubit, we can store the signs on the diagonal of D on the amplitudes, and create the state

$$\frac{1}{\|A\|_F}\sum_i^n(-1)^i\||a_i\||i\rangle|0\rangle.$$

We use this state as input register to get quantum access to matrix A. Afterward, we use quantum access to D using the second register of $\frac{1}{\|A\|_F}\sum_i^n(-1)^i\|a_i\||i\rangle|a_i\rangle$. Further on, we use the second register as index for D again. This unitary gives quantum access to the matrix $A_D = DAD^T$:

$$\frac{1}{\|A\|_F}\sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{d_i}(-1)^{d_j}a_{ij}|i\rangle|j\rangle.$$

Now, according to theorem 72, we pick $M = 2\epsilon_2^{-2}\log^2(4n^2/\delta)\ln(4/\delta)$, Let $\mathcal{M}$ be a set of M numbers sampled uniformly without repetitions from $[n]$. We create two states, $|\psi_1\rangle = |\psi_2\rangle = \frac{1}{\sqrt{M}}\sum_{j\in\mathcal{M}}^{M}|j\rangle$ upon which we apply the QFT circuit, leading to $|\psi_1\rangle = |\psi_2\rangle = \frac{1}{\sqrt{Mn}}\sum_{j\in\mathcal{M}}^{M}\sum_k^n e^{\frac{2\pi ijk}{n}}|k\rangle$. We use quantum access we just created to matrix A_D to apply matrix multiplication to $|\psi_1\rangle$, and compute the inner product between the resulting state and $|\psi_2\rangle$. We obtain a scalar T such that:

$$\begin{aligned}T :=& \langle\psi_1|A_D\psi_2\rangle = \frac{\sum_i^M\langle i|FD(A/\|A\|_F)D^TF^Ti\rangle}{M} = \frac{1}{M}\sum_{i=0}^{M}e_i^TFD(A/\|A\|_F)D^TF^Te_i\\ =&Tr[A/\|A\|_F]/M\end{aligned} \tag{7.6}$$

We get an estimate of $\overline{\langle\psi_1|A_D/\|A\|_F\,\psi_2\rangle}$ with precision ϵ_1:

$$|\overline{T} - \frac{1}{M}\sum_{i=0}^{M}e_i^TFD(A/\|A\|_F)D^TF^Te_i| \leq \epsilon_1$$

The error in matrix multiplication can be taken as small as needed as it will affect the runtime only polylogarithmically. Thanks to theorem 72 and Definition 71, we obtain an estimator for the trace $\overline{Tr[A]} = nM\overline{T}$ whose error we bound with triangle inequality as:

$$|Tr[A] - \overline{Tr[A]}| \leq |nMT - nM\overline{T}| + |nMT - Tr[A]| \leq \epsilon] \tag{7.7}$$

$$= nM\epsilon_1 + \epsilon_2 Tr[A] \leq \epsilon \tag{7.8}$$

By picking $\epsilon_1 \leq \epsilon/(2nM\|A\|_F)$ and $\epsilon_2 = \epsilon/2$, the runtime for this algorithm is: $O\left(n\frac{\mu(A)}{\epsilon^3}\right)$.

□

7.3 Quantum algorithms for the log-determinant

7.3.1 Previous work

Some years ago, another quantum algorithm for the log-determinant had been put forward [161]. This algorithm leverages quantum computers to sample from the uniform superpositions of singular values of a matrix. Then, once a certain number of samples is collected, the log-determinant is estimated as:

$$n\mathbb{E}[\log\sigma_i] \approx n\frac{1}{n}\sum_{i=0}^{n}\log(\sigma_i) \approx \log\det(A)$$

As the runtime of the algorithm was not thoroughly analyzed in [160], an analysis appeared in [161].

Lemma 75 (Sampling-based algorithm for log-determinant [161])**.** *Let $A \in \mathbb{R}^{n\times n}$ be a SPD matrix for which we have quantum access. Then, there is a quantum algorithm, that estimates $\overline{\log\det(A)}$ such that $|\overline{\log\det(A)} - \log\det(A)| < n\epsilon$ with high probability in time*

$$\widetilde{O}(\mu(A)\kappa(A)/\epsilon^3). \tag{7.9}$$

Proof. The idea of this algorithm is to estimate the log-determinant as $\overline{\log\det(A)} = n\mathbb{E}[\log\sigma_i]$, where $\mathbb{E}[\log\sigma_i] = \frac{1}{n}\sum_{i=1}^{n}\log\sigma_i$. For this, we want to sample from the uniform distribution of singular values of A. Let's recall that for a symmetric matrix A, its eigenvalue decomposition is $A = \sum_j \sigma_j|u_j\rangle\langle u_j|$. We start the algorithm from the maximally mixed state $\rho_0 = \frac{1}{n}\sum_j |j\rangle\langle j|$, which also equals $\frac{1}{n}\sum_j |u_j\rangle\langle u_j|$, as a totally mixed state is totally mixed in any orthogonal basis. By using SVE (or equivalently Hamiltonian simulation and QPE), we can obtain $\rho_1 = \frac{1}{n}\sum_j |u_j\rangle\langle u_j| \otimes |\tilde{\sigma}_j\rangle\langle\tilde{\sigma}_j|$, where $|\sigma_j - \tilde{\sigma}_j| \leq \epsilon_1$ for all j. As consequence, it is simple to check using the Taylor expansion of the logarithm function, that for each singular value, $|\log\sigma_j - \log\tilde{\sigma}_j| \leq \epsilon_1/\sigma_j$. By lemma 20, the complexity to generate ρ_1 is $\widetilde{O}(\mu(A)/\epsilon_1)$. By measuring the second register of ρ_1 we sample from the uniform distribution of the singular values, and we can therefore approximate $\mathbb{E}[\log\tilde{\sigma}_j]$. The error of the quantum procedure in estimating $\mathbb{E}[\log\tilde{\sigma}_j]$ is

$$\left|\frac{1}{n}\sum_{j=1}^{n}(\log\tilde{\sigma}_j - \log\sigma_j)\right| \leq \frac{1}{n}\sum_{j=1}^{n}|\log\tilde{\sigma}_j - \log\sigma_j| \leq \frac{\epsilon_1}{n}\sum_{j=1}^{n}\frac{1}{\sigma_j} \leq \frac{\epsilon_1}{\sigma_n} = \kappa(A)\epsilon_1. \tag{7.10}$$

To approximate $\mathbb{E}[\log\tilde{\sigma}_j]$ to precision ϵ_2, Hoeffding inequality (i.e. lemma 2) tells us that it suffices to make $\widetilde{O}(1/\epsilon_2^2)$ measurements on ρ_1. By picking $\epsilon_1 = \epsilon/2\kappa(A), \epsilon_2 = \epsilon/2$, the error to estimate $\log\det(A)$ is bounded by $n\epsilon$. The complexity is $\widetilde{O}(\kappa(A)\mu(A)/\epsilon^3)$. □

Remark: If we do not use the SVE and assume that A is s sparse and hermitian (as in the original formulation of this algorithm), then we can recover the singular values of A applying phase estimation to a unitary that performs Hamiltonian simulation on A. The complexity to implement e^{-iAt} to precision ϵ' is $O(st\|A\|_{\max} + \log 1/\epsilon')$ [104, 103], and we perform a quantum phase estimation with error $O(1/\sigma_{\max}\epsilon_1)$. Thus the complexity to obtain ρ_1 is $\widetilde{O}(s\|A\|_{\max}/\epsilon_1)$. In this setting we recover the result of [161], as the runtime becomes $\widetilde{O}(s\|A\|_{\max}\kappa(A)/\epsilon^3)$.

Lemma 76 (Variance of Zhao's estimator)**.** *Let $A \in \mathbb{R}^{n\times n}$ such that $\|A\| \leq 1$. Let $\overline{\log\det(A)}$ be the estimate of $\log\det(A)$ using the algorithm described in lemma 75 with error parameter ϵ. Then:*

$$Var(\overline{\log\det(A)}) = \frac{(n\epsilon\log\kappa(A))^2}{12}$$

.

Proof. The Bienaymé formula says that for random variables X_i, $\mathrm{Var}(\sum_i X_i) = \sum_i \mathrm{Var}(X_i)$ under the assumption that the random variables are independent. From this, as we have chosen $m = O(\epsilon^{-2})$ samples, we derive that:

$$\mathrm{Var}(\overline{\log\det(A)}) = \mathrm{Var}(n\frac{1}{m}\sum_{i}^{m}\log\sigma_i) = \frac{n^2}{m^2}\sum_{i}^{m}\mathrm{Var}(\log\sigma_i) = \frac{n^2\mathrm{Var}(\log\sigma_i)}{m}$$

As we sample the σ_i uniformly, we can equivalently assume that we sample $\log\sigma_i$ uniformly, and the variance of the uniform distribution on an interval $[a,b]$ is: $\frac{1}{12}(b-a)^2$. Therefore, for the random variable under consideration the variance is $\frac{1}{12}(\log(\sigma_n)^2)$. □

7.3.2 SVE-based quantum algorithm for the log-determinant

We start our series of quantum algorithms for the log determinant using a pretty direct approach to the problem, i.e. using a quantum algorithm for singular value estimation.

Algorithm 8 SVE-based quantum algorithm for the log-determinant

Require: Quantum access to the SPD matrix $A \in \mathbb{R}^{n\times n}$ such that $\|A\| \leq 1$, $\epsilon \in (0,1)$,.
Ensure: An estimate $|\overline{\log\det(A)} - \log\det(A)| \leq n\epsilon$.

1: Prepare

$$|A\rangle = \frac{1}{\|A\|_F}\sum_{i,j=1}^{n} a_{ij}|i,j\rangle. \tag{7.11}$$

2: Perform SVE by theorem 20 up to precision $\epsilon/\kappa\log\kappa$ and do control rotations to prepare

$$\frac{1}{\|A\|_F}\sum_{j=1}^{n}\sigma_j|u_j\rangle|u_j\rangle|\tilde{\sigma}_j\rangle\left(C\frac{\sqrt{|\log\tilde{\sigma}_j|}}{\tilde{\sigma}_j}|0\rangle + |\perp\rangle\right), \tag{7.12}$$

where $C = \min_j \tilde{\sigma}_j/\sqrt{|\log\tilde{\sigma}_j|} \approx \sigma_{\min}/\sqrt{|\log\sigma_{\min}|} = 1/\kappa(A)\sqrt{\log\kappa(A)}$.

3: Apply amplitude estimation to estimate the probability of $|0\rangle$ to precision $\epsilon/\kappa^2(A)\log\kappa(A)$. Set the result as P.

4: Return $\overline{\log\det(A)} = -\kappa^2(A)(\log\kappa(A))\|A\|_F^2 P$.

Theorem 77 (SVE-based log-determinant)**.** *Let $A \in \mathbb{R}^{n\times n}$ by a SPD matrix such that $\|A\| \leq 1$ for which we have quantum access as in Definition 8. Then, Algorithm 8 estimates $\overline{\log\det(A)}$ such that $|\log\det(A) - \overline{\log\det(A)}| < \log\det(A)\epsilon$ in time*

$$\widetilde{O}(\mu(A)\kappa^3(A)(\log\kappa(A))^2/\epsilon^2). \tag{7.13}$$

Proof. We can rewrite the quantum state encoding the representation of A (which we can create with quantum access to A) as follows:

$$|A\rangle = \frac{1}{\|A\|_F}\sum_{i,j=1}^{n} a_{ij}|i,j\rangle = \frac{1}{\|A\|_F}\sum_{j=1}^{n}\sigma_j|u_j\rangle|u_j\rangle. \tag{7.14}$$

Starting from the state $|A\rangle$, we can apply SVE by theorem 20 to A up to precision ϵ_1, and prepare the state

$$\frac{1}{\|A\|_F}\sum_{j=1}^{n}\sigma_j|u_j\rangle|u_j\rangle|\tilde{\sigma}_j\rangle.$$

Since we assume that $\|A\| \leq 1$, using controlled operations, we can prepare

$$\frac{1}{\|A\|_F}\sum_{i=1}^{n}\sigma_j|u_j\rangle|u_j\rangle\,|\tilde{\sigma}_j\rangle\left(C\frac{\sqrt{-\log\tilde{\sigma}_j}}{\tilde{\sigma}_j}\,|0\rangle + |0^{\perp}\rangle\right), \tag{7.15}$$

where $C = \min_j \tilde{\sigma}_j/\sqrt{|\log\tilde{\sigma}_j|} \approx \sigma_{\min}/\sqrt{|\log\sigma_{\min}|} = 1/\kappa(A)\sqrt{\log\kappa(A)}$. The probability of $|0\rangle$ is

$$P = -\frac{C^2}{\|A\|_F^2}\sum_{j=1}^{n}\frac{\sigma_j^2}{\tilde{\sigma}_j^2}\log\tilde{\sigma}_j. \tag{7.16}$$

(a). *Error analysis.* We choose ϵ_1 such that $\kappa(A)\epsilon_1$ is small. Note that

$$\begin{aligned}
\left|\sum_{j=1}^{n}\frac{\sigma_j^2}{\tilde{\sigma}_j^2}\log\tilde{\sigma}_j-\sum_{j=1}^{n}\log\sigma_j\right| \leq & \left|\sum_{j=1}^{n}\frac{\sigma_j^2}{\tilde{\sigma}_j^2}\log\tilde{\sigma}_j-\sum_{j=1}^{n}\frac{\sigma_j^2}{\tilde{\sigma}_j^2}\log\sigma_j\right| \\
& +\left|\sum_{j=1}^{n}\frac{\sigma_j^2}{\tilde{\sigma}_j^2}\log\sigma_j-\sum_{j=1}^{n}\log\sigma_j\right| \\
\leq & \sum_{j=1}^{n}\frac{\sigma_j^2}{\tilde{\sigma}_j^2}|\log\tilde{\sigma}_j-\log\sigma_j|+ \\
& (2\kappa(A)\epsilon_1+\kappa^2(A)\epsilon_1^2)\sum_{j=1}^{n}|\log\sigma_j| \\
\leq & \sum_{j=1}^{n}(1+\tfrac{\epsilon_1}{\tilde{\sigma}_j})^2(\tfrac{\epsilon_1}{\sigma_j}+O(\tfrac{\epsilon_1^2}{\sigma_j^2}))+ \\
& (2\kappa(A)\epsilon_1+\kappa^2(A)\epsilon_1^2)|\log\det(A)| \\
\leq & n(\kappa(A)\epsilon_1+O(\kappa^2(A)\epsilon_1^2))+(2\kappa(A)\epsilon_1+\kappa^2(A)\epsilon_1^2)|\log\det(A)| \\
= & (n+2|\log\det(A)|)(\kappa(A)\epsilon_1+O(\kappa^2(A)\epsilon_1^2)).
\end{aligned}$$

Denote P' as the ϵ_2 approximation of P obtained by amplitude estimation, then $\|A\|_F^2P'/C^2$ is an $(n+2|\log\det(A)|)(\kappa(A)\epsilon_1+O(\kappa^2(A)\epsilon_1^2))+\epsilon_2\|A\|_F^2/C^2$ approximation of $-\log\det(A)$. Note that

$$\begin{aligned}
& (n+2|\log\det(A)|)(\kappa(A)\epsilon_1+O(\kappa^2(A)\epsilon_1^2))+\epsilon_2\|A\|_F^2/C^2 \\
= & (n+2|\log\det(A)|)(\kappa(A)\epsilon_1+O(\kappa^2(A)\epsilon_1^2))+\epsilon_2\|A\|_F^2\kappa^2(A)\log\kappa(A) \\
\leq & (n+2n\log\kappa(A))(\kappa(A)\epsilon_1+O(\kappa^2(A)\epsilon_1^2))+n\epsilon_2\kappa^2\log\kappa \\
= & O(n\epsilon_1\kappa(A)\log\kappa(A)+n\epsilon_2\kappa^2(A)\log\kappa(A)). \qquad (7.17)
\end{aligned}$$

If we set $n\epsilon_1\kappa(A)\log\kappa(A)+n\epsilon_2\kappa^2(A)\log\kappa(A)=n\epsilon$, then $\epsilon_1=\epsilon/\kappa(A)\log\kappa$ and $\epsilon_2=\epsilon/\kappa^2(A)\log\kappa(A)$.

(b). *Runtime analysis.* The runtime of the algorithm comes from the using of SVE by theorem 20 and the performing of amplitude estimation on the state in equation (7.15). The complexity to obtain the state (7.15) is $\widetilde{O}(\mu(A)/\epsilon_1)$. The complexity to perform amplitude estimation (i.e. theorem 18) is $\widetilde{O}(\mu(A)/\epsilon_1\epsilon_2)=\widetilde{O}(\mu(A)\kappa^3(A)(\log\kappa(A))^2/\epsilon^2)$. We now use the hypothesis that $\|A\|<1$ (say $1/2$), and observe that $|\log\det(A)|\geq|n\log(\sigma_1)|=n\log(1/\sigma_1)$. Thus, we improve the precision in the estimate of the error, and we run Algorithm 77 with precision $\frac{\epsilon}{\log(1/\sigma_1)}$. By running our algorithm with a better precision we can assure the error is now $n\frac{\epsilon}{\log(1/\sigma_1)}\leq\epsilon\log\det(A)$. Hence the new complexity becomes $\widetilde{O}(\mu(A)\kappa(A)\epsilon^{-2}\log(1/\sigma_1))$ but we now can guarantee a relative error with respect to $\log\det(A)$. As the matrix is sub-normalized, i.e. $\|A\|<1$, this would increase the runtime only by a constant factor. □

7.3.3 SVT based algorithm for log-determinant

For computing an estimate of the log-determinant with a better dependence on the parameters that governs the runtime we explore the framework of singular value transformation, which is described in section 2.5.4. When we have a quantum access to A, i.e. when using quantum data structures in theorem 8, we can obtain an $(\mu(A),\log n,\log(\epsilon))$-block-encoding of A, as described in lemma 50 of [70]. More precisely, the error $\log(\epsilon)$ comes from the truncation error performed on the entries of a_{ij}, while creating the quantum accessible data structure (i.e Definition 8). In the algorithms presented in this thesis we implicitly considered this error to be 0. In the next algorithm we made it explicit, in order to have a more general statement.

Theorem 78. *Let $\epsilon>0$. Assume to have a $(\mu(A),a,\epsilon_1)$ block encoding of a SPD matrix $A\in\mathbb{R}^{n\times n}$ with $\|A\|<1$ with $\epsilon_1\leq\frac{\epsilon^2}{\kappa(A)^2\log(\kappa(A)\mu(A))}\mu(A)$. There is a quantum algorithm*

Algorithm 9 Log-determinant estimation using singular value transformation

Require: Block encoding $(\mu(A), \varepsilon)$ for ε as in theorem 78, for a SPD matrix $A \in \mathbb{R}^{n\times n}$ with $\|A\| \leq 1$, $\epsilon \in (0, 1)$.

Ensure: An estimate $|\overline{\log\det(A)} - \log\det(A)| \leq \log\det(A)\epsilon$.

1: Set $\epsilon_2 =$ and $\epsilon_3 = \epsilon/8\log(2\kappa(A))$,

2: Construct the polynomial $\tilde{S}(x)$ in lemma 3 with accuracy ϵ_3.

3: Prepare

$$|\psi_0\rangle = \frac{1}{\sqrt{n}}\sum_{k=1}^{n}|k\rangle|0\rangle.$$

4: Construct the block-encoding of $\tilde{S}(A/\mu(A))$ to precision $4\kappa(A)\sqrt{\epsilon_1/\mu(A)}$ to create the state:

$$|\psi_1\rangle = \frac{1}{\sqrt{n}}\sum_{k=1}^{n}\tilde{S}(A/\mu(A))|k\rangle|0\rangle + \gamma|0\rangle^{\perp}.$$

5: Apply inner product estimation subroutines, i.e. lemma 30 with precision $\epsilon_2/\log(\sigma_1)$ to estimate $\langle\psi_0|\psi_1\rangle$. Denote the result as P.

6: Return $\overline{\log\det(A)} = 2nP\log(2\kappa(A)) + n\log\mu(A)$.

that returns the estimate $\overline{\log\det(A)}$ *such that* $|\overline{\log\det(A)} - \log\det(A)| \leq \epsilon\log\det(A)$ *in time* $\widetilde{O}(\mu(A)\kappa(A)/\epsilon)$.

Proof. We define as $\tilde{S}$ the polynomial approximation of the logarithm function (which we reported in lemma 3) with error ϵ_3 . Combined with lemma 25 - which allows to create a block encoding of a matrix where we applied a polynomial function to its singular values - we can find an $(1, a+2, 4\kappa(A)\sqrt{\epsilon_1/\mu(A)}\log(1/\epsilon))$ block-encoding of $\tilde{S}(A/\mu(A))$ with circuit complexity of $O(\kappa(A)\mu(A)\text{poly}\log(1/\epsilon_3)) = \widetilde{O}(\kappa(A)\mu(A))$. Remark that the polynomial approximation induces an error in the estimate of the log-determinant. We can bound this error, by analyzing the approximation of the singular values of the matrix $\tilde{S}(A/\mu(A))$ as:

$$\begin{aligned}\left|\text{Tr}[\tilde{S}(A/\mu(A))] - \sum_{i}^{n}\frac{\log(\sigma_i/\mu(A))}{2\log(\mu(A)\kappa(A))}\right| &= \left|\text{Tr}[\tilde{S}(A/\mu(A))] - \frac{\log\det(A) - n\log(\mu(A))}{2\log(\mu(A)\kappa(A))}\right| \\ &\leq \sum_{j=1}^{n}\left|\tilde{S}(\sigma_j/\mu(A)) - \frac{\log(\sigma_j/\mu(A))}{2\log(\mu(A)\kappa(A))}\right| \\ &\leq n\epsilon_3. \end{aligned} \tag{7.18}$$

From the states defined in Algorithm 9, it is simple to note that:

$$P = \langle\psi_0|\psi_1\rangle = \frac{1}{n}\sum_{k=1}^{n}\langle k|\tilde{S}(A/\mu(A))|k\rangle = \frac{\text{Tr}\tilde{S}(A/\mu(A))}{n}.$$

Let $\overline{P}$ be an ϵ_2-approximation of P obtained by an inner product estimation subroutine, (i.e. lemma 30). We set $\overline{\log\det(A)} = 2n\overline{P}\log(2\kappa(A)\mu(A)) + n\log\mu(A)$ as an estimate of $\log\det(A)$. We use polynomial approximation of logarithm in lemma 3 with error ϵ_3.

The error is given by three things: the error in the block encoding of the polynomial approximation, i.e. theorem 25, which is $4\kappa(A)\sqrt{\epsilon_1/\mu(A)}$, and affects the creation of $|\psi_1\rangle$. The error ϵ_2 in inner product estimation subroutines, that gives $|P - \overline{P}| \leq \epsilon_2$, and the error induced by the polynomial approximation on the trace, i.e. Equation (7.18). These errors sums up additively, and thus we can use the triangle inequality to bound them, i.e.

choose ϵ_2, ϵ_3 such that the whole error is bounded by $n\epsilon$.

$$\begin{aligned}
\left|\log\det(A) - \overline{\log\det(A)}\right| &= 2n\log(2\kappa(A)\mu(A))\left|\frac{\log\det(A) - n\log\mu(A)}{2n\log(2\kappa(A)\mu(A))} - \overline{P}\right| \\
&\leq 2n\log(2\kappa(A)\mu(A))\left|\frac{\log\det(A) - n\log\mu}{2n\log(2\kappa(A)\mu(A))} - \frac{\mathrm{Tr}\tilde{S}(A/\mu(A))}{n}\right| \\
&\quad + 2n\log(2\kappa(A)\mu(A))\left|\langle\psi_0|\overline{\psi_1}\rangle - \frac{\mathrm{Tr}\tilde{S}(A/\mu)}{n}\right| \\
&\quad + 2n\log(2\kappa(A)\mu(A))\left|P - \overline{P}\right| \\
&\leq 2n\log(2\kappa(A)\mu(A))\Big(\epsilon_3 + 4\kappa(A)\sqrt{\epsilon_1/\mu} + \epsilon_2\Big). \\
&\leq n\epsilon
\end{aligned}$$

If we choose $\epsilon_2 = \epsilon_3 = \epsilon/8\log(2\kappa(A)\mu(A))$, then the error is bounded by $n\epsilon$. The cost to create a block-encoding of $\tilde{S}(A/\mu(A))$ is $O(\mu(A)\kappa(A)\log(1/\epsilon_3))$. The total cost of the algorithm to get an error of ϵn is $O(\frac{\mu(A)\kappa(A)\log(1/\epsilon_3)}{\epsilon_2}) = \widetilde{O}(\frac{\mu(A)\kappa(A)}{\epsilon})$.

We now use the hypothesis that $\|A\| < 1$ (say $1/2$), and observe that $|\log\det(A)| \geq |n\log(\sigma_1)| = n\log(1/\sigma_1)$. Thus, we improve the precision in the estimate of the error, and we run Algorithm 9 with precision $\frac{\epsilon}{\log(1/\sigma_1)}$. By running our algorithm with a better precision we can assure the error is now $n\frac{\epsilon}{\log(1/\sigma_1)} \leq \epsilon\log\det(A)$. Hence the new complexity becomes $\widetilde{O}(\mu(A)\kappa(A)\epsilon^{-1}\log(1/\sigma_1))$ but we now can guarantee a relative error with respect to $\log\det(A)$. As the matrix is sub-normalized, i.e. $\|A\| < 1$, this would increase the runtime only by a constant factor.

□

7.4 Tyler M-estimator

In this section we will discuss quantum algorithms for the problem of estimating high-dimensional covariance matrices. This problem is often classically solved by the so called Tyler M-estimator (TME), which we introduce in this section. The TME is a way to estimate covariance matrices in contexts where the sample covariance matrix, (i.e. the product of the (scaled) dataset X^TX) does not lead to a good estimator. This is the case, for instance, when the data comes from sub-Gaussians and long-tailed distribution, in high-dimensional spaces, or when the data has some outliers. TME outperforms the sample covariance in many different scenarios. Among the many, we cite [62] for finance, [119] for anomaly detection in wireless sensor networks, antenna array processing [114] and radar detection using normalized matched filter [115]. The vast majority of work on covariance matrix estimation addresses the Gaussian scenario, i.e when the points are assumed to come from a Gaussian distribution. When the number of points is bigger than the number of features ($n \gg d$), the sample covariance matrix is a maximum likelihood estimator of the covariance matrix, and it exists with high probability. The problem becomes more challenging when the underlying distribution is non-Gaussian. A first generalization of Gaussian distribution is the elliptical distribution, which has been widely studied in classical statistics, which we discuss later.

TME is the solution of an optimization problem, which is derived from a log-likelihood function (which is defined here as Definition 79. As we will see in this section, quantum algorithms for the log-determinant give a fast algorithm for evaluating the log-likelihood of the TME.

We focus on the task of estimating large covariance matrices in d dimensional spaces, using n samples, for big n and d. In classical statistic, this task has recently gained momentum, as in new datasets the number of samples might be comparable to the number of features. In its original formulation it is defined only when $d < n$. For the case when

$n < d$, there exist some regularized variants which are not studied here. The TME is a $d \times d$ matrix Σ_* which satisfies the following self-consistent equation:

$$\frac{1}{n}\sum_{i}^{n}\frac{x_i x_i^T}{x_i \Sigma_*^{-1} x_i} = \Sigma_*.$$

More formally, the TME is defined as follow:

Definition 79 (Tyler's M-estimator [158, 105, 143])**.** *For a given dataset $X \in \mathbb{R}^{n\times d}$, the Tyler's M-estimator of the covariance matrix of X is defined as:*

$$\Sigma_* = \arg \min_{\substack{Tr[\Sigma]=1 \\ \Sigma \text{ is } SPD}} F(\Sigma) \tag{7.19}$$

where

$$F(\Sigma) = \frac{1}{n}\sum_{i=0}^{n}\log(x_i^T \Sigma^{-1} x) + \frac{1}{d}\log\det(\Sigma) \tag{7.20}$$

Note that the requirement for $Tr[\Sigma] = 1$ is for requiring $F(\Sigma)$ to be invariant to scaling, such that $F(\Sigma) = F(c\Sigma)$, but in some definitions [72], the requirement for $Tr[\Sigma] = 1$ can be relaxed to be $Tr[\Sigma] = p$. The algorithm that estimate Σ_* is obtained by the limit of the sequence Σ_k, where each Σ_k is obtained by the a simple iterative procedure [158, 159, 87]:

Definition 80 (Iteration for Tyler's M-estimator)**.** *Let Σ_k be the TME at iteration k. Then, the TME at iteration $k+1$ is defined as:*

$$\Sigma_{k+1} = \sum_{i=0}^{n}\frac{x_i x_i^T}{x_i^T \Sigma_k^{-1} x_i} / Tr[\sum_{i=0}^{n}\frac{x_i x_i^T}{x_i^T \Sigma_k^{-1} x_i}] \tag{7.21}$$

The bottleneck of the classical algorithm is the calculation of the inverse of Σ_k and the estimation of $\sum_{i=0}^{n} x_i^T \Sigma_k^{-1} x_i$ and thus the cost of the whole algorithm is $O(nd^2)$.

The stopping criteria varies from case to case, and from application to application [158]. A common rule found in literature puts a threshold on the relative change of the Frobenius norm of the TME obtained from two different iterations:

$$\frac{\|\Sigma_k - \Sigma_{k-1}\|_F}{\|\Sigma_k\|_F} \leq \epsilon \tag{7.22}$$

In the next section, we propose a quantum algorithm for computing the TME, and analyze its runtime.

7.4.1 Quantum algorithm for Tyler's M-estimators

We will see that the runtime of this subroutine depends on a parameter γ, which derives from a concentration inequality of some quadratic forms. Subsequently, we analyze a specific context where this lemma is used, and argue that in practice the value of γ, should be bounded.

Assumption on data: It will be convenient, for the sake of analysis of the runtime of the algorithm, to further assume that each of the rows of X are normalized such that $1 \leq \|x_i\| \leq \sqrt{\eta}$. Therefore, $1 \leq \log(x_i^T \Sigma^{-1} x_i) \leq \log(\kappa(\Sigma)\eta)$.

Corollary 81 (Quantum algorithm for likelihood estimation of Tyler's M-estimator)**.** *Suppose we have quantum access to a matrix $X \in \mathbb{R}^{n\times d}$, and a symmetric positive definite matrix Σ such that $\|\Sigma\| \leq 1$ and $\|X\| \leq 1$. For an $\epsilon > 0$ there is a quantum algorithm that estimates $F(\Sigma)$ such that $|F(\Sigma) - \overline{F(\Sigma)}| \leq \epsilon$ in time $O(\frac{\kappa(\Sigma)\mu(\Sigma)\log\det(\Sigma)}{\epsilon})$*

Proof. Recall the formula for the likelihood of the TME is:

$$F(\Sigma) = \frac{1}{n}\sum_{i=0}^{n}\log(x_i^T\Sigma^{-1}x) + \frac{1}{d}\log\det(\Sigma)$$

We use theorem 78 and Corollary 66 to compute the log-determinant of Σ with absolute error $\epsilon/2$. For the leftmost addend in the definition of $F(\Sigma)$, we create the state:

$$\frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i\rangle\,|\overline{x_i^T\Sigma^{-1}x_i}\rangle$$

This requires $O(\frac{\kappa(\Sigma)\mu(\Sigma)\|x_i\|^2}{\epsilon})$ operations. Then, using controlled operations we create:

$$\frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i\rangle\left(\sqrt{C\log(x_i^T\Sigma^{-1}x_i)}\,|0\rangle + \gamma\,|1\rangle\right)$$

For $C = O(\frac{1}{\log(\kappa\eta)})$. Then, using amplitude estimation with precision $\frac{\epsilon}{C}$, we can obtain an estimate of $p(0) = \frac{1}{n}\sum_i^n \log(x_i^T\Sigma^{-1}x_i)$ such that $|p(0) - \overline{p(0)}| \leq p(0)\epsilon$ in time $O(\frac{\log(\kappa\eta)\kappa(\Sigma)\mu(\Sigma)\eta}{\epsilon})$. The result in the statement follows from performing amplitude estimation with precision $\frac{\epsilon}{2Cp(0)}$, and noting that the time for computing the log-determinant dominates the runtime for estimating the left addend of the log-likelihood with absolute error.

□

In the next theorem, we assume to have quantum access to the matrix Σ_k and we define a quantum procedure to estimate Σ_{k+1} , using the iterative procedure in Definition 80.

Theorem 82 (Iteration of Tyler M-estimator)**.** *Assume to have quantum access to the covariance matrix Σ_k at iteration k, and a matrix $X \in \mathbb{R}^{n\times d}$. For $\epsilon > 0$ there is a quantum algorithm that estimates $\overline{\Sigma_{k+1}}$ such that $\left\|\overline{\Sigma_{k+1}} - \Sigma_{k+1}\right\|_F \leq \epsilon\|\Sigma_{k+1}\|$ in time*

$$\tilde{O}(d^2\frac{\mu(X)\kappa(\Sigma_k)\mu(\Sigma_k)\gamma}{\epsilon^3})$$

where γ is a problem-dependent constant that depends on X and Σ_k.

Proof. First, observe that we want to compute the following quantity:

$$\begin{aligned}\Sigma_{k+1} = \widehat{\Sigma_{k+1}}/Tr[\widehat{\Sigma_{k+1}}] &= \sum_i^n \frac{x_i x_i^T}{x_i^T\Sigma_k^{-1}x_i}/Tr[\sum_{i=0}^n \frac{x_i x_i^T}{x_i^T\Sigma_k^{-1}x_i}] \\ &= \sum_i^n \frac{1}{w_i}|x_i\rangle\,|x_i\rangle\,/Tr[\sum_{i=0}^n \frac{1}{w_i}|x_i\rangle\,\langle x_i|]\end{aligned} \tag{7.23}$$

By linearity of the trace, we know that the normalizing factor at the denominator is of $Tr[\widehat{\Sigma_{k+1}}] = \sum_i^n \frac{1}{w_i} = \|W\|_1$. We start the algorithm by estimating the values of the quadratic forms, and we create the state $\frac{1}{\sqrt{n}}\sum_i^n |i\rangle\,|\overline{w_i}\rangle$ where $\overline{w_i}$ is an estimate of $\langle x_i|\Sigma_k^{-1}x_i\rangle$ with absolute error ϵ_1 (note that $1 \leq \langle x_i\Sigma_k^{-1}x_i\rangle \leq \kappa(\Sigma_k)$). This step has a cost of $O(\mu(\Sigma_k)\kappa(\Sigma_k)/\epsilon_1)$. Next, we would like to store $1/w_i$ in the amplitudes, that is, we would like to create the state $|\overline{W}\rangle = \frac{1}{\|W\|}\sum_i^n 1/\overline{w_i}\,|i\rangle$, where $\|W\|_2 = \sqrt{\sum_i 1/w_i^2}$. Note that this state is ϵ_1 far from $|W\rangle$, for an error which we bound later in the proof. For this, it suffices to perform a controlled operation:

$$\frac{1}{\sqrt{n}}\sum_{i}^{n}|i\rangle|w_i\rangle\left(\frac{1}{w_i}|0\rangle+\gamma|1\rangle\right),$$

Then, we undo the computation of the second register. The probability for measuring 0 in the ancilla qubit is $p(0)=\sum_i\frac{(1/w_i)^2}{n}$. With amplitude amplification, the cost of creating state $|W\rangle$ is thus $T_W=O(\frac{\mu(\Sigma_k)\kappa(\Sigma_k)}{\epsilon_1}\frac{\sqrt{n}}{\|W\|_2})$. We define $\delta=\frac{\sqrt{n}}{\|W\|_2}=\frac{1}{\sqrt{p(0)}}$. This parameter should be bounded by analyzing the context where Tyler 's M-estimator is applied, and we discuss a possible case in the next section. We bound the error in $\left\||W\rangle-|\overline{W}\rangle\right\|$. We do this by first estimating the error on the non-normalized vectors, i.e. $\left\|W-\overline{W}\right\|$, and then by using Claim 16 to bound the distance between unit vectors. From the error $|w_i-\overline{w_i}|\leq\epsilon_1$, we derive that $|\frac{1}{w_i}-\frac{1}{\overline{w_i}}|\leq\frac{\epsilon_1}{w_i\overline{w_i}}$. We bound $\left\|W-\overline{W}\right\|_2=\sqrt{\sum_{i=0}^{n}(\frac{1}{w_i}-\frac{1}{\overline{w_i}})^2}=\sqrt{\sum_i\left(\frac{\epsilon_1}{w_i^2}\right)^2}\leq\epsilon_1\sqrt{\sum_i\frac{1}{w_i^2}}=\epsilon\|W\|_2$. Using Claim 16 we derive a distance between normalized vectors of $\sqrt{2}\epsilon_1$. Next, we use quantum access to the rows of X using $|W\rangle$.

$$|W\rangle\mapsto\frac{1}{\|W\|_2}\sum_i\frac{1}{w_i}|i\rangle|x_i\rangle \tag{7.24}$$

We define a matrix $\hat{X}$ from the matrix X by normalizing each row x_i such that they have norm equal to 1. Quantum access to $\hat{X}$ is a weaker assumption that is satisfied from the assumption of having quantum access to X. To conclude the algorithm we use matrix multiplication with the matrix $\hat{X}$, on the index register, thus performing the following mapping:

$$\frac{1}{\|W\|}\sum_i\frac{1}{w_i}|i\rangle|x_i\rangle\mapsto\frac{1}{\|W\|}\sum_i\frac{1}{w_i}\left(|0^{\otimes t}\rangle\frac{\hat{X}^T}{\mu(\hat{X})}|i\rangle+\gamma|0^{\perp}\rangle\right)|x_i\rangle$$

Along the lines of the proof of [38, lemma 24], we measure the probability of success of measuring $|0\rangle$. For each of the basis $|i\rangle$, we have that $\|\frac{1}{\mu(A)}\hat{X}^T|i\rangle\|=\frac{1}{\mu(A)}$, as $\|\hat{X}^T|i\rangle\|=1$. Rearranging the previous equation we obtain:

$$\frac{1}{\|W\|_2}\sum_i|0^{\otimes t}\rangle\left(\frac{1}{w_i}\frac{1}{\mu(\hat{X})}|x_i\rangle|x_i\rangle\right)+\gamma|1,G\rangle$$

Measuring the $|0\rangle$ register happens with probability $p(0)=\|\frac{1}{\|W\|_2}\sum_i\frac{1}{w_i}\frac{1}{\mu(\hat{X})}|x_i\rangle|x_i\rangle\|_2^2=\|\widehat{\Sigma_{k+1}}\|_F^2$, which we approximate with precision ϵ_2, and amplify with $\mu(\hat{X})$ rounds of amplitude amplification. The state that we obtain after these operations is:

$$|\widehat{\Sigma_{k+1}}\rangle=\frac{1}{\|\widehat{\Sigma_{k+1}}\|_F}\sum_i\frac{1}{w_i}|x_i\rangle|x_i\rangle \tag{7.25}$$

We perform ℓ_2 tomography (i.e. theorem 11) on this state with error ϵ_3. We now proceed to estimate the value $\|W\|_1$. Using a similar algorithm as the one described above, we can create the state:

$$\frac{1}{\sqrt{n}}\sum_i|i\rangle\left(\frac{1}{\sqrt{w_i}}|1\rangle+\sqrt{1-\frac{1}{w_i}}|0\rangle\right)$$

Using amplitude estimation, we obtain an estimation of $\|W\|_1$. We conclude the algorithm by returning an estimate of $\overline{\Sigma_{k+1}}$ as:

$$\overline{\Sigma_{k+1}}=\frac{\overline{\left\|\widehat{\Sigma_{k1+}}\right\|_F}}{\overline{Tr[\Sigma_{k+1}]}}\overline{|\widehat{\Sigma_{k+1}}\rangle}$$

.

The runtime of the algorithm is dominated by the runtime of performing tomography and matrix multiplication on the state created in Eq. 7.24. The runtime of these operations is of:

$$O\left(\frac{d^2}{\epsilon_3}\frac{\mu(X)\kappa(\Sigma_k)\mu(\Sigma_k)\gamma)}{\epsilon_1}\right)$$

Error analysis The error in our estimate $\overline{\Sigma_{k+1}}$ is given by the error estimating $\frac{\left\|\widehat{\Sigma_{k+1}}\right\|_F}{Tr[\Sigma_{k+1}]}$ and the error in the tomography. With an absolute error estimate of $|\Sigma_{k+1}\rangle$ of ϵ_3 and an relative error on $\left\|\widehat{\Sigma_{k+1}}\right\|_F$ and $Tr[\Sigma_{k+1}]$ of ϵ_2, the final error estimate is of:

$$\left\|\Sigma_{k+1} - \frac{\overline{\left\|\widehat{\Sigma_{k+1}}\right\|_F}}{\overline{Tr[\widehat{\Sigma_{k+1}}]}}\overline{|\widehat{\Sigma_{k+1}}\rangle}\right\|_F \leq \tag{7.26}$$

$$\left(\frac{\left\|\widehat{\Sigma_{k+1}}\right\|_F}{Tr[\widehat{\Sigma_{k+1}}]} - \frac{\overline{\left\|\widehat{\Sigma_{k+1}}\right\|_F}}{\overline{Tr[\widehat{\Sigma_{k+1}}]}}\right) + \frac{\left\|\widehat{\Sigma_{k+1}}\right\|_F}{Tr[\widehat{\Sigma_{k+1}}]}\| |\widehat{\Sigma_{k+1}}\rangle - \overline{|\widehat{\Sigma_{k+1}}\rangle}\| \leq \tag{7.27}$$

$$\frac{\left\|\widehat{\Sigma_{k+1}}\right\|_F}{Tr[\widehat{\Sigma_{k+1}}]}(2\epsilon_2) + \frac{\left\|\widehat{\Sigma_{k+1}}\right\|_F}{Tr[\widehat{\Sigma_{k+1}}]}(\epsilon_1+\epsilon_3) \leq \tag{7.28}$$

$$\epsilon\|\Sigma_{k+1}\|_F. \tag{7.29}$$

$$\tag{7.30}$$

In the previous equation the error on the unit vector $|\widehat{\Sigma}_{k+1}\rangle$ is given by the ϵ_3 error in the tomography and the error ϵ_1 in the creation of the state $|W\rangle$ which we analyze now. We set $2\epsilon_2+\epsilon_1+\epsilon_3 \leq \epsilon$, and where ϵ_1 is the error in the preparation on the state $|W\rangle$, which we analyze now. In Equation 7.25 we want to create a state $|\overline{\Sigma_{k+1}}\rangle$ which is $\epsilon/2$ close to $|\Sigma_{k+1}\rangle$. We conclude by choosing $\epsilon_3 = \epsilon_1 \leq \epsilon/4$. From this we obtain the runtime in the statement of the theorem. □

7.4.2 Further remarks on the quantum algorithm for TME

In this section we discuss in more detail the quantum algorithm of TME, namely the impact of the noise in the final estimate of Γ.

Bounding γ The value of γ depends on the problem under consideration. Here we report the analysis in [72] for the case of elliptical distributions when $n, d \to \infty$ and $n/p \to \zeta \in [0,1]$. Elliptical distributions are one of the possible generalizations of normal distributions that encompass tails that are heavy.

Definition 83. *A random vector $x \in \mathbb{R}^d$ follows an elliptical distribution with location vector μ if it has the form*

$$x = \mu = uS_d^{1/2}\zeta = \mu + u\boldsymbol{z} \tag{7.31}$$

In the previous definition, μ is the center of the elliptical distribution, $u \in \mathbb{R}$ is a constant (or a random variable that does not depend on ζ), and ζ is a unit random vector distributed uniformly on the unit sphere of $\mathbb{R}^d$. Under these assumptions, the following lemma proves a concentration inequality on the value of γ.

Lemma 84 (lemma 7 of [72] - Weights of TME)**.** *Consider a sequence (n, d, S_d) where $n, d \to \infty$ with $d/n \to \chi \in (0,1)$, and S_d is an SPD matrix. For every tripled (n, d, S_d), let*

$x_i \overset{i.i.d.}{\sim} N(0, S_d)$ and let $\{1/w_i\}_{i=1}^n$ be the corresponding weights of Eq. 7.23. Then, there exist positive constants C, c, and c' depending only on γ, such that for any $0 < \epsilon < c'$, and sufficiently large n,

$$Pr\left[\max_i |n\frac{1}{w_i} - 1| \geq \epsilon\right] \leq Cne^{-c\epsilon^2 n}$$

The previous lemma says that the weights w_i concentrate around $1/n$, and as consequence, the value of γ in the case of elliptical distributions can be thought of as a constant $\Omega(1)$, but a thorough study is left for future work.

7.5 Conclusions

In this chapter we started by performing a better analysis of previous quantum algorithms for estimating the log-determinant of a symmetric positive definite matrix. Then we proceeded with the study of new quantum algorithms for the same problem, but using the state-of-the-art techniques in quantum linear algebra, leading to a qudaratic speedup with respect to the previous quantum algorithm. Remark that the structure used in Algorithm 9 can be easily adapted to estimate other spectral sums, (i.e. the sum of the singular values of a function after a function has been applied to them). Examples of other spectral sums are the Estrada's index, and the Schatten's p norms, and this is currently a work in progress. It is hard to expect faster algorithms for trace estimation. It has been shown in [132] that the problem of estimating $Tr[|A|^p]/2^n$ for a $2^n \times 2^n$ log-local Hamiltonian A up to a given accuracy is hard for the class DQC1. As the DQC1 is a class which contains a broad class of problems in BQP, we expect the runtime of estimating the trace of a matrix to be hard to improve.

The algorithm for TME is a simple application of a fast procedure to estimate the log-determinant of a big matrix. Other problems in machine learning where one needs to compute log-determinants of massive matrices are Gaussian processes (GP) and Gaussian Markov random fields (GMRF) [78]. A discussion on possible extensions of the quantum version of TME estimator is presented in the next chapter.

Chapter 8

Conclusions

In this thesis, we developed some new quantum algorithms for machine learning. Most of them (except the QFDC described in chapter 4) are the quantum analogue of a classical machine learning algorithm. The remarkable difference between the classical and the quantum algorithm is the runtime, which in the quantum case depends only polylogarithmically in the dimension of the dataset, and in the classical case is usually polynomial (and linear with some exceptions). Notably, there is a class of classical algorithms (that are usually called de-quantizations) that under similar assumptions can reach a runtime with polylogarithmic dependence on the size of the data, albeit with a much worse dependence on the other parameters, and thus we usually exclude them from the comparison between the quantum and the best classical algorithms. In order to estimate the runtimes of the quantum algorithms, we measure the parameters that govern it (like the condition number, the function μ described in definition 9), and test (through a classical simulation) whether the errors introduced in the quantum procedures can still allow getting an accurate and fast algorithm. Overall, the answer was quite positive, as we were often able to find the hyper-parameters to get both fast and accurate algorithms.

We conclude this thesis with some research directions, which might fruitfully harness quantum computers to get better algorithms, for problems where speed and precision are important.

Other EM-like algorithms There is a rich theory of classical iterative algorithms that alternate two (or more) operations until convergence. These algorithms perform a hill-climbing approach [76] in order to find local minima or maximua of a function (usually the log-likelihood). The success of these algorithms can be explained using a geometric interpretation. Let P and Q be two elements from two sets $\mathcal{P}$ and $\mathcal{Q}$. Let $d : \mathcal{P} \times \mathcal{Q} \mapsto \mathbb{R}$ be a function. The sequences $\{P_k\}_{k=0}^{\infty}$ and $\{Q_k\}_{k=0}^{\infty}$ are obtained from $k \in \mathbb{N}$ by:

$$P_{k+1} = \arg\min_{P \in \mathbb{P}} d(P, Q_k)$$

$$Q_{k+1} = \arg\min_{Q \in \mathbb{Q}} d(P_{k+1} Q)$$

It has been shown [76, 44] that, if $\mathcal{P}$ and $\mathcal{Q}$ are convex sets, and the function d is the Kullback-Leibler divergence, then all alternating minimization sequences converge monotonically to a global minimum. Some of the most important alternating minimization algorithms in ML (for which I believe quantum algorithms might be beneficial) are the following.

- *Latent Dirichlet Annotation* is a Bayesian generative model for analyzing a collection of discrete data (like documents). Each item in the collection of documents is modeled as a mixture of a set of topics, and each topic is modeled as an infinite mixture over

a set of topic probabilities. Classically, LDA is computed using a similar approach to an EM algorithm [29].

- *Non-Gaussian Information bottleneck* is the original formulation of the information bottleneck [146], which we presented in chapter 3. When the probability distribution of the variables in the feature space $\mathcal{X}$ and in the label space $\mathcal{Y}$ are not jointly Gaussian - but the probability distribution of the marginal distributions are known - it is possible to use an iterative algorithm, which was first formulated in the context of the information bottleneck. [146]. Albeit the relevance of the information bottleneck method in ML is sometimes questioned [64], a quantum algorithm for this problem might still represent an interesting technical challenge, as a single iteration of this algorithm consist in an update of 3 different probability distribution, and not just 2, as in EM.

- *Message-passing algorithms.* In graphical models (like Bayesian's networks and Markov random fields) there is an important class of algorithms for performing inference, called belief propagation. As the underlying optimization problem is often NP-hard to solve exactly, belief propagation can be approximately solved via message passing algorithms, which share many similarities with EM [47].

Quantum generative adversarial networks In recent years, some proposals for quantizing neural networks and convolutional neural networks, with provable guarantees on the runtime needed for fitting the model have been put forward [88, 6]. In classical machine learning, generative adversarial networks have revolutionized the landscape of generative models. So far, only quantum variational circuits that mirror adversarial learning have been put forward [46, 101], leaving some space for a non-variational quantum algorithm for adversarial learning.

Tyler M-estimator In statistics, TME has been extensively studied. One recent result proved that taking an entry-wise threshold on the estimated TME, allows to better bound the difference in spectral norm between the estimator and the shape matrix. As ℓ_∞ tomography might be seen as a form of thresholding, it would be interesting to see if the thresholding can be taken not only once the iterations of the TME have converged, but also at each iteration. In this way, we can hope to reduce the dependence on the dimension of the matrix, which is currently d^2. Last but not least, TME is usually improved by another form of regularization [143], which is usually the preferred estimator when the number of samples is not sufficient to give a robust estimator. In these cases, we can use the following estimator

$$\Sigma_{k+1} = \frac{1}{1+\alpha_0}\frac{d}{n}\sum_{i=1}^{n}\frac{x_i x_i^T}{x_i^T \Sigma_k^{-1} x_i} + \frac{\alpha_0}{1+\alpha_0} I$$

where α_0 is a scalar to be determined using some heuristics.

Bibliography

[1] Scott Aaronson. "Read the fine print". In: *Nature Physics* 11.4 (Apr. 2015), pp. 291–293. ISSN: 1745-2473.

[2] Scott Aaronson and Patrick Rall. "Quantum approximate counting, simplified". In: *Symposium on Simplicity in Algorithms*. SIAM. 2020, pp. 24–32.

[3] Andrey Abakumov. *Code for Domain Generation Algorithms*. accessed 24/05/2020. URL: https://github.com/andrewaeva/DGA.

[4] Dorit Aharonov et al. "Adiabatic quantum computation is equivalent to standard quantum computation". In: *SIAM review* 50.4 (2008), pp. 755–787.

[5] Hamed Ahmadi and Chen-Fu Chiang. "Quantum phase estimation with arbitrary constant-precision phase shift operators". In: *arXiv preprint arXiv:1012.4727* (2010).

[6] Jonathan Allcock et al. "Quantum algorithms for feedforward neural networks". In: *arXiv preprint arXiv:1812.03089* (2018).

[7] Andris Ambainis. "Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations". In: *arXiv preprint arXiv:1010.4458* (2010).

[8] Andris Ambainis. "Variable time amplitude amplification and quantum algorithms for linear algebra problems". In: *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*. Vol. 14. LIPIcs. 2012, pp. 636–647.

[9] Anurag Anshu et al. "Exponential separation of quantum communication and classical information". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 277–288.

[10] appliedmachinelearning.org. *Spoken Speaker Identification based on Gaussian Mixture Models : Python Implementation*. https://appliedmachinelearning.blog/2017/11/14/spoken-speaker-identification-based-on-gaussian-mixture-models-python-implementation/. accessed 20/07/2019.

[11] Pieter Arntz. *Explained: Domain Generating Algorithm*. accessed 24/05/2020. URL: https://blog.malwarebytes.com/security-world/2016/12/explained-domain-generating-algorithm/.

[12] Juan Miguel Arrazola et al. "Quantum-inspired algorithms in practice". In: *arXiv preprint arXiv:1905.10415* (2019).

[13] David Arthur and Sergei Vassilvitskii. "How slow is the k-means method?" In: *Proceedings of the twenty-second annual symposium on Computational geometry*. ACM. 2006, pp. 144–153.

[14] David Arthur and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035.

[15] Haim Avron and Sivan Toledo. "Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix". In: *Journal of the ACM (JACM)* 58.2 (2011), pp. 1–34.

[16] Francis R Bach and Michael I Jordan. "A probabilistic interpretation of canonical correlation analysis". In: *Technical Report - Department of Statistics University of California, Berkeley* (2005).

[17] Ziv Bar-Yossef, Thathachar S Jayram, and Iordanis Kerenidis. "Exponential separation of quantum and classical one-way communication complexity". In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. 2004, pp. 128–137.

[18] Albert Barkamol. *Plank solves the ultraviolet catastrophe*. https://www.webassign.net/question_assets/buelemphys1/chapter27/section27dash1.pdf. (Visited on 05/21/2020).

[19] Johannes Bausch, Sathyawageeswar Subramanian, and Stephen Piddock. "A Quantum Search Decoder for Natural Language Processing". In: *arXiv:1909.05023* (2019).

[20] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (2019), p. 043001.

[21] Charles H Bennett. "Notes on Landauer's principle, reversible computation, and Maxwell's Demon". In: *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics* 34.3 (2003), pp. 501–510.

[22] P. Berkes and L. Wiskott. "Slow feature analysis yields a rich repertoire of complex cell properties". In: *Journal of Vision* 5.6 (2005).

[23] Pietro Berkes. "Pattern Recognition with Slow Feature Analysis". In: *Cognitive Sciences EPrint Archive (CogPrints)* 4104 (2005).

[24] Dominic W Berry, Andrew M Childs, and Robin Kothari. "Hamiltonian simulation with nearly optimal dependence on all parameters". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 792–809.

[25] Dominic W Berry et al. "Simulating Hamiltonian dynamics with a truncated Taylor series". In: *Physical review letters* 114.9 (2015), p. 090502.

[26] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202.

[27] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models". In: *Computational Statistics & Data Analysis* 41.3-4 (2003), pp. 561–575.

[28] Tobias Blaschke and Laurenz Wiskott. "Independent slow feature analysis and nonlinear blind source separation". In: *International Conference on Independent Component Analysis and Signal Separation*. Springer. 2004, pp. 742–749.

[29] 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.

[30] Johannes Blömer and Kathrin Bujna. "Simple methods for initializing the EM algorithm for gaussian mixture models". In: *CoRR* (2013).

[31] Magnus Borga, Tomas Landelius, and Hans Knutsson. *A unified approach to pca, pls, mlr and cca*. Linköping University, Department of Electrical Engineering, 1997.

[32] Christos Boutsidis et al. "A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix". In: *Linear Algebra and its Applications* 533 (2017), pp. 95–117.

[33] Christos Boutsidis et al. "A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix". In: *Linear Algebra and its Applications* 533 (2017), pp. 95–117.

[34] Gilles Brassard et al. "Quantum Amplitude Amplification and Estimation". In: *Contemporary Mathematics* 305 (2002).

[35] Marin Bukov et al. "Reinforcement learning in different phases of quantum control". In: *Physical Review X* 8.3 (2018), p. 031086.

[36] Gilles Celeux and Gérard Govaert. "A classification EM algorithm for clustering and two stochastic versions". In: *Computational statistics & Data analysis* 14.3 (1992), pp. 315–332.

[37] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2019.

[38] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation". In: *arXiv preprint arXiv:1804.01973* (2018).

[39] Nai-Hui Chia et al. "Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning". In: *arXiv:1910.06151* (2019).

[40] Andrew M Childs, Robin Kothari, and Rolando D Somma. "Quantum linear systems algorithm with exponentially improved dependence on precision". In: (2015).

[41] Andrew M Childs and Nathan Wiebe. "Hamiltonian simulation using linear combinations of unitary operations". In: *arXiv preprint arXiv:1202.5822* (2012).

[42] Kenneth W Church and William A Gale. "Poisson mixtures". In: *Natural Language Engineering* 1.2 (1995), pp. 163–190.

[43] Iris Cong and Luming Duan. "Quantum Discriminant Analysis for Dimensionality Reduction and Classification". In: *arXiv preprint arXiv:1510.00113* (2015).

[44] Imre Csiszár. "Information geometry and alternating minimization procedures". In: *Statistics and decisions* 1 (1984), pp. 205–237.

[45] Ryan R Curtin et al. "Detecting DGA domains with recurrent neural networks and side information". In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 2019, pp. 1–10.

[46] Pierre-Luc Dallaire-Demers and Nathan Killoran. "Quantum generative adversarial networks". In: *Physical Review A* 98.1 (2018), p. 012324.

[47] Justin Dauwels, Sascha Korl, and H-A Loeliger. "Expectation maximization as message passing". In: *arXiv preprint cs/0508027* (2005).

[48] Tijl De Bie, Nello Cristianini, and Roman Rosipal. "Eigenproblems in pattern recognition". In: *Handbook of Geometric Computing*. Springer, 2005, pp. 129–167.

[49] Ronald De Wolf. "Quantum computing: Lecture notes". In: *arXiv:1907.09415* (2019).

[50] Jeff Desjardins. *How much data is generated each day?* URL: https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/ (visited on 04/08/2020).

[51] Jeff Desjardins. *Trends in the cost of computing*. URL: https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/ (visited on 04/08/2020).

[52] 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.

[53] AR Dexter and DW Tanner. "Packing densities of mixtures of spheres with log-normal size distributions". In: *Nature physical science* 238.80 (1972), p. 31.

[54] Sciki-learn project's documentation. *A demo of K-Means clustering on the handwritten digits data*. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html. (Visited on 11/12/2018).

[55] Alberto N Escalante-B and Laurenz Wiskott. "Slow feature analysis: Perspectives for technical applications of a versatile learning algorithm". In: *KI-Künstliche Intelligenz* 26.4 (2012), pp. 341–348.

[56] L Magnus Ewerbring et al. "Canonical correlations and generalized SVD: applications and new algorithms". In: *Advances in Parallel Computing*. Vol. 1. Elsevier, 1990, pp. 37–52.

[57] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

[58] Edward Farhi and Hartmut Neven. "Classification with quantum neural networks on near term processors". In: *arXiv preprint arXiv:1802.06002* (2018).

[59] Richard P Feynman. "Quantum mechanical computers." In: *Found. Phys.* 16.6 (1986), pp. 507–532.

[60] Richard P Feynman. "Simulating physics with computers". In: *Int. J. Theor. Phys* 21.6/7 (1999).

[61] Ronald A Fisher. "On the mathematical foundations of theoretical statistics". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 222.594-604 (1922), pp. 309–368.

[62] Gabriel Frahm and Uwe Jaekel. "Tyler's M-estimator, random matrix theory, and generalized elliptical distributions with applications to finance". In: *Random Matrix Theory, and Generalized Elliptical Distributions with Applications to Finance (October 21, 2008)* (2008).

[63] Edward Fredkin and Tommaso Toffoli. "Conservative logic". In: *International Journal of theoretical physics* 21.3-4 (1982), pp. 219–253.

[64] Marylou Gabrié et al. "Entropy and mutual information in models of deep neural networks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 1821–1831.

[65] Rong Ge et al. "Efficient algorithms for large-scale generalized eigenvector computation and canonical correlation analysis". In: *International Conference on Machine Learning*. 2016, pp. 2741–2750.

[66] ME Ghitany, Ross A Maller, and S Zhou. "Exponential mixture models with long-term survivors and covariates". In: *Journal of multivariate Analysis* 49.2 (1994), pp. 218–241.

[67] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. "Eigenvalue and generalized eigenvalue problems: Tutorial". In: *arXiv preprint arXiv:1903.11240* (2019).

[68] András Gilyén and Tongyang Li. "Distributional property testing in a quantum world". In: *arXiv preprint arXiv:1902.00814* (2019).

[69] András Gilyén, Seth Lloyd, and Ewin Tang. "Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension". In: *arXiv preprint arXiv:1811.04909* (2018).

[70] András Gilyén et al. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 193–204.

[71] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum random access memory". In: *Physical review letters* 100.16 (2008), p. 160501.

[72] John Goes, Gilad Lerman, Boaz Nadler, et al. "Robust sparse covariance estimation by thresholding Tyler's M-estimator". In: *The Annals of Statistics* 48.1 (2020), pp. 86–110.

[73] Dmitry Grinko et al. "Iterative Quantum Amplitude Estimation". In: *arXiv preprint arXiv:1912.05559* (2019).

[74] Lov K Grover. "Fixed-point quantum search". In: *Physical Review Letters* 95.15 (2005), p. 150501.

[75] Xingjian Gu, Chuancai Liu, and Sheng Wang. "Supervised Slow Feature Analysis for Face Recognition". In: 2013, pp. 178–184. URL: http://link.springer.com/10.1007/978-3-319-02961-0_22.

[76] Shane M Haas. *The expectation-maximization and alternating minimization algorithms*. 2002.

[77] Hassan Hafez-Kolahi and Shohreh Kasaei. "Information Bottleneck and its Applications in Deep Learning". In: *arXiv preprint arXiv:1904.03743* (2019).

[78] Insu Han, Dmitry Malioutov, and Jinwoo Shin. "Large-scale log-determinant computation through stochastic Chebyshev expansions". In: *International Conference on Machine Learning*. 2015, pp. 908–917.

[79] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. "Canonical correlation analysis: An overview with application to learning methods". In: *Neural computation* 16.12 (2004), pp. 2639–2664.

[80] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (Oct. 2009), p. 150502. ISSN: 0031-9007. URL: http://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[81] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Vol. 1. Springer Series in Statistics. New York, NY: Springer New York, 2009, pp. 337–387. ISBN: 978-0-387-84857-0. URL: http://www.springerlink.com/index/10.1007/b94608.

[82] MJR Healy. "A rotation method for computing canonical correlations". In: *Mathematics of Computation* 11.58 (1957), pp. 83–86.

[83] Wassily Hoeffding. "Probability inequalities for sums of bounded random variables". In: *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[84] Winston H Hsu, Lyndon S Kennedy, and Shih-Fu Chang. "Video search reranking via information bottleneck principle". In: *Proceedings of the 14th ACM international conference on Multimedia*. 2006, pp. 35–44.

[85] Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. "Disentangling gaussians". In: *Communications of the ACM* 55.2 (2012), pp. 113–120.

[86] Ravindran Kannan and Santosh Vempala. "Randomized algorithms in numerical linear algebra". In: *Acta Numerica* 26 (2017), pp. 95–135.

[87] John T Kent, David E Tyler, et al. "Redescending M-estimates of multivariate location and scatter". In: *The Annals of Statistics* 19.4 (1991), pp. 2102–2119.

[88] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. "Quantum Algorithms for Deep Convolutional Neural Networks". In: *arXiv preprint arXiv:1911.01117* (2019).

[89] Iordanis Kerenidis and Alessandro Luongo. "Classification of the MNIST data set with quantum slow feature analysis". In: *Physical Review A* 101.6 (2020), p. 062327.

[90] Iordanis Kerenidis and Anupam Prakash. "A quantum interior point method for LPs and SDPs". In: *arXiv:1808.09266* (2018).

[91] Iordanis Kerenidis and Anupam Prakash. "Quantum gradient descent for linear systems and least squares". In: *Physical Review A* 101.2 (2020), p. 022316.

[92] Iordanis Kerenidis and Anupam Prakash. "Quantum Recommendation Systems". In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference* (2017).

[93] Iordanis Kerenidis et al. "q-means: A quantum algorithm for unsupervised machine learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 4136–4146.

[94] A Yu Kitaev. "Quantum measurements and the Abelian stabilizer problem". In: *Electronic Colloq. on Computational Complexity*. 1996.

[95] Stefan Klampfl and Wolfgang Maass. "Replacing supervised classification learning by Slow Feature Analysis in spiking neural networks". In: *Advances in Neural Information Processing Systems*. 2009, pp. 988–996.

[96] Matt Langione et al. *Where Will Quantum Computers Create Value—and When?* URL: https://www.bcg.com/fr-fr/publications/2019/quantum-computers-create-value-when.aspx.

[97] François Le Gall. "Exponential separation of quantum and classical online space complexity". In: *Theory of Computing Systems* 45.2 (2009), pp. 188–202.

[98] Chuanhai Liu and Donald B Rubin. "ML estimation of the t distribution using EM and its extensions, ECM and ECME". In: *Statistica Sinica* (1995), pp. 19–39.

[99] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. "Quantum algorithms for topological and geometric analysis of data". In: *Nature communications* 7 (2016), p. 10138. URL: https://www.nature.com/articles/ncomms10138.pdf.

[100] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (2014), p. 631.

[101] Seth Lloyd and Christian Weedbrook. "Quantum generative adversarial learning". In: *Physical review letters* 121.4 (2018), p. 040502.

[102] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[103] Guang Hao Low and Isaac L Chuang. "Hamiltonian simulation by qubitization". In: *Quantum* 3 (2019), p. 163.

[104] Guang Hao Low and Isaac L Chuang. "Hamiltonian simulation by uniform spectral amplification". In: *arXiv preprint arXiv:1707.05391* (2017).

[105] Ricardo Antonio Maronna. "Robust M-estimators of multivariate location and scatter". In: *The annals of statistics* (1976), pp. 51–67.

[106] Bernard Marr. *How much data is generated each day?* URL: https://www.weforum.org/agenda/2015/02/a-brief-history-of-big-data-everyone-should-read (visited on 04/21/2020).

[107] Farinaz Koushanfar Michael Demmer Rodrigo Fonseca. *RICHARD FEYNMAN: SIMULATING PHYSICS WITH COMPUTERS*. CS294: Reading the Classics.

[108] Hideyuki Miyahara, Kazuyuki Aihara, and Wolfgang Lechner. "Quantum expectation-maximization algorithm". In: *Physical Review A* 101.1 (2020), p. 012326.

[109] Ankur Moitra. *Algorithmic aspects of machine learning*. Cambridge University Press, 2018.

[110] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[111] Andrew Ng. "CS229 Lecture notes - Machine Learning". Lecture notes CS229 Stanford. 2012.

[112] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[113] Ryan O'Donnell and John Wright. "Efficient quantum tomography". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 899–912.

[114] Esa Ollila and Visa Koivunen. "Robust antenna array processing using M-estimators of pseudo-covariance". In: *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003.* Vol. 3. IEEE. 2003, pp. 2659–2663.

[115] Esa Ollila and David E Tyler. "Regularized m-estimators of scatter matrix". In: *IEEE Transactions on Signal Processing* 62.22 (2014), pp. 6059–6070.

[116] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[117] Anupam Prakash. "Quantum Algorithms for Linear Algebra and Machine Learning." PhD thesis. EECS Department, University of California, Berkeley, Dec. 2014. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.html.

[118] Albert Pumarola et al. "3DPeople: Modeling the Geometry of Dressed Humans". In: *International Conference in Computer Vision (ICCV)*. 2019.

[119] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. "Anomaly detection in wireless sensor networks". In: *IEEE Wireless Communications* 15.4 (2008), pp. 34–40.

[120] Ran Raz. "Exponential separation of quantum and classical communication complexity". In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. 1999, pp. 358–367.

[121] Ran Raz and Avishay Tal. "Oracle separation of BQP and PH". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 13–23.

[122] Patrick Rebentrost and Seth Lloyd. "Quantum computational finance: quantum algorithm for portfolio optimization". In: *arXiv preprint arXiv:1811.03975* 98.4 (2018), p. 042308.

[123] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". In: *Physical review letters* 113.13 (2014), p. 130503.

[124] Dick Lipton & Ken Regan. "Quantum Switch-Em". In: (2019). URL: https://rjlipton.wordpress.com/2019/09/26/quantum-switch-em/.

[125] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. "Speaker verification using adapted Gaussian mixture models". In: *Digital signal processing* 10.1-3 (2000), pp. 19–41.

[126] Walter Rudin et al. *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York, 1964.

[127] Erwin Schrödinger. "An undulatory theory of the mechanics of atoms and molecules". In: *Physical review* 28.6 (1926), p. 1049.

[128] Changpeng Shao, Yang Li, and Hongbo Li. "Quantum algorithm design: Techniques and applications". In: *Journal of Systems Science and Complexity* 32.1 (2019), pp. 375–452.

[129] C. David Sherrill. *A Brief Review of Elementary Quantum Chemistry*. http://vergil.chemistry.gatech.edu/notes/quantrev/quantrev.html. (Visited on 05/21/2020).

[130] Amnon Ta-Shma. "Inverting well conditioned matrices in quantum logspace". In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 2013, pp. 881–890.

[131] 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.

[132] Peter W Shor and Stephen P Jordan. "Estimating Jones polynomials is a complete problem for one clean qubit". In: *arXiv preprint arXiv:0707.2831* (2007).

[133] Daniel R Simon. "On the power of quantum computation". In: *SIAM journal on computing* 26.5 (1997), pp. 1474–1483.

[134] Noam Slonim. "The information bottleneck: Theory and applications". PhD thesis. Citeseer, 2002.

[135] Noam Slonim and Naftali Tishby. "Document clustering using word clusters via the information bottleneck method". In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. 2000, pp. 208–215.

[136] Henning Sprekeler. "On the relation of slow feature analysis and laplacian eigenmaps". In: *Neural computation* 23.12 (2011), pp. 3287–3302.

[137] Henning Sprekeler and Laurenz Wiskott. "Understanding Slow Feature Analysis: A Mathematical Framework". In: *Cognitive Sciences EPrint Archive (CogPrints)* 6223 (2008).

[138] Henning Sprekeler, Tiziano Zito, and Laurenz Wiskott. "An extension of slow feature analysis for nonlinear blind source separation." In: *Journal of machine learning research* 15.1 (2014), pp. 921–947.

[139] Yiğit Subaşı, Rolando D Somma, and Davide Orsucci. "Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing". In: *Physical review letters* 122.6 (2019), p. 060504.

[140] Sathyawageeswar Subramanian, Stephen Brierley, and Richard Jozsa. "Implementing smooth functions of a Hermitian matrix on a quantum computer". In: *Journal of Physics Communications* 3.6 (2019), p. 065002.

[141] Liang Sun, Shuiwang Ji, and Jieping Ye. "A least squares formulation for a class of generalized eigenvalue problems in machine learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009, pp. 977–984.

[142] Lin Sun et al. "DL-SFA: deeply-learned slow feature analysis for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2625–2632.

[143] Ying Sun, Prabhu Babu, and Daniel P Palomar. "Regularized tyler's scatter estimator: Existence, uniqueness, and algorithms". In: *IEEE Transactions on Signal Processing* 62.19 (2014), pp. 5143–5156.

[144] Ewin Tang. "A quantum-inspired classical algorithm for recommendation systems". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 217–228.

[145] Ewin Tang. "Quantum-inspired classical algorithms for principal component analysis and supervised clustering". In: *arXiv preprint arXiv:1811.00414* (2018).

[146] Naftali Tishby, Fernando C Pereira, and William Bialek. "The information bottleneck method". In: *arXiv preprint physics/0004057* (2000).

[147] R Vinayakumar et al. "Improved DGA Domain Names Detection and Categorization Using Deep Learning Architectures with Classical Machine Learning Algorithms". In: *Cybersecurity and Secure Information Systems*. Springer, 2019, pp. 161–192.

[148] Voxforge.org. *Free Speech... Recognition - voxforge.org*. http://www.voxforge.org/. accessed 20/07/2019.

[149] Hao-Ting Wang et al. "Finding the needle in high-dimensional haystack: A tutorial on canonical correlation analysis". In: *arXiv preprint arXiv:1812.02598* (2018).

[150] Rue Wang. *Generalized eigenvalue problem*. http://fourier.eng.hmc.edu/e161/lectures/algebra/node7.html.

[151] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. "Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning". In: (2014).

[152] Wiskott Laurenz and Laurenz Wiskott. "Learning invariance manifolds". In: *Neurocomputing* 26-27 (1999), pp. 925–932. ISSN: 09252312. DOI: 10.1016/S0925-2312(99)00011-9. URL: https://www.ini.rub.de/PEOPLE/wiskott/Reprints/Wiskott-1998a-JSNC-InvarianceManifolds-Preprint.pdf.

[153] L. Wiskott et al. "Slow feature analysis". In: *Scholarpedia* 6.4 (2011). revision #137965, p. 5282.

[154] Jianhua Yin and Jianyong Wang. "A dirichlet multinomial mixture model-based approach for short text clustering". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 233–242.

[155] Theodore J Yoder, Guang Hao Low, and Isaac L Chuang. "Fixed-point quantum search with an optimal number of queries". In: *Physical review letters* 113.21 (2014), p. 210501.

[156] Zhang Zhang and Dacheng Tao. "Slow Feature Analysis for Human Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (Mar. 2012), pp. 436–450. ISSN: 0162-8828. URL: http://ieeexplore.ieee.org/document/6136516/.

[157] Kaining Zhang et al. "Efficient State Read-out for Quantum Machine Learning Algorithms". In: *arXiv preprint arXiv:2004.06421* (2020).

[158] Teng Zhang. "Robust subspace recovery by geodesically convex optimization". In: *arXiv preprint arXiv:1206.1386* 60.11 (2012), pp. 5597–5625.

[159] Teng Zhang. "Robust subspace recovery by Tyler's M-estimator". In: *Information and Inference: A Journal of the IMA* 5.1 (2016), pp. 1–21.

[160] Liming Zhao et al. "Compiling basic linear algebra subroutines for quantum computers". In: *arXiv preprint arXiv:1902.10394* (2019).

[161] Zhikuan Zhao et al. "Quantum algorithms for training Gaussian processes". In: *Physical Review A* 100.1 (2019), p. 012304.