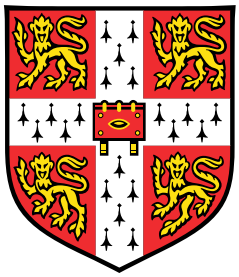
Quantum Algorithms for Matrix Problems and Machine Learning



Sathyawageeswar Subramanian

Department of Applied Mathematics and Theoretical Physics
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

St. Catharine's College

December 5, 2020

Abstract

This dissertation presents a study of quantum algorithms for problems that can be posed as matrix function tasks. In Chapter 1 we demonstrate a simple unifying framework for implementing of smooth functions of matrices on a quantum computer. This framework captures a variety of problems that can be solved by evaluating properties of some function of a matrix, and we identify speedups over classical algorithms for some problem classes. The analysis combines ideas from the classical theory of function approximation with the quantum algorithmic primitive of implementing linear combinations of unitary operators.

In Chapter 2 we continue this study by looking at the role of sparsity of input matrices in constructing efficient quantum algorithms. We show that classically pre-processing an input matrix by spectral sparsification can be profitable for quantum Hamiltonian simulation algorithms, without compromising the simulation error or complexity. Such preprocessing incurs a one time cost linear in the size of the matrix, but can be exploited to exponentially speed up subsequent subroutines such as inversion.

In Chapter 3, we give an application of this theory of matrix functions to the problem of estimating the Renyi entropy of an unknown quantum state. We combine matrix function techniques with mixed state quantum computation in the one-clean qubit model, and are able to bound of the expected runtime of our algorithm in terms of the unknown target quantity.

In addition to the theme of analysing the complexity of our algorithms, we also identify instances that are of practical relevance, leading us to some problems of machine learning. In Chapter 4 we investigate kernel based learning methods using random features. We work with the QRAM input model suitable for big data, and show how matrix functions and the quantum Fourier transform can be used to devise a quantum algorithm for sampling random features that are optimised for given input data and choice of kernel. We obtain a potential exponential speedup over the best known classical algorithm even without explicit assumptions of sparsity or low rank.

Finally in Chapter 5 we consider the technique of beamsearch decoding used in natural language processing. We work in the query model, and show how quantum search with advice can be used to construct a quantum search decoder that can find the optimal parse (which may for instance be a best translation, or text-to-speech transcript) at least quadratically faster than the best known classical algorithms, and obtain super-quadratic speedups in the expected runtime.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. It does not exceed the prescribed word limit for the relevant Maths Degree Committee.

Sathyawageeswar Subramanian
December 5, 2020

Acknowledgements

I am deeply grateful to my supervisor Richard Jozsa, for all that he has done for me over the last four years. I cannot thank him enough for his guidance and generosity, and for always being available to chat and share his ideas, advice, and wisdom.

I am indebted to the Science and Engineering Research Board (SERB, DST, Govt. Of India) and the Cambride Trust for supporting my PhD through a joint Cambridge-India Ramanujan scholarship. I must also express my deepest gratitude to The Pemanda Monappa foundation and PM Belliappa, for convincing me to take up the offer of a masters program in Cambridge, and being an inspiring and shaping force in my life these last five years. In this connection, I would also like to thank Banibrata Mukhopadhyay for his continued guidance and friendship right from my undergraduate days.

I thank all my collaborators and co-authors for teaching me a variety of subjects, and for numerous enjoyable and interesting discussions. I would also like to thank Masazumi Honda, Narsing Jha, Jose Siquiera, Sam Thomas and so many others who I have been fortunate to share my time with at the Maths Department for many interesting discussions, and much non-academic fun. My thanks also go to all the many members of my CQIF family during the last four years for the seminars, group meetings, tea break discussions and all the other fun things we have shared together.

I am very thankful to St. Catharine's College, for the warm and friendly atmosphere and community, and all the logistical and administrative help that they promptly arranged whenever it was needed.

I should also express my sincere thanks to Min-Hsiu Hsieh and the University of Technology Sydney, and Stuart Hadfield and Norman Tubman at QuAIL (NASA Ames, Mountain view) for their hospitality.

I would like to make special mention of Srinivasan Arunachalam, Koji Azuma, Johannes Bausch, Steven Herbert, and Cambyse Rouze for being friends and mentors rolled into one. Thanks also to all my friends in Cambridge over the years for enlivening my life here, and all my other friends elsewhere in the world for their continual support.

Since any attempt to list names would inevitably be incomplete, I would simply like to express my gratitude and indebtedness to all the innumerable people who have played a part in the making of this thesis and my PhD life, brightening it with their kindness and affection.

Finally, I would like to thank my parents for a lifetime of love and support; everything that is good in me is there because they planted and nurtured it. Without their affectionate encouragement, nothing I do would be possible.

ॐ श्रीमहागणपतये नमः ॥

प्र णो॑ दे॒वी सर॑स्वती॒ वाजे॑भिर्वा॒जिनी॑वती । धी॒नाम॑वि॒त्र्य॑वतु ॥ ऋ६/६१/४ ॥
इति प्रोवाचर्षिर्भरद्वाजो बार्हस्पत्यः ॥

. . . this thesis is dedicated to my parents and grandparents

Thus proclaimed the Rshi Bharadvaaja, son of Brhaspati...
"O ye shining Being, gushing forth with thy bounteous Forces, your very nature is to flow! She who is of the very substance of thought, may she protect..
and may she nourish us all in Her splendour."

Contents

Introduction

Algorithm design is a discipline that epitomises creativity. Algorithmists enjoy the opportunity to tackle abstractions of problems that instantiate themselves in a staggeringly wide range of subjects. Quantum information theory occupies a unique position in algorithmics, promising efficient algorithms on novel quantum devices that could perform computational tasks that are intractable by means of classical computing, especially in quantum chemistry and physics. To understand the extent of overlap between classical and quantum computing, and delineate their boundaries, one may take two distinct, yet intertwined, directions of research: for some chosen task, (1) constructing (efficient) quantum algorithms and analysing their complexity, and (2) proving lower bounds on the complexity of *any* quantum algorithm possible.

Since 2009, when the work of Harrow et al. showed that an exponential speedup could be expected in quantumly solving linear algebraic problems, the quantum algorithm community has been abuzz with excitement about quantum linear algebra and its applications. The research that has gone into this thesis was originally motivated by this body of work, and has focused primarily on point (1) above. We have attempted to take up the twin perspectives of devising algorithms to tackle problems that are immediately relevant (e.g. natural language processing), and studying upper bounds on the complexity for representative tasks of special importance (e.g. entropy estimation).

Quantum computing is now well known to be able to provide significant computational benefits over classical computing for a variety of tasks, including integer factorisation, search problems, and the solution of systems of linear equations, amongst others. In this thesis we expand on the capabilities of quantum computers by identifying further examples of tasks for which efficient classical algorithms are not known, but for which we are able to give efficient quantum algorithms. These tasks are drawn mainly from the areas of matrix analysis and machine learning.

What do matrix analysis and machine learning have to do with quantum computation? A decade of research has thrown up so many connections that quantum linear algebra and quantum machine learning are considered fields in their own right today. Let us take a quick look at them below.

Quantum linear algebra: Quantum computers embed information in the state of quantum systems, and then manipulate this information by engineering the quantum dynamics of the system. This quantum dynamics is mathematically manifest as the unitary evolution in a

Hilbert space. We restrict ourselves to unitary dynamics so that computations are reversible and adiabatic, in keeping with the time evolution of a closed system. The only non-unitary operation allowed in standard quantum computing is measurement.

To augment the list of potential applications of quantum computers, it is important to study the encoding of classical computational problems in a quantum framework. In such an exploration, it is useful to ask how to 'implement' or perform computations that may not be directly representable as the unitary evolution of a quantum state. The classic example of such an operation is the inversion of an arbitrary (invertible) matrix A, arising as a step in the solution to a system of linear equations, say $A\vec{x} = \vec{b}$. The matrix A here need not be unitary, or even Hermitian. We may encapsulate this and other problems of a similar spirit in the question of how to quantumly implement smooth functions of Hermitian matrices.

This leads us straight to the idea of quantum linear algebra. Linear algebraic problems can be formulated as questions about the properties of matrices and their functions, such as their traces, determinants, and quadratic forms evaluated on different vectors. On the other hand, the theory of finite dimensional quantum systems happens to be exactly complex linear algebra. Thus it does not come as a surprise that it must be possible to harness the behaviour and time evolution of quantum systems to carry out linear algebraic computations.

What does come as a non trivial observation is the fact that such quantum linear algebraic computations can sometimes be exponentially faster than their classical counterparts; in particular, for sparse and well conditioned input matrices that correspond to locally computable graphs (i.e. graphs in which each node can efficiently list all of its neighbours). What is more, there is evidence to believe that if some of these exponentially faster algorithms can be simulated by equally fast classical ones, all of quantum mechanics becomes classically simulable, an outcome that we believe to be highly unlikely.

The exponential speedup is primarily in the dimension of the input matrix, meaning that sublinear, poly-logarithmic runtimes are attained by these algorithms. It stands to be clarified that such speedups are only possible when a global property of the solution vector or matrix are desired to be evaluated — since simply writing down a vector of size N will require time that is linear in N. A further insight that has been developed over the last few years has allowed for a retention of an exponential speedup in the precision to which solutions are obtained — and the key here is to reformulate the problem in an essentially quantum way, so that quantum linear algebraic algorithms are typically described as outputting a quantum state that encodes the classical vector that solves a problem, thereby sidestepping the issue of having to write down all of its components explicitly in classical memory.

Over the last decade, several quantum algorithms demonstrating quantum speedups for linear algebra have been invented, ranging from principal component analysis and matrix inversion, to a variety of regression algorithms. Linear algebra is the life breath of a plethora of science and engineering applications. Chief among these in modern times has been machine

learning. Naturally, the quantum algorithms community has taken cognizance of this applicability, and the field of quantum machine learning in the form it is studied today was birthed around the same time as quantum linear algebra.

Quantum machine learning: Statistical learning theory studies the 'learnability' of functions from a few example input-output pairs. Researchers started enquiring over two decades ago whether learning from quantum examples, input-output pairs that are given in the form of quantum superpositions, gives any advantage over classical examples. This model is essentially one of oracles and query complexity, and although we now have numerous upper and lower bounds that illuminate quantum statistical learning theory, it appears to have so rich a structure that we are able to prove more things about it than may be useful in practice.

At the other end of the spectrum lies modern machine learning, based on large volumes of data, popularly called 'big data'. We have only glimmers of understanding of the theory behind how the hugely successful learning models, most famously those based on neural networks, learn. The landscape is peppered with heuristic algorithms and recipes that are able to achieve such astonishing feats of prediction from data that they have nearly become synonymous with the title of artificial intelligence.

The challenges of learning from big data are manifold, but the sheer volume of data and the high dimension thereof stand out among them. It is on this stage that the exponential speedups of quantum algorithms make an entrance. Early applications of the quantum linear systems solver were to problems of solving differential equations by finite difference methods. But soon, linear and other forms of regression were also studied, with promising results. Regression is a form of supervised classification, one of the cornerstones of machine learning. This lead to further studies of clustering and unsupervised learning methods. But provable speedups proved hard to come by, and where they were obtained, proved to be too straightforward an application of matrix inversion.

The breakthrough that reinvigorated quantum machine learning came in 2016, when Kerenidis and Prakash demonstrated an exponential speedup for the recommendation systems problem, a particularly useful type of matrix completion and principal component analysis problem. They considered several linear algebraic problems in an input model that mirrors classical random access memory. Data is assumed to be collected in advance and preprocessed for storage in a carefully chosen data structure. For data described by an arbitrary $N \times N$ matrix, this collection will take time at least linear in the number of non-zero entries. However, processing the data, given such a data structure, is significantly cheaper, depending only poly-logarithmically on N. This is popularly known as the QRAM input model.

Several new quantum algorithms have since been formulated in the QRAM model, achieving large speedups. However, a common thread that ran through them was the assumption that the matrices being manipulated had low rank. In the last two years, breakthrough work of Tang has shown that for low rank data, a classical oracle that mimics the ability of QRAM to

sample from the distribution defined by the squared amplitudes of a vector's components can be constructed efficiently, and used to achieve exponentially faster classical algorithms for the same problems. These have come to be known as quantum inspired classical algorithms. It is worth noting however, that while these algorithms achieve exponential speedups in asymptotic runtime, they are slower than their quantum counterparts by large polynomial factors, with degrees of over thirty. Quantum machine learning has thus already shed much light on the power of both classical and quantum computing within the last five years.

The assumption of low rank, meanwhile, has now been understood reasonably well, although the assumption of sparsity that is used by query model linear algebraic algorithms is still rather enigmatic. Figuring out how to input data into quantum algorithms has been a hotly studied question for nearly three decades, starting with the discovery of Grover's quantum search algorithm. New research into this area continues to throw up surprises, and the new age of near term devices that is expected to enable heuristic studies will give this research a further boost.

This thesis : We take up these two streams of thought, and first investigate quantum linear algebra in the query model, and subsequently study an application each of the capability to implement matrix functions in the query and QRAM models. The latter application happens to be supervised learning, and we then also consider the unsupervised learning problem of decoding data from a sequence of random variables, by a method of searching over a graph that is widely used in natural language processing.

A unifying theme that underlines our work is the development of quantum algorithms and the investigation of their asymptotic computational complexity. On top of this theme, the following chapters separate rather naturally into two parts, in two complementary ways.

First, the dominating theme of matrix functions and implementing them on quantum computers unites Chapters 1 to 4, with the former two setting out the theoretical framework and investigating the assumptions used therein, and the latter two discussing applications of this theory to practical problems that deal with potentially large volumes of high dimensional data in the two disparate input models of quantum query oracles and QRAM, respectively. In this regard, Chapter 5 forms an independent study of the application of quantum search with prior advice to sequence-to-sequence decoding, a very topical problem that has generated great interest in the last decade with the advent of machine learning for natural language processing.

Second, Chapters 1 to 3 deal with the analysis of a framework of quantum algorithms for matrix functions, limitations of this framework and a potential workaround by the use of techniques of spectral sparsification of graphs, and an application of matrix functions to approximating the entropy of unknown quantum states. Chapters 4 and 5 then naturally cluster under that most relevant buzzword of our times, machine learning.

Thesis Overview

Since we have considered a range of problems with quite a spread in background and theory, we expect that some readers may only be interested in specific portions of this work; in line with this, each chapter has been written to be readable as a standalone article, although references to other chapters are made aplenty to point out connections and remind readers of relevant results from one chapter that find use in another. Every chapter begins with a synopsis of results and techniques to help readers gauge its scope and relevance to their interests, in addition to an introduction that describes the context behind the problem that will form its focus.

Consequently, the preliminaries and notation that we will set up at the end of this introduction only touch upon those ideas and aspects that are constant throughout the thesis, and are not specific to any one problem.

Each numbered chapter in this thesis forms the content of an independent paper. Since all of the research presented here was done in collaborations, we briefly outline each of the following chapters below, with notes on collaborators' contributions.

- Chapter 1 motivates the study of matrix functions in the context of quantum computing, and gives a brief review of various works that have considered the problem of implementing functions of different kinds of matrices by means of unitary circuits. We have intended for this chapter to be a gentle introduction to the whole thesis. We also develop a method of probabilistically implementing functions of row-sparse and locally computable Hermitian matrices whose matrix entries are accessible via a quantum oracle. This method relies on approximating real functions by Chebyshev polynomials, coupled with the simple and effective technique of implementing a linear combination of unitaries (LCU) by means of a standard quantum walk construction.

  This chapter is based on joint work with Dr. Steve Brierley and Prof. Richard Jozsa [SBJ19]. Dr. Brierley suggested this project, and I did most of the of the work, in discussion with Dr. Brierley and Prof. Jozsa.

- Chapter 2 picks up on the assumption of row computability and sparsity used in the first chapter, and addresses the question of how replacing a matrix with a sparser one that approximates it well in spectral norm affects Hamiltonian simulation. Classical sampling-based algorithms for spectral sparsification reduce the total number of edges from $\mathcal{O}(n^2)$ in an input graph to $\mathcal{O}_\epsilon(n \log n)$ in the output ϵ-spectral sparsifier. Towards anwering our question, we prove that such algorithms also achieve row-sparsification, with high probability. With mild assumptions on the sampling process that generates the sparsifier, we show that the output has $\mathcal{O}(\log n)$ entries in any row, making it amenable to use in quantum algorithms that crucially utilise row-sparsity. Focusing on the specific example of Hamiltonian simlulation, which underlies numerous other

quantum algorithms as a fundamental subroutine, we show how the error propagating from sparsification can be kept under control. For sparsification by effective resistances, our assumption appears to be directly related to physical properties of the underlying system described by the Hamiltonian.

This chapter is based on joint work with Dr. Steven Herbert [HS19]. I proposed the original ideas, and the subsequent work emerged in discussions.

- Chapter 3 applies the theory of implementing matrix functions to the estimation of Renyi entropies. We use the method of quantum singular value transformations to obtain probabilistic implementations, *aka* block encodings, of ρ^α, and proceed to estimate its trace in the DQC1 model using a single clean qubit. We thus manage to ease the quantum memory overhead, requiring only relatively cheap noisy qubits. The DQC1 model is believed to be of restricted power, intermediate between BQP and BPP, and is believed to be relatively easier to build than full-fledged error corrected quantum computers. Our complexity bound demonstrates the trade-off between a worst-case $\mathcal{O}(d^2/\epsilon^2)$ independent measurements of the output of shallow circuits required by our method, as compared to circuits of depth $\mathcal{O}(d \log d/\epsilon^{1.5})$ used by phase estimation based methods. We use a recent iterative method to improve additive approximations to multiplicative ones, and are able to phrase our complexity bounds in terms of expected runtimes that depend on the unknown target quantity. This work is set in the quantum purified query access input model, wherein an unknown quantum mixed state is accessible via a unitary process that prepares a purification of the state.

  This chapter is based on joint work with Prof. Min-Hsiu Hsieh [SH19]. I proposed the original ideas, and the subsequent work is the result of discussions.

- Chapter 4 delves into kernel based methods for supervised learning, and illustrates an application of implementing matrix functions to machine learning. We work in the QRAM input model, suitable for big data. We show that random features that are optimised for the input data and its distribution, and the choice of kernel, can be sampled in time linear in the data dimension, modulo reasonable assumptions on the kernel's hyperparameters and on the input distribution. We once again use the quantum singular value transformations method, and the essential component is quantum matrix inversion. Our algorithm offers a potential exponential speedup over the best known classical one for this problem, and may be resistant to dequantisation by low-rank approximations. Our key contributions are the 'circulantisation' and subsequent diagonalisation of the full rank, possibly dense, regularised integral operator by decomposing it into a product of the quantum Fourier transform and simpler diagonal operators. Finally, we also elaborate how our quantum optimised feature sampling subroutine can be used with classical (doubly) stochastic gradient descent to achieve supervised learning while keeping our speedup intact.

This chapter is based on joint work with Dr. Hayata Yamasaki, Dr. Sho Sonoda, and Prof. Masato Koashi [YSS+20]. Dr. Sonoda suggested the problem, and together with Prof. Koashi offered help with the classical theory of kernel methods and signal processing. Dr. Yamasaki and I did the bulk of the work, in discussion.

- Chapter 5 carries forward the idea of practical algorithms for machine learning, but turns away from the earlier main theme of matrix functions and their quantum avatars. We consider the popular technique of beamsearch which interpolates between greedy search and best-first search via the beamwidth parameter, and is used in sequence-to-sequence decoding models. Working in the quantum query access model, we show that quantum search can be used to develop a decoder that can find the optimal decoded sequence (matching best-first search in this regard), while being at least quadratically faster than the best known classical beamsearch variant. Furthermore, specialising to the domain of natural language processing (NLP), we show that quantum search with a prior non-uniform advice state can be used to obtain super-quadratic speedups in the expected runtime. We provide numerical evidence based on LSTM based NLP models, specifically using Mozilla Deepspeech, that the power-law input distributions are actually ubiquitous in such language decoding problems.

  This chapter is based on joint work with Dr. Johannes Basuch and Dr. Stephen Piddock [BSP19]. The original ideas and all the work emerged in discussions between Dr. Bausch and myself. Dr. Bausch performed all the numerics and produced all the figures. He has kindly permitted me to reuse them for this thesis, where they are included for completeness. The proof of Theorem 5.6 (biased quantum sampler for formal grammars) was contributed by Dr. Piddock.

We end in Chapter 6 with a summary and some concluding remarks.

Preliminaries and Notation

We will assume that the average reader has at least as much background and familiarity with quantum computing as a final year undergraduate who has taken an introductory or elective course on the subject. We will not need to know all but the most basic ideas of quantum mechanics.

For completeness and ease of reference, we briefly review in this section some basic notions and notations in quantum computation, referring to [NC10, dWol19] for more details.

The usual unit of classical computation is the bit, a Boolean variable taking values in $\mathbb{Z}_2 = \{0, 1\}$. Its analogue in quantum computation is called the qubit, and represents the

state of a physical quantum 2-level system. A qubit can take values or *states* in $\mathbb{C}^2$, i.e. linear combinations or superpositions of two classical values (complex numbers)

$$\alpha\,|0\rangle + \beta\,|1\rangle$$

In particular we require that $|\alpha|^2 + |\beta|^2 = 1$. We have also introduced the Dirac bra-ket in the above:

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

More generally, the set of states an m-qubit quantum register can take is the set of unit vectors

$$|\psi\rangle = \sum_{i \in \{0,1\}^m} \alpha_i\,|i\rangle \text{ with } \alpha_i \in \mathbb{C}, \text{ such that } \sum_i |\alpha_i|^2 = 1$$

in the Hilbert space spanned by a set of orthonormal basis vectors $\{|i\rangle\,, i \in \{0,1\}^m\}$, known as the *computational basis*. Such an m-qubit state is said to be a superposition of the m basis states. Each α_i is called the *amplitude of basis state* $|i\rangle$, and the L_2 normalisation condition on $|\psi\rangle$ will enable us to treat the $\{|\alpha_i|^2\}$ as probabilities. We interpret the vector $|i\rangle$ as the m-dimensional complex vector v_i with entries given by $(v_i)_j = \delta_{ij}$, and also interchangeably as the integer i or the bit string that gives its binary representation $b_1 \ldots b_m$ where b_i is either 0 or 1. Furthermore, and of key importance to quantum mechanics and computation, the vector $|b_1 \ldots b_m\rangle \in \mathbb{C}^{2^m} = (\mathbb{C}^2)^{\otimes m}$ is interpreted as a tensor product

$$|b_1 \ldots b_m\rangle = |b_1\rangle \otimes |b_2\rangle \otimes \ldots \otimes |b_m\rangle$$

of the m vectors $|b_i\rangle$ in $\mathbb{C}^2$. The $\otimes$ is often dropped for convenience, and we write $|b_1\rangle\,|b_2\rangle$ for $|b_1\rangle \otimes |b_2\rangle$. We denote Hilbert spaces with a Gothic $\mathcal{H}$, and use a superscript capitalised Latin letter to indicate the register.

The conjugate transpose of a column vector $|\psi\rangle$ is a row vector denoted by $\langle\psi|$. The conjugate transpose and the transpose of an operator $\mathbf{A}$ are denoted by $\mathbf{A}^\dagger$ and $\mathbf{A}^{\mathrm{T}}$, respectively. The inner product of $|\psi\rangle$ and $|\phi\rangle$ is denoted by $\langle\psi\,|\,\phi\rangle$, while their outer product $|\psi\rangle\,\langle\phi|$ is a matrix. We will also occasionally denote rank-1 projectors by a capitalised Pi, e.g. $\Pi_\psi := |\psi\rangle\langle\psi|$.

This Dirac notation and the braket is used for vectors and inner products throughout this thesis, whether they are classical or quantum.

Physically, the computational basis states that we usually choose correspond to eigenstates of the z-component of the spin operator (i.e. the Pauli Z operator) of the underlying system. In addition to normalisation, quantum states also have a 'global phase invariance', in that for any $\theta \in \mathbb{R}$, $|\psi\rangle$ is identified with $\mathrm{e}^{\mathrm{i}\theta}\,|\psi\rangle$.

Quantum registers can also exist in probabilistic mixtures of states; to make the distinction, the simpler superposition states are called pure states, and their probabilistic mixtures are known as mixed states. Thus in the general case, a d-dimensional quantum state ρ is represented by a $d \times d$ positive semi-definite matrix with complex entries, and is normalised to have unit trace.

Unitary operators: There are two ways in which we can compute on a state $|\psi\rangle$. The first is by *unitary evolution* of the system under the Schrödinger equation with a specified Hamiltonian operator H

$$i\frac{d\,|\psi\rangle}{dt} = H\,|\psi\rangle\,,$$

where H is a hermitian matrix. Closed systems undergo reversible dynamics in quantum mechanics, and this dynamics is represented by unitary matrices. Since we can think of $|\psi\rangle$ as a vector in $\mathbb{C}^{2^m}$, a computation is represented by multiplication of this state by a $U \in \mathrm{SU}(2^m)$, i.e. $|\psi_{\text{out}}\rangle = \mathbf{U}\,|\psi_{\text{in}}\rangle$. Recall that a matrix $\mathbf{U}$ is said to be unitary if $\mathbf{U}\mathbf{U}^\dagger = \mathbb{1}$, where $\mathbf{U}^\dagger$ is the conjugate transpose of U. Unitary operators then act by left multiplication on pure states vector, and by conjugation on density matrices.

Measurements: The second kind of operation we can perform on $|\psi\rangle$ is measurement. For our purposes, the postulates of quantum mechanics say that on measuring a superposition state $|\psi\rangle$ as above *in the basis* $\{|i\rangle\}$, we obtain as outcome the random m-bit string corresponding to the basis state $|i\rangle$ with probability $p(i) = |\alpha_i|^2$. Since we have chosen states to be normalised, the measurement gives a valid probability mass function over the set of classical m-bit strings. After the measurement, the state "collapses" to the observed basis state $|i\rangle$, and no further information can be retrieved from the original state; to repeat the sampling achieved by this measurement, we need to prepare $|\psi\rangle$ for each repetition.

For two registers A and B and their state $|\psi\rangle^{AB} = \sum_{x,x'} \alpha_{x,x'}\,|x\rangle^A \otimes |x'\rangle^B \in \mathcal{H}^A \otimes \mathcal{H}^B$, a measurement of the register B for $|\psi\rangle^{AB}$ in the computational basis $\{|x'\rangle^B\}$ of $\mathcal{H}^B$ yields an outcome x' with probability $p(x') = \sum_x p(x, x')$, where $p(x, x') = |\alpha_{x,x'}|^2$. The superscripts of a state or an operator represent which register the state or the operator belongs to; we omit the superscript if it is clear from the context.

Input models: We will use two kinds of input models. The first is a quantum analogue of the classical query model, where inputs are accessed via a black-box or oracle that can be queried with an index i and returns the i-th bit of the input bit string. For a bit string $x \in \{0,1\}^n$ we assume access to a unitary $\mathcal{O}_x$ which performs the map

$$\mathcal{O}_x\,|i\rangle\,|b\rangle\,|z\rangle = |i\rangle\,|b \oplus x_i\rangle\,|z\rangle\,,$$

where the first register consists of $\lceil \log n \rceil$ qubits, the second is a single qubit register to store the output of the query, and the third is any additional workspace the quantum computer might have and is not affected by the query. Here $\oplus$ is addition on $\mathbb{Z}_2$, i.e. the XOR operation

in Boolean logic. Note that $\mathcal{O}_x$ can be used by a quantum computer to make queries in superposition:

$$\mathcal{O}_x \left(\frac{1}{n} \sum_{i=1}^{n} |i\rangle \, |b\rangle \, |z\rangle \right) = \frac{1}{n} \sum_{i} |i\rangle \, |b \oplus x_i\rangle \, |z\rangle \, ,$$

The second is the QRAM input model in which data is preprocessed and stored in a special type of binary tree data structure, which can be addressed in quantum superposition. This enables one to prepare states that encode the input data in their amplitudes, rather than as the labels of computational basis states. We shall see more about this input model in Chapter 4, which is the only chapter in this thesis that makes use of this model.

Quantum algorithms : Broadly speaking, a quantum algorithm starts by initializing m qubits in a fixed state $|0\rangle^{\otimes m}$, which we may write as $|0\rangle$ if m is clear from context. Then, we apply a 2^m-dimensional unitary operator $\mathbf{U}$ to $|0\rangle^{\otimes m}$, to prepare a state $\mathbf{U}\,|0\rangle^{\otimes m}$. Finally, a measurement on $\mathbf{U}\,|0\rangle^{\otimes m}$ is performed to sample an m-bit string from the probability distribution given by $\mathbf{U}\,|0\rangle^{\otimes m}$.

As in classical computer science, we can work in a quantum circuit model, by first fixing a basic set of operations or 'gates' (analogous to classical Boolean AND and OR), achievable on the underlying hardware. It is then possible to compile a large 'algorithm' U down into elementary unitary operations, or quantum gates, and represent it by a quantum circuit composed of sequential applications of unitaries acting on at most two qubits at a time. Each of these unitaries is called an elementary quantum gate, and we can choose to work with different basic or fundamental 'gate sets' that can be used to construct an approximate circuit for any unitary, to any arbitrary precision, via the Solovay-Kitaev theorem. The runtime of a quantum algorithm is essentially determined by the number of elementary quantum gates in its circuit. Given a circuit for a unitary, it is straightforward to implement its inverse, or control its application on ancillary registers, with only a small overhead in the gate complexity.

Complexity measures: For many theoretical studies in complexity, the query input model is a powerful setting where several results have been proven. In this model, the total number of queries made to the input oracle is the primary measure of algorithmic complexity, known as the *query complexity*.

For practical purposes, it is more important to understand the number of elementary quantum gates used to implement the unitary circuit corresponding to the algorithm in the quantum circuit model. This is known as the *gate complexity* of the algorithm. The depth of the circuit is directly related to the time complexity, and gives an idea of how parallelisable the algorithm is.

We use the standard complexity theoretic notation of $\widetilde{\mathcal{O}}$ to hide polylogarithmic factors.

Chapter 1

Implementing functions of Hermitian matrices with the LCU method

Synopsis: We consider methods for implementing smooth functions $f(A)$ of a sparse Hermitian matrix A on a quantum computer, and analyse a further combination of these techniques which has advantages of simplicity and resource consumption in some cases. Our construction uses the linear combination of unitaries method with Chebyshev polynomial approximations. The query complexity we obtain is $\mathcal{O}(\log \frac{C}{\epsilon})$ where ϵ is the approximation precision, and $C > 0$ is an upper bound on the magnitudes of the derivatives of the function f over the domain of interest. The success probability depends on the 1-norm of the Taylor series coefficients of f, the sparsity d of the matrix, and inversely on the smallest singular value of the target matrix $f(A)$.

1.1 Introduction

There are many quantum algorithms that exhibit an advantage over known classical algorithms[1]. Such a diversity of results can make it difficult to express the computational capability of a quantum computer in a clear and faithful manner to someone new to quantum computing. Here we address a large family of quantum algorithms that are often used as the main subroutine in many important applications. In particular we consider quantum algorithms that apply some smooth function of a Hermitian matrix to an input state. Examples of algorithms of this type include Hamiltonian simulation used in a variety of applications including quantum chemistry, the Quantum Linear Systems or matrix inversion algorithm [HHL09, CKS17] used in quantum machine learning applications, and sampling from Gibbs distributions used in the recent quantum semi-definite programming (SDP) solvers [BKL+19, VGG+17].

We briefly outline the methods for implementing matrix functions $f(A)$ used in these algorithms, and we further use one of them to obtain an algorithm with query complexity expressed simply in terms of properties of the matrix A and the Taylor expansion of f.

[1] at the time of writing there are over 380 papers listed in the quantum algorithms zoo

Restricting to Hermitian matrices allows one to take advantage of their spectral decomposition to naturally extend results on approximating real functions to functions of matrices. Thus, for a real valued smooth function f, we look for a quantum algorithm which, when equipped with a quantum oracle for a Hermitian matrix A and a map that prepares some state $|x\rangle$, returns a state that is close to $\frac{f(A)|x\rangle}{\|f(A)|x\rangle\|}$ in l_2-norm. If measurements are involved, we require the desired output state to be obtained with high probability.

Among the earliest work of this kind, Klappenecker and Rötteler [KR03] studied the implementation of functions of unitary matrices, under some mild assumptions. This work embodies the same principles and ideas that were later developed into a method for Hermitian matrices in works of Berry, Childs, Kothari and collaborators. Other early work in this direction was motivated by studies on Hamiltonian simulation (for example, [SRG+02] and [CW12]), and algorithms for quantumly solving ordinary differential equations (for example, [LO08] describe a method to implement non-linear transformations of the amplitudes of a given input state). Kothari [Kot14] gives a detailed description of probabilistic implementations of operators and technical lemmas on modified versions of amplitude amplification. Broadly, three different methods have emerged to realise the action of functions of Hermitian matrices on a quantum computer:

1. Using Hamiltonian simulation and Quantum Phase Estimation (QPE) as a technique to obtain a representation of the target state in the spectral basis of the matrix, followed by applying a suitably engineered unitary that computes the matrix function. This is the method used in the matrix inversion algorithm of [HHL09] and in the quantum recommendation systems algorithm of [KP17b].

2. By representing the target matrix as a Linear Combination of Unitaries (LCU). Given an algorithm to implement each of the unitaries in the summation, the target matrix can be embedded in a unitary operation on a larger state space (adjoining the necessary ancillary qubits). This necessitates a post-selection step at the end, and so produces the required state with some associated success probability. This method has been used widely for Hamiltonian simulation and matrix inversion algorithms [CW12, BCC+15, CKS17].

3. More recently, Low and Chuang [LC16] have introduced a method called Qubitisation and Quantum Signal Processing (QSP) (also referred to as the block-encoding method). They study how to implement functions of a diagonalisable complex matrix when provided with a unitary oracle and a signal state such that the action of the unitary on the subspace flagged by the signal state is proportional to the action of the target matrix. This method is thought to have optimal query complexity and optimal ancilla requirements for a large class of functions. This method has been substantially expanded and generalised to what has been termed Quantum Singular Value Transformation in a recent article of Gilyén et al. [GSL+19].

Typically, these methods have been applied to a specific function or application such as Hamiltonian simulation. The question of whether there exisits a more general result for an arbitrary function f is very natural. Recently, van Apeldoorn et al. [VGG+17] used the Linear Combination of Unitaries or LCU method [CKS17] with an approximation by Fourier series to provide a constructive approach for the implementation of bounded smooth functions of a $d-$sparse Hermitian matrix (specified by an oracle) on quantum computers. They give an algorithm for constructing the approximating linear combination, and their quantum algorithm has query complexity linear in the sparsity d. This method uses Hamiltonian simulation as a black-box subroutine and involves converting the Taylor series of the function into a Fourier series, through a sequence of approximation steps. Both [LC16] and [GSL+19] also prove theorems about the classes of functions their methods can implement and describe the approach to construct the implementation for important examples like Hamiltonian simulation.

In this chapter, we first describe the spectral method, the LCU method, and the Qubitisation method in sections 1.3, 1.4, and 1.5 respectively. We then give a constructive approach for implementing smooth functions of a Hermitian matrix on a quantum computer in section 1.6, based on the method of Chebyshev polynomial approximations used for matrix inversion in [CKS17]. The complexity of this alternative method is not directly comparable to that of [VGG+17] but the main attraction is its simplicity. The use of Chebyshev polynomials allows the query complexity of the quantum algorithm to be directly obtained from the properties of the Taylor series expansion of f.

1.2 Preliminaries

We follow the usual model in which quantum algorithms access classical data (such as matrix entries) using a unitary oracle, and use the query complexity as a measure of efficiency. Another important quantity in the circuit model of quantum algorithms is the gate complexity, the number of 2-qubit gates used. An algorithm is gate efficient if its gate complexity is larger than the query complexity by at most poly-logarithmic factors. For details on the properties of the quantum walk construction including its gate complexity, we refer to [BC12] and [BCK15].

A matrix is d-sparse if in any row, there are at most d non-zero entries. For a d-sparse $N \times N$ matrix A, we assume we have an oracle $\mathcal{P}_A$ which performs two functions:

$$|j,k,z\rangle \mapsto |j,k,z \oplus A_{jk}\rangle \tag{1.1}$$

$$|j,l\rangle \mapsto |j,\mathrm{col}(j,l)\rangle \tag{1.2}$$

for all $j,k \in \{1,\ldots,N\}$, and $l \in \{1,\ldots,d\}$. The first line simply returns the matrix entry A_{jk} in fixed-precision arithmetic. The second line computes the column index of the l^{th} non-zero

entry in row j, with the convention that it returns the column index of the first zero entry when there are fewer than l non-zero entries in row j. When this can be efficiently done, the matrix is said to be efficiently row-computable.

Functions of a matrix are defined through its spectral decomposition. An $N \times N$ Hermitian matrix A has N real eigenvalues λ_j, with a spectral decomposition as a sum of projectors onto its eigenspaces spanned by the eigenvectors $|u_j\rangle$. A function f of such a matrix is then defined as having the same eigenspaces, but with the eigenvalues $f(\lambda_j)$

$$A = \sum_{j=1}^{N} \lambda_j |u_j\rangle \langle u_j| \implies f(A) = \sum_{j=1}^{N} f(\lambda_j) |u_j\rangle \langle u_j| . \tag{1.3}$$

For finite dimensional matrices, any two functions f and g that are equal at the N eigenvalues of the matrix give rise to the same matrix function. This suggests that it might be possible to treat matrix functions as corresponding *interpolating* polynomials.

1.3 The spectral method

The first of the methods mentioned in the introduction uses Hamiltonian simulation and phase estimation as algorithmic primitives. The general framework is as follows.

Given some initial state $|\psi\rangle$, the first step is to perform Quantum Phase Estimation (QPE) on the Hamiltonian evolution under the $N \times N$ Hermitian matrix A, logically decomposing the state in the spectral basis of A, as a linear combination of product states with an ancillary register containing (approximate) eigenvalues of A normalised to $[0, 1]$. To this end, the first step in QPE is to simulate Hamiltonian evolution under A, which is the operation of (approximately) preparing the state $e^{iAt} |\psi\rangle$ for a chosen time t; for QPE, we will actually require the Hamiltonian evolution to be performed in superposition for a set of time parameters, which can be done using a control register and a conditional evolution operator of the form $\sum_t |t\rangle\langle t| \otimes e^{iAt}$. At this stage the novelty lies in constructing an operation that will use the output state from the previous step to transform the probability amplitudes to the chosen function of the eigenvalues - typically a unitary conditioned on the register containing the eigenvalues. Schematically,

$$|\psi\rangle \xrightarrow{\text{QPE on } e^{iAt}} \sum_{j=1}^{N} \psi_j |a_j\rangle |\tilde{\lambda}_j\rangle \xrightarrow{\text{Algo. (e.g. HHL)}} C \sum_{j=1}^{N} f(\tilde{\lambda}_j)\psi_j |a_j\rangle |\tilde{\lambda}_j\rangle ,$$

where $A |a_j\rangle = \lambda_j |a_j\rangle$, $|\psi\rangle = \sum_{i=1}^{N} \psi_j |a_j\rangle$, the $\tilde{\lambda}_j$ approximate the true eigenvalues λ_j, and C is a normalisation factor that is $\mathcal{O}(\max_j f(\lambda_j))$. All three steps will have some associated error, and the last step will typically only succeed probabilistically. Choosing a precision

$\epsilon > 0$, the final state $\sum_{i=1}^{N} f(\tilde{\lambda}_j)\psi_j \left|a_j\right\rangle$ (having uncomputed the eigenvalue register coming from QPE) can be made ϵ-close to $f(A)\left|\psi\right\rangle$, suitably normalised.

Harrow, Hassidim and Lloyd [HHL09] used this method to devise an algorithm for solving a system of linear equations $A\vec{x} = \vec{b}$, in a formulation called the Quantum Linear Systems Problem (QLSP). The key step is matrix inversion, corresponding to the function $f(\lambda_j) = 1/\lambda_j$. Given the oracle in (1.1) for a Hermitian matrix A, and a state preparation map for an input vector $\vec{b}$, the HHL algorithm outputs a quantum state $\left|\vec{x}\right\rangle$ that is ϵ-close in the l_2-norm to the normalised solution vector $A^{-1}\vec{b}/\|A^{-1}\vec{b}\|$. The matrix inversion routine essentially implements conditional rotations to use phase-estimated eigenvalues to create the necessary amplitude factors. The transformation achieved by a simple conditional rotation is

$$\left|\mu\right\rangle\left|0\right\rangle \xrightarrow{\mathrm{CR}_y(2\cos^{-1}(\mu))} \left|\mu\right\rangle\left(\mu\left|0\right\rangle + \sqrt{1-\mu^2}\left|1\right\rangle\right),$$

where $\mu \in [0,1]$ and is represented to fixed precision in the first qubit register. Cao et al. [CPP+13] use the HHL algorithm to solve the Poisson equation under some regularity assumptions, and give details of efficient quantum subroutines for the string of computations

$$\left|\lambda\right\rangle\left|0\right\rangle\left|0\right\rangle \to \left|\lambda\right\rangle\left|C/\lambda\right\rangle\left|0\right\rangle \to \left|\lambda\right\rangle\left|C/\lambda\right\rangle\left|\cos^{-1}(C/\lambda)\right\rangle,$$

which are useful in matrix inversion. The actual function that is implemented is not quite $1/x$, but a carefully chosen filter function that matches with the inverse on the domain of interest, and ensures that the error can be kept under control. The choice of filter functions and their error analysis forms a major part of the work in designing the conditional unitary that implements the desired function.

Complexity and hardness results for matrix inversion

The HHL algorithm was originally presented as a method for the efficient estimation of averages or other statistical quantities associated with the probability distribution corresponding to the normalised solution vector of a linear system of equations. The advantage of the algorithm lies in being able to prepare this probability distribution as a quantum state, and the state may then be used to sample from this distribution, or to compute $\vec{x}^{\dagger}M\vec{x} := \left\langle x\right| M \left|x\right\rangle$ for operators M.

When the matrix A is d-sparse and efficiently row-computable, the algorithm has query complexity $\mathcal{O}(\mathrm{poly}(d, 1/\epsilon, \kappa))$. Here, ϵ is the accuracy to which the output state approximates the normalised solution vector, and κ is the condition number of the matrix A. The condition number is given by the ratio of the largest to the smallest eigenvalues for a Hermitian matrix, and measures how invertible the matrix is: $\kappa \to \infty$ reflects an eigenvalue of the matrix going to zero, making it singular. The dependence of the algorithm on the condition number

enters through the of filter function, which is chosen such that it matches $1/x$ on the domain $[-1, -1/\kappa] \cup [1/\kappa, 1]$ and interpolates between these two pieces in $[-1/\kappa, 1/\kappa]$. Consequently the success probability of the conditional rotation step also involves κ.

In the regime where the sparsity $d = \mathcal{O}(\log N)$, the HHL algorithm can be exponentially faster than the best known classical algorithms achieving the same results. It is still unknown whether better classical algorithms exist in the slightly modified framework of QLSP, i.e., the oracular setting where the goal is to compute global or statistical properties of the solution vector.

Nevertheless, the complexity of this algorithm was shown by [HHL09] to be nearly optimal: 1) under the complexity theoretic assumption $\mathsf{BQP} \neq \mathsf{PSPACE}$ the dependence on condition number cannot be made sublinear, i.e. improved to $\kappa^{1-\delta}$ for any $\delta > 0$, and 2) under the assumption that $\mathsf{BQP} \neq \mathsf{PP}$, the error dependence cannot be improved to $\mathcal{O}(\mathrm{poly} \log 1/\epsilon)$. The proof is an efficient reduction from simulating a general quantum circuit to matrix inversion, establishing that matrix inversion is a BQP-complete problem, so that the existence of a classical algorithm for this problem implies the ability to classically simulate quantum mechanics efficiently, which is widely believed to be impossible. The class BQP consists of problems that have a bounded error polynomial time quantum algorithm that succeeds with a constant probability $p \geq c > 1/2$, while PP consists of problems with probabilistic polynomial time classical algorithms with the success probability allowed to be arbitrarily close to $1/2$.

For the problem of preparing the state encoding the probability distribution corresponding to the solution vector, avoiding errors and repetitions from the sampling step, Childs, Kothari and Somma [CKS17] designed an algorithm for QLSP with exponentially improved dependence on the precision parameter ϵ. The key ingredient in their approach is the method of writing the target operator, A^{-1} in this case, as a linear combination of unitary operators. We discuss this LCU method in the next section.

1.4 The Linear Combination of Unitaries (LCU) method

One of the disadvantages in using QPE is that achieving ϵ-precision requires $\mathcal{O}(1/\epsilon)$ uses of the matrix oracle. The LCU method offers a way to overcome this disadvantage by exploiting results from approximation theory.

The LCU method is a way to probabilistically implement an operator specified as a linear combination of unitary operators with known implementations. In essence, we construct a larger unitary matrix of which the the matrix $f(A)$ is a sub-matrix or block. Childs and Wiebe [CW12] show how to implement a sum of two unitaries. We describe this simple case below.

Suppose $A = \alpha_0 U_0 + \alpha_1 U_1$. Without loss of generality $\alpha_i > 0$, since phase factors can be absorbed into the unitaries. Consider a state preparation unitary V_α which has the action

$$|0\rangle \mapsto \frac{1}{\sqrt{\alpha}}(\sqrt{\alpha_0}\,|0\rangle + \sqrt{\alpha_1}\,|1\rangle)$$
$$|1\rangle \mapsto \frac{1}{\sqrt{\alpha}}(-\sqrt{\alpha_1}\,|0\rangle + \sqrt{\alpha_0}\,|1\rangle),$$

where $\alpha = \alpha_0 + \alpha_1$. When dealing with a linear combination of more than two unitaries, there is a lot of freedom in the choice of this V_α, as we will see later.

Assume that we can perform the unitaries U_0 and U_1 controlled by an ancillary qubit, i.e., that we can apply the conditional unitary $U = |0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes U_1$. Then for a state $|\psi\rangle$ for which we want $A|\psi\rangle$, first attach an ancillary qubit and perform the map V_α on it, followed by U, and finally uncompute the ancilla with $V_\alpha^\dagger$. This results in the following transformation

$$\begin{aligned}
|0\rangle|\psi\rangle &\xrightarrow{V_\alpha \otimes \mathbb{1}} \frac{1}{\sqrt{\alpha}}(\sqrt{\alpha_0}\,|0\rangle + \sqrt{\alpha_1}\,|1\rangle)\,|\psi\rangle \\
&\xrightarrow{U} \frac{1}{\sqrt{\alpha}}(\sqrt{\alpha_0}\,|0\rangle U_0|\psi\rangle + \sqrt{\alpha_1}\,|1\rangle U_1|\psi\rangle) \\
&\xrightarrow{V_\alpha^\dagger \otimes \mathbb{1}} \frac{1}{\alpha}\left(|0\rangle(\alpha_0 U_0 + \alpha_1 U_1)|\psi\rangle + \sqrt{\alpha_0\alpha_1}\,|1\rangle(U_1 - U_0)|\psi\rangle\right).
\end{aligned}$$

Measuring the ancilla and getting the outcome 0 will leave behind the state $A|\psi\rangle$, up to normalisation. Getting a measurement outcome of 1 means the algorithm fails. The probability of failure can be easily bounded, as $p(fail) \leq \frac{\alpha_1\alpha_0\|U_1 - U_0\|^2}{\alpha^2} \leq \frac{4\alpha_1\alpha_0}{\alpha^2}$, since the distance between two unitaries is at most 2.

Success probability and complexity

In most cases of interest, the probability of success can be increased using amplitude amplification, by repeating the procedure $\mathcal{O}(\alpha/\|A|\psi\rangle\|)$ times, when we have an estimate of the norm in the denominator. This gives a quantum algorithm that prepares the desired quantum state with constant success probability and outputs a single bit indicating whether it was successful. Indeed, when the LCU method is used to implement non-unitary operations such as in matrix inversion, a significant contribution to the complexity comes from the fact that usually the success probability is small.

This framework was investigated in detail by Berry, Childs, Kothari and coworkers. The resulting method of implementing linear combinations of unitaries makes it straightforward to translate results on approximating real functions into implementations for matrix functions: since Hermitian matrices have all real eigenvalues, we just need to find a good approximation to the real function $f(x)$. The overall query complexity of the algorithm will depend on the

1-norm or weight α of the coefficients of the linear combination, the number of terms m, and the least eigenvalue of the matrix function $f(A)$.

Thus, finding a new algorithm boils down to optimising the first two parameters, and getting good bounds on the eigenvalues of $f(A)$. We also need to choose the basis functions used in the approximation in such a way that on translating the statement to matrix functions, we get unitaries which we know how to implement - for example, a Fourier series in e^{ixt} gives rise to unitaries e^{iAt} that can be implemented via Hamiltonian simulation techniques.

[CKS17] used this method for the function $f(x) = 1/x$ to obtain improved error dependence of the algorithm for the QLSP. The dependence of the complexity of their algorithm on the precision parameter is $\mathcal{O}(\log(1/\epsilon))$, an exponential improvement over the precision dependence in [HHL09]. This is achieved by carefully choosing the approximating series and its truncation, such that with $m = \mathcal{O}(\log(1/\epsilon))$ terms, ϵ precision is achieved.

For completeness, a full description of the LCU method is included in Appendix A. We now turn to the most recent method of implementing matrix functions, the method of qubitisation and quantum signal processing, before returning to the LCU method in section 1.6.

1.5 Qubitisation or the block-encoding method

Introduced by Low and Chuang [LC16, LC17], this framework subsumes and generalises the LCU method. The method can be split into two steps - qubitisation and quantum signal processing. The idea behind quantum signal processing is to ask what kinds of 2×2 (block) unitaries can be obtained by iterating a given unitary operator, interleaving the iteration with single qubit rotations through different angles. Thus, picking a sequence of phases $\phi_0, \phi_1, \ldots, \phi_k$, we ask what class of untaries can be represented in the form

$$W_{\vec{\phi}} = e^{i\phi_0\sigma_z} U e^{i\phi_1\sigma_z} U \ldots U e^{i\phi_k\sigma_z},$$

where U is the input unitary. When the input $U = U(x)$ is parametrised by some $x \in [-1, 1]$, we can enforce $W_{\vec{\phi}}$ to have the form

$$W_{\vec{\phi}} = \begin{pmatrix} P(x) & Q(x) \\ Q^*(x) & P(x) \end{pmatrix}$$

for some functions $P(x)$ and $Q(x)$, and work out what properties these functions can have. We can also calculate what choice of phases ϕ_j give rise to a certain P and Q. This then provides a method for constructing functions of U using iteration alternating with a sequence of single qubit rotations.

Qubitisation, as the name suggests, is a technique of obtaining a suitable block encoding of an input matrix A, on which quantum signal processing can then be applied. The input is an $(m+n)$-qubit unitary matrix U that encodes an n-qubit normal operator A (i.e. $AA^\dagger = A^\dagger A$) in its top left block. More precisely, given a signal state $|G\rangle = \hat{G}|0\rangle$ that flags a subspace of the n-dimensional signal space, the input unitary U has the block form

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix},$$

i.e. $\langle G|U|G\rangle = A$ is the encoded operator (where we assume A is normalised so that $\|A\| \leq 1$). Since normal operators have a spectral decomposition, we can write $A = \sum_\lambda \lambda e^{i\theta_\lambda}$. The goal is to use U, G, their controlled versions and conjugates to obtain a unitary W that can be expressed as a direct sum over $SU(2)$ invariant subspaces, i.e., to 'qubitise' U:

$$W = \begin{pmatrix} A & -g(A) \\ g(A) & A^\dagger \end{pmatrix} = \sum_\lambda \begin{pmatrix} \lambda e^{i\theta_\lambda} & -\sqrt{1-|\lambda|^2} \\ \sqrt{1-|\lambda|^2} & \lambda e^{-i\theta_\lambda} \end{pmatrix} \otimes |\lambda\rangle\langle\lambda| .$$

The unitary W, called an iterate, has an $SU(2)$ invariant subspace that contains the signal state $|G\rangle$, and satisfies $\langle G|W|G\rangle = A$. This iterate can be used to approximately implement a wide range of matrix functions in the form $\langle G|W_{\vec{\phi}}|G\rangle = f(A) + ig(A)$, where f, g are real functions, and $\vec{\phi}$ represents the sequence of phases ϕ_j. The construction corresponds to quantum signal processing when the functions f and g have opposite parity, in which case the matrix function is implemented as a 2×2 block unitary. Furthermore, the algorithms using this technique generally achieve polylogarithmic dependence on precision, and are shown to have optimal query complexity in several cases. The ancillary qubit requirement is brought down significantly since the primitive gates used are single qubit rotations, and controlled versions of the input unitaries.

The iterate W is similar to the quantum walk operator used with the Chebyshev decompositions for the LCU method. Indeed, when A is Hermitian, W is the same as that walk operator, up to a reflection. Furthermore, the unitary on the larger system consisting of input and ancilla qubits that is constructed in the LCU approach is a particular case of a block-encoding, since it achieves the same result: given $A = \sum_i \alpha_i U_i$, the operator the LCU method constructs has A/α as the top left block (with $\alpha = \sum_i |\alpha_i|$).

The optimal Hamiltonian simulation algorithm based on the qubitisation method [LC16] can be used to speed up any algorithm that uses Hamiltonian simulation as a subroutine. Chakraborty, Gilyén and Jeffery [CGJ19] have applied the qubitisation method, also called the block encoding method, to obtain an improved Hamiltonian simulation technique for non-sparse matrices, improved quantum algorithms for linear algebra applications such as regression, and for estimating quantities like effective resistance in networks. Furthermore, they note that results in the block-encoding framework apply also when the input is specified

in the quantum data structure model (e.g. QRAM) rather than as a unitary oracle. This connection is established through a result of Kerenedis and Prakash [KP17a], which shows that if a matrix A is stored in a quantum data structure such as QRAM, an approximate block encoding for it can be implemented with polylogarithmic overhead.

The spectral and LCU methods were originally developed for and apply to Hermitian matrices, taking advantage of the fact that they have a spectral decomposition with real eigenvalues. Qubitisation and quantum signal processing can deal with normal operators as well, which have complex eigenvalues. Very recently, [GSL+19] have further generalised the qubitisation method using the idea of singular value decompositions. They show how matrix arithmetic and a variety of linear algebra applications can be achieved in a unifying framework of what they call 'quantum singular value transformation'. Given any rectangular matrix, one has a singular value decomposition $A = U\Sigma V^\dagger$, where Σ is the diagonal matrix of singular values, and U and V are orthogonal matrices with columns being the left and right eigenvectors of A respectively. Essentially, [GSL+19] construct a method to implement functions of the singular values, i.e., functions defined by $f(A) := Uf(\Sigma)V^\dagger$. Their extensive analysis also shows how the method achieves optimal complexity for a wide variety of quantum algorithms.

In the last three sections we have briefly looked at the methods used for implementing smooth functions of (Hermitian) matrices on a quantum computer. In the rest of this chapter, we analyse a certain combination of these methods, using the LCU technique with Chebyshev polynomial approximations and a quantum walk construction.

1.6 Implementing smooth functions using the LCU method

Given a Hermitian matrix A and a smooth function $f : [-1, 1] \to \mathbb{R}$, we describe below a quantum algorithm that implements an operator proportional to $f(A)$ for any input state $|\psi\rangle$.

Theorem 1.1. *Consider a quantum oracle for a d-sparse Hermitian matrix A acting on n-qubits with $\|A\| \leq 1$, and a function $f : [-1, 1] \to \mathbb{R}$. If f has a Taylor series $f(x) = \sum \alpha_i x^i$ about $x_0 = 0$, then for any n-qubit state $|\psi\rangle$, there is a quantum circuit that probabilistically prepares a state $|\tilde{\psi}\rangle = \left(\langle 0^t| \otimes \mathbb{1}\right) U |0^t\rangle |\psi\rangle$, where U is a unitary acting on t ancilla and n system qubits, such that*

$$\left\| |\tilde{\psi}\rangle - \frac{f(A)|\psi\rangle}{\|f(A)|\psi\rangle\|} \right\| < \epsilon \tag{1.4}$$

using $\mathcal{O}(L)$ queries to the oracle for A, with a success probability $p \geq \mu/\gamma$, where

$$L > \log \frac{C}{\epsilon}, \quad \mu = \inf_x |f(x)|, \quad \text{and} \quad \gamma = \sum_{i=1}^{L} d^i |\alpha_i|. \tag{1.5}$$

The circuit also outputs a flag indicating success.

Here $C > 0$ is an upper bound on the magnitudes of the derivatives of f in $(-1, 1)$, and μ is the eigenvalue of $f(A)$ with the least magnitude on the domain of interest (i.e. on the spectrum of A).

Remark. *The key property required of f in Theorem 1.1 is that it can be well approximated by a sequence of polynomials $\sum_{i=0}^{k} \alpha_i x^i$ that converge to f as the degree k is increased. This means that we do not strictly require f to be smooth, or even have bounded derivatives, as long as an approximating polynomial of sufficient precision is available. The conditions on the derivatives provide one such polynomial approximation scheme, via Taylor's theorem. As an example of a function that is not differentiable at $x = 0$ but still has a well behaved polynomial approximations, consider $f(x) := \frac{\sin x}{x}$.*

In the rest of this section, we prove this theorem by constructing the algorithm that achieves the target operation. For simplicity, we focus on the case where $\|A\| \leq 1$. The case when $1 < \|A\| \leq \Lambda$ (and correspondingly the domain of f is $[-\Lambda, \Lambda]$) can be reduced to this case by scaling appropriately.

1.6.1 Chebyshev series

The circuit in Theorem 1 is obtained using the LCU method. The first step is to derive an approximation for $f(A)$ in terms of Chebyshev polynomials. The matrix functions corresponding to these Chebyshev polynomials are then performed on a quantum computer using a quantum random walk.

The Chebyshev polynomials of the first kind, denoted T_n where n is the degree, are orthonormal polynomials on $[-1, 1]$ with weight function $(\sqrt{1 - x^2})^{-1}$. We collect a few relevant facts about these polynomials in Appendix C. In particular, we use two nice properties of Chebyshev polynomials in the quantum algorithm. The first is that monomials x^k on $[-1, 1]$ can be exactly represented as a finite sum of Chebyshev polynomials

$$x^k = \sum_{j=0}^{k} C_{kj} \mathcal{T}_j(x), \tag{1.6}$$

where the coefficients are given by

$$C_{kj} = \begin{cases} \frac{1}{2^{k-1}} \binom{k}{(k-j)/2}, & \text{if } (k-j) \text{ is even} \\ 0, & \text{otherwise,} \end{cases} \tag{1.7}$$

for $j > 0$, and $C_{k0} = \frac{1}{2^k}\binom{k}{k/2}$ is non-zero when k is even. The second useful property is that the coefficients are positive and sum up to 1

$$\sum_{j=0}^{k} C_{kj} = 1, \tag{1.8}$$

which can be seen using $\mathcal{T}_n(\cos\theta) = \cos n\theta$. We use these properties to write down a truncated Chebyshev series for $f(x)$, based on the Taylor series, which will lead to a simple expression for the success probability in the LCU method.

For a smooth function f on the interval $[-1, 1]$, consider the Taylor series about $x_0 = 0$, (also called the Maclaurin series)

$$f(x) = \sum_{i=0}^{\infty} \alpha_i x^i, \tag{1.9}$$

where $\alpha_i = \frac{f^{(i)}(0)}{i!}$ is the i^{th} Taylor coefficient, $f^{(i)}$ denoting the i^{th} derivative of f. Suppose the radius of convergence of the series is some $r > 0$. Truncating this series for some finite integer L, we get the Taylor polynomial, with truncation error bounded by Taylor's theorem

$$\tilde{f}(x) = \sum_{i=0}^{L-1} \alpha_i x^i, \tag{1.10}$$

$$\forall x \in (-1, 1),\ \left|f(x) - \tilde{f}(x)\right| \leq \frac{f^{(L)}(\xi_L)}{(L)!}|\xi_L|^L, \tag{1.11}$$

for some $\xi_L \in (-1, 1)$. Let us denote by $\alpha := \sum_{i=0}^{L-1} |\alpha_i|$ the 1-norm of the coefficients. If we have bounds on the derivatives of f, or if we can bound the tail of the series as for the exponential function, we can get a good bound on the truncation error. This will enable us to quantify the rate of convergence of the series, and to decide the order of truncation based on the desired precision.

A simple assumption that we can make about the derivatives of f is that they are bounded in magnitude by some known constant C, i.e. $\sup_x |f^{(i)}(x)| \leq C$ for all i and $\forall x \in (-1, 1)$. This holds for Schwartz functions, for example. It is then a simple calculation to show that taking $L > \log_2(C/\epsilon)$ ensures that the truncation error will be smaller than ϵ, as long as $L > 2e$, i.e. $L \geq 6$. [2].

[2] If we instead assume the weaker condition that $\sum_{i=0}^{\infty} |\alpha_i| < B$, and that the series converges in a $(1-\delta)$-ball around x_0, the corresponding truncation order is $L > \frac{1}{\delta}\log\frac{B}{\epsilon}$.

Now using (1.6) to represent the monomials x^i exactly as a finite sum of Chebyshev polynomials, we obtain a truncated Chebyshev series for f. We have

$$\begin{aligned}\tilde{f}(x) &= \sum_{i=0}^{L-1}\sum_{j=0}^{i}\alpha_i C_{ij}\mathcal{T}_j(x)\\ &= \sum_{j=0}^{L-1}\beta_j\mathcal{T}_j(x),\end{aligned} \tag{1.12}$$

with $\beta_j = \sum_{i=j}^{L-1}\alpha_i C_{ij}$. From the observation (1.8) above $\sum_{i=0}^{k} C_{ik} = 1$, so $||\beta||_1 := \beta = \sum_{i=0}^{L-1}|\beta_i| = \sum_{i=0}^{L-1}|\alpha_i|$. The probability of success in the LCU implementation depends on this sum of coefficients, and we note that rewriting the Taylor polynomial as a Chebyshev decomposition does not by itself increase the weight of the coefficients.

However, taking n steps of the quantum walk described in Appendix D results in the transformation $|0^m\rangle|\psi\rangle \mapsto |0^m\rangle\mathcal{T}_n(A/d)|\psi\rangle + |\Phi^{\perp}\rangle$. That is, the quantum walk implements the operator $\mathcal{T}_n(A/d)$ rather than $\mathcal{T}_n(A)$. To account for this, we further rewrite the series (1.12) as

$$\begin{aligned}\tilde{f}(x) &= \sum_{i=0}^{L-1} d^i\alpha_i\cdot\left(\frac{x}{d}\right)^i\\ &= \sum_{j=0}^{L-1}\gamma_j\mathcal{T}_j\left(\frac{x}{d}\right),\end{aligned} \tag{1.13}$$

with $\gamma_j = \sum_{i=j}^{L-1} d^i\alpha_i C_{ij}$. This results in an increase in the weight of the coefficients: $||\gamma||_1 := \gamma = \sum_{i=0}^{L-1}|\gamma_i| = \sum_{i=0}^{L-1}|d^i\alpha_i|$. The truncation error does not change since we are expanding around $x_0 = 0$ and simply rescaling the coefficients and argument of the series.

Finally, when $1 < \|A\| \leq \Lambda$, we can scale down the interval $[-\Lambda, \Lambda]$ to $[-1, 1]$ using the map $x \mapsto x/\Lambda$. Hence the first step is to write the Taylor series in the larger interval, and then rewrite it by scaling the coefficients as in $\tilde{f}(x) = \sum_{i=0}^{L-1}\Lambda^i f_i\cdot(x/\Lambda)^i$. Accordingly, the weight or 1-norm of the coefficients will increase to $\gamma = \sum_{i=0}^{L-1}|(\Lambda d)^i\alpha_i|$.

1.6.2 Algorithm description and complexity

By Lemma 1.2 (Appendix A), we can implement $f(A)$ approximately by using the linear combination of Chebyshev polynomials in Eq. (1.13), using the LCU method as described in Lemma 1.3 (Appendix A). We thus have a quantum algorithm which for an input state $|\psi\rangle$ produces a state $|\tilde{\phi}\rangle$ such that

$$\left\| \frac{f(A)|\psi\rangle}{\|f(A)|\psi\rangle\|} - |\tilde{\phi}\rangle \right\| \leq c'\epsilon, \tag{1.14}$$

for some constant $c' = \Omega(\frac{1}{\mu})$. The algorithm uses $\mathcal{O}(L) = \mathcal{O}(\log \frac{C}{\epsilon})$ queries to the matrix oracle, and succeeds with probability $p := \left|\frac{\|f(A)|\psi\rangle\|}{\gamma}\right|^2 \geq \left(\frac{\mu}{\gamma}\right)^2$, outputting the flag 0 on success. Here μ is the eigenvalue of $f(A)$ with the least magnitude on the domain of interest. For monotone functions, this can be estimated if a suitable upper or lower bound is known on $\|A\|$.

Typically, amplitude amplification is used to boost the probability of success to a constant. The simplest setting where this is possible is when a state preparation map for for the input state $|\psi\rangle$ is available. Using Lemma 1.3, an upper bound on the worst-case query complexity of this implementation when amplitude amplification is used to boost the probability of success is given by

$$\frac{L}{\sqrt{p}} \leq L\frac{\gamma}{||f(A)\,|\psi\rangle\,||_{min}} = \mathcal{O}\left(L(\Lambda d)^{L-1}\frac{\alpha}{\mu}\right),$$

where $\|A\| \leq \Lambda$, and $\gamma := \sum_{i=0}^{L-1} |(\Lambda d)^i f_i| = \mathcal{O}((\Lambda d)^{L-1}\alpha)$ and $\|f(A)\,|\psi\rangle\| \geq \mu$. In fact, $\gamma \approx f(\Lambda d)$. The linear factor of L comes from the fact that we need to implement the Chebyshev polynomials of degree up to L using the quantum walk. The factor γ/μ comes from using amplitude amplification to increase the success probability of obtaining the desired state. A simple lower bound on μ is $f_{min} := \inf_x |f(x)| \leq \mu \leq \|f(A)\,|\psi\rangle\|$ over $x \in [-1, 1]$. Thus if f is such that $|f(x)| \geq 1$, this factor can be omitted from the complexity. The implementation is invariably expensive when $f(A)$ has eigenvalues close to zero.

Thus, if amplitude amplification is used, plugging in the expression for L gives

$$\mathcal{O}\left(\frac{\alpha}{\mu}\left(\frac{C}{\epsilon}\right)^{\log(\Lambda d)}\log\frac{C}{\epsilon}\right) \tag{1.15}$$

queries to the oracle for A, and $\mathcal{O}\left(\frac{\alpha}{\mu}\left(\frac{C}{\epsilon}\right)^{\log(\Lambda d)}\right)$ uses of the input state preparation map.

Typically, one does not know an estimate of the success probability of the method beforehand, and needs to first use some form of (coarse) amplitude estimation in order to make sure amplitude amplification does not overshoot the target. This can be avoided by using a version of amplitude amplification that uses a fixed-point search method, which converges monotonically to the target state [Gue19].

The gate complexity can be obtained by multiplying the query complexity by the gate complexity of performing one step of the quantum walk. From [BCK15], one step of the walk costs only a constant number of queries and $\mathcal{O}\left(\log N + m^{2.5}\right)$ 2-qubit gates, where m is the number of bits of precision used for the entries of the matrix A. The factor of $m^{2.5}$ arises from the cost of black-box state preparation for implementing the quantum walk operator, and can be reduced to nearly $\mathcal{O}(m)$ as recently described in [SLS+19]. For completeness, note that the control state preparation map V in the LCU method can be constructed using the method of [GR02], since the coefficients are known.

Related work

As noted previously, [VGG+17] provide a constructive method for implementing smooth functions of Hermitian matrices, based on transforming the Taylor series for the function into a Fourier series. The algorithm is then obtained by using the LCU method and Hamiltonian simulation to implement the Fourier components. For comparison, we quote below the results as described in Theorem 40 of their paper.

[VGG+17, Theorem 40] Given the Taylor series of f about some point x_0, $f(x_0+x) = \sum_{i=0}^{\infty} a_i x^i$, with convergence radius $r > 0$, if an n-qubit Hermitian operator A satisfies $\|A - x_0\mathbb{1}\| \leq r$, then for $\epsilon \in (0, \frac{1}{2}]$ we can implement a unitary $\tilde{U}_f$ on $t+n$-qubits such that for any n-qubit state $|\psi\rangle$, we have

$$\left\| \left(\langle 0^t| \otimes \mathbb{1} \right) \tilde{U}_f \left| 0^t \right\rangle |\psi\rangle - \frac{f(A)}{B} |\psi\rangle \right\| \leq \epsilon,$$

where $\sum_{i=0}^{\infty} |a_i|(r+\delta)^i \leq B$ for some finite $B > 0$ and $\delta \in (0, r]$. If $\|A\| \leq K$, $r = \mathcal{O}(K)$, and A is d-sparse and accessible using an oracle as in (1.1), then the whole circuit can be implemented using

$$\mathcal{O}\left(\frac{Kd}{\delta} \log\left(\frac{K}{\delta\epsilon} \right) \log\left(\frac{1}{\epsilon} \right) \right)$$

queries to the oracle. The number of 3-qubit gates used for the circuit is larger by the polylogarithmic factor $\mathcal{O}\left(\log N + \log^{2.5}\left(\frac{K}{\delta\epsilon} \right) \right)$.

We notice that the query complexity in this method depends linearly on $d||A||$, in addition to the factor $\log 1/\epsilon$. In the Chebyshev method, the dependence on d and $||A||$ comes in through the success probability, leaving the query complexity dependent only on the properties of the Taylor series approximation.

The methods based on Fourier and Chebyshev series cannot be directly compared because they may not be usable in all situations - the Fourier method is possible whenever Hamiltonian simulation can be performed for the matrix A, while the Chebyshev method is possible only when the quantum walk in Appendix D is feasible to implement. Since Hamiltonian simulation can be performed using quantum walks, the Fourier method has a wider range of applicability, in general.

The probability of success in preparing the desired state by post-selection on the t-ancillary qubits is as expected in the LCU method, given by $\left| \frac{\|f(A)|\psi\rangle\|}{B} \right|^2$.

Advantages of the Chebyshev method

In comparison to the method based on Fourier series approximations, the Chebyshev series based method described here has the following advantages.

1. Since the quantum walk exactly produces the effect of Chebyshev polynomials in A/d (apart from a choice of precision in representing the entries of the matrix), and polynomials have exact representations as a finite sum of Chebyshev polynomials, we can exactly implement polynomial functions of a matrix. In the Fourier series based approach, the series truncation adds another layer to the error in the approximation of polynomials. The simplicity of the analysis also makes it apparent that the Chebyshev method could be more suitable for applications involving polynomial functions, such as iterative methods that use Krylov subspaces.
2. Leaving out amplitude amplification, the query complexity depends only on the degree of the approximating polynomial. This isolation of the dependence of the complexity on the norm of the matrix and its sparsity into the success probability could be useful in studying lower bounds on the query complexity for different matrix functions.
3. Methods based on quantum walks extend to non-sparse matrices, since they do not depend on row computability [BC12]. The complexity will generally be worse, however.
4. The classical calculation of the coefficients is particularly simple for the Chebyshev series.

There are, of course, both advantages and disadvantages of using any method. Chebyshev polynomial implementation uses quantum walk methods, and the construction of the walk requires a doubling of the input space, i.e., $\mathcal{O}(n)$ ancillary qubits. Furthermore, the rescaling of the Taylor series coefficients means that machine errors due to fixed-precision representation are magnified. To work at precision ϵ, we need to use $\Omega(\log_2(B/\epsilon))$ bits for the matrix entries.

1.7 Special function classes

The main difficulty in the approach we have described is the scaling up of the weight of the coefficients in the Taylor series approximation due to the fact that the Chebyshev polynomials obtained from the quantum walk are in A/d rather than A. This affects the success probability, potentially necessitating many rounds of repetition or amplitude amplification. The step that requires this rescaling is the reconstruction of $f(x)$ using an approximation to $f(x/d)$ for a fixed $d > 1$.

But we notice that there are simple functions for which different approaches are possible. For example, for $f(x) = 1/x$, it suffices to take $f(x) = \frac{1}{d} f(\frac{x}{d})$, which holds throughout the

domain of f. Another example is $e^x = (e^{x/d})^d$, which requires repeated application of the operator, d times. We also need to keep track of how the approximation error changes in going from $f(x/d)$ to $f(x)$.

In general, if there is a function $g : \mathbb{R} \to \mathbb{R}$ such that $f(x) = g(f(x/d))$, the efficiency of implementing of $f(A)$ using $f(A/d)$ depends on the nature of g. For some classes of functions, the composition of the function with g reduces to just scaling: $f(x) = g(f(x/d)) = g(d)f(x/d)$. Homogeneous functions are an example of this kind: $f(cx) = c^k f(x)$ for a fixed constant k for any real number c, so $g(d) := d^k$. Homogeneous functions arise mainly as (multivariate) polynomials or rational functions. Below, we make some brief remarks about matrix polynomials and exponentiation.

Polynomials

For the special case of monomials, which are homogeneous functions, we simply use the Chebyshev decomposition (1.6), which is exact. This eliminates the precision parameter ϵ. For $f(x) = x^k$ on $[-1, 1]$, the query complexity is $\mathcal{O}(k)$, or if amplitude amplification is used, $\mathcal{O}\left(kd^{k-1}\lambda^{-k}\right)$ where $\lambda = ||A^{-1}||$ is the least eigenvalue of A (where we assume A does not have 0 as an eigenvalue). For any polynomial of degree k, the complexity is the same to leading order, since the highest degree Chebyshev polynomial comes from the highest degree term in the polynomial. Computing matrix polynomials may find use in iterative methods in numerical linear algebra. These methods typically proceed by approximating a vector $f(A)\vec{v}$ in a Krylov subspace $\mathcal{K}_r(A, \vec{v_0}) := \{\vec{v_0}, A\vec{v_0}, A^2\vec{v_0}, \ldots, A^r\vec{v_0}\}$ starting with an initial guess $\vec{v_0}$, and iteratively improving it. Patel and Priyadarsini [PP18] propose a matrix inversion algorithm using this method. However, they use a different formalism to quantumly implement the monomials.

The exponential function

The matrix exponential is an immensely important function, primarily in the form of the complex exponential e^{iAt} that describes quantum evolution under the Hamiltonian operator A. Hamiltonian simulation is important in a variety of applications, ranging from quantum chemistry, to use as a subroutine in linear algebra and machine learning applications. This vast topic has received a lot of attention in the last two decades, so we shall not endeavour to elaborate on it here.

The simple exponential e^A is also an important function. For example, being able to sample from the Gibbs' state $e^{-H}/\mathrm{Tr}(e^{-H})$ has been found useful in quantum algorithms for semidefinite programming [BKL+19, vAG18]. Exponentiating density matrices can be used to construct a quantum algorithm for principal component analysis [LMR14]. Matrix

exponentiation is also expected to be useful in variational quantum chemistry algorithms, for example in implementing coupled cluster techniques [RBM+17].

Using our approach, to implement the exponential e^A of the Hermitian matrix A, we can first approximate $e^{A/d}$ and repeat this operation d times. If $||A|| \leq 1$, this leads to a query complexity of $\mathcal{O}(d \log 1/\epsilon)$. However, the success probability decays exponentially and thus many rounds of amplitude amplification may be required. Following the construction in section 1.6, we note that $\gamma = e^d$ to order ϵ, so the complexity of using amplitude amplification can only be constrained to $\mathcal{O}(e^d)$.

Many authors have considered the problem of matrix exponentiation previously. Some have considered the problem of solving a system of linear ordinary differential equations [Ber14, BCO+17], while others have focused on the problem of Gibbs' state preparation [CS17]. [VGG+17] also describe an algorithm for this problem, which they use for Gibbs' sampling in quantum SDP algorithms. [PP18] give an algorithm that also uses the Chebyshev expansion of the exponential function, but their method of implementation is based on the recursion relation for Chebyshev polynomials, and uses a digital encoding of quantum states.

1.8 Discussion

We have seen in this chapter a simple calculation motivated by the approximation of real functions by Chebyshev series to illustrate the use of quantum walks with the LCU method to implement a wide variety of smooth functions. The method is particularly simple, as the approximating linear combination is obtained from a truncated Taylor series that is transformed into a Chebyshev series. While we do not present any improvements in complexity, the simplicity of the method makes it attractive from a pedagogical viewpoint.

Although the LCU method has been widely investigated over the last few years, there are still a few interesting questions related to it - for example, the only unitaries found useful so far are tensor products of Pauli operators, Fourier basis terms, and Chebyshev polynomials, because these can be implemented using known methods (Hamiltonian simulation and quantum walks). It would be interesting to study other families of unitary circuits that can be used in conjunction with this method. In particular, for a special case such as say matrix exponentiation, is it possible to systematically determine an optimal basis of unitaries?

The theory of implementing matrix functions quantumly has been effectively unified and put on a firm grounding by the work on quantum singular value transformation of Gilyén et al. Although several attempts have been made to find applications of these methods that yield provable speedups over classical algorithms, exponential speedups have been hard to come by. Identifying problem instances that are rich enough in structure to exploit quantumly, but still be provably hard classically, is a wide open field.

Another rather enigmatic aspect of all the different kinds of quantum methods of implementing matrix functions is the seeming need for the input matrix to be either locally computable and sparse, or have low rank. While we do know of simple problems, such as identity testing, that lead to lower bounds that elucidate the limitations of these techniques and throw some light on the need for sparsity and rank assumptions, we do not have a sharp lower bounds or a clear understanding. It is also natural to wonder if finding sparse or low-rank approximations to an input matrix can still be leveraged to quantumly solve problems of interest. We explore this line of thought further in Chapter 2, and show that a one time classical preprocessing step by spectral sparsification can enable algorithms to take advantage of quantum exponential speedups, such as the one offered by the linear systems algorithm, for savings in resource guzzling iterative subroutines.

Appendix A Using the LCU method to approximate $f(A)$ for a Hermitian matrix A

We briefly describe below the LCU method for approximately implementing a linear combination of unitary matrices. This description is drawn from [CKS17], and refer readers to the proofs therein for more details.

First, we need to make sure that approximating a Hermitian operator C by another operator D in spectral norm ensures that they produce states $C|\psi\rangle / \|C|\psi\rangle\|$ and $D|\psi\rangle / \|D|\psi\rangle\|$ that are close in the Hilbert space norm, for any state $|\psi\rangle$.

Lemma 1.2. *[CKS17, Proposition 9] Let C be a Hermitian operator whose eigenvalues satisfy $|\lambda| \geq 1$, and D be an operator satisfying $\|C - D\| \leq \epsilon < 1/2$. Then for any state $|\psi\rangle$,*

$$e(\psi) := \left\| \frac{C|\psi\rangle}{\|C|\psi\rangle\|} - \frac{D|\psi\rangle}{\|D|\psi\rangle\|} \right\| < 4\epsilon. \tag{1A.1}$$

Proof. Since the assertion is about normalised states, we can consider states with $\||\psi\rangle\| = 1$ without loss of generality. Repeated application of the triangle inequality and the fact that $\|C|\psi\rangle\| \geq |\lambda_{min}|$ together imply that

$$\begin{aligned} e(\psi) &\leq \left\| \frac{C|\psi\rangle}{\|C|\psi\rangle\|} - \frac{C|\psi\rangle}{\|D|\psi\rangle\|} \right\| + \left\| \frac{C|\psi\rangle}{\|D|\psi\rangle\|} - \frac{D|\psi\rangle}{\|D|\psi\rangle\|} \right\| \\ \|C|\psi\rangle\| &\leq \|(C-D)|\psi\rangle\| + \|D|\psi\rangle\| \leq \epsilon + \|D|\psi\rangle\| \\ |\|C|\psi\rangle\| - \|D|\psi\rangle\|| &\leq \epsilon \\ \|D|\psi\rangle\| &\geq \|C|\psi\rangle\| - \epsilon \geq |\lambda_{min}| - \epsilon, \end{aligned}$$

$$e(\psi) \leq \frac{2\epsilon}{|\lambda_{min}| - \epsilon}. \tag{1A.2}$$

We can replace this with a looser bound $e(\psi) < c\epsilon$ for some constant $c > 0$; then c must satisfy $\frac{2}{c} < |\lambda_{min}| - \epsilon$, i.e., $c = \Omega(\frac{1}{|\lambda_{min}|})$. This indicates that for operators with eigenvalues that approach zero in magnitude, the approximation in normalised states is worse, because we will need to settle for a larger c.

If we assume for the eigenvalues of C, as stated in the lemma, that $|\lambda_{min}| \geq 1$, then using $\epsilon < 1/2$ gives $c \geq 4$, so that $e(\psi) < 4\epsilon$. □

We can now state precisely how a matrix function will be approximated by a finite linear combination of unitaries.

Lemma 1.3. *[CKS17, Corollary 10] Let A be a Hermitian operator acting on n-qubits with eigenvalues lying in an interval $D \subset \mathbb{R}$. Suppose the function $f : D \to \mathbb{R}$ can be approximated by the linear combination of functions $g_i : D \to \mathbb{R}$ for $i = 1, \ldots, m$ such that*

$$\sup_{x \in D} \left| f(x) - \sum_{i=1}^{m} \alpha_i g_i(x) \right| \leq \epsilon < 1/2, \tag{1A.3}$$

for coefficients $\alpha_i > 0$. Further, let $\{U_i : i = 1, \ldots, m\}$ be a set of unitaries on $(n+t)$-qubits (with $t = \mathcal{O}(\log m)$) that satisfy

$$U_i \left|0^t\right\rangle \left|\psi\right\rangle = \left|0^t\right\rangle g_i(A) \left|\psi\right\rangle + \left|\Psi_i^{\perp}\right\rangle, \tag{1A.4}$$

$\forall$ n-qubit states $|\psi\rangle$, with $\left(\left|0^t\right\rangle \left\langle 0^t\right| \otimes \mathbb{1}_n\right) \left|\Psi_i^{\perp}\right\rangle = 0$. Given an oracle $\mathcal{P}_{\vec{b}}$ that prepares a state $|b\rangle$, there is a quantum algorithm that prepares with high probability a state $|\tilde{\psi}\rangle$ such that

$$\left\| |\tilde{\psi}\rangle - |\psi\rangle \right\| \leq 4\epsilon, \quad \textit{with} \quad |\psi\rangle = \frac{f(A)\,|b\rangle}{\|f(A)\,|b\rangle\|}. \tag{1A.5}$$

The algorithm uses amplitude amplification, making $\mathcal{O}(\alpha / \|f(A)\,|\psi\rangle\|)$ queries to $\mathcal{P}_b$ (where $\alpha = \sum_i |\alpha_i|$), and to the following unitary operators

$$U := \sum_{i=0}^{m} |i\rangle\langle i| \otimes U_i, \quad V\left|0^s\right\rangle = \frac{1}{\sqrt{\alpha}} \sum_{i=0}^{m} \sqrt{\alpha_i}\,|i\rangle$$
$$W := (V^{\dagger} \otimes \mathbb{1}_n) U (V \otimes \mathbb{1}_n). \tag{1A.6}$$

A proof of this lemma can be constructed by calculating the action of the unitary W on input states of the form $\left|0^s\right\rangle \left|0^t\right\rangle \left|\psi\right\rangle$, to obtain

$$W \left|0^s\right\rangle \left|0^t\right\rangle \left|\psi\right\rangle = \frac{1}{\alpha} \left|0^s\right\rangle \left|0^t\right\rangle f(A) \left|\psi\right\rangle + \left|\Psi^{\perp}\right\rangle.$$

Since the state preparation map for $|\psi\rangle$ is available, standard amplitude amplification can be applied. We refer the reader to [CKS17] for the full proofs of lemmas 1 and 2 quoted above.

Appendix B Amplitude amplification

Given a classical input vector $\vec{b} = (b_1, \ldots, b_N)^T \in \mathbb{C}^N$, we usually assume we have an oracle $\mathcal{P}_{\vec{b}}$ that prepares a quantum state $|\psi\rangle$ corresponding to this vector, defined by

$$|0^m\rangle \mapsto |\psi\rangle := \frac{\sum_i b_i\,|i\rangle}{\|\sum_i b_i\,|i\rangle\|}, \tag{1B.1}$$

where $m = \lceil \log N \rceil + 1$. When such a unitary oracle is available, standard amplitude amplification (AA) can be applied. The initial state on which we perform AA is

$$|\Psi_i\rangle := W |0^m\rangle |\psi\rangle = \sqrt{p} |0^m\rangle |\phi\rangle + \sqrt{1-p} |\Phi^\perp\rangle , \tag{1B.2}$$

where $|\phi\rangle = \frac{A|\psi\rangle}{\|A|\psi\rangle\|}$, $p = \left|\frac{\|A|\psi\rangle\|}{\alpha}\right|^2$ and $|\Phi^\perp\rangle$ is normalised. The projection onto the desired subspace is $\Pi := |0^m\rangle\langle 0^m| \otimes \mathbb{1}_n$, which picks out the 0-flag on the ancillary register. The reflection about the target subspace can then be taken as $R_t := (\mathbb{1}_{m+n} - 2\Pi)$, since it leaves the system register unchanged, while reflecting about the ancillary success flag. Then the reflection about the initial state is

$$R_i := W(\mathbb{1}_m \otimes P_{\vec{b}}) R_t (\mathbb{1}_m \otimes P_{\vec{b}})^\dagger W^\dagger,$$

and the usual grover iterate is obtained, $G = -R_i R_t$.

Often such a state preparation map will not be available. In such cases, it may still be possible to use a version called 'oblivious' amplitude amplification, if the matrix function we are attempting to implement is close to unitary. This is done using the Grover iterate like operator $S := -W R_t W^\dagger R_t$. More details can be found in [BCC+15] and [Kot14].

Appendix C Chebyshev polynomials

It is a result in approximation theory that Chebyshev polynomials form the best polynomial basis for approximating functions on $[-1, 1]$ in the supremum or L_∞ norm. That is, they minimise the error $\sup_x |\tilde{f}(x) - f(x)|$ between the approximator $\tilde{f}$ and the target f.

The Chebyshev polynomials of the first kind, $\mathcal{T}_n(x)$, satisfy $\forall x \in [-1, 1]$

1. $\mathcal{T}_0(x) = 1$, $\mathcal{T}_1(x) = x$
2. $\mathcal{T}_{n+1}(x) = 2x\mathcal{T}_n(x) - \mathcal{T}_{n-1}(x)$
3.

$$\int_{-1}^{1} \frac{\mathcal{T}_n(x)\mathcal{T}_m(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0, & m \neq n \\ \pi/2, & m = n \neq 0 \\ \pi, & m = n = 0 \end{cases} \tag{1C.1}$$

We can exactly represent the monomials x^k on $[-1, 1]$ in the basis of Chebyshev polynomials as the finite sum of terms up to degree k as follows [Tha64]

$$x^k = \sum_{j=1}^{k} C_{kj} \mathcal{T}_j(x) + \frac{1}{2} C_{k0}. \tag{1C.2}$$

The coefficients C_{ij} can be calculated by the substitution $x = \cos(\theta)$, and using the property $\mathcal{T}_n(\cos(x)) = \cos(nx) \ \ \forall x \in [0, \pi]$, as

$$\begin{aligned} C_{kj} &= \frac{2}{\pi} \int_{-1}^{1} \frac{x^k \mathcal{T}_j(x)}{\sqrt{1-x^2}} dx \\ &= \frac{2}{\pi} \int_0^{\pi} \cos^k(x) \cos(jx) dx. \end{aligned} \tag{1C.3}$$

Writing $\cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$ and $\cos(jx) = Re(e^{ijx})$, we get

$$C_{kj} = \frac{1}{2^{k-1}\pi} Re \left(\int_0^{\pi} \sum_{l=0}^{k} \binom{k}{l} e^{i(2l-(k-j))} dx \right). \tag{1C.4}$$

The real part of the exponential integrates to zero on $[0, \pi]$ unless $2l = k - j$. In this case, the integral is just π, the length of the interval, and we get

$$C_{kj} = \begin{cases} \frac{1}{2^{k-1}} \binom{k}{(k-j)/2}, & \text{if } (k-j) \text{ is even} \\ 0, & \text{otherwise.} \end{cases} \tag{1C.5}$$

This makes sense, since for k even, x^k is an even function and will contain only Chebyshev terms of even degree in its expansion (similarly when k is odd).

The Chebyshev polynomials $\mathcal{T}$ all evaluate to 1 at $x = 1$ (can be seen from $\mathcal{T}_n(\cos(x)) = \cos(nx)$), and this gives us a useful property: for all integers $k \geq 0$

$$\sum_{j=0}^{k} C_{kj} = 1. \tag{1C.6}$$

For the quantum walk construction, we also need a few properties related to the Chebyshev polynomials of the second kind. We have that $\forall x \in [-1, 1]$, $\mathcal{T}_0(x) = U_0(x) = 1$, $T_1(x) = x$, $U_1(x) = 2x$ and both $\mathcal{T}_n(x)$ and $U_n(x)$ satisfy the same recursion relation

$$f_{n+1} = 2x f_n - f_{n-1}, \tag{1C.7}$$

for all positive integers n, where f represents either $\mathcal{T}$ or U. From this, it can be seen that $\mathcal{T}_n$ and U_n satisfy the relations

$$\begin{aligned} \lambda \mathcal{T}_{n-1}(\lambda) + (\lambda^2 - 1) U_{n-2}(\lambda) &= \mathcal{T}_n(\lambda) \\ \mathcal{T}_{n-1}(\lambda) + \lambda U_{n-2}(\lambda) &= U_{n-1}(\lambda), \end{aligned} \tag{1C.8}$$

which turn out to be useful in the next section. Furthermore,

$$\begin{aligned} \mathcal{T}_n(\cos(x)) &= \cos(nx) \\ U_n(\cos(x)) &= \frac{\sin((n+1)x)}{\sin(x)}, \end{aligned} \tag{1C.9}$$

where $x = \cos^{-1}(\lambda) \in [0, \pi]$ $\forall \lambda \in [-1, 1]$.

Appendix D Implementing Chebyshev polynomials in A using a quantum walk

To actually realise an implementation based on the LCU method, we need to be able to perform the unitary matrices in the decomposition of the target matrix. One of the families of unitaries that have been found useful is a quantum walk operator $\mathcal{W}$ which has the property that when restricted to a certain invariant subspace, $\mathcal{W}^n$ has a block form with the first block being the operator $\mathcal{T}_n(H)$, where $\mathcal{T}_n$ is the Chebyshev polynomial of the first kind and degree n, and $A = dH$ is the matrix using which the walk is constructed. We describe this quantum walk operator in this section for a normalised Hermitian matrix A. This material is drawn from [CKS17] and [BCK15].

Given a d-sparse Hermitian matrix A acting on $\mathbb{C}^N$, we consider two copies of the system, each adjoined with a single ancillary qubit to form $\mathbb{C}^N \otimes \mathbb{C}^2$, and associate to A a unitary quantum walk in this expanded space, $\mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$. The goal is to construct a method that implements a Chebyshev polynomial function $\mathcal{T}_n(A)$, but the construction below will instead produce $\mathcal{T}_n(A/d)$.

We start by considering an N-dimensional hyperplane $V \subset \mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$, spanned by the states

$$|\psi_j\rangle = |j\rangle \otimes \frac{1}{\sqrt{d}} \sum_{k:A_{jk}\neq 0} \left(\sqrt{A^*_{jk}}\,|k\rangle + \sqrt{1 - |A_{jk}|}\,|k+N\rangle \right), \tag{1D.1}$$

for $j = 1, \ldots, N$. These states are orthonormal, since we can assume that for rows with fewer than d entries, the summation is taken over enough zero entries to make up the total of d terms in the linear combination. Next, define an isometry $T : \mathbb{C}^N \to \mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$ that embeds the system register into the expanded space

$$T := \sum_{j=1}^{N} |\psi_j\rangle \langle j| . \tag{1D.2}$$

The third ingredient is a swap operator on $\mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$

$$S\,|j,k\rangle := |k,j\rangle \; \forall j,k = 1, \ldots, 2N. \tag{1D.3}$$

The unitary walk operator is then defined as

$$\mathcal{W} := S\left(2TT^\dagger - \mathbb{1}\right). \tag{1D.4}$$

Now it can be shown that in a subspace $\mathcal{B} \subset \mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$, defined as the span of the isometric mappings of the basis states $|j\rangle$, and the corresponding swapped states, the walk operator has a block form such that the blocks are Chebyshev polynomials in A/d, i.e. $\mathcal{T}_n(A/d)$ and $U_n(A/d)$.

Lemma 1.4. *The $2N$-dimensional subspace $\mathcal{B} = \text{span}\{T|j\rangle, ST|j\rangle \mid j = 1, \ldots, N\} \subset \mathbb{C}^{2N} \otimes \mathbb{C}^{2N}$ is invariant under the walk operator $\mathcal{W}$. Further, $\mathcal{W}$ can be put in a block form on $\mathcal{B}$*

$$\mathcal{W}|_{\mathcal{B}} = \begin{bmatrix} H & -\sqrt{1-H^2} \\ \sqrt{1-H^2} & H \end{bmatrix}, \tag{1D.5}$$

where $H = A/d$. We henceforth drop the subscript $\mathcal{B}$, and work only in this subspace.

Lemma 1.5. *For a matrix*

$$\mathcal{W} = \begin{bmatrix} \lambda & -\sqrt{1-\lambda^2} \\ \sqrt{1-\lambda^2} & \lambda \end{bmatrix} \tag{1D.6}$$

where $|\lambda| \leq 1$, and any positive integer n,

$$\mathcal{W}^n = \begin{bmatrix} \mathcal{T}_n(\lambda) & -\sqrt{1-\lambda^2}U_{n-1}(\lambda) \\ \sqrt{1-\lambda^2}U_{n-1}(\lambda) & \mathcal{T}_n(\lambda) \end{bmatrix}, \tag{1D.7}$$

where $\mathcal{T}_n(x)$ and $U_n(x)$ are the Chebyshev polynomials of the first and second kinds respectively, having degree n, defined on $[-1, 1]$.

Since H is Hermitian, its eigenvectors $|\lambda\rangle$ span $\mathbb{C}^N$. Thus within the invariant subspace $\mathcal{B}$ of $\mathcal{W}$, for any state $|\psi\rangle \in \mathbb{C}^N$, we combine the above two lemmas to get

$$\mathcal{W}^n T|\psi\rangle \to T\mathcal{T}_n(H)|\psi\rangle + |\psi^\perp\rangle, \tag{1D.8}$$

where $|\psi^\perp\rangle$ is orthogonal to $T|j\rangle$ for each $j = 1, \ldots, N$, but is not normalised.

Since T is an isometry, we can dilate and implement it by a unitary circuit that performs the map $|0^m\rangle|\psi\rangle \mapsto T|\psi\rangle$ for any state $|\psi\rangle \in \mathbb{C}^N$, with $m = \lceil \log 2N \rceil + 1$ ancillaries. Hence applying $T^\dagger \mathcal{W}^n T$ will perform the map

$$|0^m\rangle|\psi\rangle \mapsto |0^m\rangle \mathcal{T}_n(H)|\psi\rangle + |\Phi^\perp\rangle, \tag{1D.9}$$

where $|\Phi^{\perp}\rangle$ is not normalised but $(|0^m\rangle\langle 0^m| \otimes \mathbb{1}_N)|\Phi^{\perp}\rangle = 0$. Post-selecting on measuring the first m registers to be in the '0' state, we get a probabilistic implementation of the function $\mathcal{T}_n(H)$ with $H = A/d$.

We need only $\mathcal{O}(1)$ queries to the oracle $\mathcal{P}_A$ to implement the walk operator $\mathcal{W}$ as well as the isometry T [BC12]. So taking n steps of the walk requires $\mathcal{O}(n)$ queries to $\mathcal{P}_A$.

Chapter 2

Spectral sparsification of matrix inputs as a preprocessing step for quantum algorithms

Synopsis: We study the potential utility of classical techniques of spectral sparsification of graphs as a preprocessing step for digital quantum algorithms, in particular, for Hamiltonian simulation. Our results indicate that spectral sparsification of the adjacency matrix of a graph with n nodes and $\mathcal{O}(n^2)$ edges through a sampling method, e.g. as in [SS11] using effective resistances, gives, with high probability, a locally computable matrix $\tilde{H}$ with row sparsity at most $\mathcal{O}(\text{poly}\log n)$. For a symmetric matrix H of size n with m non-zero entries, a one-time classical runtime overhead of $\mathcal{O}(m\|H\|t\log n/\epsilon)$ expended in spectral sparsification is then found to be useful as a way to obtain a row-sparse matrix $\tilde{H}$ that can be used to approximate time evolution e^{itH} under the Hamiltonian H to precision ϵ. Once such a sparsifier is obtained, it could be used with a variety of quantum algorithms in the query model that make crucial use of row sparsity. We focus on the case of efficient quantum algorithms for sparse Hamiltonian simulation, since Hamiltonian simulation is a key subroutine that underlies several quantum algorithms, including quantum phase estimation and recent ones for linear algebra. Finally, we also give two simple quantum algorithms to estimate the row sparsity of an input matrix, which achieve a query complexity of $\mathcal{O}(n^{3/2})$ as opposed to $\mathcal{O}(n^2)$ that would be required by any classical algorithm for the task.

2.1 Introduction

In classical graph theory and signal processing, sparsifying dense matrices and performing algorithms thereon to reduce the computational load is a key idea, which has received significant attention over the past several years. For instance, in the case where the matrix is a graph adjacency matrix, one of the primary motivations for spectral sparsification of the graph Laplacian in classical computer science is its utility for attacking cut problems [BSS+13, BSS14], and as a result the spectral properties of the sparsified graph's Laplacian are frequently treated as the measure of interest to retain while reducing the number of edges

in the graph. Sparsification has also found use in designing preconditioners for linear system solvers [SS11, ST11, ST14], and in studying constraint satisfaction problems [Din06].

Recently, the idea of transferring the wealth of results on the benefits of sparsity from classical signal processing to quantum algorithms, in particular with regards to Hamiltonian simulation, has been investigated in [AZ18]. Their results for analogue simulation indicate that degree reduction and edge dilution does not work in the quantum setting for general graphs. On the other hand, digital quantum simulation may also benefit from sparsification, and this idea is yet to be explored to the best of our knowledge. For Hamiltonian simulation, we are ultimately interested in approximating the evolution $U(H,t) = e^{-iHt}$ under a Hermitian matrix H. That is, given any state $|\psi\rangle$, we want to approximate the state $U(H,t)|\psi\rangle$, which can be done by approximating U itself in spectral norm. We should then study and bound

$$\sup_{|\psi\rangle\in\mathcal{H}} \left\| U(H,t)|\psi\rangle - U(H',t)|\psi\rangle \right\|, \tag{2.1}$$

where H' is a spectral sparsifier for H. This quantity is simply the spectral norm of the difference operator $e^{-iHt} - e^{-iH't}$, but the form above reminds us that we also need to keep track of the way the eigenspaces change in the process of sparsification.

In this chapter we address three issues with transferring classical sparsification results to the quantum setting directly: firstly, the classical sparsification techniques typically yield matrices that are sparse overall, but the quantum algorithm for Hamiltonian simulation requires the more restrictive condition of row-sparsity, that is quantum algorithms for Hamiltonian simulation in the query model, in which the input Hamiltonian is accessed via a unitary oracle that computes matrix entries in place, up to fixed precision, typically require the interaction graph of the Hamiltonian matrix to be row sparse and locally computable; secondly, it is necessary that the adjacency matrix, rather than Laplacian is convergent; and finally, it is necessary that any residual error in the sparsified matrix does not cause a catastrophic break-down in the accuracy of the Hamiltonian simulation.

We show how each of these problems can be overcome, although we do still make some assumptions, namely that the Hamiltonian is real, that each row of the Hamiltonian is sampled sufficiently frequently in the sparsification method, and that the sparsified Hamiltonian commutes with actual Hamiltonian. In fact, we do not believe that any of these assumptions are necessary, rather that they are simply features of our proofs and that future analysis should reveal that the results hold when these are not made. Nevertheless, even with these assumptions, the analysis in this chapter serves the purpose of better connecting classical sparsification with Hamiltonian simulation methods that assume row-sparse matrices, and shows the likely way forward to achieving a full general and unrestricted result.

A secondary purpose of this chapter concerns the verification of row-sparsity, which is also an important question in its own right, and one which ostensibly would appear to lend itself

to being sped up by a quantum algorithm. We show this is indeed the case by proposing two simple quantum algorithms that can decide whether a given matrix is row-sparse with fewer operations than are required classically. Whilst conceptualised from quite different starting points, both of these quantum algorithms require $\mathcal{O}(n^{3/2})$ operations, compared to $\mathcal{O}(n^2)$ classically (for a $n \times n$ matrix), and this coincidence of computational complexity raises two intriguing possibilities: firstly, it may be possible to combine the two algorithms in some way to achieve the sparsity verification in $\mathcal{O}(n)$ operations; or conversely, it may be that $\mathcal{O}(n^{3/2})$ is a lower-bound.

The remainder of this chapter is organised as follows: in Section 2.2 we give precise details of the problem we are going to solve, including an analysis of the overall benefits that it brings to Hamiltonian simulation; in Section 2.3 we give our main results on relating the classical sparsification algorithm to the problem of Hamiltonian simulation; in Section 2.4 we propose two quantum sparsification verification algorithms; and finally in Section 2.5 we include a wide-ranging discussion covering, amongst other things, the physical meaning of a row-sparse Hamiltonian.

Our contributions

Our work advances in four primary directions.

1. From the literature we note that sparsification can reduce the computational load in several linear algebraic problems, whilst maintaining accuracy. However, we note a discrepancy between classical sparsification algorithms, which typically achieve edge sparsification or dilution, and quantum algorithms for sparse Hamiltonian simulation, which require row sparsity (degree reduction). To bridge this divide, we provide a necessary and sufficient condition for (general) sparsity to imply row sparsity.

2. The classical sparsification method upon which we base our analysis [SS11] provides a bound in terms of the Laplacian, whereas we require one in terms of the adjacency matrix. We therefore prove that the accuracy condition proved for the Laplacian implies that the adjacency matrix is also well-approximated by the sparsified adjacency matrix, when the above row-sparsity condition is met.

3. We then show that this condition on the adjacency matrix being well-approximated *is* sufficient for the Hamiltonian simulation with the sparsified matrix to well-approximate the actual case.

4. We are also interested in the verification of sparsity, and to this end we propose two quantum algorithms that can verify whether or not a matrix is row sparse in $\mathcal{O}(n^{3/2})$ time, for an n^2 matrix, which represents an improvement on the $\mathcal{O}(n^2)$ time required classically.

2.2 Setup and problem statement

We base our exposition on the method of spectral sparsification using effective resistance sampling developed in [SS11]. Given a graph $G = (V, E, W)$ with $|V| = n$ vertices, $|E| = m \leq \binom{n}{2}$ edges, and W an $m \times m$ diagonal matrix of edge weights, spectral sparsification generates another graph $G' = (V, E', W')$ such that

$$|E'| \in \mathcal{O}\left(\frac{n \log n}{\epsilon^2}\right),$$
$$(1 - \epsilon)L_G \preceq L_{G'} \preceq (1 + \epsilon)L_G, \tag{2.2}$$

where L represents the Laplacian matrix, and the second condition holds in the usual partial order on positive semidefinite matrices. The runtime of this classical algorithm is $\mathcal{O}(\frac{m \log n}{\epsilon^2})$.

The simplest process of this kind can be described as sampling from the edge set E according to some probability mass function (pmf) $\mathcal{P}$ and hence populating E'. If for each $e \in E$ the probability of picking e is $p_e = \mathcal{P}(e)$, we can marginalise out the edges and consider the pmf induced on the vertices, defining

$$p_v = \sum_{e \in E : B(e,v) \neq 0} p_e$$

where B denotes the edge-vertex incidence matrix of the graph. In this sampling process, edges in the new graph are reweighted so as to preserve spectral properties of the Laplacian. The weight $w(e)$ of an edge e is scaled up by a factor of $1/p_e$ each time it is drawn as a sample.

In this chapter, we show that the sparsifier generated by a spectral sparsification algorithm which uses sampling methods has under certain conditions, with high probability, the additional property that it is row-sparse, i.e., that the maximum degree of the sparsifier grows only as $\mathcal{O}(\text{poly} \log n)$.

2.3 Relating sparsification to efficient Hamiltonian simulation

In this section we prove three results that allow us to make the connection between classical sparsification and efficient Hamiltonian simulation in the query complexity model: firstly, we prove a necessary and sufficient condition for sparsity to imply row sparsity; secondly, we show that the asymptotic convergence of the Laplacian proven by [SS11] is sufficient for the asymptotic convergence of the adjacency matrices that we require; and finally we show that the propagation of error from the sparsification process into the sparsified Hamiltonian in the sense described in the sparsification bounds does not lead to a significant increase in error in the Hamiltonian simulation.

Before we get into the technical details, it may be worth discussing the relation between spectral sparsification and dimensionality reduction. Intuitively, sparsification appears to be a form of dimensionality reduction, at least in terms of the edges of the underlying graph; on the other hand, this appearance is deceptive, since the number of vertices is held fixed, the dimensions of the associated matrices are unchanged although memory can be saved in storing the matrix in data structures optimised for sparsity. Furthermore, it is not clear if an interpreation of the sparsified matrix in terms of a lower dimensional object is at all possible, while faithfully preserving the relevant (spectral) properties. As a concrete example of dimensionality reduction, the Johnson-Lindenstrauss embedding is a technique that gives a randomised embedding of a set of r vectors $v_1, \ldots, v_r \in \mathbb{R}^m$ into $\tilde{v}_1, \ldots, \tilde{v}_r \in \mathbb{R}^n$ in $n = \Theta(\log m/\epsilon^2)$ dimensions, such that pairwise ℓ_2 distances are preserved up to a factor of $(1 \pm \epsilon)$. One may wonder if such techniques can also be useful in matrix problems such as Hamiltonian simulation, but again it is unclear how we might apply the technique, since a naïve approach such as embedding the column vectors of the input matrix into lower dimensional vectors would result in a rectangular matrix as output, and also fail to preserve its spectral properties.

2.3.1 A necessary and sufficient condition for row sparsity

A matrix is row-sparse if the number of non-zero entries in a row (for all rows) grows as poly-log of the size of the matrix (at most). Clearly row-sparsity implies sparsity, but the opposite does not apply in general, for example a star network has a sparse adjacency matrix, but each element of the row corresponding to the point of the star is one (except for the leading diagonal entry), and therefore is obviously not row sparse.

In the effective resistance Hamiltonian sparsification method, each edge of a adjacency graph corresponding to the Hamiltonian is chosen with a certain probability, and a defined number of i.i.d. samples are drawn (that grows at most with $n(\text{poly}\log n)$), to construct the sparsified Hamiltonian. Marginalising over the the edges incident to each node as described in Section 2.2, to get a vertex selection probability distribution[1], it is obvious that a necessary condition for row-sparsity is that no vertex is selected with probability growing faster than $(\text{poly}\log n)/n$. We now show that this necessary condition is also sufficient, in both obvious senses of asymptotic statistical row sparsity:

1. As $n \to \infty$ the probability that any row has a number of elements greater than $\mathcal{O}(\log^{b'} n)$ tends to zero, for some b'.

2. As $n \to \infty$ the expected maximum number of non-zero elements (across the n rows of the matrix) grows as $\mathcal{O}(\log^b n)$ for some b.

[1] Note that each edge is connected to two vertices, vertex selection is not mutually exclusive so this will sum to more than one – a property later used in upper-bounding.

Hereafter these are termed the first and second properties of asymptotic row sparsity.

Proposition 2.1. *For a Hamiltonian sparsified by the effective resistance method, if no row is selected with probability greater than* $\log^b(n)/n$*, then it is satisfies the first row sparsity property.*

Proof. Let the total number of samples drawn be $n \log^a n$, therefore the expected number of times that the row with maximum occupation is selected can be upper-bounded:

$$\mu < \log^b n \log^a n = \log^c n, \tag{2.3}$$

where c is defined as $a + b$.

Let x_i be the number of non-zero entries in the i^{th} row, and X be number of non-zero entries in the row with greatest selection probability. By the Chernoff bound [Che52],

$$P(X \geq (1+\delta)\mu) \leq \left(\frac{\mathrm{e}^\delta}{(1+\delta)^{1+\delta}} \right)^\mu.$$

We see that for $\delta \geq 4$, the right hand side is less than $2^{-(1+\delta))\mu}$. Thus for $R \geq 5\mu$:

$$P(X \geq R) \leq 2^{-R}, \tag{2.4}$$

letting $R = \log^{c+1} n$ (the $c+1$ term is included to ensure that the $R \geq 5\mu$ condition is met for sufficiently large n), and noticing that all the other nodes are less likely to be chosen and thus the Union Bound can be invoked:

$$\begin{aligned} P(\max(x_i) \geq R) \leq & \sum_i p(x_i \geq R) \\ \leq & n 2^{-R} \\ = & n 2^{-\log^{c+1} n}, \end{aligned} \tag{2.5}$$

where this upper-bound also uses the fact that all other rows are chosen with probability less than the row with maximum selection probability. Let $p = n 2^{-\log^{c+1} n}$:

$$\begin{aligned} \log_2 p &= \log_2 n - \log_2^{c+1} n \\ &= \log_2 n (1 - \log^c n) \\ &< 0 \end{aligned} \tag{2.6}$$

for sufficiently large n, i.e., as $n \to \infty$, $p(X \geq R) \to 0$. □

Proposition 2.2. *For a Hamiltonian sparsified by the effective resistance method, if no row is selected with probability greater than* $\log^b(n)/n$*, then it is satisfies the second row sparsity property.*

Proof. The proof is similar to that of Proposition 2.1, and the same symbols are used. The proposition considers the expectation of the number of non-zero entries in the row with the most non-zero entries (note that this need not be the the row with highest selection probability). This can be upper-bounded:

$$\begin{aligned}\mathbb{E}(X) \leq & R \times p(X < R) + (n \log^a n) \times p(X \geq R) \\ \leq & 1 \times \log^c n + n^2 \log^a n \cdot 2^{-\log^{c+1} n},\end{aligned} \tag{2.7}$$

i.e., where the second term in the first line uses the fact that, if the maximally chosen row is chosen more than R times, it can still only be chosen a total of $n \log^a n$ times, as this is the number of samples. The first term in the RHS of eq. (2.7) is clearly $\in \mathcal{O}(\text{poly} \log n)$, letting y equal the second term in the RHS of eq. (2.7), i.e.

$$\begin{aligned}y = & n^2 \log^a n \cdot 2^{-\log^{c+1} n} \\ \implies \log y = & 2 \log n + a \log \log n - \log^{c+1} n \\ = & \log n \cdot (2 - \log^{c+1} n) + a \log \log n,\end{aligned} \tag{2.8}$$

clearly, as $n \to \infty$, $\log y \to -\infty$, therefore $y \in o(1)$. So it follows that $\mathbb{E}(X) \in \mathcal{O}(\log^c n)$. □

2.3.2 From sparsified Laplacian to sparsified adjacency matrix

The sparsification is for the Laplacian matrix, while typically we interpret an input matrix for a quantum algorithm as representing the adjacency matrix of some graph (interaction graph of a Hamiltonian, for instance). We may need to translate the spectral sparsification results from Laplacian back to Adjacency matrices.

From [SS11], we have that a weighted graph with Laplacian L is sparsified to a graph with Laplacian $\tilde{L}$, such that the following condition is met:

$$(1 - \epsilon)\mathbf{x}^T L \mathbf{x} \;\leq\; \mathbf{x}^T \tilde{L} \mathbf{x} \;\leq\; (1 + \epsilon)\mathbf{x}^T L \mathbf{x}, \tag{2.9}$$

for all vectors x and $1/\sqrt{n} < \epsilon \leq 1$.

We wish to express a similar condition for the adjacency matrix, $A = D - L$, where D is a diagonal matrix where each element is the sum of the elements in that row of the adjacency matrix. Our method to do this relies on the property $\mathbb{E}(D_{ii}) = \mathbb{E}(\tilde{D}_{ii})$ for all i – that is that the diagonal elements of the sparsified Laplacian are expected to be the same as those of the

actual Laplacian (this can be seen in the analysis in [SS11]). Additionally, we must assume that each row is expected to be selected at least $\log^\alpha n$ times, where $\alpha > 1$.

To an extent it is valid to criticise such a condition as unnecessarily restrictive, however we do expect the condition in Theorem 2.3 (or another very similar one) to hold even if this were not to be the case, albeit requiring a very different proof. The inclusion of this condition is therefore for reasons of exposition – it enables us to use a similar proof to the others in this section, and suffices to demonstrate the principle. In Section 2.5 we briefly discuss the physicality of such a restriction. With these restrictions in place, we can state the following theorem.

Theorem 2.3.

$$\mathbf{x}^T A\mathbf{x} - \epsilon'\mathbf{x}^T\mathbf{x}\max_i(D_{ii}) \;\leq\; \mathbf{x}^T\tilde{A}\mathbf{x} \;\leq\; \mathbf{x}^T A\mathbf{x} + \epsilon'\mathbf{x}^T\mathbf{x}\max_i(D_{ii}), \tag{2.10}$$

for some $\epsilon' \in \omega(1/\sqrt{\log^{\alpha-1} n})$.

Proof. We start by substituting $A = D - L$ and $\tilde{A} = \tilde{D} - \tilde{L}$ into eq. (2.9), and rearranging:

$$\begin{aligned}
(1-\epsilon)\mathbf{x}^T L\mathbf{x} \leq \quad & \mathbf{x}^T\tilde{L}\mathbf{x} \quad && \leq (1+\epsilon)\mathbf{x}^T L\mathbf{x} \\
(1-\epsilon)\mathbf{x}^T(D-A)\mathbf{x} \leq \quad & \mathbf{x}^T(\tilde{D}-\tilde{A})\mathbf{x} \quad && \leq (1+\epsilon)\mathbf{x}^T(D-A)\mathbf{x} \\
(1-\epsilon)\mathbf{x}^T A\mathbf{x} - (1-\epsilon)\mathbf{x}^T D\mathbf{x} \geq \quad & \mathbf{x}^T\tilde{A}\mathbf{x} - \mathbf{x}^T\tilde{D}\mathbf{x} \quad && \geq (1+\epsilon)\mathbf{x}^T A\mathbf{x} - (1+\epsilon)\mathbf{x}^T D\mathbf{x} \\
\mathbf{x}^T A\mathbf{x} - (1-\epsilon)\mathbf{x}^T D\mathbf{x} \geq \quad & \mathbf{x}^T\tilde{A}\mathbf{x} - \mathbf{x}^T\tilde{D}\mathbf{x} \quad && \geq \mathbf{x}^T A\mathbf{x} - (1+\epsilon)\mathbf{x}^T D\mathbf{x} \\
\mathbf{x}^T A\mathbf{x} + \epsilon\mathbf{x}^T D\mathbf{x} + \mathbf{x}^T(\tilde{D}-D)\mathbf{x} \geq \quad & \mathbf{x}^T\tilde{A}\mathbf{x} \quad && \geq \mathbf{x}^T A\mathbf{x} - \epsilon\mathbf{x}^T D\mathbf{x} - \mathbf{x}^T(D-\tilde{D})\mathbf{x} \\
\mathbf{x}^T A\mathbf{x} + \mathbf{x}^T\mathbf{x}(\epsilon+\tilde{\epsilon})\max_i(D_{ii}) \geq \quad & \mathbf{x}^T\tilde{A}\mathbf{x} \quad && \geq \mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{x}(\epsilon+\tilde{\epsilon})\min_i(D_{ii}),
\end{aligned}$$

where we define $\forall i,\ (1+\tilde{\epsilon})D_{ii} \leq \tilde{D}_{ii} \leq (1+\tilde{\epsilon})D_{ii}$, which we address using the Chernoff bound. Addressing the upper limit of $\tilde{D}_{ii}$, we have that:

$$\begin{aligned}
\forall i,\ P\left(\tilde{D}_{ii} \geq (1+\tilde{\epsilon})D_{ii}\right) &\leq \exp\left(\frac{-\tilde{\epsilon}^2 D_{ii}}{3}\right) \\
&\leq \exp\left(\frac{-\tilde{\epsilon}^2 \min_i(D_{ii})}{3}\right) \\
&= \exp\left(\frac{-\tilde{\epsilon}^2 \log^\alpha n}{3}\right),
\end{aligned} \tag{2.11}$$

using the Union bound, we have that

$$P\left(\max_i(\tilde{D}_{ii} - D_{ii}) > \tilde{\epsilon}D_{ii}\right) \leq n\exp\left(\frac{-\tilde{\epsilon}^2\log^\alpha n}{3}\right). \tag{2.12}$$

Likewise for the lower-bound on $\tilde{D}_{ii}$ we have that:

$$\begin{aligned}\forall i,\, P\left(\tilde{D}_{ii} \leq (1-\tilde{\epsilon})D_{ii}\right) &\leq \exp\left(\frac{-\tilde{\epsilon}^2 D_{ii}}{2}\right) \\ &\leq \exp\left(\frac{-\tilde{\epsilon}^2 \min_i(D_{ii})}{2}\right) \\ &= \exp\left(\frac{-\tilde{\epsilon}^2 \log^\alpha n}{2}\right), \end{aligned} \tag{2.13}$$

and again using the Union bound:

$$P\left(\max_i (D_{ii} - \tilde{D}_{ii}) > \tilde{\epsilon} D_{ii}\right) \leq n \exp\left(\frac{-\tilde{\epsilon}^2 \log^\alpha n}{2}\right). \tag{2.14}$$

So we can see that the condition $\tilde{\epsilon} \in \omega(1/\sqrt{\log^{\alpha-1} n})$ must hold in order for the closeness to hold asymptotically – i.e., as $n \to \infty$ the probability that we have a 'good' sparsifier tends to one. This dominates the condition $1/\sqrt{n} < \epsilon \leq 1$, and so we can set $\epsilon' = \epsilon + \tilde{\epsilon}$ to complete the proof. □

2.3.3 Error propagation: from spectral sparsification to Hamiltonian simulation

It remains to be shown that the fact that the adjacency matrix is well approximated by its sparsified version implies that the Hamiltonian simulation will also be well approximated when the sparsified version is used. To do so, we express eq. (2.10) in slightly different (but equivalent) terms, namely that we are given a spectral approximation $\tilde{H}$ of a Hamiltonian, H satisfying

$$(1-\epsilon')H \preceq \tilde{H} \preceq (1+\epsilon')H. \tag{2.15}$$

Consider the following

$$\begin{aligned}\left\|\left(\tilde{U} - U\right)|\psi\rangle\right\| &:= \langle\psi|\left(\tilde{U} - U\right)^\dagger \left(\tilde{U} - U\right)|\psi\rangle \\ &= \langle\psi|\mathbb{1} + \mathbb{1} - \tilde{U}^\dagger U - U^\dagger \tilde{U}|\psi\rangle\,, \end{aligned} \tag{2.16}$$

where $\tilde{U} := U(\tilde{H}, t)$ and $U := U(H, t)$ for convenience. Let us first take the simple case when H and $\tilde{H}$ commute; then $\tilde{U}^\dagger U = e^{-it(H-\tilde{H})}$, and so $\tilde{U}^\dagger U + U^\dagger \tilde{U} = 2\cos(t\Delta H)$ where $\Delta H := H - \tilde{H}$. Now we can write

$$\begin{aligned}\left\|\left(\tilde{U} - U\right)|\psi\rangle\right\| &:= 2\langle\psi|\left(\mathbb{1} - \cos(t\Delta H)\right)|\psi\rangle \\ &= t^2\epsilon'^2\langle(\Delta H)^2\rangle_\psi + \mathcal{O}\left((\epsilon' t\,\|H\|)^4\right) \\ &\leq \epsilon'^2 \cdot (t\,\|H\|)^2 + \mathcal{O}\left((\epsilon' t\,\|H\|)^4\right), \end{aligned} \tag{2.17}$$

where eq. (2.15) gives $-\epsilon' H \preceq \Delta H \preceq \epsilon' H$, which also implies $0 \leq \|(\Delta H)^2\| \leq \epsilon'^2 \|H^2\| = \epsilon'^2 \|H\|^2$. In the last line we bound $\langle H \rangle_\psi$, the expectation value of the Hamiltonian H under the state ψ, by the maximum value the energy can take, given by the spectral norm of H.

The commutator between the input and the sparsifier

When H and $\tilde{H}$ do not commute, the first order Suzuki-Trotter formula suggests that $U^\dagger U = e^{-it(H-\tilde{H})}$ up to an error of order $\mathcal{O}(t^2\|H\|)$. The error term in the sum $\tilde{U}^\dagger U + U^\dagger \tilde{U}$ is then third order in $t\|H\|$. Thus the above analysis still holds for small times $t = \mathcal{O}(\epsilon'/\sqrt{\|H\|})$. For longer evolution times, higher order product formulas along with time slicing and a more detailed error analysis would be required, with the essential conceptual ideas remaining the same.

Plugging $\tilde{H}$ into a Hamiltonian simulation algorithm (e.g. [LC17, GSL+19]) results in a circuit that will approximate evolution under $\tilde{H}$ to some desired precision ϵ through a unitary circuit $\tilde{U}'$. Noting that $\left\|U - \tilde{U}'\right\| \leq \left\|U - \tilde{U}\right\| + \left\|\tilde{U} - \tilde{U}'\right\|$, we see that $\tilde{H}$ can be used to approximate Hamiltonian evolution under H.

But do H and $\tilde{H}$ always commute? The approximation condition in (2.9) appears to be stronger than the weaker consequence (2.15) leads us to believe. If we could show that for any two vectors $\mathbf{v}, \mathbf{w}$ we have

$$(1-\epsilon)\langle \mathbf{v}|H|\mathbf{w}\rangle \leq \langle \mathbf{v}|\tilde{H}|\mathbf{w}\rangle \leq (1+\epsilon)\langle \mathbf{v}|H|\mathbf{w}\rangle,$$

then, whenever $\langle \mathbf{v}|H|\mathbf{w}\rangle = 0$ the corresponding $\langle \mathbf{v}|\tilde{H}|\mathbf{w}\rangle$ is also necessarily zero. In particular, if we denote by $\{|\mathbf{v}_i\rangle\}_i$ the eigenbasis of H, i.e. $H|\mathbf{v}_i\rangle = \lambda_i |\mathbf{v}_i\rangle$, then H is diagonal in this basis, and by the above observation, so is $\tilde{H}$. This would imply that H and $\tilde{H}$ always commute, for any H. However, standard spectral sparsification does not guarantee this condition we've asked for above, and it might be worth some additional research to investigate if sparsifiers that satisfy this condition can be constructed.

Runtime overhead:

From the above analysis of error propagation, it is clear that choosing

$$\epsilon' = \mathcal{O}\left(\frac{\sqrt{\epsilon}}{t\,\|H\|}\right) \tag{2.18}$$

ensures that $\left\|U - \tilde{U}\right\| \leq \epsilon$ to first order. Putting this back into the runtime expression for the spectral sparsification algorithm of [SS11], we can estimate the (one-time) classical runtime

overhead required in order to use sparse Hamiltonian simulation for time t, which is given by

$$\mathcal{O}\left(\frac{mt\,\|H\|\log n}{\epsilon}\right). \tag{2.19}$$

The presence of the spectral norm $\|H\|$ is to be expected as it sets the energy scales for the problem; evidently this method is only useful if $\|H\| = \mathcal{O}(\text{poly}(n))$. We expect this to be true for several systems of physical significance, e.g. molecular Hamiltonians, which typically have $\mathcal{O}(n^4)$ terms in a tensor product of Pauli basis (expecting the coupling constants also to scale polynomially for most common molecules).

2.4 Sparsity testing

Testing if an input function or vector is sparse is a problem that has recently received some attention in the context of big data and machine learning algorithms [BBG18, GOS+11]. Of course, we'd like to be able to check if the sparsifier generated by a spectral sparsification algorithm has, with high probability, the additional property that it is row-sparse, i.e., that the maximum degree of the sparsifier grows only as $\mathcal{O}(\text{poly}\log n)$.

Could a quantum algorithm for sparsity testing offer any advantages? Given an oracle that computes matrix entries in place, we could use a comparison oracle to flag an ancillary register as '1' wherever there is a zero entry, and then use quantum amplitude estimation on the ancilla to estimate the number of zero entries. We demonstrate two quantum algorithms below for testing row-sparsity of an input matrix.

2.4.1 Sparsity testing using quantum amplitude estimation

As we saw in Chapter 1 we access a matrix $A \in \mathbb{R}^{n\times n}$ via a unitary quantum oracle that computes its entries in place (to some fixed precision), i.e.

$$U_A\,|i\rangle\,|j\rangle\,|z\rangle = |i\rangle\,|j\rangle\,|z \oplus A_{ij}\rangle\,, \tag{2.20}$$

where $0 \le i, j \le n-1$ are the row and column indices, and the third register contains the matrix entry to p-bits of precision (so $0 \le z \le 2^p$). Unlike Section 1.2, we do not need the additional sparse access or row-computability oracle here. We assume real entries for simplicity, and to be consistent with the previous analysis, but the following easily generalises to complex entries.

Another oracle that we will use is the comparator

$$\textsc{comp}\,|a\rangle\,|b\rangle\,|0\rangle_{\text{flag}} = \begin{cases} |a\rangle\,|b\rangle\,|0\rangle_{\text{flag}} & \text{if } a < b \\ |a\rangle\,|b\rangle\,|1\rangle_{\text{flag}} & \text{if } a \ge b, \end{cases} \tag{2.21}$$

which can be implemented efficiently using quantum adder circuits [Gid18], and has recently been used for efficient black-box preparation of quantum states encoding amplitudes corresponding to classical input vectors [SLS+19].

Let us use the U_A oracle to prepare a superposition over the entries of a chosen row $0 \leq i \leq n-1$

$$\begin{aligned} |i\rangle |0\rangle |0\rangle_{\text{data}} &\xrightarrow{\mathbb{1}\otimes H\otimes\mathbb{1}} \frac{1}{\sqrt{n}} \sum_{j=0}^{n} |i\rangle |j\rangle |0\rangle_{\text{data}} \\ &\xrightarrow{U_A} \frac{1}{\sqrt{n}} \sum_{j=0}^{n} |i\rangle |j\rangle |A_{ij}\rangle_{\text{data}} \\ &=: |\text{row}_i\rangle \end{aligned} \tag{2.22}$$

Then we can adjoin two ancillary registers, $|\delta\rangle_{\text{ref}} |0\rangle_{\text{flag}}$, and using the $\texttt{comp}$ oracle we can make the following series of transformations

$$\begin{aligned} |\text{row}_i\rangle |\delta\rangle_{\text{ref}} |0\rangle_{\text{flag}} \xrightarrow{\mathbb{1}\otimes\mathbb{1}\otimes\ \texttt{comp}} |i\rangle \otimes \Bigg(& \frac{1}{\sqrt{n}} \sum_{j\in \text{supp}^{\delta}{}_i(A)} |j\rangle |A_{ij}\rangle_{\text{data}} |\delta\rangle_{\text{ref}} |0\rangle_{\text{flag}} \\ & + \frac{1}{\sqrt{n}} \sum_{j\notin \text{supp}^{\delta}{}_i(A)} |j\rangle |A_{ij}\rangle_{\text{data}} |\delta\rangle_{\text{ref}} |1\rangle_{\text{flag}} \Bigg), \\ =: |\text{spar}_i\rangle \end{aligned}$$

where the support of row i of A is $\text{supp}_i^{\delta}(A) = \{0 \leq j \leq n-1 : |A_{ij}| \geq \delta\}$. Here we have assumed that the data register contains the magnitude of A_{ij}, so that we can just check if it is less than a small threshold δ in order to check if it is close to zero — the magnitude can be obtained easily by taking advantage of the signed fixed-point representation of A_{ij} (e.g. by simply neglecting the sign bit). Now note that the amplitude of the $|1\rangle_{\text{flag}}$ subspace of the above state is proportional to the sparsity $s(i) := \left|\text{supp}_i^{\delta}(A)\right|$ of row i

$$\begin{aligned} \left\| \Pi_1^{\text{flag}} |\text{spar}_i\rangle \right\| &= \left\| \frac{1}{\sqrt{n}} \sum_{j\notin \text{supp}^{\delta}{}_i(A)} |j\rangle |A_{ij}\rangle_{\text{data}} |\delta\rangle_{\text{ref}} \right\| \\ &= \sqrt{\frac{s(i)}{n}}, \end{aligned} \tag{2.23}$$

where $\Pi_1^{\text{flag}} = \mathbb{1}_r \otimes \mathbb{1}_c \otimes \mathbb{1}_{\text{data}} \otimes \mathbb{1}_{\text{ref}} \otimes |1\rangle\langle 1|_{\text{flag}}$ is a projector onto the flag 1 subspace. This can be estimated to additive precision ϵ' using $\Theta(1/\epsilon)$ queries to U_A, using the method of quantum amplitude estimation [BHM+02], which would give us a quantity $\tilde{x}_i$ satisfying

$$\left| \sqrt{\frac{s(i)}{n}} - \tilde{x}_i \right| \leq \epsilon',$$

whence we see that choosing $\epsilon' = \epsilon/\sqrt{n}$ gives us an additive approximation of $\sqrt{n}\tilde{x}_i$ of $s(i)$ to precision ϵ, using $\mathcal{O}(\sqrt{n}/\epsilon)$ queries. Following the same procedure to estimate the sparsities of all n rows, the overall sparsity (which for us is the maximum number of non zeros in any row or column) can be ascertained in $\mathcal{O}(n^{3/2})$ queries; we can leave ϵ out of this consideration since it can be chosen to be of order unity.

2.4.2 Sparsity testing using quantum maximum finding

We still use oracle access as in eq. (2.20), and we assume the data register has enough qubits to store the sum of the entries in any row. We start by putting the rows in superposition:

$$|0\rangle\,|0\rangle\,|0\rangle_{\text{data}} \xrightarrow{H\otimes\mathbb{1}\otimes\mathbb{1}} \frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i\rangle\,|0\rangle\,|0\rangle_{\text{data}} \tag{2.24}$$

Now we iterate over n calls to the oracle (j is initially 0):

$$\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle\,|j\rangle\,|\sum_{k=0}^{j-1}A_{ik}\rangle_{\text{data}} \xrightarrow{U_A} \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle\,|j\rangle\,|\sum_{k=0}^{j}A_{ik}\rangle_{\text{data}}\,, \tag{2.25}$$

setting $j \leftarrow j+1$ on each iteration, until $j = n-1$, after which we have the state:

$$\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle\,|n-1\rangle\,|\sum_{k=0}^{n-1}A_{i,k}\rangle_{\text{data}}\,, \tag{2.26}$$

in which the column register, now in state $|n\rangle$, can be dispensed with. Therefore in $\mathcal{O}(n)$ operations we have created a superposition of the sum of each of the n rows, indexed accordingly. Quantum maximum finding methods [DH96, AK99] can make use of this state, preparing it $\mathcal{O}(\sqrt{n})$ times in order to find the maximum. Thus we have a quantum algorithm that takes $\mathcal{O}(n^{3/2})$ oracle queries and additional quantum arithmetic operations. By contrast, a classical algorithm to check for row sparsity would have to sum over all rows ($\mathcal{O}(n^2)$ operations) and then classically find the maximum ($\mathcal{O}(n)$ operations). (note that it may be possible to do this slightly faster, but it would still be necessary to check over a number of elements growing linearly with n for each row, and to check all of the n rows).

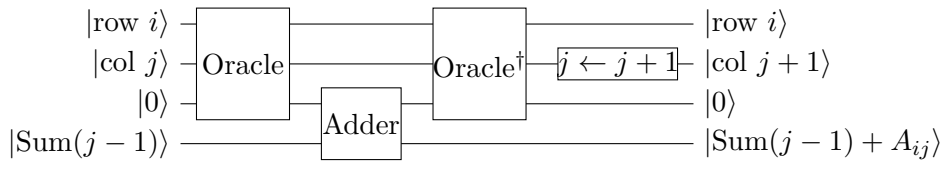


Figure 2.1: Circuit portion showing how the sum of A_{ij} can be obtained inside the ket. The oracle loads the matrix entries A_{ij} into the third register, and $\text{Sum}(j) = \sum_{k=0}^{j} A_{ij}$.

We remark that the above algorithm that uses quantum maximum finding appears to rely on A being a binary adjacency matrix. When an upper bound Λ on $\|A\|_{max}$ is available, this limitation can be overcome by normalising the matrix entries by Λ.

2.4.3 Testing Fourier sparsity

Given a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we can think of its truth table as an $n \times n$ binary matrix. A quantum query oracle for f,

$$\mathbf{U}_f |x\rangle |y\rangle |b\rangle = |x\rangle |y\rangle |b \oplus f(x,y)\rangle , \tag{2.27}$$

where $b \in \{0,1\}$ can be converted into a phase oracle by applying a Hadamard gate on the third register before and after applying $\mathbf{U}_f$

$$\mathbf{U}_f^{\text{ph}} |x\rangle |y\rangle |b\rangle = (-1)^{bf(x,y)} |x\rangle |y\rangle |b\rangle . \tag{2.28}$$

It is then possible to apply the quantum Fourier transform to the state obtained by querying the phase oracle on the uniform superposition over all the inputs x, y

$$\mathbf{U}_f^{\text{ph}} \sum_{x,y\in\{0,1\}^n} |x\rangle |y\rangle |-\rangle = \sum_{x,y\in\{0,1\}^n} (-1)^{f(x,y)} |x\rangle |y\rangle |-\rangle \tag{2.29}$$

and obtain the state encoding the Fourier amplitudes of the function f, defined by

$$\hat{f}(y) = \sum_{x\in\{0,1\}^n} (-1)^{x\cdot y} f(x). \tag{2.30}$$

We may then apply phase estimation to this state to recover a state that contains the Fourier amplitudes in the basis states, and thereby apply our sparsity testing algorithm from the previous section. This line of reasoning shows that we can test Fourier sparsity in $\mathcal{O}(n^{1.5})$ queries to f. This would be interesting to pursue further and compare with results on *learning* k-Fourier sparse functions, which it is known can be done from $\widetilde{\mathcal{O}}(k^{1.5})$ samples [ACL+19].

2.5 Discussion

In this chapter we have shown how classical sparsifying techniques can be used as a preprocessing step to obtain a row computable sparse input matrix that can then be used with efficient quantum algorithms for sparse Hamiltonian simulation. The one-time $\mathcal{O}(\frac{m \log n}{\epsilon^2})$ classical overhead in runtime may be justified by the fact that the sparsified output matrix may be used for multiple applications (e.g. for simulating time evolution of several different states) which can be efficiently performed quantumly.

Usually we require every problem instance to be row-sparse in a quantum algorithm. What we have so far, using spectral sparsification, is a guarantee that as n grows, the sparsified output is also row sparse with high probability. Therefore it is also necessary that the simulation has some sort of checking mechanism, such that the simulation is halted if too many iterations have been required (i.e., because the actual sparsified Hamiltonian that was generated was not, in fact, row sparse), and to start again with a fresh sparsified Hamiltonian. This should be easy to include in any implementation, and the first and second properties of row sparsity are sufficient in this case to guarantee good overall performance (that is, as $n \to \infty$ the probability of needing to start again vanishes).

On a more general note, it is interesting to consider Hamiltonian sparsification in the context of a suite of simulation algorithms. For example, we had remarked that star graphs are sparse, but not row sparse – thus we can see that physical systems that are dominated by a few components may yield sparsified Hamiltonian's that are essentially a superposition of a number of star graphs. Thus, whilst the techniques presented in this chapter will not apply, it may be possible to use other techniques such as low-rank approximations. Conversely, for physical systems in which a number of components barely make any impact on the whole (i.e., they have few and/or low-weight edges to other vertices), then it is likely to be safe to simply neglect these components. Informally, this can be seen as a justification for assuming that each row was sampled at least poly log many times, as in Section 2.3.2.

Open problems

As identified in the introduction, we make three assumptions in the analysis: that the Hamiltonian is real, that its rows are all sampled at least poly log many times, and that it commutes with its sparsifier. The first two of these essentially restrict the physics applications of our work, and it would therefore be beneficial to show that the same results hold when these assumptions are removed, as well as tightening the various bounds where possible.

The third assumption, however, is more fundamental – it is important to understand whether a sparsified Hamiltonian commutes with the actual Hamiltonian in general, and if not whether the discrepancy can be shown to be insignificant when the full simulation is analysed (for example by using Trotter formulas / the BCH expansion to quantify errors). However, such a question seems to be of more general relevance than simply to plug a gap in our analysis: the condition given in eq. (2.15) seems to be an eminently reasonable general measure of approximation accuracy, which may be used for myriad Hamiltonian approximation methods, and it is therefore important to show that it does indeed lead to accurate Hamiltonian simulation.

For ϵ-approximate (in l_2-norm) Hamiltonian simulation under a Hamiltonian H, i.e., for the task of approximating $e^{iHt}|\psi\rangle$, can we obtain a lower bound on the (query) complexity (in the oracle access model) that depends only on the (*overall*) sparsity or number of non-zero

entries in the matrix of H, rather than the row-sparsity? Can we demonstrate an algorithm that achieves this lower bound?

There has recently been a lot of interest in quantum algorithms for graph and distributional property testing [MdW16, GL19]. Loosely speaking, property testing relaxes a decision problem by allowing a margin of error in the answer. For example, instead of asking if an input matrix has $\mathcal{O}(\log n)$ entries in each row or not, we may ask if it has this property or is at least ϵ-far from being this sparse: this distance is defined in various ways in the theory of property testing, but a popular variant is to say a matrix is ϵ-far from having this property if changing at most $n\epsilon$ entries does not lead to a matrix that satisfies the property. This relaxation of the sparsity testing problem we saw in Section 2.4, as well as the Fourier sparsity property testing version of it, also offer intriguing avenues for further work, and may also be worth considering in different input models adapted to dense graphs (adjacency matrix) and sparser graphs (adjacency list).

In the next chapter, we pick up on a certain flavour of questions in property testing, and study an application of the quantum implementation of matrix functions — to the problem of estimating the entropy, an important property of both probability distributions and their richer cousins, quantum states.

Chapter 3

Estimating α-Renyi entropies of unknown quantum states

Synopsis: We describe a quantum algorithm to estimate the α-Renyi entropy of an unknown density matrix $\rho \in \mathbb{C}^{d\times d}$ for $\alpha \neq 1$ by combining the recent technique of quantum singular value transformations with the method of estimating normalised traces in the one clean qubit model. We consider an oracular input model where the input state is prepared via a quantum oracle that outputs a purified version of the state, assumed to be non-singular. Our method outputs an estimate of the α-Renyi entropy to additive precision ϵ, using an expected total number $\mathcal{O}(1/(x\epsilon)^2)$ of independent applications of a quantum circuit which coherently queries the input unitary $\mathcal{O}(1/\delta \log d/\epsilon)$ times, in each case measuring a single output qubit. Here δ is a lower cutoff on the smallest eigenvalue of ρ and $x = \frac{1}{d}\operatorname{Tr}(\rho^\alpha)$. The expected number of measurements made in this method can be compared to results in the sample complexity model that generally require $\Theta(d^2/\epsilon^2)$ samples. Furthermore, we also show that multiplicative approximations can be obtained by iteratively using additive approximations, with an overhead logarithmic in the dimension d.

3.1 Introduction

In this chapter we study the one parameter family of α-Renyi entropies [Rén61, MDS+13]: for $\alpha > 0$ and $\alpha \neq 1$, the α-Renyi entropy of a quantum state represented by a positive semidefinite operator $\rho \in \mathbb{C}^{d\times d}$, called its density matrix, is defined by

$$S_\alpha(\rho) := \frac{1}{1-\alpha} \log\left[\operatorname{Tr}(\rho^\alpha)\right]. \tag{3.1}$$

Taking the limit $\alpha \to 1$ gives the familiar von Neumann entropy, $S(\rho) := -\operatorname{Tr}(\rho \log \rho)$. Classical (discrete) probability distributions can be subsumed into this notation by considering a probability mass function $p = (p_1, \ldots, p_d)$ to be a density matrix that is diagonal in the

computational basis, as $\rho_p = \text{diag}(p_1, \ldots, p_d)$. $S(\rho)$ reduces to the Shannon entropy when ρ is such a 'classical state'.

The notion of entropy has played a key role in the development of a variety of scientific disciplines, ranging from thermodynamics to information theory. It gives us a way to quantify the idea of disorder in a system, and the famous second law of thermodynamics essentially states that the entropy of a closed system can never decrease. A variety of entropic functionals have operational meanings in information theory, and are closely related to the rates at which input data can be transmitted over communication channels.

Since the Renyi entropy is a generalised entropic measure and includes the von Neumann entropy as a special case, it is a problem of interest to estimate the Renyi entropy of unknown states or classical probability distributions, for different values of α. Such estimates are found to be useful, for instance, in quantifying the efficacy of an ergodic source as a random number generator [Kim18], and in the analysis of network structure, clustering, and signal processing of streams of high-frequency data [CC13]. Furthermore, as shown in [ZLO+07], the Shannon entropy can be estimated to any desired precision by interpolation using estimates of $S_\alpha(\rho)$ for values of $\alpha \in (0, 2]$.

Entropy functions are also important quantities characterising a quantum system. In entanglement theory, they can give a measure of the amount of entanglement contained in bipartite quantum systems – particularly important in this regard is the Renyi entropy for $\alpha = 2$, which is known as the entanglement entropy [CCD09]. Entropic quantities are also often used as operational measures in quantum information-processing tasks [KRS09]. As one of the most famous examples, they provide the asymptotic lower bound for compressing quantum data in a noiseless fashion, i.e. Schumacher's noiseless compression [Sch95].

In recent decades, entropy functions have also found intriguing applications in condensed matter physics [Laf16], and high energy physics. They have even had immense theoretical implications in the theory of gravity and black holes [Bek73, Don16], and their study from a quantum information theoretic viewpoint continues to be a rich source of new physical insights [AS18, AK20].

Thus it stands an important question to compute the value of these entropy functions efficiently on unknown states. In particular, given access to several copies of a quantum state, how many measurements are required to obtain estimates of a chosen entropy function of the state, to within a desired additive or multiplicative precision? Additionally, if one has access to the dynamic process that prepares the state, in the form of a unitary circuit on a larger system (the purification), does this lead to any improvement in our ability to estimate its entropies?

Related work

We can group studies of entropy estimation into four categories: (1) classical and (2) quantum algorithms for estimating entropies of classical distributions; and (3) classical and (4) quantum algorithms for estimating the entropies of quantum states. There are several studies of the first kind in classical information theory [BDK+02, WY16, JVH+15, VV11].

Coming to the third category, Hastings et al. [HGK+10], for instance, discuss a quantum Monte Carlo method to measure the 2-Renyi entropy of a many-body system by evaluating the expectation value of a unitary swap operator. Their method uses a number of samples that scales polynomially number in the system size.

More in the flavour of quantum algorithms, and in a sense straddling categories (2) and (4), Acharya et al. [AIS+17] study the sample complexity of estimating von Neumann and Renyi entropies of mixed states of quantum systems, in a model where as input one gets n independent copies of an unknown d-dimensional density matrix ρ. They allow arbitrary quantum measurements and classical post-processing, and show that in general the number of quantum samples required scales as $\Theta(d^2/\epsilon^2)$, which is asymptotically the same as the number of samples that would be required to learn the state completely via tomography methods. The experimental measurement of the entropy of specific quantum systems has also recently been investigated [IMP+15].

While it enables a tight characterisation of the sample complexity of the problem (table 3.1), other potentially stronger input models are also possible which are not captured in this picture. In this chapter, we consider an oracular input model that is popular in quantum query algorithms, wherein data is accessed in the form of a quantum state. This state may be the output of some other quantum subroutine, in which case that subroutine itself is the oracle. Such input models can capture the fact that we have access to the process generating the unknown state, which we may *a priori* expect to be useful in reducing the effort required in estimating its properties.

In this vein, and bringing us to quantum algorithms for estimating the entropies of quantum states (which as noted before subsumes the case of classical probability distributions), Li and Wu [LW19] study how to obtain additive approximations to von Neumann and Renyi entropies in an oracular model and present upper and lower bounds on the query complexity. Gilyén and Li [GL19] study another similar oracular model, known as the 'quantum purified query access' model which essentially provides a pure state, sampling from which reproduces the statistics of the original mixed state, or target classical distribution. They obtain the best upper bounds known in the literature, showing that the von Neumann entropy can be estimated with query complexity $\tilde{\mathcal{O}}(\sqrt{d}/\epsilon^{1.5})$ and $\tilde{\mathcal{O}}(d/\epsilon^{1.5})$ respectively for classical distribution and quantum density matrices. Both these papers use quantum amplitude estimation (QAE)

[BHM+02] as the means to estimate the target quantities. However, QAE requires full-fledged fault tolerant quantum computers and may not be available in the near future.

Approximation algorithms that estimate a quantity to within a multiplicative factor (i.e. such that the estimate $\tilde{x}$ lies in the interval $[x/\gamma, \gamma x]$ for some $\gamma > 1$) are particularly valuable when the target quantity might be small, and these algorithms are often harder and more complex. Batu et al. [BDK+02] consider the estimation of Shannon entropy to multiplicative precision, showing that $\mathcal{O}(d^{(1+\eta)/\gamma^2} \log d)$ samples suffice to estimate it to within a factor γ, for classical distributions with $S(p) > \gamma/\eta$. This is almost matched by a lower bound of $\Omega(d^{(1-\eta)/\gamma^2} \log d)$ later proven in [Val11]. To the best of the authors' knowledge, the problem of estimating entropies to within a multiplicative factor has not been considered in the quantum algorithms literature.

3.2 Main Results

Here, we consider the estimation of Renyi entropies in the purified quantum query access model, and approach the problem using sampling via a DQC1 method, rather than using the QAE algorithm. While being less powerful than quantum amplitude estimation, such sampling techniques have the advantage of requiring less stringent quantum resources. In particular, QAE requires long coherence times, and the application of powers of the input oracle and its inverse conditioned on large ancillary registers. Sampling methods in general trade away these requirements for a quadratic increase in the scaling with the precision parameter ϵ.

Using a recent iterative method of Chowdhury et al. [CSS19], we show how the trace of power functions of the input state can be estimated to within a suitable multiplicative precision, in order to obtain additive approximations of the Renyi entropy. This iterative algorithm has an expected asymptotic runtime that depends on the unknown quantity being estimated. Thus we can obtain better bounds on its complexity than by considering only the worst case asymptotic runtime.

Our approach is to construct a unitary that encodes (or probabilistically implements) the matrix function ρ^α. Then we can estimate its normalised trace using the DQC1 model. We will assume that α is a constant, and leave it out of complexity considerations. Our first result is an algorithm that outputs an additive approximation to the α-Renyi entropy of an unknown quantum state, for $\alpha \neq 1$.

Theorem 3.1. *Given a unitary process U_ρ on $\mathbb{C}^{d+a}$ which produces a purification $|\psi_\rho\rangle$ of a mixed state $\rho \in \mathbb{C}^{d\times d}$ with $\mathbb{1}/\delta \preceq \rho \preceq \mathbb{1}$, for $\alpha > 0$ with $\alpha \neq 1$, there exists an iterative quantum algorithm that outputs an estimate $\tilde{S}$ such that*

$$\left|\tilde{S} - S_\alpha(\rho)\right| \leq \epsilon,$$

with high probability. The algorithm runs for at most $\mathcal{O}(\log d)$ *rounds, making an expected number* $\mathcal{O}(1/(x\epsilon)^2)$ *of independent applications of a quantum circuit which coherently invokes* U_ρ *and* $U_\rho^\dagger$ *a total of* $m = \mathcal{O}(1/\delta \log d/\epsilon)$ *times, in each case measuring a single output qubit. Here,* $x = \frac{1}{d}\, Tr(\rho^\alpha)$*. The algorithm uses* $\mathcal{O}(m)$ *additional* 1- *and* 2*-qubit gates, and* $\lceil \log d \rceil + 2$ *ancillary qubits.*

Furthermore, when $\alpha \neq 1$ *is an integer,* $m = \alpha$*, making the circuit depth is independent of the dimension* d*.*

We construct the algorithm proving this claim by using the technique of block-encodings and quantum singular value transformations [CGJ19, GSL+19] to implement unitaries that are block encodings of the power functions ρ^α on the system adjoined with ancillary registers, and subsequently estimating the trace of these unitaries in the DQC1 or "one-clean qubit" model of computation [KL98] in combination with the method of [CSS19]. In contrast, in [GL19] a block encoding of $\log \rho$ is applied to a suitable input state, resulting in a state that encodes the von Neumann entropy as the amplitude of a computational basis state, which is then estimated using QAE.

We also consider the problem of estimating the entropy to multiplicative precision by using the same iterative subroutine again, improving on the additive estimate obtained, to arrive at an estimate $\tilde{S}_\alpha$ satisfying $(1 - \epsilon_{rel})S_\alpha \leq \tilde{S}_\alpha \leq (1 + \epsilon_{rel})S_\alpha$, with an overhead that is at most logarithmic in the dimension of ρ.

Theorem 3.2. *Iterating the algorithm of Theorem 3.1, we can obtain an estimate* $\hat{S}$ *which satisfies*

$$\left| \frac{\hat{S}}{S_\alpha(\rho)} - 1 \right| \leq \epsilon_{rel}$$

with high probability. The algorithm runs for at most

$$R = \mathcal{O}\left(\log\left(\frac{\log d}{\delta} \right) \right)$$

rounds, and the total expected number of runs of the DQC1 *circuit and corresponding single qubit measurements is given by*

$$\mathcal{O}\left(\frac{1}{(\epsilon x S_\alpha(\rho))^2} \right),$$

where as before, $x = \frac{1}{d}\, Tr(\rho^\alpha)$*.*

Since by our assumption and the definition of the entropy we have $\mathcal{O}(\delta) \leq S_\alpha(\rho) = \frac{1}{1-\alpha} \log dx$, we note that the the factor $1/S_\alpha(\rho)$ is large only when $dx \to 1$, so that this is at most a factor of $1/\delta^2$ worse than the expected number of measurements required in Theorem

3.1. This problem of approximating the entropy of a state to within a multiplicative precision, which is generally more difficult than additive approximation, has not been discussed in the quantum algorithm literature to the best of our knowledge. We provide a comparison between our work and some of the known results in Table 3.1.

We thus extend the investigation of evaluating entropy functions to the case of Renyi entropy, in the purified quantum query access model considered in [GL19]. Our key contributions are: (1) the use of unitary block encodings of the target operator functions obtained using quantum matrix function implementation techniques, in combination with (2) the replacement of QAE with trace estimation using one clean qubit, and (3) obtaining approximations to multiplicative precision. Since it uses the one-clean qubit model, our method does not require long coherence times or high circuit depth. Furthermore, only a single clean and well-controlled qubit is required, while the remainder can start off in the maximally mixed (highly noisy) state; apart from being a low resource requirement, this also makes our algorithm potentially feasible for testing on near-term NMR-based quantum hardware. Finally, our runtime analysis using the algorithm recapped in Appendix B allows us to bound the expected number of measurements as a function of the unknown target quantity, offering better bounds than would be given by just a worst case analysis.

S_α	Copies of ρ ($\Theta(\cdot)$)	$\mathbb{E}$ [#mmts]	Queries to U_ρ per use of circuit
$\alpha < 1$	$(d/\epsilon)^{2/\alpha}$	$\mathcal{O}(1/(x\epsilon)^2)$	$\mathcal{O}\left(\frac{1}{\delta}\log\frac{d}{\epsilon}\right)$
$\alpha > 1$ non-integer	$(d/\epsilon)^2$	〃	〃
$\alpha > 1$ integer	$\frac{d^{2-2/\alpha}}{\epsilon^{2/\alpha}}$	〃	$\mathcal{O}(\alpha)$

Table 3.1: Sample complexity from [AIS+17] for estimating the Renyi entropies of an unknown d-dimensional mixed state to additive precision ϵ for different ranges of α, contrasted with expected number of measurements in our work (Section 3.4). In the worst case, these scale as $\mathcal{O}(d^2)$ for $\alpha < 1$, and $\mathcal{O}(d^{2\alpha})$ for $\alpha > 1$. (We use 〃 to indicate the same expression as on the preceding line)

3.3 Preliminaries

3.3.1 Input model

We assume access to a unitary process U_ρ on $\mathbb{C}^d \otimes \mathbb{C}^a$ which produces a purification $|\psi_\rho\rangle$ of the actual input state ρ in $\mathbb{C}^{d\times d}$

$$U_\rho |0\rangle = |\psi_\rho\rangle = \sum_{i=1}^{n} \sqrt{p_i} |\psi_i\rangle_d |\phi_i\rangle_a , \tag{3.2}$$

so that $\mathrm{Tr}_{[]\,a}|\psi_\rho\rangle\langle\psi_\rho| = \rho$. The $\{|\phi\rangle_a\}$ and $\{|\psi\rangle_d\}$ are sets of orthonormal vectors on the ancillary and system subspaces respectively. This model, known as the purified quantum query access model, is also discussed by [GL19] and [Bel19] in the context of property testing.

Note that the case of a classical probability distribution on d points with sampling access is subsumed into this model by embedding it into the diagonal state $\rho_p = \sum_{i=1}^{d} p_i |i\rangle$.

3.3.2 Implementing power functions of Hermitian matrices

A block-encoding U_A of a Hermitian matrix A is essentially a unitary that encodes a (sub-)normalised version of A in its top left block, i.e.

$$U_A = \begin{pmatrix} A/\Lambda & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{3.3}$$

where $\Lambda \geq \|A\|$. The behaviour and use of such encodings has been explored extensively in the last few years [LC16, CGJ19, GSL+19]. Given access to U_A, a variety of smooth matrix functions (defined on the spectrum of A) may be implemented, in the sense that a new block encoding U_A^f can be obtained such that

$$U_A^f = \begin{pmatrix} f(A)/\beta & \cdot \\ \cdot & \cdot \end{pmatrix}, \tag{3.4}$$

where $\beta \geq \|f(A)\|$. In particular, here we are interested in power functions $f(x) = x^\alpha$ for an exponent $\alpha > 0$. These can be realised using e.g. Lemma 9 of [CGJ19] or Corollary 67 of [GSL+19], with minor modifications. Assuming that ρ not a pure state, and has minimum eigenvalue $\delta \in (0, 1/2)$, an ϵ-approximate block encoding of ρ^α can be created with $\mathcal{O}(\frac{1}{\delta}\log\frac{1}{\varepsilon})$ uses of U_ρ. Even if ρ has a non-trivial kernel, it is fairly easy to implement the matrix function only on the non-singular part of the input (e.g. [HHL09, GL19]), and for classical distributions, we can consider the restriction to the support of the distribution by pre-processing using e.g. sparse PCA.

The precision ε specifies how close the top left block of the new encoding is to ρ^α in the operator norm. For integral values of α we can obtain an exact encoding with $\varepsilon = 0$, e.g. using Chebyshev polynomial methods as in Chapter 1; this has the effect of removing the logarithmic factors from the complexity for integral α. We collect the necessary results about implementing matrix function in Appendix A.

3.3.3 The DQC1 Model

The one clean qubit or DQC1 (for *Deterministic Quantum Computation with one clean qubit*) paradigm was inspired by studies of NMR based quantum computing [SV99]. The model

generated interest due to the potential for physical implementation, since only a single 'clean' qubit has to be initialised to a pure state, while the remaining 'dirty' qubits can start in highly mixed (or random) states. The possibility that even such a weak model can be used to perform useful computations is striking [KL98, Jor08], because if all the qubits are in the maximally mixed state no non-trivial computation can be done (no unitary process can purify the state). Algorithms in the DQC1 model always have a fixed initial state of $n+1$ qubits, and both the problem instance and algorithm are encoded into a poly(n)-sized unitary circuit to be applied on them. The answer is encoded into the probability of obtaining the outcome zero on measuring the clean qubit at the end of the computation. The problem of computing the normalised trace of a unitary was shown to be complete for DQC1 [KL98]. Further work indicated the presence of non-classical correlations in DQC1 computations [DSC08]. More recently, it was shown that it is classically hard to sample from the output distribution of a DQC1 computer up to multiplicative error, conditional on standard complexity theoretic assumptions [MFF14, FKM$^+$18].

The DQC1 or "one-clean qubit" model of computation is based on the use of a single well-controlled or 'clean' qubit, and a number n of noisy qubits that are taken to be in the maximally mixed state [KL98, SJ08]. Algorithms in this model are embedded into some controlled n-qubit unitaries, and the outputs are encoded into the probability of observing 0 on measuring the clean qubit. Estimating the normalised trace of a unitary is known to be a DQC1-complete problem [KL98].

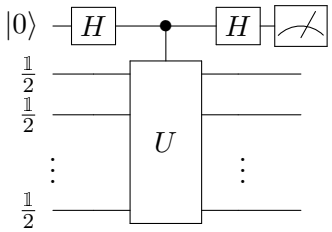


Figure 3.1: A DQC1 circuit that can be used to estimate $1/2^n \operatorname{Tr}(U)$, for which no classical efficient algorithm is known. Measurements are made in the computational basis.

The initial state consists of one qubit set to the $|0\rangle$ state, and n qubits in the maximally mixed state, i.e. $\xi_{\text{in}} = |0\rangle\langle 0| \otimes \mathbb{1}_n/2^n = \frac{\mathbb{1}+Z}{2} \otimes \mathbb{1}_n/2^n$. We can write the final state in figure 3.1 after the application of the circuit but before measurement as

$$\xi_{\text{out}} = \frac{1}{2^{n+1}} \begin{pmatrix} \mathbb{1}_n & U^\dagger \\ U & \mathbb{1}_n \end{pmatrix}, \tag{3.5}$$

from which we see that the expectation values of the Pauli X and Y operators for the clean qubit give the estimates $\langle X \rangle = 2^{-n}\text{Re}(\operatorname{Tr}(U))$ and $\langle Y \rangle = -2^{-n}\text{Im}(\operatorname{Tr}(U))$. These can be

extracted to within additive precision ϵ using $\mathcal{O}(\log \frac{1}{\eta}/\epsilon^2)$ measurements, with probability at least $1-\eta$ for $\eta > 0$. Since the dependence on η is logarithmic and the success probability can easily be boosted by repetition or other standard methods, we will drop the factor in η from further consideration.

As discussed in [Jor08], and used for example by Cade and Montanaro [CM18], the DQCk model with k clean qubits can be used to obtain the trace of a submatrix whose size is an inverse-poly sized fraction of the whole unitary. This is useful in our context, because we deal with a unitary block encoding of matrix functions such as ρ^α, so that these target matrices (whose trace we are interested in) are submatrices located in the top-left corner of the unitary.

3.4 Proof of Theorem 3.1

In order to prove Theorem 3.1 by describing the algorithm and analysing its complexity, we begin by considering how the error in estimating $x = \operatorname{Tr}(\rho^\alpha)$ propagates to the estimate of $S_\alpha(\rho)$ calculated using x.

3.4.1 Error analysis

From the definition of the α-Renyi entropy (eq. (3.1)), if we use an estimate $\tilde{x}$ of $x = \operatorname{Tr}(\rho^\alpha)$, the corresponding error in $S_\alpha(\rho)$ is given by

$$\begin{aligned}\left|\tilde{S}_\alpha(\rho) - S_\alpha(\rho)\right| &= \left|\frac{\log \tilde{x}}{1-\alpha} - \frac{\log x}{1-\alpha}\right| \\ &= \left|\frac{1}{1-\alpha}\log\frac{\tilde{x}}{x}\right|.\end{aligned}$$

From the above expression it is clear that if $\tilde{x}$ an estimate of x to a multiplicative factor $\gamma > 0$, satisfying

$$\frac{x}{\gamma} \leq \tilde{x} \leq \gamma x,$$

then we obtain an estimate $\tilde{S}_\alpha(\rho)$ to an additive precision given by

$$\left|\tilde{S}_\alpha(\rho) - S_\alpha(\rho)\right| \leq \left|\frac{\log \gamma}{1-\alpha}\right|$$

For 'tight' multiplicative approximations, with $\gamma = 1 + \epsilon_{rel}$ where we shall refer to $\epsilon_{rel} < 1$ as the multiplicative precision, we can write

$$(1-\epsilon_{rel})x \leq \tilde{x} \leq (1+\epsilon_{rel})x. \tag{3.6}$$

For such γ, since $y - \frac{y^2}{2} \leq \log(1+y) \leq y$, we have

$$\left|\tilde{S}_\alpha(\rho) - S_\alpha(\rho)\right| \leq \frac{\epsilon_{rel}}{|1-\alpha|}. \tag{3.7}$$

Thus, an upper bound on the complexity of estimating $S_\alpha(\rho)$ to additive precision ϵ is directly given by that of the method used to estimate $\mathrm{Tr}\,(\rho^\alpha)$ to multiplicative precision $\epsilon|1-\alpha|$.

Let us now consider how to obtain approximations of the (normalised) trace of ρ^α using the DQC1 method described in the previous section.

3.4.2 Estimating $\mathrm{Tr}\,(\rho^\alpha)$

From Appendix A we first use lemma 3.4 to obtain the block encoding of ρ, and then use corollary 3.6 to construct an ε-approximate block encoding for ρ^α:

$$U_\rho = \begin{pmatrix} \rho & \cdot \\ \cdot & \cdot \end{pmatrix} \mapsto \begin{pmatrix} f_\alpha(\rho) & \cdot \\ \cdot & \cdot \end{pmatrix} =: U_\alpha,$$

where $\|f_\alpha(\rho) - \rho^\alpha\| < \varepsilon$. The unitary U_α requires $\mathcal{O}\left(\frac{\max(1,\alpha)}{\delta}\log\frac{1}{\varepsilon}\right)$ uses of the block encoding of U_ρ, and two more than the number of ancillary qubits as U_ρ, where $\delta > 0$ lower bounds the least eigenvalue of ρ. Since the dependence of the complexity on ε is logarithmic, we can afford to choose exponentially small ε if necessary, incurring only a polynomial overhead. In fact, we shall choose $\varepsilon = \epsilon/d$, since the error in the trace is then bounded by $d\varepsilon = \mathcal{O}(\epsilon)$. This will only result in a factor of $\mathcal{O}(\log d/\epsilon)$ in the query complexity.

The normalised trace of this $d+a$-dimensional unitary U_α can be estimated to additive precision ϵ with high probability by making $\mathcal{O}(\log 1/\epsilon^2)$ uses of the unitary (or more precisely of the DQC1 circuit in Fig. 3.1). Since U_α has the block form

$$U_\alpha = \begin{pmatrix} \rho^\alpha & \cdot \\ \cdot & \cdot \end{pmatrix},$$

its trace contains a contribution from $\mathrm{Tr}\,(\rho^\alpha)$. What we would like is to isolate this term alone. Importantly, we know that $\mathrm{Tr}\,(\rho^\alpha)$ will be real and positive.

As mentioned in Section 3.3.3 this can be done, for example, by choosing to measure the ancillary qubits of U_α, and requiring them to be in the $|0\rangle$ state, hence projecting onto the relevant subspace to capture the trace of the submatrix ρ^α. We also describe another simple way of doing the same task below. Consider multiplying U_α by a controlled phase operator to

convert the block-encoded matrix alone to $i\rho^\alpha$, i.e. consider the unitary $U'_\alpha = U_\alpha V$ where the unitary V applies a conditional phase of $e^{i\pi/2} = i$,

$$V = i\,|0\rangle\langle 0| \otimes \mathbb{1} + \sum_{k=1}^{a-1} |k\rangle\langle k| \otimes \mathbb{1}, \tag{3.8}$$

which can easily be arranged using an ancillary qubit initialised to the $|+\rangle$ state to which a conditional rotation of $R_y(\pi/2)$ is applied. Now, exploiting the fact that $\mathrm{Tr}\,(\rho^\alpha)$ is purely real, we can estimate it by $\mathrm{Re}(\,\mathrm{Tr}\,(U_\alpha)) - \mathrm{Re}(\,\mathrm{Tr}\,(U'_\alpha))$, using only twice as many measurements and uses of U_α as required for estimating $\mathrm{Re}(\,\mathrm{Tr}\,(U_\alpha))$ itself.

Finally, recall that with DQC1 we get the *normalised* trace. However as discussed in Section 3.4.1, since we are interested in a multiplicative approximation to the trace, this does not pose any issue — approximations to within a multiplicative factor are unaffected when scaled by a constant. Thus an estimate of $\frac{1}{d}\,\mathrm{Tr}\,(\rho^\alpha)$ to multiplicative precision ϵ is also a valid estimate of $\mathrm{Tr}\,(\rho^\alpha)$ to within the same multiplicative precision.

Estimating the trace to multiplicative precision

In Appendix B, we review an iterative method introduced in [CSS19] for improving an additive precision estimate to a multiplicative one, by starting off with inaccurate additive estimates (with large ϵ) and progressively improving the precision, without requiring too many iterations. Combining this idea encapsulated in Algorithm 3.1 with the estimation of normalised trace, we obtain an estimate $\tilde{x}$ that satisfies

$$\left|\frac{\tilde{x}}{x} - 1\right| < \epsilon_{rel}.$$

It is worth mentioning that the only valid multiplicative approximation to *any* precision of a quantity that takes value zero is zero itself. We do not need to worry about the possibility of $x = 0$ here, because the trace of ρ^α is non-zero for all quantum states ρ and finite values of α.

From the discussion in Appendix B, we conclude that at most

$$\mathcal{O}\left(\frac{x_{\max}}{x_{\min}}\right)$$

iterations are required, and the total expected number of uses of the block encoding of ρ^α and subsequent single qubit measurements is given by

$$\mathcal{O}\left(\frac{1}{(x\epsilon)^2}\right),$$

where $x = \frac{1}{d}\operatorname{Tr}(\rho^\alpha)$ is the unknown quantity being estimated. The minimum and maximum values of the normalised trace are straightforward to compute, and we discuss them below for completeness.

Minimum and maximum values of $\operatorname{Tr}(\rho^\alpha)$

Since we know states are normalised so that $\operatorname{Tr}(\rho) = 1$, the eigenvalues $p_1, \ldots, p_d$ of ρ form a probability distribution. The minimum and maximum values of the trace of ρ^α can then easily be calculated (e.g. see Lemma 5 of [AIS+17]). Using the definition of the trace

$$\operatorname{Tr}(\rho^\alpha) := \sum_{i=1}^{d} p_i^\alpha,$$

we have for $\alpha < 1$

$$1 \leq \operatorname{Tr}(\rho^\alpha) \leq d^{1-\alpha},$$

and for $\alpha > 1$

$$d^{1-\alpha} \leq \operatorname{Tr}(\rho^\alpha) \leq 1,$$

Correspondingly, for the normalised trace we have the ranges

$$\frac{1}{d}\operatorname{Tr}(\rho^\alpha) \in \begin{cases} [d^{-1}, d^{-\alpha}] & 0 < \alpha < 1 \\ [d^{-\alpha}, d^{-1}] & 1 < \alpha \end{cases} \tag{3.9}$$

Hence, for the $\alpha < 1$ case we can also consider scaling up the block encoding of ρ^α by d using QSVT techniques for "pre-amplification" which we recall in Lemma 3.7 of Appendix A. This will make the normalised trace larger than unity and hence any additive approximation is also a multiplicative approximation to the same precision, whence the cost for estimating the entropy becomes $1/\epsilon^2$ measurements in the DQC1 of a unitary block encoding circuit of $d\rho^\alpha$ that uses U_ρ and $U_\rho^\dagger$ around $\mathcal{O}\left(\frac{d\log(d/\epsilon)}{\epsilon^2\delta}\right)$.

3.5 Proof of Theorem 3.2

Minimum and maximum values of $S_\alpha(\rho)$ given ρ has non-trivial eigenvalues

Suppose the least eigenvalue of ρ is known to be some fixed $p_{\min} = \delta \in (0, 1/2)$. We only consider the case where $\delta < 1/d$, since otherwise the actual minimum eigenvalue will be zero

since the state is normalised.

Renyi Entropies: For the case of the Renyi entropies, we can simply use the minimum and maximum values of $\operatorname{Tr}(\rho^\alpha)$ from (3.9). With the minimum eigenvalue fixed, this is achieved by the minimisers and maximisers of the entropy functional for the remaining $d-1$ dimensional distribution.

Note that when the density matrix has a non-trivial kernel, the maximum entropy can be $\mathcal{O}(1)$ independent of the dimension. For example if we take $\delta = 1/4$, the uniform distribution over the remaining $d-1$ dimensions is excluded from our consideration for $d \geq 5$. However, we will not need to make such a detailed analysis.

Since the factor of $1-\alpha$ in the definition of the entropy changes sign at $\alpha = 1$, we do not need to consider $\alpha > 1$ and $\alpha < 1$ separately; the minimum and maximum entropy are the same in both cases. Taking the $\alpha > 1$ case for illustration, we have in this case $1-\alpha < 0$ and $(d-1)^{1-\alpha} < 1$, so

$$\begin{aligned} S_\alpha^{\min} &= \frac{1}{1-\alpha}\log\left(\delta^\alpha + (1-\delta)^\alpha\right) \\ &\geq \frac{\alpha}{1-\alpha}\log(1-\delta) \geq \frac{\delta\alpha}{|\alpha-1|} \\ S_\alpha^{\max} &= \frac{1}{1-\alpha}\log\left(\delta^\alpha + (d-1)^{1-\alpha}(1-\delta)^\alpha\right) \\ &\leq \log d \end{aligned} \tag{3.10}$$

Thus we then have, since $0 < \delta < 1/2$,

$$\frac{S_\alpha^{\max}}{S_\alpha^{\min}} = \mathcal{O}\left(\frac{\log d}{\delta}\right). \tag{3.11}$$

Thus, from the analysis of Appendix B we see that Algorithm 3.1 runs for at most

$$R = \mathcal{O}\left(\log\left(\frac{\log d}{\delta}\right)\right) \tag{3.12}$$

rounds, and the total expected number of runs of the DQC1 circuit and corresponding single qubit measurements is given by (3B.4) with $x = S_\alpha(\rho)$ and with the algorithm of Theorem 3.1 playing the role of the additive estimation subroutine. That is

$$\text{number of mmts.} = \mathcal{O}\left(\frac{1}{(\epsilon x S_\alpha(\rho))^2}\right), \tag{3.13}$$

where $x = \frac{1}{d}\operatorname{Tr}(\rho^\alpha)$.

3.6 Discussion

The method we have proposed here is interesting primarily due to the use of the recently developed algorithmic technique of block encodings along with the one-clean qubit model which is believed to be relatively easier to implement than full fledged error corrected quantum computers. These block encoding methods allow the implementation of several other matrix functions, which may facilitate the estimation of other entropy-like matrix functionals; there are also several possible applications of estimating entropic functionals as a subroutine in algorithmic procedures for pattern matching, compression tasks, and network analysis.

Our algorithm is certainly not optimal in either the dimension or the inverse precision, and both improving the query complexity and obtaining lower bounds in the purified access input model would be interesting. As can be seen from Table 3.1, the sample complexity bounds obtained by [AIS+17] using Empirical Young Diagram methods also scale quadratically in the dimension and inverse precision, and they show that their bounds are tight in most cases; it is worth noting though that they allow arbitrary measurements and classical post-processing, which are stronger resources than demanded by our methods. We are currently working on proving lower bounds in our model, in order to understand whether the complexity of our algorithm can be improved as a function of either of these parameters, or if it is in fact tight.

Trace estimation using the DQC1 method can be motivated by the so-called HADAMARD test. The advantage in using the DQC1 method is that only a constant number of well controlled, 'clean' qubits are required, and the task of initial state preparation is significantly eased. On the other hand, using the HADAMARD test to estimate measurement outcome frequencies requires the preparation of suitable initial states which introduces additional sources of error and complexity. Similarly, the amplitude estimation methods which have previously been used for entropy estimation in quantum property testing algorithms require long coherence times and the application of powers of U_ρ and $U_\rho^\dagger$ controlled on ancillary registers, leading to deep circuits, and in general need full fledged fault-tolerant quantum computers.

Using the iterative method of Algorithm 3.1 brings the advantage of obtaining approximations to multiplicative precision, with its stopping condition ensuring that the algorithm runs for at most $\mathcal{O}(\log d/\delta)$ iterations for states with entropy at least δ. As discussed in Appendix B, the expected number of iterations in fact depends logarithmically on the ratio of the unknown target quantity S to its maximum possible value, and consequently the total number of runs of the DQC1 subroutine is depends quadratically on S^{-1} — this is particularly valuable since we can explicitly bound the expected runtime by a function of the unknown target quantity, while this kind of analysis is not often possible. Entropies can indeed take values in $[0, 1] \subseteq [0, \log d]$ where d is the dimension of the system. It is important to note that it would be useful to have some other method to test if the input state is pure, since then it would have zero entropy and our algorithm for obtaining multiplicative estimates would fail.

The definition of the complexity class DQC1 requires that the number of runs or measurements made scales polynomially in the total number of dirty qubits. Thus we clarify that our method, which requires ϵ to scale inverse polynomially with d, does not place the (decision variant of) task of entropy estimation in DQC1. However, it would be interesting question to see if the close connection between entropy estimation and trace estimation can be exploited to solve the problem in DQC1, or if it would be possible to prove lower bounds showing that it is strictly harder. Entropy estimation commonly falls under the category of problems of distributional property testing. It is worth investigating what kinds of property testing tasks can be solved within DQC1, and what lower bounds can be proved in this model. Such studies would also throw further light on the power and limitations of this restricted model of quantum computation.

Appendix A Implementing power functions of density matrices

Gilyén et al. [GSL+19] give a series of lemmas showing how to implement block encodings of different kinds of inputs, of which we will be interested in the case of density operators.

Definition 3.3 (Definition 43, [GSL+19]). *An (α, a, ϵ) block encoding of an operator A acting on s qubits is a unitary U acting on $a+s$ qubits, such that*

$$\|A - \alpha\Pi^\dagger U \Pi\| \leq \epsilon, \tag{3A.1}$$

where the first register consists of ancillary qubits, $\Pi := |0\rangle^{\otimes a} \otimes \mathbb{1}_s$ is an isometry $\Pi : (\mathbb{C}^2)^{\otimes s} \mapsto span_{\mathbb{C}}\{|0\rangle^{\otimes a}\} \otimes (\mathbb{C}^2)^{\otimes s}$, and $\alpha, \epsilon \in (0, \infty)$.

The conversion of a purified access oracle as in eq. (3.2) into a block encoding for ρ can be achieved using the following result.

Lemma 3.4 (Lemma 45, [GSL+19]). *Given a unitary U_ρ acting on $a+s$-qubits, which prepares a purification $U_\rho |0\rangle |0\rangle = |\rho\rangle$ of an s-qubit density operator ρ, such that $Tr([]\, a]|\rho\rangle\langle\rho| = \rho$, the unitary*

$$U = \left(U_\rho^\dagger \otimes \mathbb{1}_s\right) (\mathbb{1}_a \otimes \mathsf{SWAP}_s) (U_\rho \otimes \mathbb{1}_s)$$

gives an exact $(1, a+s, 0)$ block encoding of ρ.

This means that the unitary U has the block form

$$U = \begin{pmatrix} \rho & \cdot \\ \cdot & \cdot \end{pmatrix},$$

where we have not specified the $a+s-1$ other s-qubit blocks on the diagonal.

Such block encodings can be used to implement smooth functions of the input matrix via polynomial approximations, with the following theorem.

Theorem 3.5 (Theorem 56, [GSL+19]). *Given an (α, a, ϵ) block encoding U of a Hermitian matrix A, for any degree m polynomial $P(x)$ that satisfies $\forall\ x \in [-1, 1],\ |P(x)| < 1/2$, there exists a $(1, a+2, 4m\sqrt{\epsilon/\alpha} + \delta)$ block encoding U_p of $P(A/\alpha)$. We can construct U_p using m applications of U and $U^\dagger$, a single application of controlled-U, and $\mathcal{O}((a+1)m)$ additional 1- and 2-qubit gates. A description of the circuit of U_p can be calculated in $\mathcal{O}(\mathrm{poly}(m, \log 1/\delta))$ time on a classical computer.*

Using Theorem 3.5, we can implement ϵ-approximate block encodings of power functions ρ^α on the part of the spectrum of ρ that is contained in $[\delta, 1]$ for $\delta > 0$ by using polynomial approximations. The lower cutoff δ is necessary because power functions for non-integer α are

not differentiable at $x = 0$. Monomials for $\alpha = 1, 2, \ldots$ can be implemented exactly on the entire domain $[0, 1]$.

Lemma 3.6 (Corollary 67, [GSL+19]). *Given an exact unitary block encoding U of a d-qubit density matrix ρ, that uses a- ancillary qubits, we can implement an ε-approximate block encoding of ρ^α for $\alpha > 0$ using $\mathcal{O}(\frac{\max(1,\alpha)}{\delta} \log \frac{1}{\varepsilon})$ applications of U, and $a + 2$- ancillary qubits. Here we assume that the spectrum of ρ is contained in $[\delta, 1]$.*

We also quote the following useful theorem that shows how to amplify the singular values of a block encoding.

Lemma 3.7 (Theorem 30, [GSL+19]). *Given an exact unitary block encoding U of a Hermitian operator A, that uses a ancillary qubits, for $\gamma > 1$ we can implement a block encoding U_γ such that every eigenvalue $\lambda_i < \frac{1-\delta}{\gamma}$ of A is amplified to $\gamma\lambda_i$ to multiplicative precision ϵ*

$$\left| \frac{\tilde{\lambda}_i}{\gamma \lambda_i} - 1 \right| < \epsilon.$$

U_γ requires a single ancillary qubit, and m applications of U and $U^\dagger$, and $\mathcal{O}((a + 1)m)$ additional 1- and 2-qubit gates, where

$$m = \mathcal{O}\left(\frac{\gamma}{\delta} \log \frac{\gamma}{\epsilon} \right) \tag{3A.2}$$

Here we assume that the spectrum of A is contained in $[\delta, 1]$.

Appendix B Obtaining multiplicative approximations using additive approximations

Suppose we have an algorithm to obtain an ϵ-additive approximation $\tilde{A}$ to some unknown quantity A, i.e. $|A - \tilde{A}| \leq \epsilon$. Often we may be interested in an ϵ_{rel}-multiplicative approximation that satisfies $(1 - \epsilon_{rel})A \leq \tilde{A} \leq (1 + \epsilon_{rel})A$; this is clearly useful when the target quantity could be small, potentially making it difficult to choose an additive precision in advance. Given a lower bound $0 < \lambda \leq |A|$, an appropriate additive precision can be chosen to get a desired precision multiplicative approximation. The complexity of estimating the multiplicative approximation increases by a factor of $\mathcal{O}(\lambda^{-1})$ over that required for additive approximation. If we know independent of the problem size that $|A| > 1$, then any ϵ-additive approximation is also a good $\epsilon' < \epsilon$ multiplicative approximation.

Chowdhury et al. [CSS19] have described a procedure to produce a multiplicative approximation of a quantity x that has known upper and lower bounds $x_{\max}$ and $x_{\min}$ to precision

ϵ_{rel} using a series of additive precision approximations with exponentially increasing precision in the form

$$\epsilon_r = \frac{\epsilon_{rel} x_{\max}}{2^{r+1}}.$$

They show that

$$R = \lceil \log \frac{x_{\max}}{x_{\min}} \rceil \tag{3B.1}$$

iterations suffice to obtain a multiplicative approximation $\tilde{x}$ satisfying

$$\left| \frac{\tilde{x}}{x} - 1 \right| < \epsilon_{rel}, \tag{3B.2}$$

with high probability. Furthermore, their algorithm has an expected runtime that depends on $\log\left(\frac{x_{\max}}{x}\right)$, which could be significantly better than the worst case if x is close to its maximum value. For a proof that this algorithm, which we recap in Algorithm 3.1, indeed returns an estimate satisfying eq. (3B.2), we refer to [CSS19]. We shall discuss its expected runtime below, which will be useful in our analysis.

Algorithm 3.1 Approximating x to multiplicative precision.

function MULTESTIMATE(ADDESTIMATE, c, ϵ_{rel}, $x_{\max}$, $x_{\min}$)
  $c' \leftarrow 1 - {}^{(1-c)}/_{R}$ ▷ success probability for additive estimate
  $r \leftarrow 0$
  repeat
    $\epsilon_r \leftarrow \frac{\epsilon_{rel}}{2^{r+1}} \cdot x_{\max}$ ▷ precision for additive estimate
    $x_r \leftarrow x_{\max}/2^r$
    $\tilde{x}_r \leftarrow$ ADDESTIMATE(ϵ_r, c') ▷ obtain $\tilde{x}_r$ *s.t.* $\Pr\left(|x_r - \tilde{x}_r| < \epsilon_r\right) > c'$
    $r \leftarrow r + 1$
  until $\tilde{x}_r > x_r$
return $\tilde{x}_r$
end function

From the choice (3B.1) and the stopping condition in Algorithm 3.1, the number of iterations is always $1 \leq r \leq R$. Without loss of generality, $\exists\ q \geq 1$ such that

$$\frac{x_{\max}}{2^{q-1}} > x > \frac{x_{\max}}{2^q}. \tag{3B.3}$$

To bounded the expected number of rounds, consider the case where the algorithm fails to terminate for $r < q + 1$. Then for $r \geq q + 1$ the probability that the algorithm does not stop at step r is given by

$$\Pr\left(\tilde{x}_r < x_r\right) \leq 1 - c',$$

where we get the upper bound by noting that since $r \geq q + 1 \implies x_r > x$, and the estimate $\tilde{x}_r$ is ϵ_r close to x. This requires the mild underlying assumption that $\epsilon_{rel} < 2$.

Hence the net probability that the algorithm terminates at step $r = q + k$ for $k \geq 1$ is at most $(1 - c')^{k-1}c'$. If $c' > 1/2$, the expected value of k under this geometric distribution is bounded above by $q + 2$, and from eq. (3B.3) we have

$$q \leq \log_2 \left(\frac{2x_{\max}}{x} \right).$$

Since we will use the DQC1 normalised trace estimation technique as the underlying subroutine to obtain the additive estimate, we can also calculate the total expected number of measurements. This will also give us a bound on the number of uses of the input unitary. The k^{th} iteration requires $\mathcal{O}(\epsilon_k^{-2})$ measurements, and so if the algorithm terminates at step r, the cumulative number of measurements scales as

$$\begin{aligned} M_{\leq r} &\approx \sum_{k=0}^{r} \frac{1}{\epsilon_k^2} = \left(\frac{1}{x_{\max}\epsilon_{rel}} \right)^2 \sum_{k=0}^{r} 4^{k+1} \\ &= \left(\frac{1}{x_{\max}\epsilon_{rel}} \right)^2 \frac{4^{r+1} - 1}{3} \\ &= \mathcal{O}\left(\left(\frac{2^r}{x_{\max}\epsilon_{rel}} \right)^2 \right). \end{aligned}$$

Now we can upper bound the expected value of $M_{\leq r}$ under the geometric distribution for the number of iterations; we have

$$\begin{aligned} \mathbb{E}[M] &\leq \sum_{k=1}^{\infty} c'(1 - c')^{k-1} M_{\leq q+k} \\ &\leq \left(\frac{1}{x_{\max}\epsilon_{rel}} \right)^2 c' \sum_{k=1}^{\infty} (1 - c')^{k-1} 4^{q+k} \\ &= 4^{q+1} \left(\frac{1}{x_{\max}\epsilon_{rel}} \right)^2 \cdot c' \sum_{k=0}^{\infty} (4(1 - c'))^k \\ &= \mathcal{O}\left(\left(\frac{2^q}{x_{\max}\epsilon_{rel}} \right)^2 \right), \end{aligned}$$

where we make the mild assumption that $c' > 3/4$. Recalling that $\frac{x_{\max}}{2^{q-1}} > x$, we finally have

$$\mathbb{E}[M] = \mathcal{O}\left(\left(\frac{1}{x\epsilon_{rel}} \right)^2 \right). \tag{3B.4}$$

Thus, the total expected number of measurements, or equivalently uses of the input unitary, scales quadratically in the inverse of the target quantity.

This is very useful for our problem: as we saw in Section 3.4.1, knowing $x = \mathrm{Tr}(\rho^\alpha)$ to multiplicative precision ϵ allows us to obtain $S_\alpha(\rho)$ to additive precision ϵ.

Appendix C Using Quantum Amplitude Estimation

The central concept in using QAE to estimate any functional $\varphi(\vec{p}) := \sum_i f(p_i)$ of an input vector $\vec{p}$ (which is generally a probability mass function) is to use the input unitary that performs the map

$$U_\rho \left|0\right\rangle^{\otimes d+a} = \left|\psi_\rho\right\rangle = \sum_{i=1}^{n} \sqrt{p_i} \left|\phi_i\right\rangle_a \left|\psi_i\right\rangle_d$$

to construct a new unitary circuit which on a chosen, easy to prepare initial state performs a map of the type

$$W \left|0\right\rangle^{\otimes a} \left|\text{in}\right\rangle^{\otimes d} \left|0\right\rangle_{\text{flag}} = \left|\psi_\rho\right\rangle = \sum_{i=1}^{n} \sqrt{f(p_i)} \left|\phi_i\right\rangle_a \left|\psi_i\right\rangle_d \left|0\right\rangle_{\text{flag}} + \left|...\right\rangle \left|1\right\rangle_{\text{flag}},$$

so that value of the target functional φ is encoded in the amplitude of the part of the output state marked by the $|0\rangle$ subspace of the flag register.

It is easy and straightforward to repeat the calculations of [GL19] performed for the von Neumann entropy, to α-Renyi entropies, by simply changing the matrix function being implemented using the singular value transformation technique to ρ^α. Using the block encoding in Lemma 3.4 of the purified density operator with the result of corollary 3.6, choosing the power function with exponent $\frac{\alpha-1}{2}$, we get a unitary that on input

$$\left|\rho\right\rangle = \sum_{i=1}^{d} \sqrt{p_i} \left|\phi_i\right\rangle_a \left|\psi_i\right\rangle_d$$

performs the map

$$U \left|\rho\right\rangle \left|0\right\rangle \mapsto \sum_{i=1}^{d} \sqrt{p_i^\alpha} \left|\phi_i\right\rangle_a \left|\psi_i\right\rangle_d \left|0\right\rangle + \left|\perp\right\rangle \left|1\right\rangle,$$

where the second term is orthogonal to all states with $|0\rangle$ in the ancillary register. Now we can apply amplitude estimation to obtain an approximation to the amplitude of the ancillary register in the $|0\rangle$ state, which in fact is equal to $\text{Tr}\,(\rho^\alpha)$.

We also need to perform additional error analysis, similar to that in [GL19], since the power function is not implemented exactly but with some precision. If we implement ρ^c for $c = \frac{\alpha-1}{2}$ to precision η', the error in the trace can be bounded by $d\eta'$. Thus we would need to choose $\eta' = \eta/d$ in order to obtain some chosen constant precision $\eta \in (0, 1)$. This means QAE requires $\mathcal{O}(d/\eta)$ queries to the block encoding for ρ^c, which from corollary 3.6, requires $\mathcal{O}(\frac{1}{\delta} \log \frac{d}{\eta})$ uses of the purified access oracle U_ρ.

Appendix D von Neumann entropy

The von Neumann entropy too can be estimated in a fashion similar to that outlined for Renyi entropies. The discussion is slightly complicated by technicalities due to the non-smooth behaviour of the function $S(x) := x \log x$ at the origin $x = 0$. To mitigate this, [GL19] use a piecewise approximation to $S(x)$ on $[0, 1]$, using a lower cut-off $\delta > 0$ and bounding the error caused by using a relatively bad approximation for $S(x)$ on $[0, \delta)$.

We discuss below a few different ways to approach the problem, and the possible pitfalls in these approaches, and end by presenting the approximation used by [GL19] and some variations of it that yield good results.

Constant precision approximations

In [ZGH+18] it is shown that for the function

$$Q(s) = (1 + s) \log(1 + s) - s, \tag{3D.1}$$

the function

$$B(s) = \frac{s^2}{2(1 + \frac{s}{3})} \tag{3D.2}$$

gives a fairly tight lower bound for small values of s. Using this inequality and taking s to be $\rho - \mathbb{1}$ for some density matrix ρ, we have

$$\rho \log \rho \geq \rho - \mathbb{1} + \frac{3(\rho^2 - 2\rho + \mathbb{1})(2\mathbb{1} + \rho)^{-1}}{2}. \tag{3D.3}$$

Defining the function on the rhs as $f(\rho)$, we see that it can be implemented by LCU techniques. In particular when ρ only has small eigenvalues $0 < \lambda \ll 1$ we could approximate $(2\mathbb{1} + \rho)^{-1} \approx \frac{1}{2}(\mathbb{1} - \frac{\rho}{2})$, and the resulting cubic polynomial can be implemented with asymptotic complexity $\mathcal{O}(d^2/\varepsilon^3)$, where d is the sparsity of the matrix ρ, and ε is its smallest eigenvalue (assuming the state to be of full rank).

Thus using $f(x) = \frac{-1}{4} - \frac{7x}{8} + \frac{3x^2}{2} - \frac{3x^3}{8}$ as the approximation for $S(x)$, we may implement it in one of several ways: for example, we have the LCU unitary W that approximately implements the operator $\rho \log \rho$ using two ancillary qubits, and queries to the oracle for ρ,

$$W_{\rho \log \rho} |0\rangle |\psi\rangle = \frac{1}{\sqrt{3}} |0\rangle \rho \log \rho |\psi\rangle + |\perp\rangle, \tag{3D.4}$$

since the sum of coefficients in the LCU is $|f| = 3$ (the orthogonal junk state is not normalised here). The gate complexity for the unitary

The error function $x \log x - f(x)$ is monotonically decreasing and is hence bounded by its value at $x = 0$, which is 0.25.

Series for $x \log x$

There are various series approximations[1] we can consider for the function $g(x) := x \log x$. Estimating tail bounds will give us a way to control the error as a function of the order of truncation of the series.

Taylor series

Consider the following Taylor series about the point $x = 1$

$$x \log x = -x \sum_{k=1}^{\infty} \frac{(-1)^k (x-1)^k}{k}, \qquad |x-1| < 1. \tag{3D.5}$$

If we choose to truncate it at order n to obtain an approximating polynomial $P_n(x)$, we can write down the tail of the series as

$$|x \log x - P_n(x)| = \left| -x \sum_{k=n+1}^{\infty} \frac{(-1)^k (x-1)^k}{k} \right|, \tag{3D.6}$$

and using the fact that the infinite series $\sum_{k=1}^{\infty} \frac{(-1)^k}{k}$ sums to $\ln 2$, since we are interested in the domain $x \in (0, 1)$, we know the quantity on the rhs is at most $\left| \ln 2 - x \sum_{k=1}^{n} \frac{(-1)^k (x-1)^k}{k} \right|$. Requiring this tail to be less than some error threshold ϵ, we end up with the inversion problem.

Let $P_n(x)$ be the n-th order Taylor polynomial for $g(x)$ at $x_0 \in (0, 1)$, and pick a lower threshold $\varepsilon < x_0$. Then $\forall x \in (\varepsilon, 1)$

$$|g(x) - P_n(x)| \leq \frac{K_{n+1}}{(n+1)!} |x - x_0|^{n+1} \tag{3D.7}$$

where K_{n+1} is chosen so that

$$|g^{(n+1)}(x)| \leq K_{n+1}. \tag{3D.8}$$

Since $g^{(n+1)}(x) = (-1)^{n-1}(n-1)! x^{-n}$ for $n \geq 1$, a trivial choice of K_{n+1} is

$$K_{n+1} = \varepsilon^{-n}(n-1)!. \tag{3D.9}$$

[1] Since $S(x)$ has zeros at $x = 0, 1$, approximations of the kind $x(x-1)P_n(x)$ for which the coefficients of the polynomial are obtained by minimising an integral quadratic (least-squares type) error are found useful; for instance see Stackexchange.

Then for $x_0 < 1/2$ and $x \in [\varepsilon, 1]$, we have

$$|g(x) - P_n(x)| \leq \frac{\varepsilon^{-n}}{(n+1)n}|x - x_0|^{n+1} \tag{3D.10}$$

$$\leq \frac{\varepsilon^{-n}}{(n+1)n}. \tag{3D.11}$$

we can think of the right hand side as the error as a function of the truncation order and the threshold $\varepsilon > 0$. Denoting by $\delta(n, \varepsilon)$, note that it monotonically *increases* with n for small values of the threshold, since

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}n} \log \delta(n, \varepsilon) &= -\log \varepsilon - \frac{1}{n} - \frac{1}{n+1} \\ &\geq -\log \varepsilon - \frac{3}{2}, \end{aligned} \tag{3D.12}$$

since $n \geq 1$ and $\log \varepsilon < 0$. This means that unless $\varepsilon \geq e^{-1.5} \approx 0.223$, the truncation error *grows* with the number of terms we include in the Taylor series. In fact, even for the cases when $\varepsilon \geq 0.223$, the error initially decreases but once again starts increasing beyond a certain value of n, the truncation order that minimises the error being given by

$$n^* = -\frac{2 + b + \sqrt{4 + b^2}}{2b}, \qquad b = \log \varepsilon,$$

which takes a value of $n^* = 1$ for $\varepsilon = e^{-1.5}$, and diverges as $\frac{2}{1-x} - \frac{1}{2}$ for $x \to 1^-$.

Piecewise approximations

As we saw in the last two sections, constant error approximations give polynomials of fixed degree but no control over the precision. If we divide the domain into $[0, \varepsilon]$ and $(\varepsilon, 1]$ and use a piecewise approximation in these two intervals, better results can be obtained.

For small enough ε, a simple linear approximation $f_1(x) = -x$ can be used for $x \log x$. The error caused in the contribution to the entropy by eigenvalues of ρ in the interval $[0, \varepsilon]$ is given by

$$\left| \sum_{p_i < \varepsilon} (p_i \log p_i + p_i) \right| \leq d\varepsilon \left(\log 1/\varepsilon + 1 \right), \tag{3D.13}$$

since the function $x \log 1/x + x$ monotonically increases in the interval $(0, 1)$. Here d is the dimension of ρ. For the overall error to be bounded by some ϵ, we require that

$$\varepsilon \left(\log 1/\varepsilon + 1 \right) < \frac{\epsilon}{d}$$

Chapter 4

Quantum Algorithm for Learning with optimised Random Features

Synopsis: In this chapter, we elaborate on an accelerated framework for kernel method based machine learning that uses optimised random features. Kernel methods augmented with random features give scalable algorithms for learning from big data. But it has been found to be computationally hard to sample random features from a probability distribution that is optimised for the data, so as to minimise the required number of features for a desired learning accuracy. We develop a quantum algorithm for sampling from this optimised distribution over features, in runtime $\mathcal{O}(D)$, linear in the dimension D of the input data. Our algorithm achieves an exponential speedup in D compared to any known classical algorithm for this sampling task. In contrast to standard quantum machine learning algorithms, we do not assume sparsity or low rank of the operator being inverted, but take advantage of its sparsity in a Fourier transformed representation, in conjunction with the efficiency of the quantum Fourier transform. We also show that the sampled features can be combined with regression by stochastic gradient descent to achieve the learning without cancelling out our exponential speedup.

4.1 Introduction

Random features [RR08] provide a powerful technique for scaling up kernel methods [SS01] applicable to a variety of machine learning tasks, such as ridge regression [RR17], kernel learning [SD16], and principle component analysis [UMM+18]. Recently, Bach [Bac17] has constructed a data-optimised probability distribution of random features, sampling features from which can drastically improve the runtime of learning algorithms based on random features. However, this sampling task runs into computational difficulties, requiring the inversion of a high-dimensional operator [Bac17]. The new field of quantum machine learning (QML) [BWP+17, CHI+18, DB18] that has burgeoned over the last decade offers shoots of hope for accelerating exactly such linear algebraic problems. In this chapter, we focus on an efficient quantum algorithm for sampling from this data-optimised distribution over features.

Learning with random features

Supervised learning deals with the problem of obtaining an approximation to an unknown function $y = f(x)$ using a set of 'labelled' examples or pairs $(x_i, f(x_i))$. We will consider D-dimensional input $x \in \mathbb{R}^D$ and real-valued output $y \in \mathbb{R}$. Given N input-output pairs, we want to learn f to a desired accuracy $\epsilon > 0$. Kernel methods use the reproducing kernel Hilbert space (RKHS) associated with a symmetric, positive semidefinite function $k(x', x)$, called *the kernel*, to model the function f [SS01]. Traditional kernel methods may not be scalable as the number of input data points N gets large. The use of *random features* has emerged as a popular method for developing scalable learning algorithms based on kernel methods, along with other techniques for scaling-up via low-rank matrix approximation such as [SS00, WS01, FS02].

Algorithms using random features are based on the fact that we can represent any translation-invariant kernel $k(x, y)$ as the expectation of the product of *features* $\varphi(v, x)\varphi(v, y)$ over a probability measure $d\tau$ on the associated *feature space* $\mathcal{V}$, specially engineered for the kernel; $v \in \mathcal{V}$ here functions as a parameter. This allows for the kernel (and its feature space) to be approximated by replacing the expectation with a sample average that can be evaluated via a quadrature, using sufficiently many random samples.

Conventional algorithms using random features [RR08, RR09] sample a set of D-dimensional parameters $v_0, \ldots, v_{M-1} \in \mathbb{R}^D$ from $d\tau$, and use these to decide M features or feature vectors $\varphi(v_m, \cdot)$. These algorithms typically have a runtime scaling asymptotically as $\mathcal{O}(MD)$. For special classes of kernels such as Gaussian kernels, this runtime can be reduced to $\mathcal{O}(M \log D)$ [LSS13, YSC+16]. We *learn* the function f, or rather an approximation to it, using a linear combination of the M features, i.e.

$$f(x) \approx \sum_{m=0}^{M-1} \alpha_m \varphi(v_m, x) =: \hat{f}(x). \tag{4.1}$$

To achieve the learning to an accuracy $\mathcal{O}(\epsilon)$, we need to settle on a sufficiently large number M of features. Once we fix the M features, we obtain the coefficients α_m by applying linear (or ridge) regression to minimise an error between f and $\hat{f}$, using the N data points [RR09, RR17, CRR18]. Using the method of doubly stochastic gradients [DXH+14], the sampling of features and the regression of coefficients can be performed simultaneously.

The problem

These conventional methods for obtaining random features from the data-independent distribution defined by $d\tau$ require a large number of features to learn the function f, with M typically scaling exponentially in D. Consequently deciding on the M features and the regression over M coefficients requires long runtimes and intense computational efforts.

To better this situation, we aim to minimise M required for the learning. To do this, rather than sampling from $d\tau$, we will sample features from a probability distribution that puts greater weight on *important features optimised for the data* via a probability density function $q(v)$ for $d\tau$. Bach constructs such an optimised probability density function $q_\epsilon^*(v)$ for $d\tau$ (eq. (4.8), Section 4.4), in order to minimise M while achieving learning accuracy $\mathcal{O}(\epsilon)$ [Bac17]. Furthermore, he proves sampling from $q_\epsilon^*(v)$ achieves minimal M (up to a logarithmic gap) among all algorithms using random features, for a fixed accuracy ϵ. It *significantly improves* M compared to sampling from $d\tau$ [Bac17, RR17, SGT18] — for instance, to achieve learning with the Gaussian kernel from data given according to a sub-Gaussian distribution, compared to sampling from the data-independent $d\tau$ of [RR08, RR09], the required number M of features sampled from the optimised distribution $q_\epsilon^*(v)d\tau(v)$ can be *exponentially small* in ϵ [Bac17]. We call features sampled from $q_\epsilon^*(v)d\tau(v)$ *optimised random features*.

However, sampling from $q_\epsilon^*(v)d\tau(v)$ has been found to be "hard in practice" [Bac17] for two reasons. Firstly, its definition eq. (4.8) involves an *infinite-dimensional operator* $(\Sigma + \epsilon\mathbb{1})^{-1}$ on the space of functions $f : \mathbb{R}^D \to \mathbb{R}$ with D-dimensional input data, which is intractable to calculate by computer without approximation. Secondly, even if we approximate $\Sigma + \epsilon\mathbb{1}$ by an operator on a finite-dimensional space, the *inverse operator* approximating $(\Sigma + \epsilon\mathbb{1})^{-1}$ is still hard to calculate; in particular, for achieving a desired accuracy in the approximation, the required dimension of this finite-dimensional space can be exponentially large in D, contributing an $\mathcal{O}(\exp(D))$ factor to the runtime [SGT18, SK19]. *No known classical algorithm* can calculate the inverse of such an $\mathcal{O}(\exp(D))$-dimensional operator *within sub-exponential time* in D, unless there are sufficiently strong low rank and well-conditioned approximations available for it.

Related work: We remark that several authors have proposed probability density functions similar to $q_\epsilon^*(v)$, from which a sample can be obtained in $\mathcal{O}(\text{poly}(D))$ time [AKM+17, LTO+19, LHC+19]; however, none of these are known to minimise M. In particular, [Bac17] defines $q_\epsilon^*(v)$ using an integral operator (eq. (4.8)), and proves its otimality in minimising M using the properties of this integral operator. On the other hand, [AKM+17] and [LHC+19] use different formalisms involving Gram matrices, and the distributions they define do not achieve optimal M. Similarly, whereas sampling from an importance-weighted distribution can be used in column sampling for scaling-up kernel methods via low-rank matrix approximation [Bac13, AM15, RCC+18], such algorithms are not applicable to the setting of random features, as discussed in [Bac17]. Quasi-Monte Carlo techniques [ASY+16, CLY+17] also improve M, but they are heuristic and offer no guarantees on M.

Our solution

To overcome these difficulties, we approach the problem in the framework of quantum algorithms. We follow the obvious motivation to use the highly successful quantum linear systems algorithm which, as we have seen in Chapter 1, can be up to exponentially faster than the best known classical algorithms for this problem [HHL09]. Under the $\mathcal{O}(\exp(D))$-dimensional discretised approximation to the integral operator, Bach's classical algorithm [Bac17] requires as much as $\mathcal{O}(\exp(D))$ time for drawing a single sample from $q_\epsilon^*(v)d\tau(v)$, for D-dimensional data; we construct a quantum algorithm achieving this sampling in as fast as *linear* time $\mathcal{O}(D)$.

We thus identify the bottleneck that is faced by classical algorithms in obtaining samples from the data-optimised distribution, and resolve it by the use of a quantum algorithm. This enables us to reap the benefits of learning with a minimal number of random features – since, in contrast to sampling from a data-independent distribution $d\tau(v)$, we can use our quantum algorithm sampling from $q_\epsilon^*(v)d\tau(v)$ for learning with a nearly minimal number M of features.

4.2 Main Results

Our main contributions are threefold.

1. (Theorem 4.3) We construct a quantum algorithm for **sampling an optimised random feature from $q_\epsilon^*(v)d\tau(v)$ in as fast as linear runtime $\mathcal{O}(D)$ in the data dimension** D. Significantly, we achieve this without assuming sparsity or low rank of relevant operators.

2. To construct this quantum algorithm, we **circumvent the difficulty of infinite dimension** by formulating a discrete approximation of the problem of sampling a real-valued feature from $q_\epsilon^*(v)d\tau(v)$. This approximation is equivalent to using fixed-point number representation with rescaling.

3. (Theorem 4.4) We show that we can combine M features sampled by our algorithm with regression by stochastic gradient descent **to achieve supervised learning in time $\mathcal{O}(MD)$, i.e. *without cancelling out our exponential speedup***. This M is expected to be nearly *minimal* since we use optimised random features.

The rest of this chapter is organised as follows. Our primary result is the efficient quantum subroutine of Algorithm 4.1 for sampling an optimised random feature, in the discretised setting of Section 4.5; we also prove Theorem 4.3 bounding its runtime. As we show in Section 4.7, it is crucial for our algorithm to use the *perfect reconstruction* of the kernel, i.e., an exact representation of the kernel on the data domain as the *finite* sum of feature map φ

evaluations weighted by a function $Q^\tau(\tilde{v})$ over a *finite* set of features $\tilde{v} \in \tilde{\mathcal{V}}$ (Proposition 4.1). Like the diagonal operator $\mathbf{q}^\rho$ in Table 4.2, we will denote a diagonal operator for $Q^\tau(\tilde{v})$ as $\mathbf{Q}^\tau$, the maximum of $Q^\tau(\tilde{v})$ as $Q^\tau_{\max}$, and a probability mass function on $\tilde{\mathcal{V}}$ proportional to $Q^\tau(\tilde{v})$ as $P^\tau(\tilde{v})$. In Appendix A, we clarify how to input this representation of the kernel to our algorithm, via a quantum oracle Orc_τ. In Section 4.8, correspondingly to the optimised distribution $\tilde{q}^*_\epsilon(v)\, d\tau(v)$ of real-valued features in $\mathcal{V}$, we provide an optimised probability distribution $Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})$ of our features in the finite set $\tilde{\mathcal{V}}$, and construct a quantum state $|\Psi\rangle$ to sample the optimised random feature from $Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})$ (Proposition 4.2). In Algorithm 4.1, we efficiently prepare the state $|\Psi\rangle$, and perform a measurement on $|\Psi\rangle$ to achieve the sampling. In Section 4.10, we also show how to perform linear regression by a classical algorithm to achieve the learning without cancelling out our quantum speedup (Theorem 4.4).

4.2.1 Comparison with previous works on quantum machine learning

The novelty of our contributions lies in our QML algorithm that can be *exponentially faster* than any known classical algorithm in some parameter regimes, but is still *free from sparsity or low-rank assumptions* on the operators involved. We exploit what essentially amounts to the operator to be inverted being sparse in the Fourier basis, in order to achieve HHL type speedups for the sampling task at the centre of obtaining features of the data. The learning algorithm inherits the speedup from our algorithm for efficiently sampling the optimised random features, and requires careful handling to make sure that the classical regression using the sampled features does not dominate the runtime.

Despite several recent efforts to apply QML to kernel methods [MD19], super-polynomial speedups have been rare. Typical QML algorithms that work in the oracle model [HHL09, WBL12, LGZ16, ZFF19] achieve exponential speedups over classical algorithms only if matrices involved in the algorithms satisfy a row-sparsity and local computability assumption; in particular, $N \times N$ matrices can have only $\operatorname{poly}\log(N)$ nonzero elements in each row and column. The other popular class of QML algorithms works with data that is presented in QRAM [LMR14, RML14, KP17b, WZP18], and do not require sparsity — but may attain large speedups only if the involved matrices have low rank. Motivated by the quantum recommendations system algorithm of Kerenidis and Prakash [KP17b], Tang recently showed that using a classical analogue of the QRAM data structure, one can construct "quantum-inspired" classical algorithms, which also assume low rank and achieve exponential speedups over the previous classical state-of-the-art. These "de-quantised" algorithms are, however, slower than their quantum counterparts by very large polynomial factors. Several QRAM based quantum algorithms are now known to be de-quantisable in this manner [Tan19, JLS19, CGL$^+$19]. The framework of Quantum singular value transformations (QSVT) [GSL$^+$19] that we briefly touched upon in Chapter 1 has recently emerged as a fundamental subroutine that

can be used to implement all these different quantum algorithms, rooted in different input models, in a unified way.

However, these assumptions of sparsity or low rank certainly restrict the power and the applicability of the QML algorithms [Aar15], and we do not yet have a thorough theoretical understanding of the role played by either of these. That the computational bottlenecks posed by input matrices that have high rank or are dense can be dealt with by identifying low rank approximations or by preprocessing the input by spectral sparsification are ideas that have been explored, as we saw in Chapter 2. Here, we present another potential workaround for obtaining genuine, 'non-de-quantisable', quantum speedups for problems that do not come with guaranteed sparsity or low rank.

We work in the QRAM input model, and circumvent the sparsity and low-rank assumptions by an amalgam of the QSVT with another fundamental subroutine, the quantum Fourier transform (QFT) [HH00], broadening the applicability of our QML algorithm. While we do not present any hardness proofs for the problem we study, we make the following broad observation. QFT and QSVT serve as essential subroutines for universal quantum computation. We know that the quantum linear systems (QLS) problem, and hence the QSVT which is capable of implementing an optimal QLS algorithm, is BQP-complete[1] [HHL09]; indeed, as we shall see, an essential step in our algorithm is the inversion of the square root of the regularised integral operator. Similarly, we also know that the BQP-hardness of the local Hamiltonian eigenvalue sampling problem coupled with its efficient solution by quantum phase estimation using the QFT gives strong evidence against classical simulability of the QFT [Zha12]. These powerful subroutines thus make our algorithm hard to simulate numerically by classical computation, and hard to perform on near-term quantum computers [HCT+19, AAB+19] that cannot implement universal quantum computation due to noise. Furthermore, we speculate that this evidence for the hardness of the subroutines we use in combination with the large speedups we are able to obtain indicate that our algorithm is unlikely to be classically simulable, and hence resilient to de-quantisation by existing methods.

We venture to claim that our algorithm's potential for wide applicability, and its resilience to de-quantisation, makes it a promising candidate for "killer applications" of universal quantum computers.

4.3 Preliminaries and Notation

Before delving into the detailed description of the problem and our results, we briefly recall some preliminaries.

[1] Or more to be more precise , PromiseBQP-complete

Since this chapter has proven to be a veritable notational house of horrors, we also set up and collect some underlying notational guidelines that we shall follow, that will help the reader in deciphering some of the cranky and overworked symbols when they appear at a distance from where they were first defined. Some of the basic objects and their multitudinous discretised, rescaled, approximate, and quantised avatars are listed in Tables 4.1 and 4.2, which may be worth keeping bookmarked and ready at hand at all times.

To start with pleasantries, we denote the complex conjugation of $z \in \mathbb{C}$ by $\overline{z}$. Components of vectors such as $x \in \mathbb{R}^D$ will be denoted with a superscript, as $x = (x^1, \ldots, x^D)$; bold letters will be reserved for finite-dimensional square matrices, and will always be quantum operators of some kind.

We will overload and abuse notation for some objects.

- $\mathcal{I}$ $(= \mathcal{I}_{b,G})$ will refer to the set $\{0, \delta_b, 2\delta_b, \ldots, G - \delta_b\}$ of points that represent a real interval $[0, G]$ to b-bits of precision ($\delta_b = 2^{-b}$), but will also interchangeably be used to refer to the set $\{0, 1, \ldots, G-1\}$ with which it is in bijection under $\mathcal{I} \ni x \mapsto 2^b x$; basically, we interpret the bit strings representing the fixed-point notation as both discretised reals in $[0, G]$ and as all the integers in $[0, 2^b G]$. Furthermore, we will overload G to mean $2^b G$ when the distinction is either clear from context or unimportant.

- The kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a function of two variables, but translation invariance imbues it with the property $\forall x', x \in \mathcal{X},\ k(x', x) = k(x' - x, 0)$. We will use the same symbol k to denote the function of a single variable defined by $k(x) := k(x, 0)$.

When we wish to be painfully pedantic, the runtimes for oracles, both classical and quantum, will be denoted by T with a subscript indicating the type of data delivered by the oracle: for example $T_{\tilde{x}}$, T_y and T_ρ for the oracles in eqs. (4.18) and (4.19).

Discretisation incurs a tilde for classical objects, as in $\mathcal{X} \to \tilde{\mathcal{X}}$ and $x \to \tilde{x}$, and the quantum correspondents of operators that have been discretised to become finite dimensional matrices acquire a certain boldness, as with $k \to \mathbf{k}$ or $q^\rho \to \mathbf{q}^\rho$. Empirical approximations of these objects, whether classical or quantum, go further and wear a hat: $q^\rho \to \hat{q}^\rho$ and $\mathbf{q}^\rho \to \hat{\mathbf{q}}^\rho$.

Probability measures are functions that map from the set of power sets of a domain, to real numbers. A probability measure $d\rho$ on a space $\mathcal{X}$ defines the linear functional of integration on the space of (measurable) functions over $\mathcal{X}$, and is associated with a probability density function $q^\rho(x)$ via the relation

$$d\rho(x) = q^\rho(x) dx,$$

for $\forall\ x \in \mathcal{X}$. This essentially describes the probability distribution over points in the sample space $\mathcal{X}$, as

$$\Pr\left[y \in (x, x + dx)\right] = d\rho(x).$$

We will follow the practice illustrated here, of indicating measures with a d followed by a greek letter, and their density functions by q with the same Greek letter in superscript – in this chapter, *there will be no context wherein a Greek letter appears in the superscript to denote an exponent.* Probability density functions will be normalised against their measures by the condition

$$\int_{\mathcal{X}} q^{\rho}(x) d\rho(x) = 1. \tag{4.2}$$

Quantum computation : To apply non-unitary operators $\mathbf{A}$ in quantum computation, we use the technique of *block encodings* [GSL+19] that we'd briefly touched upon in Chapters 1 and 3. A block encoding of $\mathbf{A}$ is a unitary operator $\mathbf{U} = \begin{pmatrix} \mathbf{A} & \cdot \\ \cdot & \cdot \end{pmatrix}$ that encodes $\mathbf{A}$ in its top-left (or $|0\rangle\langle 0|$) subspace. Suppose that we apply $\mathbf{U}$ to a state $|0\rangle \otimes |\psi\rangle = \begin{pmatrix} |\psi\rangle \\ \mathbf{0} \end{pmatrix}$, where $\mathbf{0}$ is a zero column vector, and $|0\rangle \in \mathbb{C}^d$ for some d. Then, we obtain $\mathbf{U}(|0\rangle \otimes |\psi\rangle) = \sqrt{p}\,|0\rangle \otimes \frac{\mathbf{A}|\psi\rangle}{\|\mathbf{A}|\psi\rangle\|_2} + \sqrt{1-p}\,|\perp\rangle$, where $p = \|\mathbf{A}\,|\psi\rangle\|_2^2$. The state $|\perp\rangle$ satisfies $(|0\rangle\langle 0| \otimes \mathbb{1})\,|\perp\rangle = 0$ and is of no interest. We can prepare the state $\frac{\mathbf{A}|\psi\rangle}{\|\mathbf{A}|\psi\rangle\|_2}$ with high probability using this process for preparing $\mathbf{U}(|0\rangle \otimes |\psi\rangle)$ repeatedly $\mathcal{O}(1/\sqrt{p})$ times, by means of amplitude amplification.

4.4 Supervised learning by optimised random features

We now proceed to introduce the supervised learning setting which we shall work with in this chapter. We will also formulate an approximate version of it in Section 4.5. Suppose we are given N ordered pairs of data points, $(x_0, y_0), \ldots, (x_{N-1}, y_{N-1}) \in \mathcal{X} \times \mathcal{Y}$, where $y_n = f(x_n)$, $f : \mathcal{X} \to \mathcal{Y}$ is an unknown function to be learned, $\mathcal{X} = \mathbb{R}^D$ is the domain for D-dimensional input data, $\mathcal{Y} = \mathbb{R}$ is the range for output data. Each x_n is an observation of an independently and identically distributed (IID) random variable on $\mathcal{X}$ equipped with a probability measure $d\rho$.

We choose a continuous, positive definite, and translation-invariant kernel of the form $k(x', x) = k(x' - x)$. Here we overload the notation, setting $k(x) := k(x, 0)$. Any such kernel can be represented as the inner product of *features*

$$k(x', x) = \int d\tau(v) \overline{\varphi(v, x')} \varphi(v, x), \tag{4.3}$$

where $\varphi : \mathcal{V} \times \mathcal{X} \to \mathbb{C}$ is a 'feature map', which we shall choose to be

$$\varphi(v, x) = \mathrm{e}^{-2\pi \mathrm{i} v \cdot x}. \tag{4.4}$$

We think of $x \mapsto \varphi(v, x)$ as a one-dimensional random feature. The feature space $\mathcal{V} = \mathbb{R}^D$ is a parameter space equipped with a probability measure $d\tau$, and q^τ is (expressible as) the Fourier transform of k [RR08], i.e.

$$q^\tau(v) = \int_{\mathcal{X}} dx\, \mathrm{e}^{-2\pi \mathrm{i} v \cdot x}\, k(x). \tag{4.5}$$

The kernel is then normalised by requiring that

$$k(0,0) = \int_{\mathcal{V}} d\tau\,(v) = 1.$$

To specify a model of f, we use the reproducing kernel Hilbert space $\mathcal{F}$ (RKHS) associated with the kernel k. Assuming that the map $x \mapsto k(x,x)$ is integrable with respect to the measure $d\rho$, $\mathcal{F}$ becomes a subset of the set of functions that are square-integrable for $d\rho$, i.e. $L_2(d\rho)$ (we leave out the domain $\mathcal{X}$ and co-domain $\mathcal{Y}$ for brevity in this notation). We make the further technical assumption that $\mathcal{F}$ is dense in $L_2(d\rho)$, enabling us to approximate any function in $L_2(d\rho)$ by a function in $\mathcal{F}$, for any desired accuracy; this assumption can, however, be removed by careful functional analytic arguments, which it will be unneccesary for us to discuss.

Before proceeding further, we introduce a so-called integral operator, $\Sigma : L_2\,(d\rho) \to L_2\,(d\rho)$, which will be the focus of much of the analysis in this chapter and is defined by [CS02]

$$(\Sigma f)\,(x') := \int_{\mathcal{X}} d\rho(x)\, k\,(x', x)\, f(x). \tag{4.6}$$

The inner product of two square-integrable functions $f, g \in L_2(d\rho)$ is given by their standard inner product in $L_2(d\rho)$

$$\langle f \,|\, g \rangle := \int_{\mathcal{X}} d\rho(x) \overline{f(x)} g(x).$$

Inner products on $\mathcal{F} \subset L_2(d\rho)$ can then be defined using the integral operator to pull functions back to the whole of $L_2(d\rho)$: for $f, g \in \mathcal{F}$ we have

$$\langle f \,|\, g \rangle_{\mathcal{F}} := \left\langle \Sigma^{-1/2} f \,\middle|\, \Sigma^{-1/2} g \right\rangle.$$

We correspondingly have two norms, the usual one on $L_2(d\rho)$, and one on our space of interest

$$\|f\|_{\mathcal{F}}^2 := \langle f \,|\, f \rangle_{\mathcal{F}}.$$

We aim to learn an approximation of f from the given data, so that the generalisation error between f and our approximant can be bounded to a desired accuracy $\epsilon > 0$. Our

approximant is $\hat{f}$ of eq. (4.1), the linear combination over features $\varphi(v_m, \cdot)$, to which we shall append a subscript α to indicate that it is parametrised by the coefficient vector $\alpha \in \mathbb{R}^M$:

$$\hat{f}_\alpha(x) := \sum_{m=0}^{M-1} \alpha_m \varphi(v_m, x)$$

and the generalisation error is related primarily to the deviation $\left|f(x) - \hat{f}(x)\right|$ on values of x outside the input data set; in particular, we will be interested in the expectation value of this difference over $\mathcal{X}$, i.e.

$$\begin{aligned} \mathrm{err}(f, \hat{f}_\alpha) &:= \int_{\mathcal{X}} d\rho(x) \left|f(x) - \hat{f}_\alpha(x)\right|^2 \\ &= \left\|f - \hat{f}_\alpha\right\|^2 \end{aligned} \tag{4.7}$$

To achieve this learning to an accuracy $\mathcal{O}(\epsilon)$ with a minimal number M of random features, rather than sampling from $d\tau$, Bach proposes to sample features according to a probability density q_ϵ^* that is optimised for $d\tau$

$$q_\epsilon^*(v) \propto \left\langle \varphi(v, \cdot) \,\middle|\, (\Sigma + \epsilon \mathbb{1})^{-1} \varphi(v, \cdot) \right\rangle, \tag{4.8}$$

where ϵ is a regularisation parameter, $\mathbb{1}$ is the identity operator, and Σ is the integral operator that was defined above. q_ϵ^* is then normalised with respect to $d\tau$ as in eq. (4.2).

[Bac17] shows that for any f satisfying $\|f\|_{\mathcal{F}} \leq 1$, it suffices to sample a nearly optimal number scaling as

$$M = \mathcal{O}\left(\mathrm{d}(\epsilon) \log\left(\frac{\mathrm{d}(\epsilon)}{\delta}\right)\right) \tag{4.9}$$

of features from $q_\epsilon^*(v) d\tau(v)$ to achieve, for any $\delta > 0$

$$\Pr\left[\min_\alpha \{\mathrm{err}(f, \hat{f}_\alpha)\} \leq 4\epsilon\right] > 1 - \delta \tag{4.10}$$

where

$$\mathrm{d}(\epsilon) := \mathrm{Tr}\left(\Sigma(\Sigma + \epsilon \mathbb{1})^{-1}\right) \tag{4.11}$$

is the known as the 'degree of freedom', and represents the effective dimension of the data seen as a submanifold of the whole space.

Our interest in this chapter is to sample these optimised random features according to the density q_ϵ^*. Computationally speaking, we will only ever be able to sample from a distribution that is *close* to $q_\epsilon^*(v) d\tau(v)$, and we shall refer to these as our optimised random features.

Everything we have seen so far is in the setting of Banach spaces and continuous variables. Let us now see how to deconstruct these into simpler objects for computational purposes.

4.5 Discretised setting for random feature sampling

Since we work with digital (quantum) computers, it becomes necessary to think about how to represent and use continuous real valued data. Delicate properties of the infinite-dimensional operators on Banach spaces that are used in kernel methods can nevertheless be suitably retained in their discretised forms, without compromising the provable guarantees of the learning algorithm. Discretisation involves standard techniques of fixed point number representation, but introduces an overhead in precision, and raises questions of convergence. We thus devote some space and discussion to these issues in this section.

Throughout the rest of this chapter, we will assume that the input data comes from a bounded domain; in particular, $\mathrm{supp}\,(d\rho(x)) \subseteq [0, x_{\max}]^D$ for some $0 < x_{\max} < \infty$. Thus we expect no data points to lie outside this D-dimensional hypercube of side length $x_{\max}$.

To learn from such an input distribution, we may without loss of generality pick a kernel that is zero (or rapidly decaying) outside the same hypercube. Assuming $k(x', x) \to 0$ sufficiently fast when $|x - x'|$ deviates from zero (a property enjoyed, for example, by Gaussian kernels), we fix a known $G \gg x_{\max}$, and define a periodic function $\tilde{k}$ that approximates k well $\forall\ x', x \in [0, x_{\max}]^D$ by

$$\begin{aligned} \tilde{k}(x', x) &:= \sum_{n \in \mathbb{Z}^D} k(x', x + Gn) \\ &\approx k(x', x). \end{aligned} \tag{4.12}$$

Notice that $\tilde{k}$ is also translation invariant. We shall restrict our attention to the interval $[0, G]$ and use $\tilde{k}$ as the kernel in place of k.

While at first sight, $\tilde{k}$ appears to be more expensive to compute, we shall see in Section 4.7 that it is the addition of this periodicity to shift invariant kernels that will enable us to diagonalise the matrix of the kernel via the Discrete (and hence Quantum) Fourier Transform, by exploiting its circulant form[2].

4.5.1 Discretising continuous data

As is standard in digital computation, we represent real numbers approximately, with a finite number of bits. Such fixed-point number representation to b bits has precision $\delta_b := 2^{-b} > 0$, and uses a finite set $\mathcal{I} := \{0, \delta_b, 2\delta_b, \ldots, G - \delta_b\}$ to represent the real interval $[0, G]$.

The data domain $\mathcal{X} = \mathbb{R}^D$, an infinite set, is then represented by the finite grid $\tilde{\mathcal{X}} = \mathcal{I}^D$. We quotient out real-valued points $x \in \mathcal{X}$ by the equivalence relation of being closest to a

[2] A circulant matrix is a square matrix in which each row is shifted by one element to the right relative to the preceding row.

unique grid point $\tilde{x} \in \tilde{\mathcal{X}}$, which will subsequently represent x. It will not be necessary for us to discretise the data range $\mathcal{Y}$, since none of the operations we study use values in $\mathcal{Y}$.

Vectors in $\tilde{\mathcal{X}}$ will become states of D quantum registers $\mathcal{H}_{\mathcal{I}}$ of $\lceil \log_2 G \rceil$ qubits each

$$\begin{aligned}\mathcal{H}^X &:= \operatorname{span}\{|\tilde{x}\rangle : \tilde{x} \in \tilde{\mathcal{X}}\} \\ &= (\mathcal{H}_{\mathcal{I}})^{\otimes D}.\end{aligned}$$

Each sub-register $\mathcal{H}_{\mathcal{I}}$ corresponds to a copy of $\mathcal{I}$. Classical data is thus stored in the 'ket labels' of quantum states (as opposed to being encoded in the amplitudes); the correspondence between classical vectors and quantum states is

$$\tilde{x} = (\tilde{x}^1, \ldots, \tilde{x}^D)^{\mathrm{T}} \in \tilde{\mathcal{X}} \qquad \longleftrightarrow \qquad |\tilde{x}\rangle^X = \bigotimes_{i=1}^{D} \left|\tilde{x}^i\right\rangle \in \mathcal{H}^X$$

where each $|\tilde{x}^i\rangle \in \mathcal{H}_{\mathcal{I}}$.

Accordingly, functions on the continuous space $\mathcal{X}$ become vectors on the finite-dimensional $\mathcal{H}^X$, and operators acting on functions on $\mathcal{X}$ become matrices that act on $\mathcal{H}^X$ (cf. Table 4.1). Inner products in the Banach space of functions over $\mathcal{X}$ are approximated by quadratures given by the inner products of the corresponding vectors in the Hilbert space $\mathcal{H}^X$:

$$\int_{\mathcal{X}} d\rho(x)\, \overline{f(x)} g(x) \approx \langle f \,|\, \mathbf{q}^{\rho} \,|\, g \rangle \,. \tag{4.13}$$

The diagonal operator $\mathbf{q}^{\rho}$ is now exactly the kind of "classical state" encoding a classical probability mass function on the diagonal that we touched upon in the introduction to Chapter 3.

With this scheme in place, we have a discretised version of the optimised probability density function q_ϵ^* (cf. eq. (4.8))

$$\tilde{q}_\epsilon^*(v) \propto \left\langle \varphi(v,\cdot) \,\middle|\, \mathbf{q}^{\rho}(\boldsymbol{\Sigma} + \epsilon \mathbb{1})^{-1} \,\middle|\, \varphi(v,\cdot) \right\rangle. \tag{4.14}$$

Strictly speaking, we will also be discretising the feature space, so that this $\tilde{q}_\epsilon^*$ will become a probability mass function rather than a density function, and is then (exactly) normalised by the corresponding quadrature for $\int_{\mathcal{V}} \tilde{q}_\epsilon^*(v)\, d\tau(v) = 1$.

4.5.2 Discretised representation of our inputs

Real-valued input data points $x \in \mathcal{X}$ sampled according to $d\rho$ get replaced by the closest grid point $\tilde{x} \in \tilde{\mathcal{X}}$. Each $\tilde{x}$ occurs with a probability $\int_{\Delta_{\tilde{x}}} d\rho(x)$, where $\Delta_{\tilde{x}}$ is the D-dimensional hypercube of side δ_b centred at $\tilde{x}$.

Function / operator on $\mathcal{X}$	Vector / operator on $\mathcal{H}^X$
$f : \mathcal{X} \to \mathbb{C}$	$\lvert f\rangle \propto \sum_{\tilde{x}} f(\tilde{x}) \lvert\tilde{x}\rangle$
$\varphi(v, \cdot) : \mathcal{X} \to \mathbb{C}$	$\lvert\varphi(v,\cdot)\rangle \propto \sum_{\tilde{x}} \varphi(v, \tilde{x}) \lvert\tilde{x}\rangle$
$\tilde{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	$\mathbf{k} := \sum_{\tilde{x}', \tilde{x}} \tilde{k}(\tilde{x}', \tilde{x}) \lvert\tilde{x}'\rangle\langle\tilde{x}\rvert$
$q^\rho : \mathcal{X} \to \mathbb{R}$	$\mathbf{q}^\rho := \sum_{\tilde{x}} q^\rho(\tilde{x}) \lvert\tilde{x}\rangle\langle\tilde{x}\rvert$
Σ acting on $f : \mathcal{X} \to \mathbb{C}$	$\mathbf{\Sigma} := \mathbf{k}\mathbf{q}^\rho$
$\Sigma f : \mathcal{X} \to \mathbb{C}$	$\mathbf{\Sigma} \lvert f\rangle$

Table 4.1: Discretised representation of $\mathcal{X}$ by $\mathcal{H}^X$. Note $\tilde{x}', \tilde{x} \in \tilde{\mathcal{X}}$. Note that normalisation constant is $\|f\|$ for $\lvert f\rangle$, and $\sqrt{G^D}$ for $\lvert\varphi(v,\cdot)\rangle$.

The true probability distribution of the data is unknown in our setting. In its place, we use the N given data points to calculate an empirical approximation to $d\rho(x) = q^\rho(x)dx$. For any $\tilde{x} \in \tilde{\mathcal{X}}$, let $n(\tilde{x})$ denote the number of input data points that fall within the hypercube $\Delta_{\tilde{x}}$. We approximate the distribution $d\rho(x)$ for all $x \in \Delta_{\tilde{x}}$ by the empirical frequencies of the data points, i.e.

$$\hat{q}^\rho(\tilde{x}) := \frac{n(\tilde{x})}{N}. \tag{4.15}$$

Just as we had with the discretised $\mathbf{\Sigma} = \mathbf{k}\mathbf{q}^\rho$ of Table 4.1, we now have hat-wearing *empirical* integral and probability density operators, given by

$$\hat{\mathbf{\Sigma}} := \mathbf{k}\hat{\mathbf{q}}^\rho, \quad \hat{\mathbf{q}}^\rho := \sum_{\tilde{x} \in \tilde{\mathcal{X}}} \hat{q}^\rho(\tilde{x}) \lvert\tilde{x}\rangle\langle\tilde{x}\rvert . \tag{4.16}$$

We aim to analyse the asymptotic runtime of our algorithm when the number N of data points is large. For $N \to \infty$, statistical errors in the empirical distribution caused by the finiteness of N vanish; detailed analyses of statistical errors for finite N can be found in [Bac17].

4.5.3 Why does this discretisation work?

To justify our discretisation scheme, we assume that functions involved in the learning, such as the target f and density $q^\rho(x)$ of data, are L-Lipschitz continuous[3] for some Lipschitz constant L. This is generally considered a mild assumption in the signal processing literature, and is for instance satisfied when there is a finite upper bound on the high frequency domain of the Fourier transform of the target function (i.e. a frequency cut-off).

[3] A function $h : \mathcal{X} \to \mathbb{C}$ is said to be L-Lipschitz continuous if $\forall\, x, x' \in \mathcal{X}$, we have $|h(x) - h(x')| \leq L \|x - x'\|_2$.

Given this Lipschitz continuity, we have, for any relevant function h

$$\begin{aligned} |h(x) - h(\tilde{x})| &= \mathcal{O}\left(L \left\|x - \tilde{x}\right\|_2\right) \\ &= \mathcal{O}\left(L\delta_b\sqrt{D}\right), \end{aligned} \tag{4.17}$$

since x lies in the D-dimensional hypercube of side δ_b centred at $\tilde{x}$. Errors caused by discretisation, such as $|f(x) - f(\tilde{x})|$ and $|q^\rho(x) - q^\rho(\tilde{x})|$, hence become negligible if $L\delta_b\sqrt{D} \to 0$ as the data dimension D gets large.

These errors can then be made as small as desired by having $L\delta_b$ decay faster than $1/\sqrt{D}$, i.e. $L\delta_b = \mathrm{o}(D^{-1/2})$. It is clear that L should be beyond our control, depending on the input data and model. Thus we can choose $\delta_b = 2^{-b} = \mathrm{o}(D^{-1/2})$ by increasing the number of bits of precision for the fixed-point representation. In particular we need to choose $b = \Omega(\log D)$ bits, which is a reassuringly familiar conclusion.

It turns out that we can also control L in a sense, by defining rescaled versions the data and relevant functions on a larger domain. In particular, to achieve $L = \mathrm{o}(D^{-1/2})$, we need to rescale G to $G_r = \Omega(L\sqrt{D})$, thereby *artificially* expanding the domain interval $[0, G]$ (i.e. without changing the input data, but by simply zooming in on it, so to speak). This rescaling can be done such as to leave both the model $\hat{f}$ and the learning accuracy *invariant*. Table 4.2 summarises how different quantities and objects behave under this rescaling scheme.

That increasing the precision of fixed point representations and allowing larger cutoff thresholds for input functions and data both lead to suppression of discretisation errors in essentially the same fashion does not come as a surprise. Our goal was simply to illustrate that these two slightly different flavoured techniques are available, and touch upon how they work.

Original data and functions	Rescaled by $r > 1$
Inputs x	rx
Upper bound G of interval $[0, G]$	$G_r = rG$
Kernel $k(x', x)$	$k_r(rx', rx) := k(x', x)$
Target $y = f(x)$ to be learned	$f_r(rx) := f(x)$
Density of data $q^\rho(x)$	$q^\rho_r(rx) := q^\rho(x)/r$
Lipschitz constant L of $f(x)$	L/r
Lipschitz constant L of $q^\rho(x)$	L/r^2

Table 4.2: Rescaling data by a parameter $r > 1$.

How rescaling changes Lipschitz constants

Fix a parameter $r > 1$ and set $\xi = rx$. For probability density functions $q(x)$ on the interval $[0, G]$, consider $q_r(x)$ defined by

$$q_r(\xi) = \frac{1}{r}\, q\left(\frac{\xi}{r}\right),$$

and for all other functions h, define h_r by

$$h_r(\xi) = h\left(\frac{\xi}{r}\right).$$

Given that the original functions are L-Lipschitz continuous, it is easy to check that q_r and f_r have Lipschitz constants $L/r^2 < L/r$ and L/r respectively; at the same time, their domain[4] is scaled up from $[0, G]$ to $[0, rG]$. It is clear that $r = \Omega(L\sqrt{D})$ ensures $\lim_{D\to\infty} L_r\sqrt{D} = 0$. Evidently, we can replace all the functions we are interested in by the rescaled versions defined as above, and then error of discretisation will vanish, just as if we had made the functions "more continuous" by making their Lipschitz constants smaller!

This scheme of rescaling for probability densities and functions is designed to map these objects isometrically from $L_2(d\rho, [0, G])$ to $L_2(d\rho_r, [0, G_r])$, since it preserves inner products:

$$\begin{aligned}
\langle f_r \,|\, g_r \rangle &= \int_{[0,G_r]} d\rho_r(\xi)\ \overline{f_r(\xi)} g_r(\xi) \\
&= \int_{[0,G_r]} d\xi\ q_r^\rho\,(\xi)\, \overline{f_r(\xi)} g_r(\xi) \\
&= \int_{[0,G]} d(rx)\ \frac{1}{r} q^\rho(x) \overline{f(x)} g(x) \\
&= \langle f \,|\, g \rangle\,,
\end{aligned}$$

where $d\rho_r(\xi) = q_r^\rho\,(\xi)\, d\xi$ is the probability measure obtained from rescaling $d\rho$.

We thus focus on asymptotic runtime analysis of our algorithm for $G_r = \Omega(\sqrt{D})$, to suppress the discretisation error. With this understood, we will omit the subscript and write G interchangeably for G_r. The error due to discretisation using a finite and bounded G is well studied in the classical literature, wherein it is often called 'quantisation': for example, [Pro07] analyses this error using established procedures in signal processing.

[4] Note that the range of q remains the set of positive reals, and we also do not touch the output labels $y = f(x) = f_r(rx)$.

4.6 Input model

The time required for accessing data is an important consideration in machine learning, where large amounts of data are concerned. This crucially depends on the computational architecture and data structures used. Turing machines rooted in theory can only access data sequentially, while modern computers actually have random access memory (RAM). We shall assume the availability of classical random access query oracles

$$\mathrm{Orc}_{\tilde{x}}(n) = \tilde{x}_n, \quad \mathrm{Orc}_y(n) = y_n, \tag{4.18}$$

that map addresses or indices $n \in \{0, \ldots, N-1\}$ to the corresponding data point. For all practical purposes, the runtime per query is $\mathcal{O}(1)$ and does not depend on the data.

Here we have another opportunity to clarify the setting for kernel-based learning via feature expansions that we summarised in Section 4.5: the input that such algorithms require is N points $x \in \mathcal{X}$ sampled according to q^ρ, along with their function evaluations $f(x)$. It is important to note here that the function values or labels, y, only come into play *after* the random feature expansion has been decided upon. In sampling good features only the points x have a role to play.

Analogous to this ability to request samples $\tilde{x}$ with probability $\hat{q}(\tilde{x})$ (after discretisation), we allow a quantum computer to use a unitary oracle Orc_ρ to initialise a quantum register to the state

$$\begin{aligned} \mathrm{Orc}_\rho |0\rangle &= \sum_{\tilde{x} \in \tilde{\mathcal{X}}} \sqrt{\hat{q}^\rho(\tilde{x})} \, |\tilde{x}\rangle \\ &= \sqrt{\hat{\mathbf{q}}^\rho} \left(\sum_{\tilde{x} \in \tilde{\mathcal{X}}} |\tilde{x}\rangle \right), \end{aligned} \tag{4.19}$$

where the second line indicates that this state can be considered as the square root of the (positive) operator $\hat{\mathbf{q}}^\rho$ applied to an unnormalised uniform superposition over all grid points in $\tilde{\mathcal{X}}$.

Access to this state allows us to sample $\tilde{x}$ with probability $\hat{q}(\tilde{x})$ by a measurement on this state in the computational basis $\{|\tilde{x}\rangle\}$, but also potentially do much more since we have these samples in superposition. At first sight, what this oracle provides may look similar to "quantum examples" that were introduced by Bshouty and Jackson [BJ95] and are commonly considered in the quantum statistical learning literature [AdW17]; however, these are not quantum examples in that sense. Rather, all we need here is the ability to request samples from the input domain $\mathcal{X}$ of the target function f, and the corresponding quantum state does not carry information about $f(x)$.

The implementation of the oracle Orc_ρ incurs a preprocessing overhead. From the N input data points, [KP17b] build a binary tree that stores the signed inputs in its leaves, and the squared sums of its children in non-leaf nodes. This results in the root note containing the 2-norm of the data vector. This classical data structure can be initialised in $\mathcal{O}(N(D\log G)^2)$ time using $\mathcal{O}(N(D\log G)^2)$ bits of memory, and can be updated efficiently with incoming data. It has a RAM-like $\mathcal{O}(1)$ lookup time for individual data points. Loosely speaking, QRAM [GLM08a, GLM08b] uses this data structure along with state preparation subroutines to implement the unitary oracle Orc_ρ that can initialise a register in a superposition whose amplitudes encode the input data. Querying the underlying binary tree, Orc_ρ can be implemented to precision δ with a gate complexity that scales as

$$T_\rho = \widetilde{O}\left(TD\log G\right)$$

using the basic method of preparing states using controlled rotations [GR02], hiding a $\mathcal{O}(\mathrm{poly}\log\frac{1}{\delta})$ overhead. We do not include the time for collecting the data or preparing the above data structure in runtime of our learning algorithm. The use of QRAM is fairly in QML, seeing as there are no other real alternatives for dealing with big data. Even with a data structure as powerful as QRAM, achieving quantum speedups has proven to be nontrivial. The physical realisation of QRAM is an active area of research [JPC+19, HZZ+19].

The runtime T_ρ of the data access oracle is independent of N since we keep track of frequencies rather than individual points for our quantum algorithm, and this quantity, rather than N explicitly, will appear in the runtime of our algorithm.

4.7 Perfect reconstruction of kernel

One of our technical contributions, and a key ingredient that makes our quantum algorithm possible, is the *perfect reconstruction* of the kernel from its Fourier modes. Let us now look at this in more detail.

We mentioned in passing in Section 4.4 that our kernel k can be expressed in the form of eq. (4.3), namely

$$k(x',x) = \int d\tau(v)\overline{\varphi(v,x')}\varphi(v,x).$$

The right hand side in this equation is an expectation of the inner product $\langle\varphi(\cdot,x')\,|\,\varphi(\cdot,x)\rangle$, over the measure $d\tau$ defined on the feature space $\mathcal{V}$. For the discretised kernel $\tilde{k}$, we can go one step further and exactly represent it in the form

$$\tilde{k}(x',x) = \sum_{v\in\mathbb{Z}^D} q^\tau(v)\overline{\varphi(v,x')}\varphi(v,x), \tag{4.20}$$

using Shannon's sampling theorem [Sha49]. Recalling eq. (4.4), where we defined $\varphi(v,x) = \mathrm{e}^{-2\pi \mathrm{i} v\cdot x}$, we see that this is actually a weighted Fourier series

$$\tilde{k}(x',x) = \sum_{v\in\mathbb{Z}^D} q^\tau(v)\ \mathrm{e}^{-2\pi\mathrm{i}v\cdot(x-x')}.$$

Moreover, on our bounded data domain $[0,G]$, it suffices to sum over a *finite* set of features

$$\tilde{\mathcal{V}} := \left\{0, \frac{1}{G}, \ldots, 1-\frac{1}{G}\right\}^D, \tag{4.21}$$

along with a function Q^τ over this set, designed to accumulate the weight q^τ puts on grid points originally outside $\tilde{\mathcal{V}}$

$$Q^\tau(\tilde{v}) := \sum_{v\in\mathbb{Z}^D} q^\tau(\tilde{v}+v). \tag{4.22}$$

For many density functions q^τ of interest, Q^τ can be calculated analytically in closed form; we present a couple of examples in Table 4.3. As usual, we have the diagonal operator

$$\mathbf{Q}^\tau := \sum_{\tilde{x}\in\tilde{\mathcal{X}}} Q^\tau(\tilde{x}/G)\,|\tilde{x}\rangle\langle\tilde{x}|\,, \tag{4.23}$$

where we use the fact that $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{X}} = \{0,1,\ldots,G-1\}^D$ are in one-to-one correspondence under the bijection $\tilde{x} = G\tilde{v}$, so as to interchangeably interpret the ket labels as $\tilde{v}\in\tilde{\mathcal{V}}$ or the corresponding $\tilde{x} = G\tilde{v}\in\tilde{\mathcal{X}}$. Putting these ingredients together, we have the following proposition.

Proposition 4.1 (Perfect reconstruction of kernel)**.** *For any periodic, translation-invariant, symmetric, positive definite kernel $\tilde{k}$ as defined in eq. (4.12), we have $\forall\ \tilde{x}',\tilde{x}\in\tilde{\mathcal{X}}$*

$$\begin{aligned}\tilde{k}\left(\tilde{x}',\tilde{x}\right) &= \frac{1}{G^D}\sum_{\tilde{v}\in\tilde{\mathcal{V}}} Q^\tau(\tilde{v})\overline{\varphi(\tilde{v},\tilde{x}')}\varphi(\tilde{v},\tilde{x}) \\ &= \left\langle \tilde{x}' \middle| \mathbf{F}_D^\dagger \mathbf{Q}^\tau \mathbf{F}_D \middle| \tilde{x}\right\rangle.\end{aligned}$$

$\mathbf{F}$ here is the one-dimensional Discrete Fourier Transform (DFT) on $\mathbb{Z}_G$, which is unitary and hence the same as the QFT), with the action

$$\mathbf{F}\,|\tilde{x}\rangle = \frac{1}{\sqrt{G}}\sum_{\tilde{x}'=0}^{G-1} \mathrm{e}^{-2\pi\mathrm{i}\tilde{x}'\tilde{x}/G}\,|\tilde{x}'\rangle\,, \tag{4.24}$$

where it will be convenient for us to relabel the kets by $\tilde{v} = \tilde{x}'/G$, enabling the rewrite

$$\mathbf{F}\,|\tilde{x}\rangle = \frac{1}{\sqrt{G}}\sum_{\tilde{v}\in\{0,1/G,\ldots,1-1/G\}} \mathrm{e}^{-2\pi\mathrm{i}\tilde{v}\tilde{x}}\,|\tilde{v}\rangle\,. \tag{4.25}$$

$\mathbf{F}_D := \mathbf{F}^{\otimes D}$ is then the D-dimensional DFT between the spaces $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{V}}$.

Note that since $\tilde{k}$ is a real-valued symmetric function, i.e., $\tilde{k}(x', x) = \tilde{k}(x, x')$ and $\overline{\tilde{k}(x', x)} = \tilde{k}(x', x)$, we also have

$$\left\langle \tilde{x}' \middle| \mathbf{F}_D^\dagger \mathbf{Q}^\tau \mathbf{F}_D \middle| \tilde{x} \right\rangle = \left\langle \tilde{x}' \middle| \mathbf{F}_D \mathbf{Q}^\tau \mathbf{F}_D^\dagger \middle| \tilde{x} \right\rangle .$$

We delegate the proof of this claim to Appendix A.

Rahimi and Recht discuss a method of sampling random sinusoids from the Fourier transform kernel function we seek to approximate, and show its utility for interpolation tasks [RR08]. Along the same lines, if we could sample M features from the probability mass function

$$P^\tau(\tilde{v}) := \frac{Q^\tau(\tilde{v})}{\sum_{\tilde{v}' \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v}')}, \tag{4.26}$$

corresponding to $d\tau$, we could combine them with the DFT to learn a target f with kernel $\tilde{k}(\tilde{x}', \tilde{x})$, for some sufficiently large M. But $P^\tau(\tilde{v})$ is not optimised for the data.

4.8 Quantum state of optimised random features

So instead, we construct a density function $Q^*_\epsilon(\tilde{v})$ that optimally weights the probability $P^\tau(\tilde{v})$ on the finite set $\tilde{\mathcal{V}}$ of features, and corresponds to the optimised density $\tilde{q}^*_\epsilon(v)$ for $d\tau$ on the set $\mathcal{V}$ of real-valued features. In place of $\tilde{q}^*_\epsilon$ of eq. (4.14), we define

$$Q^*_\epsilon(\tilde{v}) \propto \left\langle \varphi(\tilde{v}, \cdot) \middle| \hat{\mathbf{q}}^\rho (\hat{\mathbf{\Sigma}} + \epsilon \mathbb{1})^{-1} \middle| \varphi(\tilde{v}, \cdot) \right\rangle, \tag{4.27}$$

and normalise the probability distribution $Q^*_\epsilon(\tilde{v}) P^\tau(\tilde{v})$ by

$$\sum_{\tilde{v} \in \tilde{\mathcal{V}}} Q^*_\epsilon(\tilde{v}) P^\tau(\tilde{v}) = 1.$$

Recall that $\hat{\mathbf{\Sigma}} = \mathbf{k}\hat{\mathbf{q}}$ is the empirical integral operator (eq. (4.16)). In order to describe a quantum state that can be used for sampling from $Q^*_\epsilon(\tilde{v}) P^\tau(\tilde{v})$, we pad $\hat{\mathbf{\Sigma}}$ with a regularisation parameter ϵ and define a symmetrised, full-rank, positive semidefinite operator (cf. eqs. (4.8) and (4.14))

$$\hat{\mathbf{\Sigma}}_\epsilon := \frac{1}{Q^\tau_{\max}} \left(\sqrt{\hat{\mathbf{q}}^\rho} \mathbf{k} \sqrt{\hat{\mathbf{q}}^\rho} + \epsilon \mathbb{1} \right), \tag{4.28}$$

where we assume the maximum value of Q^τ is efficiently precomputed and available[5],

$$Q^\tau_{\max} := \max \{ Q^\tau(\tilde{v}) : \tilde{v} \in \tilde{\mathcal{V}} \}. \tag{4.29}$$

[5]For more on this point, please see Appendix A

For convenience, we also set up notation for a normalised version of the Q^τ operator (eq. (4.23))

$$\underline{\mathbf{Q}}^\tau := \frac{1}{Q^\tau_{\max}} \mathbf{Q}^\tau. \tag{4.30}$$

We can now write down the quantum state that Algorithm 4.1 will use for sampling from $Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})$.

Proposition 4.2 (Quantum state for sampling an optimised random feature)**.** *Performing a measurement in the computational basis* $\{|\tilde{v}\rangle^{X'} : \tilde{v} \in \tilde{\mathcal{V}}\}$ *on the second register of the bipartite quantum state*

$$|\Psi\rangle \propto \sum_{\tilde{v}\in\tilde{\mathcal{X}}} \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} |\tilde{v}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}}^\tau} \mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} |\tilde{v}\rangle^{X'} \tag{4.31}$$

defined on two registers X *and* X' *of* $\lceil D \log G \rceil$ *qubits each produces the outcome* $\tilde{v}$ *with probability*

$$\Pr(\tilde{v}) = Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v}). \tag{4.32}$$

Samples $\tilde{v}$ obtained by measuring this state are therefore optimised random features. Note that as mentioned before, we interchangeably interpret $|\tilde{v}\rangle$ as the corresponding $|\tilde{x}\rangle$.

Proving that this state does what we profess it to do involves only a straightforward linear algebraic calculation, which we relegate to Appendix B.

4.9 Quantum algorithm for sampling optimised random features

The route is now clear to present our main subroutine — Algorithm 4.1 prepares, with high probability, a copy of the state $|\Psi\rangle$ presented in Proposition 4.2, and measures it to obtain an optimised random feature $\tilde{v} \in \tilde{\mathcal{V}}$ sampled from a probability distribution $Q\,(\tilde{v})\,P^\tau(\tilde{v})$ that is within total variation distance Δ of the true optimised one:

$$\sum_{\tilde{v}\in\tilde{\mathcal{V}}} |Q(\tilde{v})P^\tau(\tilde{v}) - Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})| \leq \Delta$$

We present the algorithm below in pseudocode, and the following discussion will lead us on towards bounding the asymptotic runtime (measured by a combination of circuit size and number of QRAM/oracle invocations).

As we discussed in detail in the introduction to this chapter, sampling from $Q^*_\epsilon P^\tau$ poses a bottleneck for classical algorithms, arising from the need to invert the G^D-dimensional operator $\hat{\mathbf{\Sigma}}_\epsilon$; furthermore, $\hat{\mathbf{\Sigma}}_\epsilon$ being regularised has full rank *by design*, and also may not

Algorithm 4.1 Quantum algorithm for sampling an optimised random feature.

Require: Learning accuracy, sampling precision, quantum oracles of eq. (4.19) & eq. (4A.14)

function QUOPTRF(ϵ, Δ, Orc_ρ, Orc_τ, $Q^\tau_{\max}$)

$|0\rangle^X \otimes |0\rangle^{X'} \longmapsto \sum_{\tilde{v}} |\tilde{v}\rangle^X \otimes \sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{v}\rangle^{X'}$ ▷ Initialise and load data

$\ldots \xrightarrow{\mathbb{1}\otimes\left(\mathbf{F}_D^\dagger\right)^{X'}} \sum_{\tilde{v}} |\tilde{v}\rangle^X \otimes \mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{v}\rangle^{X'}$ ▷ Inverse QFT

$\ldots \xrightarrow{\mathbb{1}\otimes\left(\sqrt{\underline{\mathbf{Q}}^\tau}\right)^{X'}} \sum_{\tilde{v}} |\tilde{v}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}}^\tau}\mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{v}\rangle^{X'}$ ▷ Block enc. + Ampl. Amp.

$\ldots \xrightarrow{\left(\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}}\right)^X \otimes \mathbb{1}} |\Psi\rangle$ ▷ Block enc. + Ampl. Amp

$\tilde{v} \leftarrow \textsc{measure}\left(|\Psi\rangle, X'\right)$. ▷ $\Pr(\tilde{v}) = Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})$

Return $\tilde{v}$.

end function

be sparse. Recent techniques for developing "quantum-inspired" classical algorithms [Tan19, JLS19, CGL+19] are not applicable either, since they make use of low-rank approximations.

But having defined the quantum state $|\Psi\rangle$, we can immediately see that given an efficient implementation of a block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$, the techniques for implementing matrix functions that we saw in Chapter 1 come to our aid once again. In particular, the quantum singular value transformation (QSVT) method [GSL+19] gives the most efficient way in the literature for implement a block encoding of $\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}}$. However, it turns out that constructing an efficient block encoding for $\hat{\mathbf{\Sigma}}_\epsilon$ is by no means straightforward, as long as we use conventional methods that take advantage of sparsity, row-computability, or low-rank.

One of our technical contribution is to overcome this difficulty by constructing an efficient block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$ using Quantum Fourier Transform (QFT). The perfect reconstruction of the kernel described in Proposition 4.1 allows us to explicitly decompose $\hat{\mathbf{\Sigma}}_\epsilon$ into a product of simpler building blocks: diagonal operators $\sqrt{\underline{\mathbf{Q}}^\tau}$ and $\sqrt{\hat{\mathbf{q}}^\rho}$ (efficiently implementable by block encodings), and the unitary operator $\mathbf{F}_D$ (and $\mathbf{F}_D^\dagger$) representing the QFT (DFT) on $(\mathbb{Z}_G)^{\times D}$ (i.e. $\lceil D \log G \rceil$ qubits).

The discovery that the DFT on several groups of interest can be implemented with a quantum circuit of size that scales logarithmically in the size of the group led to the development of Shor's algorithm, and a flurry of related results for the Hidden Subgroup Problem. $\mathbf{F}_D$ (and $\mathbf{F}_D^\dagger$) can be implemented to precision δ' by a quantum circuit with a gate complexity that scales as [HH00]

$$\mathcal{O}\left(D \log G \cdot \log\log G \cdot \operatorname{poly}\log \frac{1}{\delta'}\right).$$

The QSVT can be applied to construct a block encoding of $\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}}$ with precision δ, using the block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$

$$\text{reps} = \widetilde{O}\left(\frac{Q^\tau_{\max}}{\epsilon}\text{poly}\log\frac{1}{\delta}\right)$$

times, where $\frac{Q^\tau_{\max}}{\epsilon}$ is within a constant factor of the condition number of $\hat{\mathbf{\Sigma}}_\epsilon$ [GSL+19]. Combining these ingredients lets us go beyond all known classical algorithms, and achieve this sampling with complexity sub-exponential, in fact linear, in the data dimension D.

Theorem 4.3 (Complexity of quOptRF)**.** *Given D-dimensional data discretised by $G > 0$, for any accuracy $\epsilon > 0$ and any sampling precision $\delta > 0$, Algorithm 4.1 samples a feature $\tilde{v} \in \tilde{\mathcal{V}}$ from a weighted distribution QP^τ that is δ-close to the true optimised distribution in total variation distance, i.e.*

$$\sum_{\tilde{v}\in\tilde{\mathcal{V}}} |(Q(\tilde{v}) - Q^*_\epsilon(\tilde{v}))P^\tau(\tilde{v})| \leq \delta$$

, with a net runtime that scales as

$$T = \widetilde{\mathcal{O}}\left(D \cdot \log G \cdot \frac{Q^\tau_{\max}}{\epsilon} poly\log\frac{1}{\Delta}\right) \times$$

where $Q^\tau_{\max}$ is defined as in eq. (4.29).

Recall that we choose G to scale polynomially in D for convergence (Section 4.5.3). $Q^\tau_{\max}$ can be made $\mathcal{O}(\text{poly}\, D)$ as well, a point that we have discussed further in Appendix A. We relegate the proof of this theorem to Appendix C.

4.10 Putting things together: supervised learning with quOptRF

Now that we can sample to our heart's content $v_0, \ldots, v_{M-1} \in \tilde{\mathcal{V}}$ some M optimised random features using Algorithm 4.1, let us see how to achieve supervised learning to accuracy $\mathcal{O}(\epsilon)$. The idea is to perform linear regression efficiently by a classical algorithm, to obtain coefficients $\alpha = (\alpha_0, \ldots, \alpha_{M-1})^{\mathrm{T}} \in \mathbb{R}^M$ for learning $\sum_{m=0}^{M-1} \alpha_m \varphi(v_m, \cdot) \approx f$. As discussed in Section 4.5, we aim to clarify the runtime of the learning in the large-scale limit. The optimal coefficient α minimises the generalisation error

$$I(\alpha) := \sum_{\tilde{x}\in\tilde{\mathcal{X}}} p^\rho(\tilde{x})\left|f(\tilde{x}) - \sum_{m=0}^{M-1} \alpha_m \varphi(v_m, \tilde{x})\right|^2. \tag{4.33}$$

where the data are IID sampled from $p^\rho(\tilde{x}) := \int_{\Delta_{\tilde{x}}} d\rho(x)$.

We use stochastic gradient descent (SGD) shown in Algorithm 4.2 [JNN19] for the regression to obtain α minimising I, which is a common practice in large-scale machine learning. The

Algorithm 4.2 Stochastic gradient descent (SGD).

Require: Loss function $I : \mathbb{R}^M \to \mathbb{R}$, a projection Π to a convex parameter region $\mathcal{W} \subset \mathbb{R}^M$, a fixed number of iterations $T \in \mathbb{N}$, an initial point $\alpha^{(1)} \in \mathcal{W}$, T-dependent hyperparameters representing step sizes $\left(\eta^{(t)} : t = 1, \ldots, T\right)$ given in [JNN19].

Ensure: Approximate solution α minimising $I(\alpha)$.

1: **for** $t \in \{1, \ldots, T\}$ **do**

2: Calculate unbiased gradient estimator $\hat{g}^{(t)}$ satisfying $\mathbb{E}\left[\hat{g}^{(t)}\right] = \nabla I(\alpha^{(t)})$.

3: $\alpha^{(t+1)} \leftarrow \Pi(\alpha^{(t)} - \eta^{(t)} \hat{g}^{(t)})$.

4: **end for**

5: **Return** $\alpha \leftarrow \alpha^{(T+1)}$.

performance of SGD with random features is extensively studied in [CRR18]. Nevertheless, our contribution is to clarify its *runtime* by explicitly evaluating the runtime per iteration. We assume sufficiently large number N of data points are given; in particular, $N > T$ where T is the number of iterations in the SGD. Then, the sequence of input data points $(\tilde{x}_0, y_0), (\tilde{x}_1, y_1), \ldots$ provides observations of an IID random variable as assumed in Section 4.5.2, and SGD converges to the minimum of the generalisation error I.[6]

To bound the runtime of the SGD, we show that Algorithm 4.2 after $\mathcal{O}((\epsilon Q_{\min})^{-2} \log(1/\delta))$ iterations returns α minimising I to accuracy ϵ with probability at least $1 - \delta$ [JNN19], where $Q_{\min}$ is the minimum of $Q(v_0), \ldots, Q(v_{M-1})$ in Theorem 4.3.[7] In the tth iteration of the SGD for each $t \in \{1, \ldots, T\}$, we calculate an (unbiased) estimate $\hat{g}^{(t)}$ of the gradient ∇I. Using the tth IID sampled data $(\tilde{x}_t, y_t)$ and an integer $m \in \{0, \ldots, M-1\}$ sampled uniformly, we calculate $\hat{g}^{(t)}$ within time $\mathcal{O}(MD)$ in addition to one query to each of the classical oracles $\mathrm{Orc}_{\tilde{x}}$ and Orc_y to get $(\tilde{x}_t, y_t)$ in time $T_{\tilde{x}}$ and T_y, respectively; that is, the runtime per iteration of the SGD is $\mathcal{O}(MD + T_{\tilde{x}} + T_y)$. Combining Algorithm 4.1 with this SGD, we achieve the learning by Algorithm 4.3 within the following overall runtime.

Theorem 4.4 (Overall runtime of supervised learning by optimised random features)**.** *The runtime of Algorithm 4.3 is*

$$\mathcal{O}(MT) + \mathcal{O}\left(MD \cdot \frac{1}{\epsilon^2 Q_{\min}^2} \log \frac{1}{\delta}\right) \quad = \quad \widetilde{\mathcal{O}}(MD).$$

[6] Bach [Bac17] analyses regularised least-squares regression exploiting Q_ϵ^* rather than least-squares of I, but Q_ϵ^* is hard to compute. We may replace this regularisation with L_2 regularisation $R(\alpha) = \lambda \|\alpha\|_2^2$. SGD minimising $I + R$ needs $\mathcal{O}(1/(\epsilon\lambda))$ iterations due to strong convexity [JNN19], while further research is needed to clarify how this affects the learning accuracy.

[7] If important features minimising M have been sampled, their weight $Q_{\min}$ is expected to be large, not dominating the runtime.

Algorithm 4.3 Supervised learning by quOptRF.

Require: Algorithms 4.1 and 4.2, required number M of features for achieving the learning to accuracy $\mathcal{O}(\epsilon)$, classical oracles $\mathrm{Orc}_{\tilde{x}}, \mathrm{Orc}_y$ in eq. (4.18).

Ensure: optimised random features $v_0, \ldots, v_{M-1}$ and coefficients $\alpha_0, \ldots, \alpha_{M-1}$ for $\sum_m \alpha_m \varphi(v_m, \cdot)$ to achieve the learning with probability greater than $1 - \delta$.

1: **for** $m \in \{0, \ldots, M-1\}$ **do**

2: $\quad v_m \leftarrow$ quOptRF. ▷ By Algorithm 4.1.

3: **end for**

4: minimise $I(\alpha)$ to accuracy $\mathcal{O}(\epsilon)$ by SGD to obtain $\alpha_0, \ldots, \alpha_{M-1}$. ▷ By Algorithm 4.2.

5: **Return** $v_0, \ldots, v_{M-1}, \alpha_0, \ldots, \alpha_{M-1}$.

where T is the runtime of quOptRF, using which we sample M optimised random features by Algorithm 4.1, and the second term is runtime of the SGD.

In particular, this is as fast as linear in M and D, i.e. $\mathcal{O}(MD)$. The proof of this final learning algorithm primarily involves classical elements, and we present it in Appendix D.

Since we've used optimised random features, M the number of features required is expected to be nearly minimal; and since Algorithm 4.1 or quOptRF ('kwop-turf') resolves the classical bottleneck of sampling optimised random features, this marriage of SGD and quOptRF thus speeds up learning with random features.

4.11 Discussion

We have constructed a quantum algorithm for sampling an *optimised random feature* within time that scales linearly in the data dimension D, achieving an exponential speedup in D compared to the best known classical algorithms for this sampling task. Combining M features sampled by this quantum algorithm with stochastic gradient descent, we can achieve supervised learning in time $\mathcal{O}(MD)$, where this M is expected to be nearly minimal since we use the optimised random features. M depends explicitly on the so-called degree of freedom of the data, which as described in eq. (4.11) depends on the trace of the inverse of the regularised integral operator. For realistic input distributions and kernels of interest, such as sub-Gaussian inputs with Gaussian kernels, this degree of freedom is expected to be polynomial in D. Since our quantum algorithm does not assume sparsity or low-rank, our results open a route to a widely applicable framework of kernel-based quantum machine learning with a potential exponential speedup.

There are several interesting questions that arise out of our work in this chapter. Most interesting among these is the wider applicability of using periodicity and translation invariance to diagonalise an operator by Fourier transforms. Can we find other examples where a full rank and dense input operator can be decomposed in this form? This is what we referred to as circulantisation in the introduction, and it is also noteworthy that it is also unknown if this can be taken advantage of in classical algorithms to speed them up. In this regard, since the sampling tasks are rather well defined, we also wonder if proving hardness results about the simulability of our algorithm is a plausible goal. We also speculate that we can reduce the runtime to $\mathcal{O}(M \log D)$, by using the notion of orthogonal random features in [LSS13, YSC+16] — but modifying them to *optimised* orthogonal random features to achieve minimal M.

With quOptRF, we draw the curtains on our tryst with matrix functions in this thesis. In the next, and final, chapter of this thesis, we carry forth the ideas of machine learning and big data. But we turn away from supervised learning, to the wilder and richer landscape of unsupervised learning in the context of Natural Language Processing. We will attach this problem with quantum search, and we shall see some expected runtime analyses that are reminiscent of Chapter 3.

Appendix A The kernel as a Fourier sum: Proof of Proposition 4.1

We now turn to proving how we can rewrite a translation-invariant kernel map in terms of a Fourier transform. To recall, our claim is that:

Proposition (Perfect reconstruction of kernel)**.** *For any periodic, translation-invariant, symmetric, positive definite kernel $\tilde{k}$ as defined in eq. (4.12), we have $\forall\, \tilde{x}', \tilde{x} \in \tilde{\mathcal{X}}$*

$$\begin{aligned}\tilde{k}\left(\tilde{x}', \tilde{x}\right) &= \frac{1}{G^D} \sum_{\tilde{v} \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v}) \overline{\varphi(\tilde{v}, \tilde{x}')} \varphi(\tilde{v}, \tilde{x}) \\ &= \left\langle \tilde{x}' \middle| \mathbf{F}_D^\dagger \mathbf{Q}^\tau \mathbf{F}_D \middle| \tilde{x} \right\rangle .\end{aligned}$$

Proof. To begin, recall that any translation-invariant kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be written as

$$k(x', x) = k\left(x' - x\right) = \int_{\mathcal{V}} d\tau(v) \overline{\varphi\left(v, x'\right)} \varphi\left(v, x\right), \tag{I'm 4.3}$$

where eq. (4.4) defines $\varphi(v, x) := \mathrm{e}^{-2\pi \mathrm{i} v \cdot x}$, and the density function

$$q^\tau(v) = \int_{\mathcal{X}} dx\, \mathrm{e}^{-2\pi \mathrm{i} v \cdot x}\, k(x) \tag{I'm 4.5}$$

is the Fourier transform of the kernel. Also recall that we approximate $k(x', x)$ by

$$\tilde{k}(x', x) := \sum_{n \in \mathbb{Z}^D} k(x', x + Gn), \tag{I'm 4.12}$$

whence $\tilde{k}$ inherits the translation invariance of k, in addition to being periodic; in particular $\forall\, n' \in \mathbb{Z}^D$

$$\begin{aligned}\tilde{k}(x' + Gn', x) &= \sum_{n \in \mathbb{Z}^D} k(x' + Gn', x + Gn) \\ &= \sum_{n \in \mathbb{Z}^D} k(x' + Gn' - x - Gn) \\ &= \sum_{n \in \mathbb{Z}^D} k(x', x + G(n - n')) \\ &= \tilde{k}(x', x),\end{aligned}$$

where the second line uses the translation invariance of k.

Shannon's sampling theorem [Sha49, Pro07] is used in signal processing to show that we can perfectly reconstruct any one-dimensional kernel function $\tilde{k}$ on a *continuous* domain

$I = \left[-\frac{G}{2}, \frac{G}{2}\right]$ using a *discrete* set of frequencies from its Fourier transform. The Fourier transform of a 1D kernel on I is given by

$$\begin{aligned}\int_{-\frac{G}{2}}^{\frac{G}{2}} dx\, \mathrm{e}^{-2\pi \mathrm{i} v x}\, \tilde{k}(x) &= \int_{-\infty}^{\infty} dx\, \mathrm{e}^{-2\pi \mathrm{i} v x}\, k(x) \\ &= q^\tau(v). \end{aligned} \tag{4A.1}$$

The inverse Fourier transform then allows us to reconstruct $\tilde{k}$ from q^τ

$$\tilde{k}(x) = \int_{-\infty}^{\infty} dv\, \mathrm{e}^{2\pi \mathrm{i} v x}\, q^\tau(v).$$

The sampling theorem says that $\forall\, x \in I$, using only discrete frequencies $v \in \mathbb{Z}$ we can still recover $\tilde{k}$ exactly, i.e.

$$\tilde{k}(x) = \frac{1}{G} \sum_{v=-\infty}^{\infty} q^\tau\left(v/G\right) \mathrm{e}^{2\pi \mathrm{i} v x/G}. \tag{4A.2}$$

Furthermore, since $\tilde{k}(x)$ is periodic, the above equality actually holds for any $x \in \mathbb{R}$. Analogously, for any $D \geq 1$, we have $\forall\, x \in \mathbb{R}^D$

$$\tilde{k}(x) = \frac{1}{G^D} \sum_{v \in \mathbb{Z}^D} q^\tau\left(v/G\right) \mathrm{e}^{2\pi \mathrm{i} v \cdot x/G}. \tag{4A.3}$$

From $\mathbb{Z}^D$ to $\tilde{\mathcal{X}}$: Since our data domain $\tilde{\mathcal{X}} = \{0, 1, \ldots, G-1\}^D$ is actually *discretised* and finite, $\tilde{k}(x)$ can be perfectly reconstructed on $\tilde{\mathcal{X}}$ via the D-dimensional DFT, using only a *finite* set of discrete frequency modes. As in the recipe for the sampling theorem, first consider the DFT of $\tilde{k}$

$$\begin{aligned}\left(\mathrm{DFT}\, \tilde{k}\right)(\tilde{x}) &= \frac{1}{\sqrt{G^D}} \sum_{\tilde{x}' \in \tilde{\mathcal{X}}} \tilde{k}(\tilde{x}') \mathrm{e}^{-2\pi \mathrm{i} \tilde{x}' \cdot \tilde{x}/G} \\ &= \frac{1}{\sqrt{G^D}} \sum_{\tilde{x}' \in \tilde{\mathcal{X}}} \left(\frac{1}{G^D} \sum_{v' \in \mathbb{Z}^D} q^\tau\left(v'/G\right) \mathrm{e}^{2\pi \mathrm{i} v' \cdot \tilde{x}'/G} \right) \mathrm{e}^{-2\pi \mathrm{i} \tilde{x}' \cdot \tilde{x}/G} \\ &= \frac{1}{\sqrt{G^D}} \sum_{v \in \mathbb{Z}^D} q^\tau\left(\frac{\tilde{x}}{G} + v\right), \end{aligned} \tag{4A.4}$$

since the sum over $\tilde{x}'$ in the second line is nonzero iff $v' = \tilde{x} + Gv$ for $v \in \mathbb{Z}^D$. Denoting $\tilde{x}/G$ by $\tilde{v}$, we are thus lead naturally to the definition eq. (4.22) of $Q^\tau : \tilde{\mathcal{V}} \to \mathbb{R}$

$$Q^\tau(\tilde{v}) := \sum_{v \in \mathbb{Z}^D} q^\tau\left(\tilde{v} + v\right), \tag{4A.5}$$

where $\tilde{\mathcal{V}}$ is the finite set of features

$$\tilde{\mathcal{V}} := \left\{0, \frac{1}{G}, \ldots, 1 - \frac{1}{G}\right\}^D, \tag{I'm 4.21}$$

and the bijection $\tilde{v} \mapsto G\tilde{v}$ lets us go back and forth between $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{X}}$.

Finally, inverting the DFT of eq. (4A.4), we obtain $\forall \tilde{x}', \tilde{x} \in \tilde{\mathcal{X}}$ the perfect reconstruction of the kernel $\tilde{k}(\tilde{x}', \tilde{x})$ using the feature points in $\tilde{\mathcal{V}}$ and the function Q^τ

$$\begin{aligned}
\tilde{k}\left(\tilde{x}', \tilde{x}\right) &= \tilde{k}\left(\tilde{x}' - \tilde{x}\right) \\
&= \frac{1}{G^D} \sum_{\tilde{v} \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v})\, \mathrm{e}^{2\pi \mathrm{i} \tilde{v} \cdot (\tilde{x}' - \tilde{x})} \\
&= \frac{1}{G^D} \sum_{\tilde{v} \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v}) \overline{\varphi\left(\tilde{v}, \tilde{x}'\right)} \varphi\left(\tilde{v}, \tilde{x}\right),
\end{aligned} \tag{4A.6}$$

To put this in matrix form, we defined in eq. (4.23) the diagonal operator

$$\begin{aligned}
\mathbf{Q}^\tau &:= \sum_{\tilde{v} \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v}) \left|\tilde{v}\right\rangle\!\left\langle\tilde{v}\right| \\
&\equiv \sum_{\tilde{x} \in \tilde{\mathcal{X}}} Q^\tau\left(\tilde{x}/G\right) \left|\tilde{x}\right\rangle\!\left\langle\tilde{x}\right|
\end{aligned}$$

corresponding to $Q^\tau(\tilde{v})$. Denote by $\mathbf{F}$ the one-dimensional DFT on $\mathbb{Z}_G$, with matrix elements

$$\mathbf{F}_{\tilde{x}', \tilde{x}} := \frac{1}{\sqrt{G}}\, \omega^{-\tilde{x}' \tilde{x}},$$

written in terms of the G^{th} root of unity

$$\omega = \mathrm{e}^{2\pi \mathrm{i}/G}.$$

$\mathbf{F}_D := \mathbf{F}^{\otimes D}$ is then the D-dimensional DFT on $(\mathbb{Z}_G)^{\times D}$, with matrix entries

$$\left(\mathbf{F}_D\right)_{\tilde{x}', \tilde{x}} := \frac{1}{\sqrt{G^D}}\, \omega^{-\tilde{x}' \cdot \tilde{x}}. \tag{4A.7}$$

These matrix entries are values of the feature map $\varphi : \mathcal{V} \times \mathcal{X} \to \mathbb{C}$ evaluated on $\tilde{\mathcal{V}} \times \tilde{\mathcal{X}}$, i.e.

$$\begin{aligned}
\varphi\left(\tilde{v}, \tilde{x}\right) = \mathrm{e}^{-2\pi \mathrm{i} \tilde{v} \cdot \tilde{x}} &= \sqrt{G^D}\, \left\langle \tilde{v} \,\middle|\, \mathbf{F}_D \,\middle|\, \tilde{x} \right\rangle \\
&= \sqrt{G^D}\, \left\langle \tilde{x} \,\middle|\, \mathbf{F}_D \,\middle|\, \tilde{v} \right\rangle.
\end{aligned} \tag{4A.8}$$

The second line captures the invariance of $\mathbf{F}_D$ under transpose with respect to the computational basis. From eqs. (4A.6) to (4A.8), we finally have $\forall\, \tilde{x}', \tilde{x} \in \tilde{\mathcal{X}}$

$$\tilde{k}\left(\tilde{x}', \tilde{x}\right) = \left\langle \tilde{x}' \middle| \mathbf{F}_D^\dagger \mathbf{Q}^\tau \mathbf{F}_D \middle| \tilde{x} \right\rangle. \tag{4A.9}$$

□

Which kernels can we use?

Our quantum algorithm can use any translation-invariant kernel $\tilde{k}$ in the form of its perfect reconstruction, Proposition 4.1. However, speedups may only be possible in some parameter regimes, and to this end we sneak in some additional (mild) assumptions below.

$Q^\tau(\tilde{v})$ ought to be a classically efficiently computable, in time at most $\mathcal{O}(\mathrm{poly}(D))$ time. We also assume efficient precomputation of its maximum,

$$Q^\tau_{\max} := \max\left\{Q^\tau(\tilde{v}) : \tilde{v} \in \tilde{\mathcal{V}}\right\}. \tag{I'm 4.29}$$

We further assume that the kernel's parameters can be adjusted appropriately to ensure $\tilde{k}(0,0) = \Omega(1)$, and that q^τ has no pathological peaks, so that $Q^\tau_{\max} = \mathcal{O}(\mathrm{poly}(D)))$.

Remark. *As touched upon in Section 4.7, let P^τ be the normalised probability mass function induced by Q^τ on $\tilde{\mathcal{V}}$*

$$P^\tau(\tilde{v}) := \frac{Q^\tau(\tilde{v})}{\sum_{\tilde{v}' \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v}')}. \tag{4A.10}$$

We obtain from eq. (4A.6)

$$\tilde{k}(0,0) = \frac{1}{G^D} \sum_{\tilde{v} \in \tilde{\mathcal{V}}} Q^\tau(\tilde{v}), \tag{4A.11}$$

and hence, we can regard $\tilde{k}(0,0)$ as the normalisation factor in

$$P^\tau(\tilde{v}) = \frac{1}{\tilde{k}(0,0)} \frac{Q^\tau(\tilde{v})}{G^D}. \tag{4A.12}$$

This yields a lower bound on the maximum value of Q^τ (cf. (4.29))

$$Q^\tau_{\max} = G^D \times \frac{Q^\tau_{\max}}{G^D} \geq \sum_{\tilde{v} \in \tilde{\mathcal{V}}} \frac{Q^\tau(\tilde{v})}{G^D} = \tilde{k}(0,0). \tag{4A.13}$$

With Q^τ precomputed[8], we quantumly access it where needed with an oracle Orc_τ

$$\mathrm{Orc}_\tau \ |v\rangle\,|0\rangle = |v\rangle\,|Q^\tau(v)\rangle\,, \tag{4A.14}$$

[8] Orc_τ can then be efficiently implemented with either quantum arithmetics, or QRAM and a lookup table.

and denote its runtime T_τ per query. In fact, in practice many important cases of Q^τ are classically efficiently computable, so that we can have an efficient subroutine that efficiently computes $Q^\tau(v)$ on the fly as and when needed. We assume $Q^\tau(v) \in \mathbb{R}$ is computed to sufficient precision, and omit the overheads, which only contribute weak logarithmic factors to the runtime and ancillary qubit requirement.

	$k(x', x)$	$Q^\tau(\tilde{v})$	$\tilde{k}(0,0)$
Gaussian	$\exp\left(-\gamma \left\|x' - x\right\|_2^2\right)$	$\prod_{i=1}^{D} \vartheta\left(\pi\tilde{v}^i; e^{-\gamma}\right)$	$\frac{1}{1-\mathrm{e}^{-\gamma}} \geq 1$
Laplacian	$\exp\left(-\gamma \left\|x' - x\right\|_1\right)$	$\prod_{i=1}^{D} \frac{\sinh(\gamma)}{\cosh(\gamma) - \cos(2\pi\tilde{v}^i)}$	$''$

Table 4.3: Examples of the distribution function $Q^\tau(\tilde{v})$ for typical kernels. $\tilde{v} = (\tilde{v}^1, \ldots, \tilde{v}^D)^{\mathrm{T}}$, and ϑ is the theta function $\vartheta(u; q) := 1 + 2\sum_{n=1}^{\infty} q^{n^2} \cos(2nu)$.

Remark. *Table 4.3 illustrates that representative choices of kernels, such as the Gaussian and Laplacian, satisfy our assumptions. For these kernels, Q^τ is a product of D special functions, computable classically in time $T_\tau = \mathcal{O}(D)$. Also, $Q^\tau_{\max} = Q^\tau(0)$, and can be made $\mathcal{O}(\mathrm{poly}(D))$ by reducing the parameter $\gamma > 0$. In fact, decreasing γ enlarges the class of learnable functions. These examples indicate that the assumptions made in this section are weak and easy to satisfy. We reiterate that our algorithm does not impose sparsity or low rank on $\mathbf{k}$ or $\hat{\mathbf{q}}^\rho$, making it more widely applicable than previous QML algorithms.*

Appendix B The state that encodes Q^*P: Proof of Proposition 4.2

We had claimed that the bipartite quantum state

$$|\Psi\rangle \propto \sum_{\tilde{v}\in\tilde{\mathcal{X}}} \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} |\tilde{v}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}^\tau}} \mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} |\tilde{v}\rangle^{X'} \tag{4B.1}$$

defined on two registers X and X' of $\lceil D \log G \rceil$ qubits each is our means to sampling optimised random features. In particular, measuring the register X' of $|\Psi\rangle$ in the computational basis $\{|\tilde{v}\rangle^{X'} : \tilde{v} \in \tilde{\mathcal{V}}\}$, we obtain a measurement outcome $\tilde{v}$ with probability

$$\Pr(\tilde{v}) = Q_\epsilon^*(\tilde{v}) P^\tau(\tilde{v}). \tag{4B.2}$$

Recall eq. (4.26) and the definition of the optimised probability distribution $Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})$

$$Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v}) = \frac{1}{\Gamma}Q^\tau(\tilde{v})\left\langle \varphi\left(\tilde{v},\cdot\right)\middle|\,\hat{\mathbf{q}}^\rho\Big(\hat{\boldsymbol{\Sigma}}+\epsilon\mathbb{1}\Big)^{-1}\,\middle|\,\varphi\left(\tilde{v},\cdot\right)\right\rangle, \tag{4B.3}$$

where the normalisation factor is just

$$\Gamma = \sum_{\tilde{v}'\in\tilde{\mathcal{V}}} Q^\tau(\tilde{v}')\left\langle \varphi\left(\tilde{v}',\cdot\right)\middle|\,\hat{\mathbf{q}}^\rho\Big(\hat{\boldsymbol{\Sigma}}+\epsilon\mathbb{1}\Big)^{-1}\,\middle|\,\varphi\left(\tilde{v}',\cdot\right)\right\rangle. \tag{4B.4}$$

$\hat{\boldsymbol{\Sigma}} = \mathbf{k}\hat{\mathbf{q}}^\rho$ is the empirical integral operator, with the discretised kernel matrix and empirical data density given by (Section 4.5.2)

$$\begin{aligned}\mathbf{k} &= \sum_{\tilde{x}',\tilde{x}\in\tilde{\mathcal{X}}} \tilde{k}\left(\tilde{x}',\tilde{x}\right)\left|\tilde{x}'\right\rangle\left\langle\tilde{x}\right|,\\ \hat{\mathbf{q}}^\rho &= \sum_{\tilde{x}\in\tilde{\mathcal{X}}} \hat{q}^\rho(\tilde{x})\left|\tilde{x}\right\rangle\left\langle\tilde{x}\right|,\end{aligned}$$

and we will interchangeably interpret the ket labels as $\tilde{v}\in\tilde{\mathcal{V}}$ or the corresponding $\tilde{x} = G\tilde{v}\in\tilde{\mathcal{X}}$. Since functions become vectors under discretisation (Table 4.1), using eq. (4A.8) we have

$$\begin{aligned}\left|\varphi\left(\tilde{v},\cdot\right)\right\rangle &= \frac{1}{\sqrt{G^D}}\sum_{\tilde{x}\in\tilde{\mathcal{X}}}\varphi\left(\tilde{v},\tilde{x}\right)\left|\tilde{x}\right\rangle\\ &= \sum_{\tilde{x}\in\tilde{\mathcal{X}}}\left(\mathbf{F}_D\right)_{\tilde{v},\tilde{x}}\left|\tilde{x}\right\rangle\\ &= \mathbf{F}_D\left|\tilde{v}\right\rangle,\end{aligned}$$

exploiting the symmetry $\mathbf{F}_D^{\mathrm{T}} = \mathbf{F}_D$ enjoyed by the QFT (DFT) in the computational basis, and as usual, using the equivalence of the two finite sets $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{X}}$.

Then, we have

$$\begin{aligned}\Pr(\tilde{v}) &= \frac{1}{\Gamma}Q^\tau(\tilde{v})\left\langle \varphi\left(\tilde{v},\cdot\right)\middle|\,\hat{\mathbf{q}}^\rho\Big(\hat{\boldsymbol{\Sigma}}+\epsilon\mathbb{1}\Big)^{-1}\,\middle|\,\varphi\left(\tilde{v},\cdot\right)\right\rangle\\ &= \frac{1}{\Gamma}Q^\tau(\tilde{v})\left\langle \tilde{v}\middle|\,\mathbf{F}_D^\dagger\hat{\mathbf{q}}^\rho\Big(\hat{\boldsymbol{\Sigma}}+\epsilon\mathbb{1}\Big)^{-1}\mathbf{F}_D\,\middle|\,\tilde{v}\right\rangle.\end{aligned} \tag{4B.5}$$

Then, plugging in $\mathbf{Q}^\tau$ from eq. (4.23), we obtain

$$\Pr(\tilde{v}) = \frac{1}{\Gamma}\left\langle \tilde{v}\middle|\,\sqrt{\mathbf{Q}^\tau}\mathbf{F}_D^\dagger\hat{\mathbf{q}}^\rho\Big(\hat{\boldsymbol{\Sigma}}+\epsilon\mathbb{1}\Big)^{-1}\mathbf{F}_D\sqrt{\mathbf{Q}^\tau}\,\middle|\,\tilde{v}\right\rangle. \tag{4B.6}$$

Multiplying and dividing by $Q^\tau_{\max}$, we can manipulate this into the form

$$\Pr(\tilde{v}) = \frac{1}{\Gamma}\left\langle \tilde{v}\middle|\,\sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger\hat{\mathbf{q}}^\rho\left[\frac{1}{Q^\tau_{\max}}\Big(\hat{\boldsymbol{\Sigma}}+\epsilon\mathbb{1}\Big)\right]^{-1}\mathbf{F}_D\sqrt{\underline{\mathbf{Q}^\tau}}\,\middle|\,\tilde{v}\right\rangle, \tag{4B.7}$$

where we denote $\mathbf{Q}^\tau$ normalised by $Q^\tau_{\max}$ as

$$\underline{\mathbf{Q}}^\tau := \frac{1}{Q^\tau_{\max}}\mathbf{Q}^\tau \tag{I'm 4.30}$$

to keep the notation manageable. We had previously defined in Section 4.8 a positive definite operator with full support on $\mathcal{H}^X$

$$\hat{\mathbf{\Sigma}}_\epsilon := \frac{1}{Q^\tau_{\max}}\left(\sqrt{\hat{\mathbf{q}}^\rho}\mathbf{k}\sqrt{\hat{\mathbf{q}}^\rho} + \epsilon\mathbb{1}\right). \tag{I'm 4.28}$$

Since $\hat{\mathbf{q}}^\rho$ may not have full rank, we define on the support of $\mathbf{q}^\rho$ the corresponding operator

$$\begin{aligned}\hat{\mathbf{\Sigma}}^\rho_\epsilon &:= \frac{1}{Q^\tau_{\max}}\cdot\sqrt{\hat{\mathbf{q}}^\rho}\left(\hat{\mathbf{\Sigma}} + \epsilon\mathbb{1}\right)(\hat{\mathbf{q}}^\rho)^{-\frac{1}{2}} \\ &= \frac{1}{Q^\tau_{\max}}\left(\sqrt{\hat{\mathbf{q}}^\rho}\mathbf{k}\sqrt{\hat{\mathbf{q}}^\rho} + \epsilon\Pi^\rho\right)\end{aligned}$$

where $(\hat{\mathbf{q}}^\rho)^{-\frac{1}{2}}$ denotes the square root of the Moore-Penrose pseudoinverse of $\hat{\mathbf{q}}^\rho$, and Π^ρ projects onto the support of $\mathbf{q}^\rho$. Clearly, $\hat{\mathbf{\Sigma}}^\rho_\epsilon = \Pi^\rho\hat{\mathbf{\Sigma}}_\epsilon\Pi^\rho$, and since $\mathbf{k}$ is a symmetric matrix in the computational basis, both $\hat{\mathbf{\Sigma}}_\epsilon$ and $\hat{\mathbf{\Sigma}}^\rho_\epsilon$ are also symmetric.

We have by definition

$$\left(\hat{\mathbf{\Sigma}}^\rho_\epsilon\right)^{-1} = (\hat{\mathbf{q}}^\rho)^{\frac{1}{2}}\left[\frac{1}{Q^\tau_{\max}}\left(\hat{\mathbf{\Sigma}} + \epsilon\mathbb{1}\right)\right]^{-1}(\hat{\mathbf{q}}^\rho)^{-\frac{1}{2}}. \tag{4B.8}$$

We can now proceed to further simplify eq. (4B.7):

$$\begin{aligned}\Pr(\tilde{v}) &= \frac{1}{\Gamma}\left\langle\tilde{v}\,\middle|\,\sqrt{\underline{\mathbf{Q}}^\tau}\mathbf{F}^\dagger_D\sqrt{\hat{\mathbf{q}}^\rho}\left(\hat{\mathbf{\Sigma}}^\rho_\epsilon\right)^{-1}\sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D\sqrt{\underline{\mathbf{Q}}^\tau}\,\middle|\,\tilde{v}\right\rangle \\ &= \frac{1}{\Gamma}\left\langle\tilde{v}\,\middle|\,\sqrt{\underline{\mathbf{Q}}^\tau}\mathbf{F}^\dagger_D\sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}^{-1}_\epsilon\sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D\sqrt{\underline{\mathbf{Q}}^\tau}\,\middle|\,\tilde{v}\right\rangle,\end{aligned} \tag{4B.9}$$

since $\Pi^\rho\sqrt{\hat{\mathbf{q}}^\rho} = \sqrt{\hat{\mathbf{q}}^\rho}$, and $\left(\hat{\mathbf{\Sigma}}^\rho_\epsilon\right)^{-1} = \Pi^\rho\left(\hat{\mathbf{\Sigma}}_\epsilon\right)^{-1}\Pi^\rho$, by definition of Π^ρ.

Let us now turn our attention to the quantum state in eq. (4B.1). For any two operators A on $\mathcal{H}^X$ and B on $\mathcal{H}^{X'}$, where $\dim\mathcal{H}^X = \dim\mathcal{H}^{X'}$, we have what is often called the 'transpose trick' [NC10]

$$\sum_{x\in X} A\,|x\rangle^X \otimes B\,|x\rangle^{X'} = \sum_{x\in X}|x\rangle^X \otimes BA^{\mathrm{T}}\,|x\rangle^{X'}, \tag{4B.10}$$

a property that is extensively exploited in manipulations of maximally entangled bipartite states, and the Choi-Jamiołkowski isomorphism. Applying this to eq. (4B.1),

$$\begin{aligned}|\Psi\rangle &\propto \sum_{\tilde{v}\in\tilde{\mathcal{V}}} \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} |\tilde{v}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\, |\tilde{v}\rangle^{X'} \\ &= \sum_{\tilde{v}\in\tilde{\mathcal{V}}} |\tilde{v}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} |\tilde{v}\rangle^{X'}, \end{aligned} \tag{4B.11}$$

using the symmetry of $\hat{\mathbf{\Sigma}}_\epsilon$. If we measure $|\Psi\rangle$ in the computational basis $\left\{|\tilde{v}\rangle^X \otimes |\tilde{v}'\rangle^{X'}\right\}$, the probability distribution of measurement outcomes is

$$\begin{aligned} p(\tilde{v},\tilde{v}') &= \left|\left(\langle\tilde{v}|^X \otimes \langle\tilde{v}'|^{X'}\right)|\Psi\rangle\right|^2 \\ &\propto \left|\left\langle \tilde{v}' \middle| \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \middle| \tilde{v}\right\rangle\right|^2 \\ &= \left\langle \tilde{v}' \middle| \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \middle| \tilde{v}\right\rangle \left\langle \tilde{v} \middle| \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D \sqrt{\underline{\mathbf{Q}^\tau}} \middle| \tilde{v}'\right\rangle. \end{aligned} \tag{4B.12}$$

Measuring the register X' alone yields an outcome $\tilde{v}'$ with probability given by tracing out the register X, i.e. marginalising over $\tilde{v}$,

$$\begin{aligned} p\left(\tilde{v}'\right) &\propto \sum_{\tilde{v}\in\tilde{\mathcal{X}}} p\left(\tilde{v},\tilde{v}'\right) \\ &= \sum_{\tilde{v}\in\tilde{\mathcal{X}}} \left\langle \tilde{v}' \middle| \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \middle| \tilde{v}\right\rangle \left\langle \tilde{v} \middle| \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D \sqrt{\underline{\mathbf{Q}^\tau}} \middle| \tilde{v}'\right\rangle \\ &= \left\langle \tilde{v}' \middle| \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \left(\sum_{\tilde{v}\in\tilde{\mathcal{X}}} |\tilde{v}\rangle\langle\tilde{v}|\right) \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D \sqrt{\underline{\mathbf{Q}^\tau}} \middle| \tilde{v}'\right\rangle \\ &= \left\langle \tilde{v}' \middle| \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \mathbb{1} \hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}} \sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D \sqrt{\underline{\mathbf{Q}^\tau}} \middle| \tilde{v}'\right\rangle \\ &= \left\langle \tilde{v}' \middle| \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho}\hat{\mathbf{\Sigma}}_\epsilon^{-1} \sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D \sqrt{\underline{\mathbf{Q}^\tau}} \middle| \tilde{v}'\right\rangle. \end{aligned} \tag{4B.13}$$

Finally, the normalisation condition $\langle\Psi\,|\,\Psi\rangle = 1$ ensures that the normalisation factor for $p(\tilde{v}')$ is the same as Γ of eq. (4B.4). Therefore, eq. (4B.9) and eq. (4B.13) together yield

$$p(\tilde{v}) = Q_\epsilon^*(\tilde{v}) P^\tau(\tilde{v}), \tag{4B.14}$$

completing the proof.

Appendix C The runtime of quOptRF: Proof of Theorem 4.3

In this appendix we look at the step-by-step resource requirements of Algorithm 4.1 for sampling an optimised random feature by preparing and measuring the state defined in

Proposition 4.2, and thereby establish Theorem 4.3. Note that each line of Algorithm 4.1 is performed approximately with a sufficiently small precision to achieve the overall sampling precision $\Delta > 0$, in the same way as classical algorithms that deal with real numbers using fixed- or floating-point number representation with a sufficiently small precision.

Theorem (Runtime of our quantum algorithm for sampling an optimised random feature)**.** *Given D-dimensional data discretised by $G > 0$, for any accuracy $\epsilon > 0$ and any sampling precision $\Delta > 0$, Algorithm 4.1 samples a feature $\tilde{v} \in \tilde{\mathcal{V}}$ from a weighted distribution $Q(\tilde{v})P^\tau(\tilde{v})$ with $\sum_{\tilde{v}\in\tilde{\mathcal{V}}} |(Q(\tilde{v}) - Q^*_\epsilon(\tilde{v}))P^\tau(\tilde{v})| \leq \delta'$, in runtime*

$$T = \mathcal{O}\left(D \log G \log\log G + T_\rho + T_\tau\right) \times \widetilde{\mathcal{O}}\left(\frac{Q^\tau_{\max}}{\epsilon} \mathit{poly}\log\frac{1}{\delta'}\right),$$

where T_ρ and T_τ are the runtimes of the oracles Orc_ρ and Orc_τ per query, and $Q^\tau_{\max}$ and Orc_τ are defined as eq. (4.29) and eq. (4A.14), respectively. In particular, T is linear in D.

Algorithm 4.1 again: Quantumly sampling an optimised random feature (quOptRF).

Require: A desired accuracy $\epsilon > 0$ in the supervised learning, sampling precision $\delta' > 0$, quantum oracles Orc_ρ in eq. (4.19) and Orc_τ in eq. (4A.14), and $Q^\tau_{\max} > 0$ in eq. (4.29).

Ensure: An optimised random feature $\tilde{v} \in \tilde{\mathcal{V}}$ sampled from a probability distribution $Q(\tilde{v})P^\tau(\tilde{v})$ with $\sum_{\tilde{v}\in\tilde{\mathcal{V}}} |Q(\tilde{v})P^\tau(\tilde{v}) - Q^*_\epsilon(\tilde{v})P^\tau(\tilde{v})| \leq \delta'$.

1: Initialise quantum registers X and X', and load data $|0\rangle^X \otimes |0\rangle^{X'} \longmapsto \sum_{\tilde{x}\in\tilde{\mathcal{X}}} |\tilde{x}\rangle^X \otimes \sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{x}\rangle^{X'}$.

2: Perform a D-dimensional quantum Fourier transform $\mathbf{F}_D^\dagger$ on X' to obtain $\sum_{\tilde{x}\in\tilde{\mathcal{X}}} |\tilde{x}\rangle^X \otimes \mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{x}\rangle^{X'}$.

3: Apply the block encoding of $\sqrt{\underline{\mathbf{Q}^\tau}}$ to X' followed by amplitude amplification to obtain a state proportional to $\sum_{\tilde{x}\in\tilde{\mathcal{X}}} |\tilde{x}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}^\tau}}\mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{x}\rangle^{X'}$.

4: Apply the block encoding of $\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}}$ to X to obtain the quantum state $|\Psi\rangle^{XX'}$ in Proposition 4.2.

5: Perform a measurement of X' in the computational basis to obtain $\tilde{x}$ with probability $Q^*_\epsilon\left(\frac{\tilde{x}}{G}\right)P^\tau\left(\frac{\tilde{x}}{G}\right)$.

6: **Return** $\tilde{v} = \frac{\tilde{x}}{G}$.

The two primary subroutines used in Algorithm 4.1 are (1) the quantum Fourier transform (QFT); and (2) quantum singular value transformation (QSVT).

The unitary QFT operator $\mathbf{F}$ of eq. (4.24) can be implemented to precision δ' by a quantum circuit composed of

$$f_1 = \mathcal{O}\left(\log G \cdot \log\log G \cdot \operatorname{poly}\log\left(\frac{1}{\delta'}\right)\right) \tag{4C.1}$$

controlled NOTs and single qubit rotation gates [HH00]. Using this, we get a circuit for $\mathbf{F}_D = \mathbf{F}^{\otimes D}$ with gate complexity $\mathcal{O}(Df_1)$. Multiplying two $\mathcal{O}(\log G)$-bit numbers classically requires $\mathcal{O}(\log G \log\log G)$ operations [HV19]; quantum arithmetics will have the same scaling to leading order. We could also use the exact QFT [NC10] or grammar-school-method multiplication instead of these algorithms, only introducing an extra $\mathcal{O}(\log G)$ factor.

Block encodings and QSVT

To discuss the complexity of steps 1, 3, 4 of Algorithm 4.1, we construct efficient block encodings for the data density and integral operators; in particular, we first exhibit a block encoding of $\sqrt{\underline{\mathbf{Q}}^\tau}$, and use it to exhibit one for $\hat{\mathbf{\Sigma}}_\epsilon$. We continue to use the definition Definition 3.3 of block encodings that we saw in the appendices to Chapter 3.

Our construction is based on a method of building block encodings for positive operator-valued measures (POVM) [GSL+19]. In particular, for any precision $\delta > 0$ and any POVM element $\mathbf{\Lambda}$, satisfying $0 \preceq \mathbf{\Lambda} \preceq \mathbb{1}$, let $\mathbf{U}$ be a unitary operator a-ancilla plus s-system qubits, that satisfies for any state $|\psi\rangle$

$$\left|\operatorname{Tr}\left[\Pi_\psi \mathbf{\Lambda}\right] - \operatorname{Tr}\left[\mathbf{U}\left(\Pi_0^A \otimes \Pi_\psi\right)\mathbf{U}^\dagger\left(\Pi_0 \otimes \mathbb{1}_{a+s-1}\right)\right]\right| \leq \delta, \tag{4C.2}$$

where $\Pi_\psi = |\psi\rangle\langle\psi|$ etc. are projectors, and A is the a-qubit ancillary register. The second term here corresponds to the probability of the measurement outcome being 0 on measuring the first ancilla qubit of the state $\mathbf{U}\left(|0\rangle^{\otimes n} \otimes |\psi\rangle\right)$, for any input state $|\psi\rangle$, i.e.

$$\Pr(A = 0) = \operatorname{Tr}\left[\mathbf{U}\left(\Pi_0^A \otimes \Pi_\psi\right)\mathbf{U}^\dagger\left(\Pi_0 \otimes \mathbb{1}_{a+s-1}\right)\right]. \tag{4C.3}$$

We can then use [GSL+19, Lemma 49]:

Lemma. *Using one query each to* $\mathbf{U}$ *and* $\mathbf{U}^\dagger$*, and one additional* CNOT *gate, we obtain the* $(1, 1+a, \delta)$*-block encoding of* $\mathbf{\Lambda}$ *given by*

$$\left(\mathbb{1}_1 \otimes \mathbf{U}^\dagger\right)\left(\text{CNOT} \otimes \mathbb{1}_{a+s-1}\right)\left(\mathbb{1}_1 \otimes \mathbf{U}\right).$$

We now present an explicit construction of the circuit $\mathbf{U}$ for the diagonal POVM operator $\mathbf{\Lambda} = \sqrt{\underline{\mathbf{Q}}^\tau}$ in the following lemma, using the quantum oracle Orc_τ. Since a diagonal operators are 1-sparse, conventional methods of implementing block encodings of sparse operators [GSL+19] would also be applicable to $\sqrt{\underline{\mathbf{Q}}^\tau}$; nonetheless our key contribution here

is to use the the block encoding of $\sqrt{\underline{\mathbf{Q}^\tau}}$ as a building block of a more complicated one: that of $\hat{\mathbf{\Sigma}}_\epsilon$, which is not necessarily sparse or of low rank. By the definition of Orc_τ,

$$\mathrm{Orc}_\tau(\,|v\rangle \otimes |0\rangle\,) = |v\rangle \otimes |Q^\tau(v)\rangle\,. \tag{I'm 4A.14}$$

As we have always done, we implicitly use the bijection between the discretised set of data points $\tilde{\mathcal{X}} = \{0, 1, \ldots, G-1\}$ and feature points $\tilde{\mathcal{V}} = \{0, 1/G, \ldots, 1 - 1/G\}$ while computing values of the function Q, which actually has domain $\tilde{\mathcal{X}}$ but divides its arguments by G before using them; we assume that the oracle Orc_τ has these niceties built into it. Achieving overall precision δ will require the use of sufficient ancillary qubits to perform floating point operations with $Q^\tau(v)$ etc. , but we will neglect the weak logarithmic overheads coming from this.

Lemma 4.5 (Block encoding of a diagonal POVM operator)**.** *For any diagonal positive semidefinite operator* $\mathbf{Q}^\tau$ *as in eq. (4.23), we can implement a* $(1, a, \delta)$ *block encoding of* $\sqrt{\underline{\mathbf{Q}^\tau}}$ *using*

$$s = \mathcal{O}\left(D \cdot \log G \cdot \mathrm{poly}\log\frac{1}{\delta}\right)$$

ancillary qubits, a single query to the quantum oracle Orc_τ*, and* $\mathcal{O}\left(s \log\log G\right)$ *additional one and two qubit gates.*

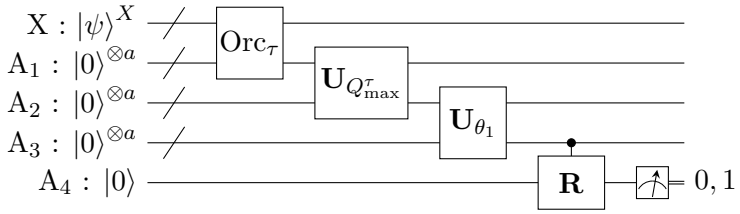


Figure 4.1: A quantum circuit representing a unitary operator $\mathbf{U}$ that achieves eq. (4C.2) for $\mathbf{\Lambda} = \sqrt{\underline{\mathbf{Q}^\tau}}$, which can be used for implementing a block encoding of $\sqrt{\underline{\mathbf{Q}^\tau}}$. This circuit achieves the transformations shown in eq. (4C.9). The last controlled gate represents C$\mathbf{R}$. Here, the system register X consists of $D\lceil\log G\rceil$ qubits, and ancillary registers A_{1-3} consist of $a = \mathcal{O}(\mathrm{poly}\log 1/\delta)$ qubits each.

Proof. First we construct a quantum circuit $\mathbf{U}$ that achieves eq. (4C.2) for $\mathbf{\Lambda} = \sqrt{\underline{\mathbf{Q}^\tau}}$. Denote an arbitrary input system state by

$$|\psi\rangle^X = \sum_{\tilde{v}\in\tilde{\mathcal{X}}} \alpha_{\tilde{v}}\,|\tilde{v}\rangle^X \in \mathcal{H}^X. \tag{4C.4}$$

Define a function

$$\theta_1\,(q) := \arccos\left(q^{\frac{1}{4}}\right). \tag{4C.5}$$

Let $\mathbf{R}_\theta$ denote the single qubit rotation $\mathrm{e}^{-\mathrm{i}\theta\sigma_y}$

$$\mathbf{R}_\theta := \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{4C.6}$$

and let $\mathrm{C}\mathbf{R}$ be the controlled rotation

$$\mathrm{C}\mathbf{R} = \sum_\theta |\theta\rangle\langle\theta| \otimes \mathbf{R}_\theta. \tag{4C.7}$$

Also assume we have circuits for unitary operators $\mathbf{U}_{Q^\tau_{\max}}$ and $\mathbf{U}_{\theta_1}$

$$\begin{aligned} \mathbf{U}_{Q^\tau_{\max}} |q\rangle |0\rangle &= |q\rangle \left| \frac{q}{Q^\tau_{\max}} \right\rangle \\ \mathbf{U}_{\theta_1} |q\rangle |0\rangle &= |q\rangle |\theta_1(q)\rangle . \end{aligned} \tag{4C.8}$$

The circuit of fig. 4.1 then achieves the following transformation (to precision δ). First load the periodi sed data density Q of eq. (4.22)

$$|\psi\rangle |0\rangle |0\rangle |0\rangle |0\rangle \xrightarrow{\mathrm{Orc}_\tau} \sum_{\tilde{v}} \alpha_{\tilde{v}} |\tilde{v}\rangle |Q^\tau(\tilde{v})\rangle |0\rangle |0\rangle |0\rangle , \tag{4C.9}$$

then normalise by the maximum of Q, doing some quantum artihmetics,

$$\ldots \xrightarrow{\mathbf{U}_{Q^\tau_{\max}}} \sum_{\tilde{v}} \alpha_{\tilde{v}} |\tilde{v}\rangle |Q^\tau(\tilde{v})\rangle \left| \frac{Q^\tau(\tilde{v})}{Q^\tau_{\max}} \right\rangle |0\rangle |0\rangle , \tag{4C.10}$$

and then doing some inverse trigonometry,

$$\ldots \xrightarrow{\mathbf{U}_{\theta_1}} \sum_{\tilde{v}} \alpha_{\tilde{v}} |\tilde{v}\rangle |Q^\tau(\tilde{v})\rangle \left| \frac{Q^\tau(\tilde{v})}{Q^\tau_{\max}} \right\rangle \left| \theta_1 \left(\frac{Q^\tau(\tilde{v})}{Q^\tau_{\max}} \right) \right\rangle |0\rangle \tag{4C.11}$$

and finally using the angle obtained to perform controlled rotations, we obtain

$$\begin{aligned} \ldots \xrightarrow{\mathrm{C}\mathbf{R}} \quad & \sum_{\tilde{v}} \alpha_{\tilde{v}} |\tilde{v}\rangle |Q^\tau(\tilde{v})\rangle \left| \frac{Q^\tau(\tilde{v})}{Q^\tau_{\max}} \right\rangle \left| \theta_1 \left(\frac{Q^\tau(\tilde{v})}{Q^\tau_{\max}} \right) \right\rangle \\ & \left(\left(\frac{1}{Q^\tau_{\max}} Q^\tau(\tilde{v}) \right)^{\frac{1}{4}} |0\rangle + \sqrt{1 - \sqrt{\frac{1}{Q^\tau_{\max}} Q^\tau(\tilde{v})}} |1\rangle \right). \end{aligned} \tag{4C.12}$$

The runtime of the quantum oracle Orc_τ queried in eq. (4C.9) is $\mathcal{O}(1)$ modulo preprocessing cost for the QRAM; let us call it T_τ. The quantum arithmetics for normalising by $Q^\tau_{\max}$ in eq. (4C.10), and calculating the inverse trigonometric function θ_1 in eq. (4C.11) incurs a circuit of size $a = \mathcal{O}(\mathrm{poly}\log 1/\delta')$ for a suitable precision δ' [HRS18]. In eq. (4C.12), we rotate the single qubit register A_4 controlled on the angle computed and stored in A_3, needing $\mathcal{O}(a)$

gates since A_3 stores θ to a-bits of precision. Measuring the A_4 register, we get outcome 0 with probability

$$\Pr(\mathrm{A}_4 = 0) = \sum_{\tilde{v}\in\tilde{\mathcal{X}}} |\alpha_{\tilde{v}}|^2 \sqrt{\frac{1}{Q^\tau_{\max}} Q^\tau(\tilde{v})} = \mathrm{Tr}\left[\Pi_\psi \sqrt{\underline{\mathbf{Q}^\tau}}\right], \tag{4C.13}$$

which achieves eq. (4C.2) for $\mathbf{\Lambda} = \sqrt{\underline{\mathbf{Q}^\tau}}$ within the claimed runtime. □

With the block-encoded $\sqrt{\underline{\mathbf{Q}^\tau}}$ as a building block, we next construct a block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$. Note that while the following proposition provides a

$$\left(1, \mathcal{O}\left(D \log G \mathrm{poly}\log\frac{1}{\delta}\right), \delta\right)$$

block encoding of $(1+\epsilon/Q^\tau_{\max})^{-1}\hat{\mathbf{\Sigma}}_\epsilon$, it is equivalently a

$$\left(1+\epsilon/Q^\tau_{\max}, \mathcal{O}\left(D \log G \mathrm{poly}\log\frac{1}{\delta}\right), (1+\epsilon/Q^\tau_{\max})\,\delta\right)$$

block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$ by definition.

Lemma 4.6 (Block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$)**.** *For any $\epsilon > 0$ and any operator $\hat{\mathbf{\Sigma}}_\epsilon$ of the form in eq. (4.28), we can implement a $(1, s, \delta)$-block encoding of*

$$\frac{1}{1+\epsilon/Q^\tau_{\max}}\hat{\mathbf{\Sigma}}_\epsilon$$

using a single query each to the quantum oracles $\mathrm{Orc}^\dagger_\rho$ and Orc_τ, and $\mathcal{O}(s \log\log G)$ additional one- and two-qubit gates, with an ancillary register is of size

$$s = \mathcal{O}\left(D \cdot \log G \cdot \mathrm{poly}\log\frac{1}{\delta}\right)$$

Proof. We will first construct a circuit $\mathbf{U}$ that achieves eq. (4C.2) for $\mathbf{\Lambda} = \hat{\mathbf{\Sigma}}_\epsilon$, using the same notation as in lemma 4.5. Let

$$|\psi\rangle^X = \sum_{\tilde{x}\in\tilde{\mathcal{X}}} \alpha_{\tilde{x}} |\tilde{x}\rangle^X \in \mathcal{H}^X. \tag{4C.14}$$

be an arbitrary input system state, and define functions

$$\theta_2(q) := \arccos\left(\sqrt{q}\right), \tag{4C.15}$$

$$\theta_3(\epsilon) := \arccos\left(\sqrt{\frac{\epsilon/Q^\tau_{\max}}{1+\epsilon/Q^\tau_{\max}}}\right). \tag{4C.16}$$

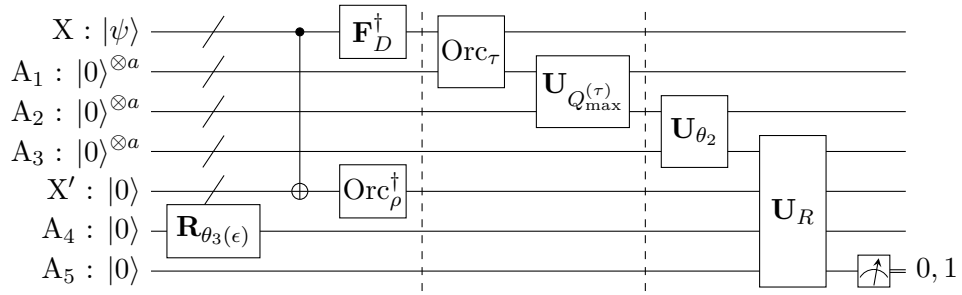


Figure 4.2: Quantum circuit for $\mathbf{U}$ that achieves eq. (4C.2) for $\mathbf{\Lambda} = \hat{\mathbf{\Sigma}}_\epsilon$, which can be used for implementing a block encoding of $\hat{\mathbf{\Sigma}}_\epsilon$. The first controlled-not collectively represents CNOT gates on corresponding pairs of qubits of X and X'. This circuit achieves the series of transformations starting from eq. (4C.19). Measuring register A_5 produces outcomes in accordance with eq. (4C.23). Here, $a = \mathcal{O}(\mathrm{poly}\log 1/\epsilon)$.

Since these are efficiently computable elementary functions, we again have efficient circuits for the quantum arithmetics, for instance

$$\mathbf{U}_{\theta_2} |q\rangle |0\rangle = |q\rangle |\theta_2(q)\rangle . \tag{4C.17}$$

We will also need the multi-controlled rotation (as before, $\Pi_\theta := |\theta\rangle\langle\theta|$ etc. are projectors)

$$\begin{aligned}\mathbf{U}_R = \left(\mathbb{1}^{\mathrm{A}_3} \otimes \mathbb{1}^{X'} \otimes \Pi_0^{\mathrm{A}_4} \otimes \mathbb{1}^{\mathrm{A}_5}\right) + \left(\sum_\theta \Pi_\theta^{\mathrm{A}_3} \otimes (\mathbb{1} - \Pi_0)^{X'} \otimes \Pi_1^{\mathrm{A}_4} \otimes \sigma_x^{\mathrm{A}_5}\right) \\ + \left(\sum_\theta \Pi_\theta^{\mathrm{A}_3} \otimes \Pi_0^{X'} \otimes \Pi_1^{\mathrm{A}_4} \otimes \mathbf{R}_\theta^{\mathrm{A}_5}\right) .\end{aligned} \tag{4C.18}$$

Let us now analyse the quantum circuit of fig. 4.2, with these ingredients in place. The part of this circuit in between the vertical dashed lines also appears in fig. 4.1. The preprocessing before the first vertical line effects the transformation

$$|\psi\rangle^X |0\rangle |0\rangle |0\rangle |0\rangle^{X'} |0\rangle^{\mathrm{A}_4} |0\rangle^{\mathrm{A}_5} \tag{4C.19}$$

$$\xrightarrow{\text{CNOT}} \sum_{\tilde{x}} \alpha_{\tilde{x}} |\tilde{x}\rangle^X |0\rangle |0\rangle |0\rangle |\tilde{x}\rangle^{X'} |0\rangle^{\mathrm{A}_4} |0\rangle^{\mathrm{A}_5} \tag{4C.20}$$

$$\begin{aligned}\xrightarrow{\mathbf{F}_D^\dagger \otimes \mathrm{Orc}_\rho^\dagger \otimes \mathbf{R}_{\theta_3(\epsilon)}} \sum_{\tilde{x}} \alpha_{\tilde{x}} \mathbf{F}_D^\dagger |\tilde{x}\rangle^X |0\rangle |0\rangle |0\rangle \mathrm{Orc}_\rho^\dagger |\tilde{x}\rangle^{X'} \\ \left(\sqrt{\frac{\epsilon/Q_{\max}^\tau}{1+\epsilon/Q_{\max}^\tau}} |0\rangle^{\mathrm{A}_4} + \sqrt{\frac{1}{1+\epsilon/Q_{\max}^\tau}} |1\rangle^{\mathrm{A}_4}\right) \otimes |0\rangle^{\mathrm{A}_5} .\end{aligned} \tag{4C.21}$$

In eq. (4C.20), CNOT gates on each of the $\mathcal{O}(D \log G)$ qubits of the quantum registers X and X' copy out the basis states into the register X'. $\mathbf{F}_D^\dagger$ in eq. (4C.21) requires $\widetilde{\mathcal{O}}(D \log G)$ gates as we saw in eq. (4C.1). $\mathrm{Orc}_\rho^\dagger$ queries QRAM and so it costs only $\mathcal{O}(1)$ in lookup (addressing)

time, but say the preprocessing (state preparation) time is T_ρ. The single qubit rotation $\mathbf{R}_{\theta_3(\epsilon)}$ of eq. (4C.6) can be implemented with precision $O(\delta')$ using $\mathcal{O}\left(\text{poly}\log\frac{1}{\delta'}\right)$ gates [NC10].

Next comes the part that was also in fig. 4.1. Following this, we implement $\mathbf{U}_{\theta_2}$ as we did in eq. (4C.11), using $\mathcal{O}\left(\text{poly}\log\frac{1}{\delta'}\right)$ gates. We can implement $\mathbf{U}_R$ using $\mathcal{O}\left(D\log G\,\text{poly}\log\frac{1}{\delta'}\right)$ gates since $|\theta\rangle$ is stored in $\mathcal{O}\left(\text{poly}\log\frac{1}{\delta'}\right)$ qubits and X' consists of $\mathcal{O}\left(D\log G\right)$ qubits.

After performing $\mathbf{U}_R$, we measure the last qubit A_5 in the computational basis. We will calculate the probability of obtaining the outcome 0 in this measurement of A_5 using conditional probabilities; suppose that we performed a measurement of the one-qubit register A_4 in the computational basis. The outcome probabilities are

$$\begin{aligned}\Pr(\mathrm{A}_4=0)&=\frac{\epsilon/Q^\tau_{\max}}{1+\epsilon/Q^\tau_{\max}}\\ \Pr(\mathrm{A}_4=1)&=\frac{1}{1+\epsilon/Q^\tau_{\max}}.\end{aligned}\tag{4C.22}$$

Conditioned on seeing $\mathrm{A}_4=0$, measuring A_5 gives outcome 0 with certainty, corresponding to the first term of eq. (4C.18). Similarly, the third term of eq. (4C.18) tells us that

$$\begin{aligned}\Pr(\mathrm{A}_5=0\mid\mathrm{A}_4=1)&=\sum_{\tilde{x}'\in\tilde{\mathcal{X}}}\left|\sum_{\tilde{x}\in\tilde{\mathcal{X}}}\alpha_{\tilde{x}}\left\langle\tilde{x}'\right|\mathbf{F}_D^\dagger\left|\tilde{x}\right\rangle\left\langle 0\right|\mathrm{Orc}_\rho^\dagger\left|\tilde{x}\right\rangle\sqrt{\frac{1}{Q^\tau_{\max}}Q^\tau\left(\frac{\tilde{x}'}{G}\right)}\right|^2\\ &=\sum_{\tilde{x}'\in\tilde{\mathcal{X}}}\frac{1}{Q^\tau_{\max}}Q^\tau\left(\frac{\tilde{x}'}{G}\right)\left|\sum_{\tilde{x}\in\tilde{\mathcal{X}}}\alpha_{\tilde{x}}\left\langle\tilde{x}'\right|\mathbf{F}_D^\dagger\left|\tilde{x}\right\rangle\left\langle 0\right|\mathrm{Orc}_\rho^\dagger\left|\tilde{x}\right\rangle\right|^2.\end{aligned}\tag{4C.23}$$

The second term of eq. (4C.18) makes no contribution to eq. (4C.23) because σ_x flips $|0\rangle$ to $|1\rangle$. Recall that by definition of Orc_ρ,

$$\mathrm{Orc}_\rho|0\rangle=\sum_{\tilde{x}\in\tilde{\mathcal{X}}}\sqrt{\hat{q}^\rho(\tilde{x})}\,|\tilde{x}\rangle=\sqrt{\hat{\mathbf{q}}^\rho}\sum_{\tilde{x}\in\tilde{\mathcal{X}}}|\tilde{x}\rangle\,.\tag{I'm 4.19}$$

This means that

$$\left|\left\langle 0\right|\mathrm{Orc}_\rho^\dagger\left|\tilde{x}\right\rangle\right|=\sqrt{\hat{q}^\rho(\tilde{x})}.$$

We then have

$$\begin{aligned}\textit{eq. (4C.23)}&=\sum_{\tilde{x}'\in\tilde{\mathcal{X}}}\frac{1}{Q^\tau_{\max}}Q^\tau\left(\frac{\tilde{x}'}{G}\right)\left|\sum_{\tilde{x}\in\tilde{\mathcal{X}}}\alpha_{\tilde{x}}\left\langle\tilde{x}'\right|\mathbf{F}_D^\dagger\left|\tilde{x}\right\rangle\sqrt{\hat{q}^\rho(\tilde{x})}\right|^2\\ &=\sum_{\tilde{x}'\in\tilde{\mathcal{X}}}\frac{1}{Q^\tau_{\max}}Q^\tau\left(\frac{\tilde{x}'}{G}\right)\left|\sum_{\tilde{x}\in\tilde{\mathcal{X}}}\alpha_{\tilde{x}}\left\langle\tilde{x}'\right|\mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\left|\tilde{x}\right\rangle\right|^2.\end{aligned}\tag{4C.24}$$

By definition eq. (4C.14) of $|\psi\rangle$, we have

$$\begin{aligned}
\textit{eq. (4C.24)} &= \sum_{\tilde{x}'\in\tilde{\mathcal{X}}} \frac{1}{Q^\tau_{\max}} Q^\tau\left(\frac{\tilde{x}'}{G}\right) \left|\left\langle \tilde{x}' \middle| \mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} \middle| \psi \right\rangle\right|^2 \\
&= \sum_{\tilde{x}'\in\tilde{\mathcal{X}}} \frac{1}{Q^\tau_{\max}} Q^\tau\left(\frac{\tilde{x}'}{G}\right) \left\langle \tilde{x}' \middle| \mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} \Pi_\psi \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{F}_D \middle| \tilde{x}' \right\rangle \\
&= \frac{1}{Q^\tau_{\max}} \operatorname{Tr}\left[\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} \Pi_\psi \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{F}_D \left(\sum_{\tilde{x}'\in\tilde{\mathcal{X}}} Q^\tau\left(\frac{\tilde{x}'}{G}\right) |\tilde{x}'\rangle \langle \tilde{x}'| \right) \right]. \qquad \text{(4C.25)}
\end{aligned}$$

By definition eq. (4.23) of $\mathbf{Q}^\tau$, this is the same as

$$\begin{aligned}
\textit{eq. (4C.25)} &= \frac{1}{Q^\tau_{\max}} \operatorname{Tr}\left[\mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} \Pi_\psi \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{F}_D \mathbf{Q}^\tau \right] \\
&= \frac{1}{Q^\tau_{\max}} \operatorname{Tr}\left[\Pi_\psi \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{F}_D \mathbf{Q}^\tau \mathbf{F}_D^\dagger \sqrt{\hat{\mathbf{q}}^\rho} \right] \\
&= \frac{1}{Q^\tau_{\max}} \operatorname{Tr}\left[\Pi_\psi \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{k} \sqrt{\hat{\mathbf{q}}^\rho} \right], \qquad \text{(4C.26)}
\end{aligned}$$

where the last equality follows from the perfect reconstruction of the kernel k shown in Proposition 4.1. Therefore, plugging in the probabilities for measurements on A_4 from eq. (4C.22), we obtain

$$\begin{aligned}
\Pr(\mathrm{A}_5 = 0) &= \frac{\epsilon/Q^\tau_{\max}}{1+\epsilon/Q^\tau_{\max}} \times 1 + \frac{1}{1+\epsilon/Q^\tau_{\max}} \times \left(\frac{1}{Q^\tau_{\max}} \operatorname{Tr}\left[\Pi_\psi \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{k} \sqrt{\hat{\mathbf{q}}^\rho} \right] \right) \\
&= \frac{1}{1+\epsilon/Q^\tau_{\max}} \times \operatorname{Tr}\left[\Pi_\psi \left(\frac{\epsilon}{Q^\tau_{\max}} \mathbb{1} \right) \right] + \frac{1}{1+\epsilon/Q^\tau_{\max}} \times \operatorname{Tr}\left[\Pi_\psi \left(\frac{1}{Q^\tau_{\max}} \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{k} \sqrt{\hat{\mathbf{q}}^\rho} \right) \right] \\
&= \frac{1}{1+\epsilon/Q^\tau_{\max}} \times \operatorname{Tr}\left[\Pi_\psi \left(\frac{1}{Q^\tau_{\max}} \sqrt{\hat{\mathbf{q}}^\rho} \mathbf{k} \sqrt{\hat{\mathbf{q}}^\rho} + \frac{\epsilon}{Q^\tau_{\max}} \mathbb{1} \right) \right] \\
&= \operatorname{Tr}\left[\Pi_\psi \left(\frac{1}{1+\epsilon/Q^\tau_{\max}} \hat{\mathbf{\Sigma}}_\epsilon \right) \right], \qquad \text{(4C.27)}
\end{aligned}$$

where the last equality follows from the definition eq. (4.28) of $\hat{\mathbf{\Sigma}}_\epsilon$ (Section 4.8). This achieves eq. (4C.2) for $\mathbf{\Lambda} = \frac{1}{1+\epsilon/Q^\tau_{\max}} \hat{\mathbf{\Sigma}}_\epsilon$ within the claimed circuit size. □

We can now bound the runtime of Algorithm 4.1 using the analyses for these block encodings in lemmas 4.5 and 4.6.

Proof of Theorem 4.3. The most resource intensive step of Algorithm 4.1 is Step 5, as can be seen in the following.

In Step 2, we prepare $\sum_{\tilde{x}} |0\rangle^X \otimes \sqrt{\hat{\mathbf{q}}^\rho} |\tilde{x}\rangle^{X'}$ by one query to the oracle Orc_ρ, followed by $\mathcal{O}(D \log G)$ CNOT gates to prepare $\sum_{\tilde{x}} |\tilde{x}\rangle^X \otimes \sqrt{\hat{\mathbf{q}}^\rho} |\tilde{x}\rangle^{X'}$, since $\mathcal{H}^X$ consists of $\mathcal{O}(D \log G)$ qubits. Step 3 performs $\mathbf{F}_D^\dagger$, which is implemented using $\widetilde{\mathcal{O}}(D \log G)$ gates as shown in eq. (4C.1). Step 4 is the block encoding of $\sqrt{\mathbf{Q}^\tau}$, requiring $\widetilde{\mathcal{O}}(D \log G + T_\tau)$ gates as

shown in lemma 4.5. The runtime at this moment is $\widetilde{\mathcal{O}}\left(D\log G + T_\rho + T_\tau\right)$. After Step-4, the part of the state we are interested in is

$$\sum_{\tilde{x}\in\tilde{\mathcal{X}}} |\tilde{x}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}}^\tau}\mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{x}\rangle^{X'}\,. \tag{4C.28}$$

The norm of this term is

$$\begin{aligned}\left\|\sum_{\tilde{x}\in\tilde{\mathcal{X}}} |\tilde{x}\rangle^X \otimes \sqrt{\underline{\mathbf{Q}}^\tau}\mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\,|\tilde{x}\rangle^{X'}\right\|_2 &= \sqrt{\frac{\operatorname{Tr}\left[\sqrt{\hat{\mathbf{q}}^\rho}\mathbf{F}_D\mathbf{Q}^\tau\mathbf{F}_D^\dagger\sqrt{\hat{\mathbf{q}}^\rho}\right]}{Q^\tau_{\max}}} \\ &= \sqrt{\frac{\operatorname{Tr}\left[\mathbf{F}_D\mathbf{Q}^\tau\mathbf{F}_D^\dagger\hat{\mathbf{q}}^\rho\right]}{Q^\tau_{\max}}} \\ &= \sqrt{\frac{\operatorname{Tr}\hat{\mathbf{\Sigma}}}{Q^\tau_{\max}}},\end{aligned} \tag{4C.29}$$

where the last equality uses $\hat{\mathbf{\Sigma}} = \mathbf{k}\hat{\mathbf{q}}^\rho = \mathbf{F}_D\mathbf{Q}^\tau\mathbf{F}_D^\dagger\hat{\mathbf{q}}^\rho$ obtained from Proposition 4.1. For any translation-invariant kernel $\tilde{k}(x',x) = \tilde{k}\left(x'-x\right)$, we can evaluate $\operatorname{Tr}\hat{\mathbf{\Sigma}}$ as

$$\operatorname{Tr}\hat{\mathbf{\Sigma}} = \operatorname{Tr}\left[\mathbf{k}\hat{\mathbf{q}}^\rho\right] = \tilde{k}(0)\operatorname{Tr}\hat{\mathbf{q}}^\rho = \tilde{k}(0,0) = \Omega(1), \tag{4C.30}$$

where we use the mild assumption $\tilde{k}(0,0) = \Omega(k(0,0)) = \Omega(1)$ that we'd discussed in Appendix A. Thus, to obtain the normalised quantum state proportional to the term eq. (4C.28), Step 4 is followed by standard amplitude amplification, repeating the above steps

$$\mathcal{O}\left(\sqrt{\frac{Q^\tau_{\max}}{\operatorname{Tr}\hat{\mathbf{\Sigma}}}}\right) = \mathcal{O}\left(\sqrt{Q^\tau_{\max}}\right)$$

times. Therefore, at the end of Step 4 including the amplitude amplification, the runtime is

$$\mathcal{O}\left(\left(D\log G\log\log G\operatorname{poly}\log\frac{1}{\delta} + T_\rho + T_\tau\right)\times\sqrt{Q^\tau_{\max}}\right). \tag{4C.31}$$

Step 5 performs a block encoding of $\hat{\mathbf{\Sigma}}_\epsilon^{-\frac{1}{2}}$, which is obtained from quantum singular value transformation (QSVT) of the block encoding $\mathbf{U}_{\hat{\mathbf{\Sigma}}}$ of $\left(1+\epsilon/Q^\tau_{\max}\right)^{-1}\hat{\mathbf{\Sigma}}_\epsilon$ constructed in lemma 4.6. The QSVT combined with variable-time amplitude amplification [Amb12, CGJ19] yields a block encoding of $\left(\frac{1}{1+(\epsilon/Q^\tau_{\max})}\hat{\mathbf{\Sigma}}_\epsilon\right)^{-\frac{1}{2}}$, using $\mathbf{U}_{\hat{\mathbf{\Sigma}}}$

$$\widetilde{O}\left(\left(\frac{Q^\tau_{\max}}{\epsilon}+1\right)\operatorname{poly}\log\frac{1}{\delta'}\right)$$

times [GSL+19]. This includes amplitude amplification, and $\frac{Q^\tau_{\max}}{\epsilon}+1$ is the condition number of $\frac{1}{1+(\epsilon/Q^\tau_{\max})}\hat{\mathbf{\Sigma}}_\epsilon$ since it holds that

$$\frac{1}{1+(\epsilon/Q^\tau_{\max})}\frac{\epsilon}{Q^\tau_{\max}}\mathbb{1} \leq \frac{1}{1+(\epsilon/Q^\tau_{\max})}\hat{\mathbf{\Sigma}}_\epsilon \leq \mathbb{1}. \tag{4C.32}$$

Thus, the net runtime/circuit size for Step 5 is

$$\mathcal{O}\left(D\log G\log\log G+T_\rho+T_\tau\right)\times\widetilde{O}\left(\frac{Q^\tau_{\max}}{\epsilon}\operatorname{poly}\log\frac{1}{\delta}\right). \tag{4C.33}$$

Finally, from eq. (4C.31) and eq. (4C.33), the total runtime of quOptRF at the end of Step 5, including all amplitude amplification steps, is

$$\begin{aligned}&\mathcal{O}\left(D\log G\log\log G+T_\rho+T_\tau\right)\times\widetilde{O}\left(\left(\sqrt{Q^\tau_{\max}}+\frac{Q^\tau_{\max}}{\epsilon}\right)\operatorname{poly}\log\frac{1}{\delta}\right)\\&=\widetilde{\mathcal{O}}\left(D\log G\cdot\frac{Q^\tau_{\max}}{\epsilon}\cdot\operatorname{poly}\log\frac{1}{\delta}\right),\end{aligned} \tag{4C.34}$$

establishing our claim. □

Appendix D The runtime of SGD: Proof of Theorem 4.4

Finally, we prove Theorem 4.4 bounding the overall runtime of the supervised learning with optimised random features. Algorithms 4.2 and 4.3 for stochastic gradient descent (SGD) and achieving the supervised learning respectively are also repeated in the following.

Theorem (Overall runtime of supervised learning by optimised random features)**.** *The runtime of Algorithm 4.3 is*

$$O\left(MT\right)+O\left(\left(MD+T_{\tilde{x}}+T_y\right)\frac{1}{\epsilon^2Q^2_{\min}}\log\frac{1}{\delta}\right),$$

where T is the runtime of quOptRF, M is the number of optimised random features to be sampled using by Algorithm 4.1, and the second term is runtime of the SGD. In particular, this is as fast as linear in M and D, i.e. $\mathcal{O}\left(MD\right)$.

Recall that to perform the SGD, we invoke classical RAM oracles for accessing data

$$\mathrm{Orc}_{\tilde{x}}(n)=\tilde{x}_n,\quad \mathrm{Orc}_y(n)=y_n. \tag{I'm 4.18}$$

Algorithm 4.2 again: Descending stochastically using the gradient

Require: A function $I : \mathcal{W} \to \mathbb{R}$, a projection Π to a convex parameter region $\mathcal{W} \subset \mathbb{R}^M$, a fixed number of iterations $T \in \mathbb{N}$, an initial point $\alpha^1 \in \mathcal{W}$, T-dependent hyperparameters representing step sizes $\left(\eta^{(t)} : t = 1, \ldots, T\right)$ given in [JNN19].

Ensure: Approximate solution α minimising $I(\alpha)$.

1: **for** $t \in \{1, \ldots, T\}$ **do**

2: Calculate $\hat{g}^{(t)}$ satisfying $\mathbb{E}\left[\hat{g}^{(t)}\right] = \nabla I(\alpha^{(t)})$.

3: $\alpha^{(t+1)} \leftarrow \Pi(\alpha^{(t)} - \eta^{(t)}\hat{g}^{(t)})$.

4: **end for**

5: **Return** $\alpha \leftarrow \alpha^{(T+1)}$.

Algorithm 4.3 again: quOptRF for supervised learning.

Require: Inputs to Algorithms 4.1 and 4.2, required number M of features for achieving the learning to accuracy $O(\epsilon)$, classical oracles $\mathrm{Orc}_{\tilde{x}}, \mathrm{Orc}_y$ in eq. (4.18).

Ensure: Optimised random features $v_0, \ldots, v_{M-1}$ and coefficients $\alpha_0, \ldots, \alpha_{M-1}$ for $\sum_m \alpha_m \varphi(v_m, \cdot)$ to achieve the learning with probability greater than $1 - \delta$.

1: **for** $m \in \{0, \ldots, M-1\}$ **do**

2: $v_m \leftarrow$ quOptRF. ▷ by Algorithm 4.1.

3: **end for**

4: Minimise $I(\alpha)$ to accuracy $O(\epsilon)$ by SGD to obtain $\alpha_0, \ldots, \alpha_{M-1}$. ▷ by Algorithm 4.2.

5: **Return** $v_0, \ldots, v_{M-1}, \alpha_0, \ldots, \alpha_{M-1}$.

Proof. Let us look at Algorithm 4.3 step-by-step. In Step 2, using Algorithm 4.1 repeatedly M times, we can obtain M optimised random features within time

$$O(MT_q), \tag{4D.1}$$

where T_q is the runtime of quOptRF as given by Theorem 4.3. As for Step 4, we bound the runtime of the SGD in Algorithm 4.2. In the following, we show that the runtime of each iteration of the SGD is $\mathcal{O}\left(MD + T_{\tilde{x}} + T_y\right)$, and the required number of iterations in the SGD is upper bounded by $\mathcal{O}\left(\frac{1}{\epsilon^2 Q_{\min}^2} \log \frac{1}{\delta}\right)$.

Consider the t^{th} iteration of the SGD for $t \in \{0, \ldots, T-1\}$. The most intensive step is the calculation of an unbiased estimate $\hat{g}^{(t)}$ of the gradient $\mathbb{E}\left[\hat{g}^{(t)}\right] = \nabla I\left(\alpha^{(t)}\right)$ for

$$I(\alpha) := \sum_{\tilde{x} \in \tilde{\mathcal{X}}} p^{\rho}(\tilde{x}) \left| f(\tilde{x}) - \sum_{m=0}^{M-1} \alpha_m \varphi(v_m, \tilde{x}) \right|^2, \tag{4D.2}$$

where $\varphi(v, x) = \mathrm{e}^{-2\pi \mathrm{i} v \cdot x}$, $p^{\rho}(\tilde{x}) = \int_{\Delta_{\tilde{x}}} d\rho(x)$, and we write

$$\alpha = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{M-1} \end{pmatrix}. \tag{4D.3}$$

The gradient of I is given by

$$\begin{aligned} \nabla I(\alpha) &= \sum_{\tilde{x} \in \tilde{\mathcal{X}}} p^{\rho}(\tilde{x}) \begin{pmatrix} 2\Re\left[\mathrm{e}^{-2\pi \mathrm{i} v_0 \cdot \tilde{x}} \left(f(\tilde{x}) - \sum_{m=0}^{M-1} \alpha_m \mathrm{e}^{2\pi \mathrm{i} v_m \cdot \tilde{x}}\right)\right] \\ \vdots \\ 2\Re\left[\mathrm{e}^{-2\pi \mathrm{i} v_{M-1} \cdot \tilde{x}} \left(f(\tilde{x}) - \sum_{m=0}^{M-1} \alpha_m \mathrm{e}^{2\pi \mathrm{i} v_m \cdot \tilde{x}}\right)\right] \end{pmatrix} \\ &= \sum_{m=0}^{M-1} \frac{1}{M} \sum_{\tilde{x} \in \tilde{\mathcal{X}}} p^{\rho}(\tilde{x}) \begin{pmatrix} 2\Re\left[\mathrm{e}^{-2\pi \mathrm{i} v_0 \cdot \tilde{x}} \left(f(\tilde{x}) - M\alpha_m \mathrm{e}^{2\pi \mathrm{i} v_m \cdot \tilde{x}}\right)\right] \\ \vdots \\ 2\Re\left[\mathrm{e}^{-2\pi \mathrm{i} v_{M-1} \cdot \tilde{x}} \left(f(\tilde{x}) - M\alpha_m \mathrm{e}^{2\pi \mathrm{i} v_m \cdot \tilde{x}}\right)\right] \end{pmatrix}, \end{aligned} \tag{4D.4}$$

where $\Re$ represents the real part. In the t^{th} iteration, Algorithm 4.2 estimates the gradient at the point $\alpha_0^{(t)} \in \left\{\alpha^{(1)}, \ldots, \alpha^{(t)}\right\}$

Using a pair of given data points $(\tilde{x}_t, y_t = f(\tilde{x}_t)) \in \{(\tilde{x}_0, y_0), (\tilde{x}_1, y_1), \ldots, \}$ sampled with probability $p^{\rho}(\tilde{x})$ as observations of an independently and identically distributed (IID) random variable, and an integer $m \in \{0, \ldots, M-1\}$ uniformly sampled with probability $\frac{1}{M}$, we give an unbiased estimate $\hat{g}^{(t)}$ of this gradient at each point $\alpha^{(t)}$ by

$$\hat{g}^{(t)} = \begin{pmatrix} 2\Re\left[\mathrm{e}^{-2\pi \mathrm{i} v_0 \cdot \tilde{x}_t} \left(y_t - M\alpha_m^{(t)} \mathrm{e}^{2\pi \mathrm{i} v_m \cdot \tilde{x}_t}\right)\right] \\ \vdots \\ 2\Re\left[\mathrm{e}^{-2\pi \mathrm{i} v_{M-1} \cdot \tilde{x}_t} \left(y_t - M\alpha_m^{(t)} \mathrm{e}^{2\pi \mathrm{i} v_m \cdot \tilde{x}_t}\right)\right] \end{pmatrix}. \tag{4D.5}$$

By construction, we have

$$\mathbb{E}\left[\hat{g}^{(t)}\right] = \nabla I\left(\alpha^{(t)}\right). \tag{4D.6}$$

We obtain $\tilde{x}_t$ and $y_t = f(\tilde{x}_t)$ using the classical RAM oracles $\mathrm{Orc}_{\tilde{x}}$ and Orc_y; say we bundle their query and update times all together into $T_{\tilde{x}}$ and T_y respectively. We can represent the integer m using $\mathrm{len}_m = \lceil \log_2 M \rceil$ bits, where $\lceil x \rceil$ is the least integer greater than or equal to x, so we can sample m from a uniform distribution standard random number generators within time $\mathcal{O}(\mathrm{poly}\log M)$. Note that even if in context it is expensive to use classical randomness,

quantum computation can efficiently sample m uniform len_m-bit strings in $O(\log M)$ time. To do this, len_m qubits are initially prepared in the all $|0\rangle$ state, and the Hadamard gate is applied to each qubit to obtain

$$\frac{1}{\sqrt{2^{\mathrm{len}_m}}}(|0\rangle + |1\rangle)^{\otimes \mathrm{len}_m}, \tag{4D.7}$$

followed by a measurement of this state in the computational basis. Given $\tilde{x}_t$, y_t, and m, we can calculate each of the M elements of $\hat{g}$ in eq. (4D.5) within time $O(D)$ for calculating the inner product of D-dimensional vectors, and hence the calculation of all the M elements takes time $O(MD)$. Note that without sampling m, $O(M^2D)$ runtime per iteration would be needed, because each of the M elements of the gradient in eq. (4D.4) includes the sum over M terms. Therefore, each iteration takes time

$$O\left(T_{\tilde{x}} + T_y + \mathrm{poly}\log M + MD\right) = \mathcal{O}\left(MD + T_{\tilde{x}} + T_y\right). \tag{4D.8}$$

To bound the required number of iterations, we use an upper bound of the number of iterations in Algorithm 4.2 given in [JNN19], which shows that if we have for any $\alpha \in \mathcal{W}$

$$\|\nabla I(\alpha)\|_2 \leq L, \tag{4D.9}$$

the unbiased estimate $\hat{g}$ for any point $\alpha \in \mathcal{W}$ almost surely satisfies

$$\|\hat{g}\|_2 \leq L, \tag{4D.10}$$

and the diameter of $\mathcal{W}$ is bounded by some $\mathrm{diam}\,\mathcal{W} \leq d$, then after T iterations, with probability greater than $1-\delta$, Algorithm 4.2 returns α satisfying

$$\epsilon = O\left(dL\sqrt{\frac{\log\frac{1}{\delta}}{T}}\right), \tag{4D.11}$$

where we write

$$\epsilon = I(\alpha) - \min_{\alpha\in\mathcal{W}}\{I(\alpha)\}. \tag{4D.12}$$

In the following, we bound d and L in eq. (4D.11) to clarify the upper bound of the required number of iterations T in our setting.

To show a bound of d, recall the assumption that we are given a sufficiently large number M of features for achieving the learning in our setting. Then, [Bac17] has shown that with the M features sampled from the weighted probability distribution $Q(v_m)P^\tau(v_m)$ by Algorithm 4.1, the learning to the accuracy $O(\epsilon)$ can be achieved with coefficients satisfying

$$\|\beta\|_2^2 = O\left(\frac{1}{M}\right), \tag{4D.13}$$

where $\beta = (\beta_0, \ldots, \beta_{M-1})^{\mathrm{T}}$ is defined for each m as

$$\beta_m = \sqrt{Q(v_m)}\alpha_m. \tag{4D.14}$$

This bound yields

$$\sum_{m=0}^{M-1} Q(v_m)\alpha_m^2 = O\left(\frac{1}{M}\right). \tag{4D.15}$$

In the worst case, a lower bound of the left-hand side is

$$\sum_{m=0}^{M-1} Q(v_m)\alpha_m^2 \geq Q_{\min} \|\alpha\|_2^2, \tag{4D.16}$$

where $Q_{\min}$ is given by

$$Q_{\min} = \min\{Q(v_m) : m \in \{0, 1, \ldots, M-1\}\}. \tag{4D.17}$$

Note that as discussed in Appendix A, in the parameter region of sampling optimised random features that are weighted by importance and that nearly minimise M, the minimal weight $Q_{\min}$ of these features is expected to be sufficiently large compared to ϵ, not dominating the runtime. We still keep $Q_{\min}$ in our analysis to bound the worst-case runtime. From eq. (4D.15) and eq. (4D.16), we obtain an upper bound of the norm of α minimising I

$$\|\alpha\|_2^2 = \mathcal{O}\left(\frac{1}{MQ_{\min}}\right). \tag{4D.18}$$

Thus, it suffices to choose the parameter region $\mathcal{W}$ of α as an M-dimensional ball of centre 0 and of radius $\mathcal{O}\left(\frac{1}{\sqrt{MQ_{\min}}}\right)$, which yields the diameter

$$d = \mathcal{O}\left(\frac{1}{\sqrt{MQ_{\min}}}\right). \tag{4D.19}$$

As for a bound of L, we obtain from eq. (4D.5)

$$\|\hat{g}\|_2 = \mathcal{O}\left(M\|\alpha\|_2 + \sqrt{M}\right) = \mathcal{O}\left(\sqrt{\frac{M}{Q_{\min}}} + \sqrt{M}\right) = \mathcal{O}\left(\sqrt{\frac{M}{Q_{\min}}}\right), \tag{4D.20}$$

where we take the worst case of small $Q_{\min}$, and we use bounds

$$\sqrt{\sum_{m=0}^{M-1} |M\alpha_m e^{2\pi i v_m \cdot \tilde{x}_t}|^2} = \mathcal{O}(M\|\alpha\|_2), \tag{4D.21}$$

$$\sqrt{\sum_{m=0}^{M-1} y_t^2} = \mathcal{O}(\sqrt{M}). \tag{4D.22}$$

Since this upper bound of $\|\hat{g}\|_2$ is larger than $\|\nabla I(\alpha)\|_2$, we have

$$L = \mathcal{O}\left(\sqrt{\frac{M}{Q_{\min}}}\right). \tag{4D.23}$$

Using eq. (4D.19) and eq. (4D.23), we bound the right-hand side of eq. (4D.11)

$$\epsilon = \mathcal{O}\left(dL\sqrt{\frac{\log\frac{1}{\delta}}{T}}\right) = \mathcal{O}\left(\frac{1}{Q_{\min}}\sqrt{\frac{\log\frac{1}{\delta}}{T}}\right). \tag{4D.24}$$

Therefore, it follows that

$$T = \mathcal{O}\left(\frac{1}{\epsilon^2 Q_{\min}^2}\log\frac{1}{\delta}\right). \tag{4D.25}$$

Combining eq. (4D.1), eq. (4D.8), and eq. (4D.25), we obtain the claimed overall runtime. □

Chapter 5

A quantum search decoder for Natural Language Processing

Synopsis: Probabilistic language models, e.g. those based on an LSTM, often face the problem of finding a high probability prediction from a sequence of random variables over a set of words. This is commonly addressed using a form of greedy decoding such as beam search, where a limited number of highest-likelihood paths (the beam width) of the decoder are kept, and at the end the maximum-likelihood path is chosen. The resulting algorithm has linear runtime in the beam width. However, the input is not necessarily distributed such that a high-likelihood input symbol at any given time step also leads to the global optimum. Limiting the beam width can thus result in a failure to recognise long-range dependencies. In practice, only an exponentially large beam width can guarantee that the global optimum is found: for an input of length n and average parser branching ratio R, the baseline classical algorithm needs to query the input on average R^n times.

In this work, we construct a quantum algorithm to find the globally optimal parse with high constant success probability. Given the input to the decoder is distributed like a power-law with exponent $k > 0$, our algorithm yields a runtime $R^{nf(R,k)}$, where $f \leq 1/2$, and $f \to 0$ exponentially quickly for growing k. This implies that our algorithm always yields a super-Grover type speedup, i.e. it is more than quadratically faster than its classical counterpart. The algorithm is based on a recent quantum maximum finding algorithm, which we combine with an advice-based query analysis for quantum search; it is known that the latter cannot be used to speed up an equivalent classical algorithm. The quantum search decoder requires a quantum procedure that can sample from the grammar to be parsed, but in a biased fashion: the weight of each word in the sequence is determined by the sequence of random variables given as input. We explicitly construct such a quantum sampling subroutine for the case where a classical uniform sampler is known (e.g. for regular or context-free languages).

We further modify our procedure to recover a quantum beam search variant, which enables an even stronger empirical speedup, while sacrificing accuracy. Finally, we

apply this quantum beam search decoder to Mozilla's implementation of Baidu's *DeepSpeech* neural net, which we show to exhibit such a power law word rank frequency, underpinning the applicability of our model.

5.1 Background and Context

A recurring task in the context of parsing and neural sequence to sequence models—such as machine translation [SMH11, SVL14], natural language processing [Sch14] and generative models [Gra13]—is to find an optimal path of tokens (e.g. words or letters) from a sequential list of probability distributions. Such a distribution can for instance be produced at the output layer of a recurrent neural network, e.g. a long short-term memory (LSTM). The goal is to decode these distributions by scoring all viable output sequences (paths) under some language model, and finding the path with the highest score.

Nowadays, the de-facto standard solution is to use a variant of beam search [STN94, VCS+16, WR16, KMC+18] to traverse the list of all possible output strings. Beam search stores and explores a constant sized list of possible decoded hypotheses at each step, compared to a greedy algorithm that only considers the top element at each step. Beam search thus interpolates between a simple greedy algorithm, and best-first search; but just like greedy search, beam search is not guaranteed to find a global optimum. Furthermore, beam search suffers from sensitivity to the predicted sequence length; improving the algorithm itself [MC18, YHM18], as well as finding new decoding strategies [FLD18, HBF+19], is an ongoing field of research.

A related question is found in transition based parsing of formal languages, such as context-free grammars [HMU01, ZC08, ZN11, ZQH15, DBL+15]. In this model, an input string is processed token by token, and a heuristic prediction (which can be based on various types of classifiers, such as feed forward networks) is made on how to apply a transition at any one point. As in generative models and decoding tasks, heuristic parsing employs beam search, where a constant sized list of possible parse trees is retained in memory at any point in time, and at the end the hypothesis optimising a suitable objective function is chosen. Improvements of beam-search based parsing strategies are an active field of research [BBD16, BMP+16, VG18].

In essence, the problem of decoding a probabilistic sequence with a language model—or probabilistically parsing a formal grammar—becomes one of performing search over an exponentially-growing tree, since at each step the list of possible sequences branches with degree up to the number of predicted words. The goal is to find a path through this search space with the highest overall score. Due to runtime and memory constraints, a tradeoff has to be made which limits any guarantees on the performance of the search strategy.

As we have seen throughout the previous chapters, quantum computing has shown promise as an emerging technology to efficiently solve some instances of difficult computing tasks in fields ranging from optimisation [GAW19], linear algebra [HHL09, BCO+17], simulation of quantum systems [Llo96], distributional property testing [MdW16], and language processing [WBS+19, AGS18], to machine learning [CT17, JZW+18, DLD17, CL18, Bau18, BL18]. While quantum computers are not yet robust enough to evaluate any of these applications on sample sizes large enough to claim an empirical advantage, a structured search problem such as language decoding is a prime candidate for a quantum speedup.

While the most naïve search problems can be sped up using Grover's search algorithm (or one of its variants, such as fixed point/oblivious amplitude amplification), finding good applications for quantum algorithms remains challenging, and super-quadratic speedups (such as Shor's for prime factoring [NC10]) are rare. As we saw in Chapter 4, recently several exponentially-faster algorithms (such as quantum recommender systems [KP17b], or dense low rank linear algebra [WZP18]) have been fully or partially dequantised, since their reliance on the QRAM input model and low rank assumptions if classically available, can yield an exponential speedup without the need for quantum computing [Tan19].

Our quantum search decoder *does not* rely on QRAM. Our novel algorithmic contribution is to analyse a very recent quantum maximum finding algorithm [VGG+17] and its expected runtime when provided with a biased quantum sampler for a formal grammar that we developed, under the promise that at each step the input tokens are non-uniformly distributed.

For the case of finding the most likely parsed string, the close connection between decoding a probabilistic sequence and sampling from it yields precisely the quadratic speedup expected from applying quantum amplitude amplification to an unstructured search problem.

We obtain a more striking advantage in the case that the input sequence is just serving as advice on where to find the top scoring parse under a secondary metric—i.e. where the element with the highest score is *not necessarily* the one with the highest probability of occurring when sampled. In that case, our proposal is always more than quadratically faster than its classical counterpart, and the speedup becomes more pronounced the better the advice state.

5.2 Main Results

In this chapter, we address the question of decoding a probabilistic sequence of words, letters, or generally tokens, obtained e.g. from the final softmax layer of a recurrent neural network, or given as a probabilistic list of heuristic parse transitions. These models are essentially identical from a computational perspective. Hence, we give the following formal setup, and will speak of a decoding task, leaving implicit the two closely-related applications.

Given an alphabet Σ, we expect as input a sequence of random variables $X_1, X_2, \ldots, X_n$ over Σ, distributed as $X_i \sim \mathcal{D}_i^\Sigma$. The distributions $\mathcal{D}_i^\Sigma$ can in principle vary for each i; furthermore, the X_i can either be independent, or include correlations. The input model is such that we are given this list of distributions explicitly, e.g. as a table of floating point numbers; for simplicity of notation we will continue to write X_i for such a table. The decoding machine M is assumed to ingest the input one symbol at a time, and branch according to some factor R at every step; for simplicity we will assume that R is constant (e.g. an upper bound to the branching ratio at every step). As noted, M can for instance be a parser for a formal grammar (such as an Earley parser [Ear70]) or some other type of language model; it can either accept good input strings, or reject others that cannot be parsed. The set of configurations of M that lead up to an accepted state is denoted by Ω; we assume that everything that is rejected is mapped by the decoder to some type of sink state $\omega \neq \Omega$.

We allow M to make use of a heuristic that attempts to guess good candidates for the next decoding step. Furthermore, this heuristic can also depend on the input, i.e. we have a function $H : \Sigma \times \Omega \mapsto \Omega$. We allow H to itself be an automaton, possibly with a stack, and even a full-fledged Turing machine is a natural extension of this model. Since we are not interested in the complexity of the heuristic itself, we simply distinguish between a stateful and stateless heuristic by regarding them as randomised automata with or without correlations respectively, but otherwise assume they always produces the expected output in unit time.

It is not difficult to see that the randomised input setting is more generic than employing a heuristic at the decoding step. In this light, we will restrict our discussion to a decoder M that processes a token sequence step by step, and such that its state itself now simply becomes a sequence $(M_i)_{i\leq n}$ of random variables. Described as a stochastic process, the M_i are random variables over the set Ω of internal configurations after the automaton has ingested X_i, given that it has ingested $X_{i-1}, \ldots, X_1$ prior to that, with a distribution $\mathcal{D}_i^\Omega$. The probability of decoding a specific accepted string $x = (x_1, \ldots, x_n)$ is then given by the product of the conditional probabilities

$$\begin{aligned}\Pr(M_n = x) :&= \mathcal{N} \prod_{i=1}^{n} \Pr(X_i = x_i | X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2}, \ldots, X_1 = x_1) \\ &= \mathcal{N} \Pr(X_n = x_n, X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \ldots, X_1 = x_1) \\ &= \mathcal{N} \Pr(X = x)\end{aligned} \tag{5.1}$$

where $\mathcal{N} = 1/(\sum_{x\in\Omega} \Pr(X = x))$, and in a slight abuse of notation we write $M_n = x$ when we mean $M_n = y(x)$, where $y(x)$ is the configuration of the parser M that was provided with some input to produce the parsed string x; similarly we will write $x \in \Omega$ for an accepted string/decoded path.[1]

[1] Since there is a one-to-one mapping between accepted strings and parser configurations $y(x)$ that lead up to it, it is unambiguous to write $\Pr(M_n = x)$ as the probability that the decoder M after n steps produced

The obvious question is: which final accepted string of the decoder is the most likely? This is captured in the following computational problem.

Most Likely Parse

Input: Decoder M over alphabet Σ and with set of accepting configurations Ω. Sequence of random variables $(X_i)_{i\leq n}$ over sample space Σ.

Question: Find $\sigma = \operatorname{argmax}_{x\in\Omega} \Pr(M_n = x)$.

Classically, it is clear that if we have a procedure that can sample the random variable M_n efficiently, then we can find the most likely element with an expected runtime of $1/\Pr(M_n = \sigma)$, as this is the number of samples we are expected to draw to see the element once. While such sampling algorithms might be inefficient to construct in general, we emphasise that the question of drawing samples from strings over a formal language is an active field of research, and algorithms to sample uniformly are available for a large class of grammars: in linear time for regular languages [BG12, ODG13], and context-free grammars/restrictions thereof can be sampled uniformly [McK97, GPS01, HC83, GJK+97, Den96] and also with word bias [RPW13, LP13, DRT00, Pon12].

In Theorem 5.6 and Section 5.3.1, we lift a classical uniform sampler (e.g. given as a bounded-error probabilistic poly-time BPP algorithm with coin flips as a source of randomness) to a biased quantum sampler, which we can use to obtain a quantum advantage when answering Most Likely Parse. We note that the techniques therein may well be used to obtain a classical Monte Carlo procedure to sample from M_n. In what follows, we will therefore assume that obtaining a sample of M_n within a sufficiently-small error margin can be done with a (uniform) family of classical poly-time randomised circuits denoted $(S_n)_{n\geq 1}$ that is given to us. Yet in order to be precise, we will explicitly keep the sampling runtime separate from the rest of the complexity analysis.

We prove the following result:

Theorem 5.1. *For an input sequence of length n of random variables to a parser with a classical sampling runtime $T(n)$, there exists a quantum search algorithm answering* Most Likely Parse *with certainty, using $\pi/4\sqrt{\Pr(M_n = \sigma)}$ iterations. In each iteration, it runs a quantum circuit for the sampler in $\mathcal{O}(T(n)^{1.6})$ time.*

As explained, this theorem formalises the expected quadratic speedup of the runtime as compared to a classical algorithm based on sampling from M_n. This works because we know that the element to search for is the one that also occurs with the highest likelihood within the distribution. Given the input to the parser is power-law distributed (see Definition 5.8), this allows us to formulate the following corollary.

the output string (or path) x, which essentially identifies Ω and the subset of strings in the language of length n that can be accepted by the decoder.

Corollary 5.2. *If the* $X_i \sim \text{Power}_R(k)$*, then answering* Most Likely Parse *requires at most* $1/H_R(k)^{n/2}$ *queries.*[2]

Yet *a priori*, it is not clear that the weight of a decoded path (e.g. the product of probabilities of the input tokens) also corresponds to the highest score we wish to assign to such a path. This becomes obvious in the setting of a heuristic applied to a live translation. While at every point in time the heuristic might be able to guess a good forward transition, it might well be that long range correlations strongly affect the likelihood of prior choices. Addressing these long-distance "collocations" is an active field of research [ZQH15].

To give an exemplary illustration, consider the sentence

$$\text{Who does Bill want to } \begin{cases} \text{replace} \\ \text{win} \end{cases} \text{ ?}$$

This example contains a so-called clausally unbounded long distance dependency [Dąb08]. In the top and bottom branches respectively, the word 'Who' is either the direct object of the verb 'replace', or the direct object of the verb 'want' and hence by implication the subject of the verb 'win'. The parser cannot distinguish between the two cases until it has seen the verb, so both choices of interpretation have to be retained. Several recent studies evaluate how parsers perform in the presence of such dependencies in the input. The findings of [KHQ+18] indicate that LSTM models are capable of using about 200 tokens of context on average, but that they sharply distinguish nearby context (≈ 50 tokens) from the distant past. Furthermore, such models appear to be very sensitive to word order within the most recent context, but ignore word order in the long-range context (more than 50 tokens away). On the other hand, specialised dependency parsers (such as the MSTParser and MaltParser) which are equipped with simple post-processing to extract unbounded dependencies from the basic dependency tree are able to correctly recall unbounded dependencies only roughly 50% of the time [NRM+10].

To address this setting formally, we assume there is a scoring function $F : \Omega \longrightarrow \mathbb{R}$, which assigns scores to all possible decoded paths. Without loss of generality, there will be one optimal string which we denote with $\tau = \text{argmax}_{x \in \Omega} F(x)$. Furthermore, we order all decoded strings Ω in some fashion, and index them with numbers $i = 1, \ldots, |\Omega|$. Within this ordering, τ can now be in different places—either because the heuristic guesses differently at each step, or because the input sequence varied a little. We denote the probability that the marked element τ is at position i with p_i. In essence, the position where τ is found is now a random variable itself, with probability mass function $\Pr(\text{finding } \tau \text{ at index } i) = p_i$.

[2] $H_R(k) = \sum_{i=1}^{R} i^{-k}$ denotes the R^{th} harmonic number of order k.

For the decoder probabilities $\Pr(M_n = x)$ to serve as *good advice* on where to find the highest-score element under the metric F, we demand that

$$\Pr(M_n = \text{string with index } i) = p_i. \tag{5.2}$$

Loosely speaking, what eq. (5.2) means is that the final distribution over the states of the decoder puts high mass where the highest-scoring element often occurs.

To be precise, we define the following problem.

Highest Score Parse

Input: Decoder M over alphabet Σ and with state space Ω. Sequence of random variables $(X_i)_{i \leq n}$ over sample space Σ. Scoring function $F : \Omega \longrightarrow \mathbb{R}$.

Promise: Eq. (5.2).

Question: Find $\tau = \operatorname{argmax}_{x \in \Omega} F(x)$.

What is the classical baseline for this problem? As mentioned in [Mon11], if p_x is the probability that x is the highest-scoring string, then in expectation one has to obtain $1/p_x$ samples to see x at least once. Any procedure based on sampling from the underlying distribution p_x thus has expected runtime

$$\sum_{x \in \Omega} \frac{1}{p_x} \times p_x = |\Omega|. \tag{5.3}$$

In a sense this is as bad as possible; the advice gives zero gain over iterating the list item by item and finding the maximum in an unstructured fashion. Yet provided with the same type of advice, a quantum computer can exhibit tremendous gains over unstructured search.

Theorem 5.3. *With the same setup as in Theorem 5.1 but under the promise that the input tokens are iid with $X_i \sim \text{Power}_{|\Sigma|}(k)$ over alphabet Σ (Definition 5.8), that the decoder has a branching ratio $R \leq |\Sigma|$, and that we can uniformly sample from the grammar to be decoded, there exists a quantum algorithm* QuantumSearchDecode *answering* Highest Score Parse *with an expected number of iterations*

$$\text{RT}_1(R, k, n) = \mathcal{O}\left(R^{nf(R,k)}\right), \quad \textit{where } f(R,k) = \log\left(\frac{H_R(k/2)}{H_R(k)^{1/2}}\right) \Big/ \log R,$$

and where $H_R(k)$ denotes the R^{th} harmonic number of order k. Each iteration runs a quantum circuit for the sampler in time $\mathcal{O}(T(n)^{1.6})$.

There exists no classical algorithm to solve this problem based on taking stochastic samples from the decoder M that requires less than $\Omega(R^n)$ samples.

While the runtime for Algorithm 5.1 used to prove Theorem 5.3 is based on an analytical bound we found that numerically it comes close to the true expected query complexity of the search decoding algorithm. The exponent $f(R,k)$ indicates the speedup over a classical implementation of the decoding algorithm (which would have to search over R^n elements). We find that $f(R,k) < 1/2$ for all $R, k > 0$, and in fact $f(R,k) \longrightarrow 0$ exponentially quickly with k; we formulate the following corollary.

Corollary 5.4. *For $k > 0$,* QuantumSearchDecode *is always faster than plain Grover search (with runtime $\propto R^{n/2}$); the extent of the speedup depends on the branching ratio R and the power law exponent k, and is plotted in fig. 5.1.*

Finally, in Section 5.5 we modify the full quantum search decoder by only searching over the paths with likelihood above some given threshold (that we allow to depend on n in some fashion), effectively turning the decoder into a type of beam search, but where the pruning only happens at the very end. This means that in contrast to beam search, the top scoring element is found over the *globally* most likely parsed paths, avoiding the risk early beam pruning brigs. We analyse the runtime of Algorithm 5.2 for various choices of beam width numerically, and analyse its performance on a concrete example—Mozilla's *DeepSpeech* implementation, a speech-to-text LSTM which we show to follow a power-law token distribution at each output frame.

*F*or *DeepSpeech*, we empirically find that input sequence lengths of up to 500 tokens can realistically be decoded, with an effective beam width of 10^{15} hypotheses—while requiring $\approx 3e6$ search iterations (cf. fig. 5.9).

We want to emphasise that the fact the letters a-z follow Zipf's law with respect to their occurrence in English sentences (see e.g. [Egg00, Pia14]) plays no role in attaining the speedup. In addition to fig. 5.7, we verified that when only collecting those output frames of *DeepSpeech* where, say, "t" is the most likely prediction, the distribution over all letters—sorted by rank, i.e. sorted from most to least likely prediction—is already a power-law. This is a feature of the output of the model, and not necessarily a property of the underlying data the model was trained on. In our context this means that the Softmax output layer of the LSTM has to yield a power-law probability distribution. How frequently a given letter is the most likely prediction—which is itself known to be a power-law, as mentioned—is not important.

5.3 Quantum Search Decoding

In this section, we give an explicit algorithm for quantum search decoding. As mentioned before (see Section 5.2), we assume we have access to a classical algorithm that, given a list of transition probabilities determined by the inputs $X_1, \ldots, X_n$, yields a random sample drawn from the distribution—either uniformly, or with weights corresponding to eq. (5.1). Since

either way this sampler is given as a classical probabilistic program, we first need to translate it to a quantum algorithm. We start with the following lemma.

Lemma 5.5. *For a probabilistic classical circuit with runtime $T(n)$ and space requirement $S(n)$ on an input of length n, there exists a quantum algorithm that runs in time $\mathcal{O}(T(n)^{\log_2 3})$ and requires $\mathcal{O}(S(n)\log T(n))$ qubits.*

Proof. Follows from [BTV01, Th. 1]: any non-reversible computation requiring time T and space S can be simulated reversibly in time $T' = 3^k 2^{\mathcal{O}(T/2^k)}$ and space $S' = (1+\mathcal{O}(k))S$, for a $0 \leq k \leq \log_2 T$ chosen arbitrarily. Choose $k = \log_2 T$, then $S' = (1+\mathcal{O}(\log_2 T))S$, and $T' = \mathcal{O}(T^{\log_2 3})$. Now translate this reversible probabilistic classical circuit into a quantum circuit—e.g. using the Solovay-Kitaev theorem [NC10], which incurs an at most logarithmic runtime overhead. □

5.3.1 Biased Quantum Sampling from a Regular or Context-Free Grammar

As an immediate consequence, given a classical probabilistic sampling algorithm that *can* produce strings $a_1 a_2 \cdots a_n$ of a language such that each string is weighted by the probability of the symbol a_i occurring at site i (i.e. eq. (5.1)), we can obtain a quantum circuit that produces a quantum state which is a weighted superposition over all such strings.

In addition to the weighted superposition, however, we would like to have the weight of each state in the superposition spelled out as an explicit number in an extra register (e.g. as a fixed precision floating point number), i.e. in the form

$$\mathbf{U}_\mu |0\rangle = |\mu\rangle \propto \sum_{q\in\Omega} \sqrt{p_q}\, |h_q\rangle\, |p_q\rangle\, |q\rangle\,, \tag{5.4}$$

where Ω is the set of accepted strings reachable by the decoder in n steps, $|h_q\rangle$ is an ancillary state that depends on q and is contained in the decoder's work space, and if q is a state reached by reading the input sequence $a_{1j_1}, a_{2j_2}, \ldots, a_{nj_n}$. The weights $p_q = \prod_{i=1}^n p_{ij_i}$.

However, it is not clear that such a weighted sampler (dependent input or not) is available at all. As outlined in the introduction, we know there exist *uniform* classical probabilistic samplers for large classes of grammars, e.g. for regular languages in linear time (e.g. [ODG13]) and polynomial time for variants of CFGs (e.g. [GPS01]). Again keeping the uniform sampler's runtime separate from the rest of the algorithm, we can raise classical uniform samplers to obtain a biased quantum state preparator for $|\mu\rangle$.

Theorem 5.6. *Given a classical probabilistic algorithm that, in time $T(n)$, produces uniform samples of length n from a language, and given a list of independent random variables $X_1, \ldots, X_n$ with pdfs $p_{i,j}$ for $i = 1, \ldots, n$ and $j = [\Sigma]$, we can construct a quantum circuit*

$\mathbf{U}_{\mu'}$ *that produces a state* $|\mu'\rangle$ ϵ*-close to the one in eq. (5.4). The algorithm runs in time* $\mathcal{O}(T(n)^{1.6} \times n^3\kappa/\epsilon^2)$*, where* κ *is an upper bound on the relative variance of the conditional probability* $Pr(a|s_1 \dots s_i)$*.*

Proof. Using lemma 5.5, translate the parser—which takes its input step by step—into a sequence of unitaries $\mathbf{U} = \mathbf{U}_n \cdots \mathbf{U}_1$. Considering a single unitary $\mathbf{U}_i$ at the i^{th} step, it is clear that it can be broken up into a family of unitaries $(\mathbf{U}_i^a)_{a\in\Sigma}$, such that each $\mathbf{U}_i^a$ is a specialization of $\mathbf{U}_i$ when given a fixed input symbol $a \in \Sigma$. We define $\mathbf{V}_i^a$ to perform $\mathbf{U}_i^a$, and in addition store the input a in some ancillary workspace, e.g. via $\mathbf{V}_i^a |\phi\rangle |\xi\rangle = (\mathbf{U}_i^a |\phi\rangle) |\xi \oplus a\rangle$. Then define the block-diagonal unitary $\mathbf{V}_i := \text{diag}(\mathbf{V}_i^a)_{a\in\Sigma}$, which acts like a controlled matrix, meaning that if $\mathbf{V}_i$ acts on some state $|\psi\rangle = |a\rangle |\phi\rangle$, then $\mathbf{V}_i |\psi\rangle = |a\rangle \mathbf{V}_i^a |\phi\rangle$. Naturally this works in superposition as well, e.g. $\mathbf{V}_i(\alpha |a\rangle + \beta |b\rangle) |\phi\rangle = \alpha |a\rangle \mathbf{V}_i^a |\phi\rangle + \beta |b\rangle \mathbf{V}_i^b |\phi\rangle$. We further assume that the $\mathbf{V}_0^a$ take as initial state $|0\rangle |q_0\rangle$.

The final step in augmenting the parser is to extend $\mathbf{V}_i$ to carry out a controlled multiplication: for a finite set of numbers $F \subset \mathbb{R}$ (e.g. fixed precision), and $d_1, d_2 \in F$, we write $\mathbf{V}_i(d_1) |a\rangle |d_2\rangle |\phi\rangle = |a\rangle |d_1 \times d_2\rangle \mathbf{V}_i^a |\phi\rangle$. We denote this extended unitary for step i with $\mathbf{U}_i'$.

The next ingredient we take is the classical uniform language sampler. Once again using lemma 5.5, we raise it to a unitary $\mathbf{W}$, which takes as input a prefix $s_m := a_1 \cdots a_m$ of the m previously-seen tokens, and a list of distributions over the future weights $W_m := (p_{i,j})_{m<j\leq n}$. These are the distribution of tokens for each of the X_j. We then augment $\mathbf{W}$ to a circuit $\mathbf{W}'$ that quantumly performs the following classical calculations, in superposition over its input:

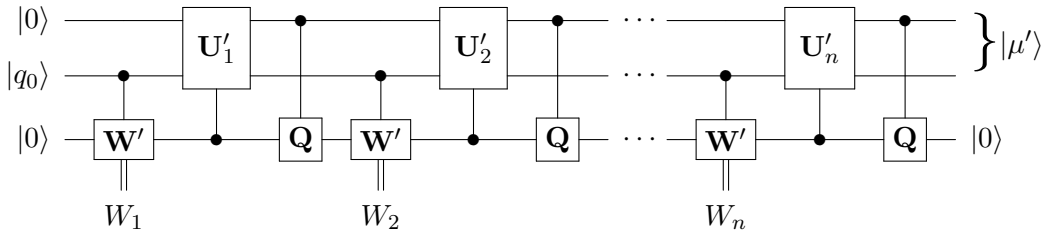
1. Draw S samples uniformly at random from the grammar starting at strings prefixed with s_m; denote this list with $B := \{b_1, \dots, b_S\}$.

2. Group the samples B into bins C_a of samples with the same first token $a \in \Sigma$, i.e. $C_a = \{b \in B : b = a?? \cdots ?\}$, where ? stands for any token in the alphabet Σ.

3. Calculate the total of the probabilities of each bin C_a where each element is weighted with respect to the future probabilities given in list W_m, which yields a distribution $D = (d_a)_{a\in\Sigma}$.

It is straightforward to write the unitary $\mathbf{W}'$ that then takes a state $|00\rangle \in \mathcal{H}_F \otimes \mathbb{C}^\Sigma$—the first register for storing a number in F, and the second for storing a letter—and a list of such weights D to a weighted superposition $\mathbf{W}'(D) |0\rangle = \sum_{a\in\Sigma} \sqrt{d_a} |d_a\rangle |a\rangle$ (where for the sake of simplicity we drop the scratch space register that is certainly required). Furthermore, we need a controlled unitary $\mathbf{Q}$ that, given some state $|h\rangle |a\rangle$ where $h = h(a)$ in some specified fashion—which we can demand the $\mathbf{V}_i^a$ produce—uncomputes a and d_a from the second

register, i.e. $\mathbf{Q}\,|h\rangle\,|d_a\rangle\,|a\rangle = |h\rangle\,|00\rangle$. Together with the sequence of parser unitaries $\mathbf{U}'_i$, the overall quantum circuit $\mathbf{U}_\mu$ can then be constructed as follows:



For a partial string $s_1 s_2 \cdots s_i$ of length i, we denote the set of all strings in the grammar prefixed with letters of s with $\mathcal{A}(s_1 \dots s_i)$. At every step i in the algorithm we sample the expectation value of a future hypothesis continuing with some token a, weighted by their individual likelihood p_{ij}. The sampling procedure then yields an empirical distribution $(d_a)_{a\in\Sigma}$, which we denote with

$$d_a = f_*^{si}(a) = \sum_{j=1}^{S} \chi\left[b_j \in \mathcal{A}(s_1 \dots s_i a)\right] p(b_j) \Big/ \sum_{j=1}^{S} p(b_j), \tag{5.5}$$

where the S sampled hypothesis are given in list $B = \{b_1, \dots, b_S\}$ with individual letters $b_j = b_{j,1} \cdots b_{j,n}$). As usual, $\chi[\cdot]$ denotes the indicator function, and

$$p(b_j) := \prod_{k=1}^{n} p_{k,b_{jk}}.$$

Our goal is to show that the algorithm reproduces the desired weight distribution given in eq. (5.4), i.e.

$$\Pr(s) = \prod_{i=0}^{n-1} \Pr(s_{i+1}|s_1 \dots s_i) \quad \text{where } \Pr(s_{i+1}|s_1 \dots s_i) = \frac{\sum_{x\in\mathcal{A}(s_1\dots s_i)} p(x)}{\sum_{x\in\mathcal{A}(s_1\dots s_{i+1})} p(x)}$$

To estimate the total probability distribution to error ϵ in total variation distance, it suffices to approximate each conditional distribution to error ϵ/n, and thus we must show how many samples S are required for d_a to be a good estimator for $\Pr(a|s_1 \dots s_i)$.

First note that $f^{si}(a) = u^{si}(a)/v^{si}$ for

$$u^{si}(a) := \frac{1}{S}\sum_{j=1}^{S} \chi\left[b_j \in \mathcal{A}(s_1 \dots s_i a)\right] p(b_j) \quad \text{and} \quad v^{si} := \frac{1}{S}\sum_{j=1}^{S} p(b_j) = \sum_{a\in\Sigma} u^{si}(a).$$

It is straightforward to calculate that

$$\mathbb{E}(u^{si}(a)) = \frac{1}{|\mathcal{A}(s_1 \dots s_i)|} \sum_{x\in\mathcal{A}(s_1\dots s_i a)} p(x) \quad \text{and} \quad \mathbb{E}(v^{si}) = \frac{1}{|\mathcal{A}(s_1 \dots s_i)|} \sum_{x\in\mathcal{A}(s_1\dots s_i)} p(x)$$

and so $\mathbb{E}(u^{si}(a))/\mathbb{E}(v^{si}) = \Pr(a|s_1 \dots s_i)$, the value we are trying to estimate.

Therefore it suffices to take enough samples S such that the $u^{si}(a)$ are close to their mean in relative error (and thus v^{si} is also close in relative error, since $v^{si} = \sum_a u^{si}(a)$).

Noting that $u^{si}(a) = \frac{1}{S}\sum_{j=1}^{S} Y_j$ for i.i.d. random variables Y_j, we have that $\mathrm{Var}(u^{si}(a)) = \frac{1}{S}\mathrm{Var}(Y)$. Therefore by Chebyshev's inequality, to get a ϵ/n relative error approximation requires the number of samples S to be at least

$$S \geq \frac{\mathrm{Var}(Y)}{\mathbb{E}(Y)^2(\epsilon/n)^2}.$$

By assumption $\mathrm{Var}(Y)/\mathbb{E}(Y)^2 \leq \kappa$, and so the total number of uses of the sampler over all n steps of the algorithm is $O(\kappa n^3/\epsilon^2)$ as claimed.

□

We remark that getting a precise handle on κ strongly depends on the grammar to be parsed and the input presented to it; it seems unreasonable to claim any general bounds as it will most likely be of no good use for any specific instance. However, we note that it is conceivable that if the input is long and reasonably independent of the language to be sampled, then κ should be independent of n, and $\kappa \approx 1/p(r_{\min})$, where $p(r)$ is the distribution of the input tokens at any point in time—e.g. $p(r) \propto r^{-k}$ as in a power law.[3]

We note that variants of this sampling algorithm are certainly possible: a naïve approach would be to just sample from the product-of-powerlaws distribution and postselect on the resulting strings being in the grammar; the performance of this will then depend on the number of strings in the grammar vs. the number of all possible strings. Another method could be to execute the uniform sampler in superposition, and perform amplitude amplification on the resulting quantum state to reintroduce the power-law bias. The number of amplification rounds will again depend on the distribution of the strings in the grammar.

5.3.2 The Quantum Search Decoder

The quantum algorithm underlying the decoder is based on the standard maximum finding procedure from [DH96, AK99], and its extension in [VGG+17] used in the context of SDP solvers.

The procedure takes as input a unitary operator $\mathbf{U}_\mu$ which prepares the advice state, and a scoring function F which scores its elements, and returns as output the element within the

[3] This should make intuitive sense: the branching ratios are already biased with respect to the number of future strings possible with prefix s; if the input sequence is independent of the grammar, then we would expect them to weigh the strings roughly uniformly; the extra factor of $1/p(r_{\min})$ simply stems from the weighing of the token we bin by, a.

Algorithm 5.1 Quantum search decoding.

function QuantumSearchDecode$_m(\mathbf{U}_\mu, F)$
  $bestScore \leftarrow -\infty$
  $counter \leftarrow 0$
  repeat
    $cmp \leftarrow [(\cdot) \mapsto (bestScore < \cdot)]$ ▷ comparator against current best score
    $|\psi\rangle \leftarrow$ ExponentialSearch$(\mathbf{U}_\mu, cmp \circ F)$ ▷ amplify elements $\geq$ pivot
    $bestScore \leftarrow \mathbf{M}_{\text{score}} |\psi\rangle$ ▷ measure new best score
    $counter \leftarrow counter + 1$
  until $counter = m$
end function

advice state that has the maximum score under F. As in Section 5.3.1, we assume that F can be made into a reversible quantum circuit to be used in the comparison operation. We also note that reversible circuits for bit string comparison are readily available [SR07], and can be implemented using quantum adder circuits [Gid18].

Algorithm 5.1 lists the steps in the decoding procedure. As a subroutine within the search loop, we perform exponential search with oblivious amplitude amplification [BCC+14].

As in the maximum finding algorithm, the expected query count for quantum search decoding is given as follows.

Theorem 5.7 ([VGG+17])**.** *If x is the highest-scoring string, the expected number of iterations in* QuantumSearchDecode *to find the maximum is* $\mathcal{O}(\min\{1/|\langle x|\mu\rangle|, \sqrt{n}\})$.

5.4 Power Law Decoder Input

In this section we formally prove that if the decoder is fed independent tokens that are distributed like a power law, then the resulting distribution over the parse paths yields a super-Grover speedup—meaning the decoding speed is faster than applying Grover search, which itself is already quadratically faster than a classical search algorithm that traverses all possible paths individually.

A power law distribution is the discrete variant of a Pareto distribution, also known as Zipf's law, which ubiquitously appears in the context of language features [Jäg12, SB16, Egg00, Pia14]. This fact has already been exploited by some authors in the context of generative models [GGJ11].

Formally, we define it as follows.

Definition 5.8. *Let A be a finite set with $|A| = R$, and $k > 1$. Then* $\text{Power}_R(k)$ *is the power law distribution over R elements: for $X \sim \text{Power}_R(k)$ the probability density function satisfies*

$\Pr(X = x) = r^{-k}/H_R(k)$ *for an element of rank* r*, where* $H_R(k)$ *denotes the* R^{th} *harmonic number of order* k*.*

We are interested in the Cartesian product of power law random variables, i.e. sequences of random variables of the form $(X_1, \ldots, X_n)$. Assuming the random variables $X_i \sim \text{Power}_R(k)$ are all independent and of rank r_i with pdf $q(r_i) = r_i^{-k}/H_R(k)$, respectively, it is clear that

$$p(r_1, \ldots, r_n) = \prod_{i=1}^{n} q(r_i) = \frac{1}{H_R(k)^n} \frac{1}{(r_1 \cdots r_n)^k}. \tag{5.6}$$

As in [Mon11], we can upper bound the number of decoder queries in QuantumSearchDecode by calculating the expectation value of the iterations necessary—given by Theorem 5.7—with respect to the position of the top element.

We assume that at every step, when presented with choices from an alphabet Σ, the parsed grammar branches on average $R \leq |\Sigma|$ times. Of course, even within a single time frame, the subset of accepted tokens may differ depending on what the previously-accepted tokens are. This means that if the decoder is currently on two paths β_1 (e.g. corresponding to "I want") and β_2 ("I were"), where the next accepted token sets are $\Sigma_1, \Sigma_2 \subset \Sigma$ (each different subsets of possible next letters for the two presented sentences), respectively, then we do *not* necessarily have that the total probability of choices for the two paths—$\Pr(\Sigma_1)$ and $\Pr(\Sigma_2)$—are equal. But what does this distribution over all possible paths of the language, weighted by eq. (5.1), look like?

Certainly this will depend on the language and type of input presented. Under a reasonable assumption of independence between input and decoded grammar, this becomes equivalent to answering the following question: let X a product-of-powerlaw distributions with pdf given in eq. (5.6), where every term is a powerlaw over Σ. Let Y be defined as X, but with a *random subset* of elements deleted; in particular, such that R^n elements are left, for some $R < |\Sigma|$. Is Y distributed as a product-of-powerlaws as in eq. (5.6), but over R elements at each step? In the case of continuous variables this is a straightforward calculation, and we postpone it to Appendix B; numerics suggest it also holds true for the discrete case.

But even if the input that the parser given is independent of the parsed grammar, it is not clear whether the *sample distribution* over R (i.e. sampling R out of $|\Sigma|$ power-law distributed elements) follows the same power law as the original one over Σ; this is in fact not the case in general [ZQH15]. However, it is straightforward to numerically estimate the changed power law exponent of a sample distribution given R and $|\Sigma|$—and we note that the exponent shrinks only marginally when $R < |\Sigma|$.

In this light and to simplify the runtime analysis, we therefore assume the decoder accepts exactly R tokens at all times during the parsing process (like an R-ary tree over hypotheses) with a resulting product-of-powerlaw distribution, and give the runtimes in terms of the

branching ratio, and not in terms of the alphabet's size. This indeed yields a fair runtime for comparison with a classical variant, since any classical algorithm will *also* have the aforementioned advantage (i.e. we assume the size of final elements to search over is R^n, which precisely corresponds to the number of paths down the R-ary tree).

5.4.1 Most Likely Parse: Query Bound

In this case F simply returns p_q as the score in eq. (5.4). It thus suffices to calculate the state overlap $|\langle x|\mu\rangle|$, under the assumption that x is the highest mass point of the probability density function. By eq. (5.6), we have $|\langle x|\mu\rangle|^2 = H_R^{-n}(k)$. The claim of Corollary 5.2 follows from these observations.

5.4.2 Highest Score Parse: Simple Query Bound

We aim to find a top element scored under some function F under the promise that $|\mu\rangle$ (eq. (5.4)) presents good advice on where to find it, in the sense of eq. (5.2). The expected runtimes for various power law falloffs k can be obtained by taking the expectation with respect to p_x as in [Mon11].

In order to do so, we need to be able to calculate expectation values of the cartesian product of power law random variables, where we restrict the domain to those elements with probability above some threshold. We start with the following observation.

Lemma 5.9. *If* QuantumSearchDecode *receives as input iid random variables* $X_1, \ldots, X_n$*, with* $X_i \sim \mathrm{Power}_R(k)$*, then the number of queries required to the parser is*

$$\mathrm{RT}_1(R, k, n) = \mathcal{O}\left(\frac{H_R(k/2)^n}{H_R(k)^{n/2}}\right).$$

Proof. The expectation value of $1/\langle x|\mu\rangle$ is straightforward to calculate; writing $\vec{r} = (r_1, \ldots, r_n)$, by eq. (5.6), we have

$$\begin{aligned}\mathbb{E}(1/\langle x|\mu\rangle) &= \sum_{\vec{r}} p(\vec{r}) \times \frac{1}{\sqrt{p(\vec{r})}} \\ &= \frac{1}{H_R(k)^{n/2}} \sum_{r_1=1}^{R} \cdots \sum_{r_n=1}^{R} \frac{1}{(r_1 \cdots r_n)^{k/2}}.\end{aligned}$$

The claim then follows from $\mathcal{O}(\min\{1/\langle x|\mu\rangle, \sqrt{n}\}) \leq \mathcal{O}(1/\langle x|\mu\rangle)$. □

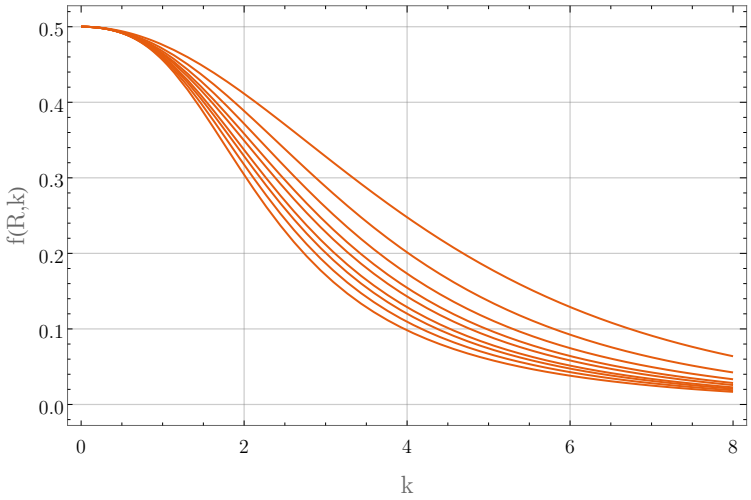


Figure 5.1: Exponent $f(R,k)$ of expected runtime of QuantumSearchDecode, when fed with a power law input with exponent k, over R alphabet tokens; plotted are individual curves for the values $R \in \{3,5,10,15,20,30,40,60,100\}$, from top to bottom. For all R, $f(R,k)$ drops off exponentially with growing k.

We observe that the runtime in lemma 5.9 is exponential in n. Nevertheless, as compared to a Grover algorithm—with runtime $R^{n/2}$—the base is now dependent on the power law's falloff k. We can compare the runtimes if we rephrase $\mathrm{RT}_1(R,k,n) = R^{nf(R,k)}$, by calculating

$$\left(\frac{H_R(k/2)}{H_R(k)^{1/2}}\right)^n = R^{nf(R,k)} \quad \Longleftrightarrow \quad f(R,k) = \log\left(\frac{H_R(k/2)}{H_R(k)^{1/2}}\right) \Big/ \log R.$$

We observe that the exponent $f(R,k) \in (0,1/2]$; its precise dependency on k for a set of alphabet sizes R is plotted in fig. 5.1. For growing k, $f(R,k)$ falls off exponentially.

5.4.3 Most Likely Parse: Full Query Bound

A priori, it is unclear how much we lose in lemma 5.9 by upper-bounding $\mathcal{O}(\min\{1/\langle x|\mu\rangle, \sqrt{n}\})$ by $\mathcal{O}(1/\langle x|\mu\rangle)$—so let us be more precise. In order to evaluate the expectation value of the minimum, we will break up the support of the full probability density function $p(\vec{r})$ into a region where $p(\vec{r}) > 1/R^n$, and its complement. Then, for two constants C_1 and C_2, we have for the full query complexity

$$\mathrm{RT}_2(R,k,n) = \mathbb{E}\left[\mathcal{O}(\min\{1/\langle x|\mu\rangle, \sqrt{n}\})\right] = C_1 \sum_{\vec{r}:p(\vec{r})>1/R^n} \sqrt{p(\vec{r})} + C_2\sqrt{n} \sum_{\vec{r}:p(\vec{r})\leq 1/R^n} p(\vec{r}). \tag{5.7}$$

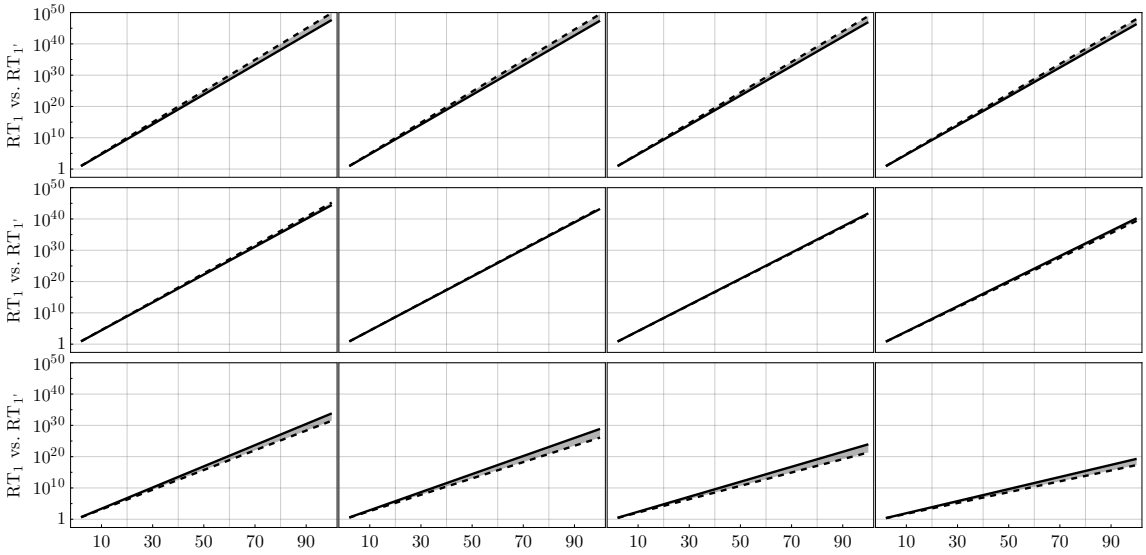


Figure 5.2: Expected runtime $\mathrm{RT}_1(R,k,n)$ as evaluated for $R = 10$ and various k (top row: $k \in \{0.2, 0.4, 0.6, 0.8\}$, middle row: $k \in \{1.2, 1.4, 1.6, 1.8\}$, bottom row: $k \in \{2.5, 3, 3.5, 4\}$, always from left to right), vs. the same parameters used for $\mathrm{RT}_{1'}(R,k,n)$ (dashed line), where the discrete probabilities from the power law are approximated with a continuous Pareto distribution. On the x-axis is the length of the input sequence n.

In order to calculate sums over sections of the pdf $p(\vec{r})$, we first move to a truncated Pareto distribution by making the substitution

$$\sum_{r\in A} \frac{1}{r^k} \longrightarrow \int_A \frac{1}{r^k}\mathrm{d}r \quad \text{and} \quad H_R(k) \longrightarrow h_R(k) := \int_1^R \frac{1}{r^k}\mathrm{d}r = \begin{cases} \frac{R^{1-k}-1}{1-k} & k \neq 1 \\ \log R & \text{otherwise.} \end{cases}$$

While this does introduce a deviation, its magnitude is minor, as can be verified numerically throughout (see fig. 5.2, where we plot both RT_1 and the continuous variant $\mathrm{RT}_{1'}(R,k,n) := h_R^n(k/2)/h_R^{n/2}(k)$).

The type of integral we are interested in thus takes the form

$$M(R,k_1,k_2,c,n) := \frac{1}{h_R^n(k_1)} \iiint_1^R \frac{\chi(r_1\cdots r_n \leq c)}{(r_1\cdots r_n)^{k_2}}\mathrm{d}r_1\cdots\mathrm{d}r_n, \tag{5.8}$$

where k_1 is not necessarily equal to k_2, and typically $c = (R/h_R(k_1))^{n/k_1}$, which would reduce to the case we are seeking to address in eq. (5.7). Here, $\chi(\cdot)$ denotes the characteristic function of a set, i.e. it takes the value 1 where the premise is true, and 0 otherwise. It is possible to integrate eq. (5.8) numerically for small n; however, due to the high dimensionality and the flat tail, convergence suffers drastically already for $n > 6$. Similarly, evaluating the integral with a computer algebra system takes significant time for larger n and produces ever growing

expressions that are hard to handle, as the reader is welcome to verify. To address this problem, we derive the following closed-form expression.

Lemma 5.10. *For $k \neq 1$, eq. (5.8) becomes*

$$M(R,k_1,k_2,c,n) = \frac{(-1)^n}{k'^n h_R^n(k_1)} \sum_{j=0}^{\min\{n,\lfloor c'/a' \rfloor\}} \binom{n}{j} \left(\mathrm{e}^{a'k'j} - \mathrm{e}^{-c'k'} \sum_{l=0}^{n-1} \frac{(a'k'j - c'k')^l}{l!} \right),$$

where $k' = 1 - k_2$, $c' = \log c$, $a' = \log R$.

Proof. As a first step, we perform a log substitution $z_i = \log r_i$, $\mathrm{e}^{z_i}\mathrm{d}z_i = \mathrm{d}r_i$ which yields

$$M(R,k_1,k_2,c,n) = \frac{1}{h_R^n(k_1)} \iiint_0^{\log R} \mathrm{e}^{(1-k_2)(z_1+\ldots+z_n)} \chi(z_1+\ldots+z_n \leq \log c)\mathrm{d}z_1 \cdots \mathrm{d}z_n.$$

The characteristic function is now supported on a rescaled unit simplex, and writing $\bar{z} := \sum_i z_i$ we can take its Fourier transform

$$\begin{aligned}
\mathcal{F}_t\chi(\bar{z} \leq c') &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \chi(\bar{z} \leq c')\mathrm{e}^{\mathrm{i}\bar{z}t}\mathrm{d}\bar{z} \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c'} \mathrm{e}^{\mathrm{i}\bar{z}t}\mathrm{d}\bar{z} \\
&= \frac{\pi}{2}\delta(t) + \frac{\mathrm{e}^{\mathrm{i}c't}}{\sqrt{2\pi}\mathrm{i}t} \\
&=: \tilde{\chi}_{c'}(t).
\end{aligned}$$

We of course have $\mathcal{F}_{\bar{z}}^{-1}\mathcal{F}_t\chi \equiv \chi$. Then

$$\begin{aligned}
h_R^n(k_1)M(R,k_1,k_2,c,n) &= \iiint_0^{a'} \mathrm{e}^{k'\bar{z}}\chi(\bar{z} \leq c')\mathrm{d}z_1 \cdots \mathrm{d}z_n \\
&= \iiint_0^{a'} \mathrm{e}^{k'\bar{z}} \int_{\mathbb{R}} \mathrm{e}^{-\mathrm{i}t\bar{z}} \frac{\tilde{\chi}_{c'}(t)}{\sqrt{2\pi}}\mathrm{d}t\ \mathrm{d}z_1 \cdots \mathrm{d}z_n \\
&\overset{*}{=} \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \tilde{\chi}_{c'}(t) \left(\prod_{l=1}^{n} \int_0^{a'} \mathrm{e}^{(k'-\mathrm{i}t)z_l}\mathrm{d}z_l \right) \mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \tilde{\chi}_{c'}(t) \left(\frac{\mathrm{e}^{a'(k'-\mathrm{i}t)} - 1}{k' - \mathrm{i}t} \right)^n \mathrm{d}t \\
&= \frac{(-1)^n}{\sqrt{2\pi}} \int_{\mathbb{R}} \sum_{j=0}^{n} \binom{n}{j} (-1)^j \frac{1}{(k'-\mathrm{i}t)^n} \mathrm{e}^{a'(k'-\mathrm{i}t)j} \tilde{\chi}_{c'}(t)\mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}} \sum_{j=0}^{n} \binom{n}{j} \mathrm{e}^{a'k'j} \frac{(-1)^j}{\mathrm{i}^n} \underbrace{\int_{\mathbb{R}} \frac{\mathrm{e}^{-\mathrm{i}a'jt}}{(t+\mathrm{i}k')^n} \tilde{\chi}_{c'}(t)\mathrm{d}t}_{=:J_n}.
\end{aligned} \tag{5.9}$$

In the step marked with $*$, we applied Fubini's theorem, for which we implicitly assumed a smooth limiting argument for the step function. To evaluate the integral J_n, we observe that the denominator has a root of order n at

$$t_0 = -\mathrm{i}k' = \begin{cases} +\mathrm{i}|k'| & k > 1 \Leftrightarrow k' < 0 \\ -\mathrm{i}|k'| & k < 1 \Leftrightarrow k' > 0 \\ 0 & k = 1 \Leftrightarrow k' = 0. \end{cases}$$

We further expand the Fourier-transformed characteristic function—and again glossing over the details of Fubini's theorem to swap the integration order—to obtain

$$J_n = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c'} \int_{\mathbb{R}} \frac{\mathrm{e}^{\mathrm{i}t(x-ja')}}{(t+\mathrm{i}k')^n} \mathrm{d}t \; \mathrm{d}x. \tag{5.10}$$

We handle the integrand's three pole cases separately.

$\mathbf{k > 1}$. We have $k' < 0$ and an order n pole at $\mathrm{i}|k'|$; the integrand $g(t) := \mathrm{e}^{\mathrm{i}t(x-ja')}/(t+\mathrm{i}k')^n$ is holomorphic in the lower half plane. The exponent of the exponential, $x - ja'$, assumes signs

$$x - ja' \begin{cases} > 0 & x > ja' \\ < 0 & x < ja' \\ = 0 & x = ja'. \end{cases}$$

In the latter case, the integral (over t) evaluates to zero.

In the middle case, for $t = -\mathrm{i}s$ we have $\exp(\mathrm{i}(-\mathrm{i})s(x - ja')) = \exp(s(x - ja')) \longrightarrow 0$ as $s \longrightarrow \infty$; by Jordan's lemma we can thus write

$$\int_{\mathbb{R}} g(t)\mathrm{d}t = \lim_{r\to\infty} \oint_{\gamma_1(r)} g(t)\mathrm{d}t = 0,$$

where $\gamma_1(r)$ contains the real interval $[-r, r]$ and a half circle connecting the end points in the lower half complex plane.

In the first case, for $t = \mathrm{i}s$, we have $\exp(\mathrm{i}^2 s(x - ja')) = \exp(-s(x - ja')) \longrightarrow 0$ as $s \longrightarrow \infty$; however now the corresponding upper half plane loop encircles the pole of $g(x)$. We apply the residue theorem for a flipped path $\gamma_2(r) = -\gamma_1(r)$:

$$\begin{aligned}
\int_{\mathbb{R}} \frac{\mathrm{e}^{\mathrm{i}t(x-ja')}}{(t+\mathrm{i}k')^n}\mathrm{d}t &= \lim_{r\to\infty} \oint_{\gamma_2(r)} \frac{\mathrm{e}^{\mathrm{i}t(x-ja')}}{(t+\mathrm{i}k')^n}\mathrm{d}t \\
&= 2\pi\mathrm{i}\mathrm{Res}_t(g, t_0) \\
&= \frac{2\pi\mathrm{i}}{(n-1)!} \lim_{t\to t_0} \frac{\mathrm{d}^{n-1}}{\mathrm{d}t^{n-1}}((t-t_0)^n g(t)) \\
&= \frac{2\pi\mathrm{i}}{(n-1)!} \lim_{t\to t_0} (\mathrm{i}(x-ja'))^{n-1}\mathrm{e}^{\mathrm{i}t(x-ja')} \\
&= \frac{2\pi\mathrm{i}^n}{(n-1)!}(x-ja')^{n-1}\mathrm{e}^{k'(x-ja')}.
\end{aligned}$$

For the case $x - ja' < 0$ we are left to perform the outer integration in eq. (5.10). If $c' \leq ja'$ we necessarily have $x \leq ja'$ and $J_n = 0$. For the case $c' > ja'$ we have

$$\begin{aligned}
\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c'} \frac{2\pi\mathrm{i}^n}{(n-1)!}(x-ja')^{n-1}\mathrm{e}^{k'(x-ja')}\mathrm{d}x &= \frac{\sqrt{2\pi}\mathrm{i}^n}{(n-1)!} \int_0^{c'-ja'} y^{n-1}\mathrm{e}^{k'y}\mathrm{d}y \\
&= \frac{\sqrt{2\pi}\mathrm{i}^n}{(n-1)!} \frac{1}{(-k')^n}(\Gamma(n) - \Gamma(n, a'k'j - c'k')) \\
&= \frac{\sqrt{2\pi}\mathrm{i}^n}{(-k')^n}\left(1 - \frac{\Gamma(n, a'k'j - c'k')}{\Gamma(n)}\right),
\end{aligned}$$

Where $\Gamma(n, \cdot)$ is the lower incomplete gamma function. Putting it all together, we get

$$J_n = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c'} \int_{\mathbb{R}} \frac{\mathrm{e}^{\mathrm{i}t(x-ja')}}{(t+\mathrm{i}k')^n}\mathrm{d}t\ \mathrm{d}x = \frac{\sqrt{2\pi}\mathrm{i}^n}{(-k')^n} \begin{cases} 0 & c' \leq ja' \\ 1 - \frac{\Gamma(n, a'k'j - c'k')}{\Gamma(n)} & \text{otherwise.} \end{cases}$$

Finally, we insert the last expression back into eq. (5.9), and obtain

$$\begin{aligned}
M(R, k_1, k_2, c, n) &= \frac{1}{h_R^n(k_1)} \frac{1}{\sqrt{2\pi}} \sum_{j=0}^{n} \binom{n}{j} \mathrm{e}^{a'k'j} \frac{(-1)^j}{\mathrm{i}^n} J_n \\
&= \frac{(-1)^n}{k'^n h_R^n(k_1)} \sum_{j=0}^{\min\{n, \lfloor c'/a' \rfloor\}} \binom{n}{j} \mathrm{e}^{a'k'j} \left(1 - \frac{\Gamma(n, a'k'j - c'k')}{\Gamma(n)}\right).
\end{aligned}$$

The second term in the sum we can further simplify using the identity $\Gamma(n, x)/\Gamma(n) = \mathrm{e}^{-x} \sum_{l=0}^{n-1} x^l/l!$ which holds for integer j, which yields

$$M(R, k_1, k_2, c, n) = \frac{(-1)^n}{k'^n h_R^n(k_1)} \sum_{j=0}^{\min\{n, \lfloor c'/a' \rfloor\}} \binom{n}{j} \left(\mathrm{e}^{a'k'j} - \mathrm{e}^{-c'k'} \sum_{l=0}^{n-1} \frac{(a'k'j - c'k')^l}{l!}\right).$$

$k < 1$. We have $k' > 0$ and the order n pole of eq. (5.10) lies at $-\mathrm{i}|k'|$. The integrand $g(t) = \mathrm{e}^{\mathrm{i}t(x-ja')}/(t+\mathrm{i}k')^n$ is holomorphic in the upper half plane; and analogous to before, this time when $x - ja' > 0$, we have

$$\int_{\mathbb{R}} g(t)\mathrm{d}t = \lim_{r\to\infty} \oint_{\gamma_2(r)} g(t)\mathrm{d}t = 0.$$

In the opposite case we can again apply the residue theorem and obtain

$$\int_{\mathbb{R}} g(t)\mathrm{d}t \overset{*}{=} -2\pi\mathrm{i}\mathrm{Res}_t(g, t_0) = -\frac{2\pi\mathrm{i}^n}{(n-1)!}(x-ja')^{n-1}\mathrm{e}^{k'(x-ja')},$$

where the negative sign in step $*$ stems from the clockwise orientation of the contour γ_2. The outer integration in eq. (5.10) is now

$$\begin{aligned}
J_n &= -\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c'} \frac{2\pi\mathrm{i}^n}{(n-1)!} \begin{cases} (x-ja')^{n-1}\mathrm{e}^{k'(x-ja')} & x < ja' \\ 0 & \text{otherwise} \end{cases} \mathrm{d}x \\
&= -\frac{\sqrt{2\pi}\mathrm{i}^n}{(n-1)!} \int_{-\infty}^{\min\{c', ja'\}} (x-ja')^{n-1}\mathrm{e}^{k'(x-ja')}\mathrm{d}x \\
&= -\frac{\sqrt{2\pi}\mathrm{i}^n}{(n-1)!} \int_{-\infty}^{\min\{c'-ja, 0\}} y^{n-1}\mathrm{e}^{k'y}\mathrm{d}y \\
&= -\frac{\sqrt{2\pi}\mathrm{i}^n}{(n-1)!}(-1)^{n+1}k'^{-n}\Gamma(n, -k'\min\{c'-ja', 0\}) \\
&= \frac{\sqrt{2\pi}\mathrm{i}^n}{(-k')^n} \frac{\Gamma(n, \min\{a'k'j - c'k', 0\})}{\Gamma(n)} \\
&= \frac{\sqrt{2\pi}\mathrm{i}^n}{(-k')^n} \begin{cases} 1 & c' \geq ja' \\ \frac{\Gamma(n, a'k'j - c'k')}{\Gamma(n)} & \text{otherwise.} \end{cases}
\end{aligned}$$

Inserting the expression back into eq. (5.9) we obtain

$$\begin{aligned}
M(R, k_1, k_2, c, n) &= \frac{1}{h_R^n(k_1)} \frac{1}{\sqrt{2\pi}} \sum_{j=0}^{n} \binom{n}{j} \mathrm{e}^{a'k'j} \frac{(-1)^j}{\mathrm{i}^n} J_n \\
&= \frac{(-1)^n}{k'^n h_R^n(k_1)} \Bigg[\sum_{j=0}^{\min\{n, \lfloor c'/a' \rfloor\}} \binom{n}{j} (-1)^j \mathrm{e}^{a'k'j} \; + \\
&\qquad \sum_{j=\lfloor c'/a' \rfloor + 1}^{n} \sum_{l=0}^{n-1} \binom{n}{j} (-1)^j \mathrm{e}^{c'k'} \frac{(a'k'j - c'k')^l}{l!} \Bigg].
\end{aligned}$$

To reduce the last sum to the previous expression, we note that

$$\sum_{j=0}^{n}\sum_{l=0}^{n-1}\binom{n}{j}(-1)^j\frac{(xj-y)^l}{l!} = \sum_{l=0}^{n-1}\frac{x^l}{l!}\sum_{j=0}^{n}\binom{n}{j}(-1)^j\sum_{m=0}^{l}\binom{l}{m}j^m\left(-\frac{y}{x}\right)^{l-m}$$
$$= \sum_{l=0}^{n-1}\frac{x^l}{l!}\sum_{m=0}^{l}\binom{l}{m}\left(-\frac{y}{x}\right)^{l-m}\underbrace{\sum_{j=0}^{n}\binom{n}{j}(-1)^j j^m}_{=(-1)^n n! S_m^{(n)}},$$

where $S_m^{(n)}$ is the Stirling number of the second kind, which denotes the number of ways to partition a set of size m into n non-empty subsets. Since $m \leq l \leq n-1$, $S_m^{(n)} \equiv 0$ here, and thus

$$\sum_{l=0}^{n-1}\sum_{j=\lfloor c'/a'\rfloor+1}^{n}\binom{n}{j}(-1)^j\frac{(a'k'j-c'k')^l}{l!} = -\sum_{l=0}^{n-1}\sum_{j=0}^{\min\{n,\lfloor c'/a'\rfloor\}}\binom{n}{j}(-1)^j\frac{(a'k'j-c'k')^l}{l!}.$$

The claim follows. □

We leave the $k = 1$ case as an exercise to the reader.

With lemma 5.10, we can now evaluate the terms in eq. (5.7) efficiently. The first term is

$$\mathrm{rt} = \sqrt{h_R^n(k)}M\left[R, k, \frac{k}{2}, \left(\frac{R}{h_R(k)}\right)^{\frac{n}{k}}, n\right], \tag{5.11}$$

and the second

$$\mathrm{rt}' = 1 - M\left[R, k, k, \left(\frac{R}{h_R(k)}\right)^{\frac{n}{k}}, n\right]. \tag{5.12}$$

Of interest is whether taking this full expectation value and splitting it to fall back to Grover search whenever the probability dips below $1/R^n$ yields a significant improvement of the runtime bound. We found this to not be the case, as fig. 5.3 demonstrates; while for smaller n there is a significant improvement, as n grows the ratio $\mathrm{rt}/\mathrm{RT}_1 \longrightarrow 1$ exponentially fast.

5.5 Quantum Beam Search Decoding

The goal of this section is to modify the QuantumSearchDecode decoder such that it behaves more akin to a classical beam search algorithm. More specifically, instead of searching for the top scored element which could sit *anywhere* within the advice distribution, we make the assumption that wherever the advice probability lies below some threshold $p(x) < p_0$—where p_0 can be very small—we discard those hypotheses. This is done by dovetailing a few rounds

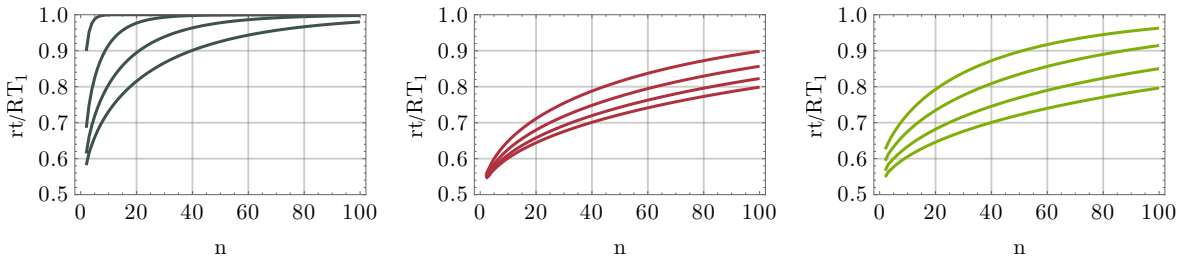


Figure 5.3: Ratios of $\mathrm{rt}/\mathrm{RT}_1$ for various power law exponents k. left: $\{0.2, 0.4, 0.6, 0.8\}$ from top to bottom, middle: $\{1.2, 1.4, 1.6, 1.8\}$ from top to bottom, right: $\{2.5, 3.0, 3.5, 4.0\}$ from bottom to top. In all cases the runtime ratios approach 1 exponentially fast with growing n.

Algorithm 5.2 Algorithm for beam search decoding.

function QuantumBeamDecode$_m$($\mathbf{U}_\mu$, F, p_0)
  $bestScore \leftarrow -\infty$
  $counter \leftarrow 0$
  repeat
    $cmp_1 \leftarrow [(\cdot) \mapsto (p_0 < \cdot)]$ ▷ comparator against threshold
    $cmp_2 \leftarrow [(\cdot) \mapsto (bestScore < \cdot)]$ ▷ comparator against current best score
    $amp \leftarrow [(\cdot) \mapsto \text{AmplitudeAmplification}(\cdot, cmp_1)]$ ▷ prune hypotheses
    $|\psi\rangle \leftarrow \text{ExponentialSearch}(amp \circ \mathbf{U}_\mu, cmp_2 \circ F)$ ▷ select elements $\geq$ pivot
    $bestScore \leftarrow \mathbf{M}_{\text{score}} |\psi\rangle$ ▷ measure new best score
    $counter \leftarrow counter + 1$
  until $counter = m$
end function

of amplitude amplification to suppress all beam paths with probability less than p_0 (which we can do, since we have those probabilities written out as numbers within the advice state $|\mu\rangle$ in eq. (5.4)); a schematic of the algorithm can be found in Algorithm 5.2.

Of course we only want to do this if the number of amplification rounds, given as the square root of the inverse of the leftover probability $\sum_{x:p(x)\geq p_0} p(x)$, is small (i.e. constant, or logarithmic in n). We note that this expression is, as before, well-approximated by $M(R, k, k, p_0, n)$ given in lemma 5.10.

In beam search, only the top scoring hypotheses are kept around at any point in time; the difference to our method is of course that we can score the elements *after every hypothesis has been built*. This is not possible in the classical case, since it would require an exponential amount of memory or post selection. As in Section 5.3, we have the two cases of finding the top scoring path and the most likely parse. Deriving a runtime bound for the most likely parse is straightforward—and does not, in fact, gain anything. This is because when finding the maximum likelihood path τ, one performs amplitude amplification on that element

anyhow, and $p(\tau) > p_0$—so it is within the set of elements with probability kept intact by the post-amplification.[4]

The only interesting case of amplifying the advice state in QuantumSearchDecode to raise it to a beam search variant is thus for finding the top scoring element under a secondary scoring function, using the decoder's output as advice distribution. The relevant questions to ask here is what choice of p_0 will

1. only require a constant—or logarithmic—number of rounds of amplitude amplification,
2. retain a large number of hypotheses, and
3. improve runtime for the post-amplified QuantumSearchDecode variant.

We address all these questions in the next sections.

5.5.1 Constant Post-Amplification

In light of simplicity, we will take RT_1 as an upper runtime bound to the full expected number of rounds, RT_2; as we amplify away all paths with weights below the cutoff we never expect to find an element therein—meaning we can drop the fallback to Grover search in our analysis, and treat the search as if the advice state was purely on those paths with weight $\geq p_0$.

We first address the question for which choice of p_0 the cumulative leftover probability $M(R, k, k, p_0, n)$ can be lower-bounded by a quantity independent of n, which means we have to perform only a *constant* number of amplitude amplification rounds on the advice state. In order to do so, we solve the implicit inequality

$$\text{minimise } f_{\text{split}} \text{ subject to } M\left[R, k, k, \underbrace{\left(\frac{R}{h_R(k)}\right)^{\frac{n}{k} f_{\text{split}}}}_{=p_0}, n\right] \geq C_0. \tag{5.13}$$

As M is monotonically decreasing for a decreasing splitting exponent f_{split}, and since M can be computed in $\mathcal{O}(n^2)$ many arithmetic operations, we can perform the minimisation efficiently. For a choice of $C_0 = 1/4$ (which implies a single amplitude amplification round) and $C_0 = 1/100$ (ten rounds of amplification) we plot f_{split} in fig. 5.4. As can be seen, f_{split} tends towards a limiting value $\in (0, 1)$ for $n \longrightarrow \infty$.

The next step in our analysis is to take the modified splitting exponent f_{split} and count how many hypotheses N_{hyp} remain to be searched over; this is important because it is not clear a priori how many paths we can still search over, and if that quantity is low—or even tends towards zero—then we retained too few elements. Our hope is of course that in contrast to beam search, where generally the beam's width, i.e. the number of hypotheses retained at

[4] If anything, p_0 introduces some prior knowledge about the first pivot to pick for maximum finding.

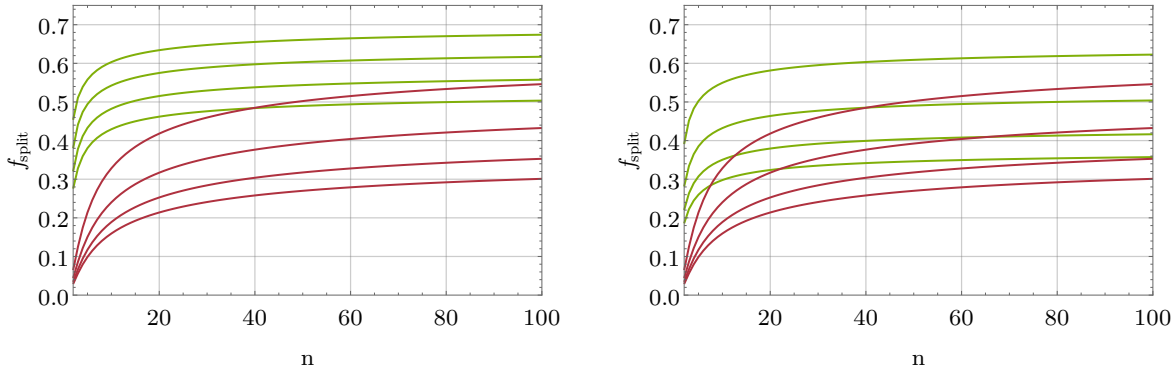


Figure 5.4: minimised value of the splitting exponent f_{split} as defined in eq. (5.13). Plotted are the values for $R = 6$ (left) and $R = 24$ (right), as well as $C_0 = 1/4$ (green, upper family of lines) which implies exactly one extra round of amplitude amplification, and $C_0 = 1/100$ (red, lower family of lines) which implies ten extra rounds of amplification. The power law exponents chosen are $k \in \{1.5, 2.0, 2.5, 3.0\}$ (bottom to top, respectively).

any point in time, is capped at some possibly large but constant value, we have a growing number of hypotheses to search over.

In order to count this number of hypotheses given a cutoff probability p_0, we can evaluate $M(R, k, k, p_0, n)$ in the limit of the power law exponent $k \longrightarrow 0$, and finally multiply $h_R^n(k_1)$ in eq. (5.8) to make the integral *count* instead of calculating a cumulative density. We again choose a series of values for R, k and C_0 and plot the results in fig. 5.5. While the number of leftover hypotheses is indeed reduced drastically as compared to performing a full search over R^n elements, it is still growing exponentially with n, which results in a significant number of hypotheses to search over, many more than possible in the classical setting.

As a last step, we want to analyse the modified runtime given the changed probability cutoff, which corresponds to evaluating the integral $M(R, k, k/2, p_0, n)$ with the p_0 derived from the optimisation eq. (5.13). The results are collected in fig. 5.6. As one can verify, the runtime does remain asymptotically exponential in the sequence length n; however the base of the exponential is reduced accordingly.

5.5.2 Non-Constant Post-Amplification

The analysis of Section 5.5.1 can of course be repeated for a non-constant f_{split}; however, one has to be aware that these extra amplitude amplification rounds factor into the overall

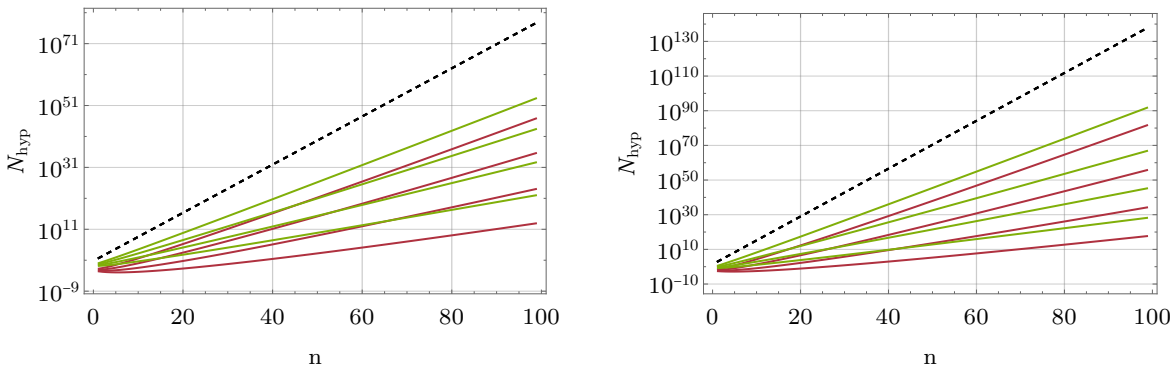


Figure 5.5: Number of hypotheses N_{hyp} left for a specific choice of splitting exponent f_{split} to retain $C_0 > 1/4$ (green, one extra round of amplification) and $C_0 > 1/100$ (red, ten extra rounds of amplification) total probability weight for the hypotheses. The value of f_{split} is obtained numerically from eq. (5.13) (cf. fig. 5.4). Plotted is the case $R = 6$ (left) and $R = 24$ (right), and $k \in \{1.5, 2.0, 2.5, 3.0\}$ (from top to bottom in each plot and each colour, respectively). The dashed line is the total number of possible hypotheses R^n as reference.

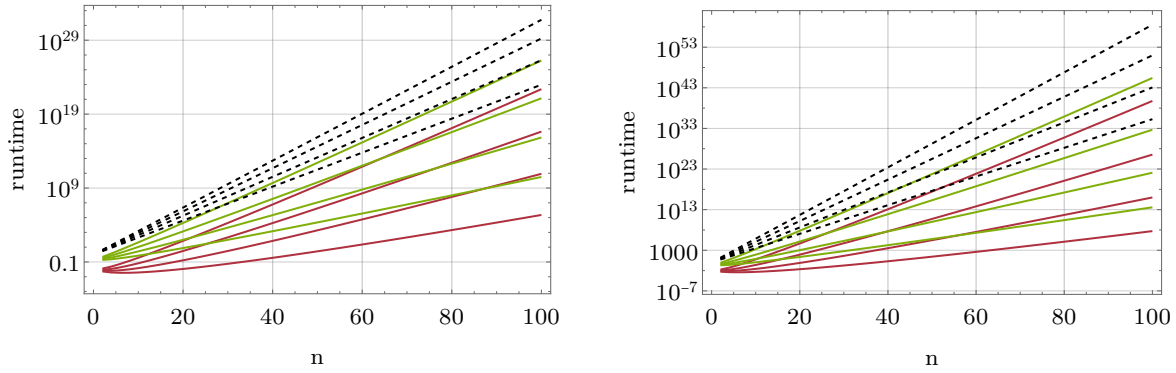


Figure 5.6: Runtime when post-amplifying to retain only a fraction $C_0 \geq 1/4$ of weight (green, one extra round of amplification) or $C_0 \geq 1/100$ (red, ten extra rounds of amplification) on the hypotheses. The value of f_{split} is obtained numerically from eq. (5.13) (cf. fig. 5.4). Plotted is the case $R = 6$ (left) and $R = 24$ (right), and $k \in \{1.5, 2.0, 2.5, 3.0\}$ (from top to bottom in each plot and for each colour, respectively). The dashed line is the full search runtime $\text{RT}_1(R, k, n)$ from lemma 5.9 as reference.

runtime. For a retained fraction $g(n)$ of the total probability weight, the optimisation thus reads

$$\text{minimise } p_0 \text{ subject to } M(R, k, k, p_0, n) \geq g(n) \tag{5.14}$$

$$\text{which retains } \lim_{k \to 0} M(R, k, k, p_0, n) \text{ hypotheses,} \tag{5.15}$$

$$\text{and has runtime bound } g(n)^{-1/2} M(R, k, k/2, p_0, n). \tag{5.16}$$

Instead of listing a series of results for a range of parameters, we provide an explicit example of this analysis with real-world parameters derived from Mozilla's DeepSpeech neural network in the next section.

5.6 DeepSpeech

5.6.1 Analysis of the Output Rank Frequency

To support the applicability of our model, we analysed our hypothesis that the output probabilities of an LSTM used to transcribe voice to letters—which can then be used e.g. in a dialogue system with an underlying parser—is distributed in a power-law like fashion. More specifically, we use *DeepSpeech*, Mozilla's implementation of Baidu's *DeepSpeech* speech recognition system [HCC+14, Moz19b].

The neural network processes mel-frequency cepstral coefficients extracted from a sliding window of 25 miliseconds, with a stride of 20 miliseconds; for each such frame, the LSTM is invoked, and yields a distribution over the letters of the English alphabet "a" to "z", as well as a few special symbols, e.g. "silence". For the specific architecture of the LSTM we refer the reader to the original paper [HCC+14]. Our hypothesis was that these letter probabilities follow a power-law distribution; our data supports this claim, as can be seen in fig. 5.7.

5.6.2 Runtime Bounds for Quantum Beam Search Decoding

As outlined in Section 5.5.2 we take the power law exponent derived from Mozilla's DeepSpeech neural network, $k = 3.03$, and derive runtime bounds for decoding its output with a parser under the assumption that, on average, we take $R = 3$ branches in the parsing tree at every time step. As discussed in Section 5.4, the sampling distribution over three elements only yields a slightly lower exponent of $k = 2.91$. How does quantum beam search perform in this setting, and how many hypotheses are actually searched over? And what if we fix the beam's width to a constant, and increase the sequence length? We summarise our findings in figs. 5.8 and 5.9.

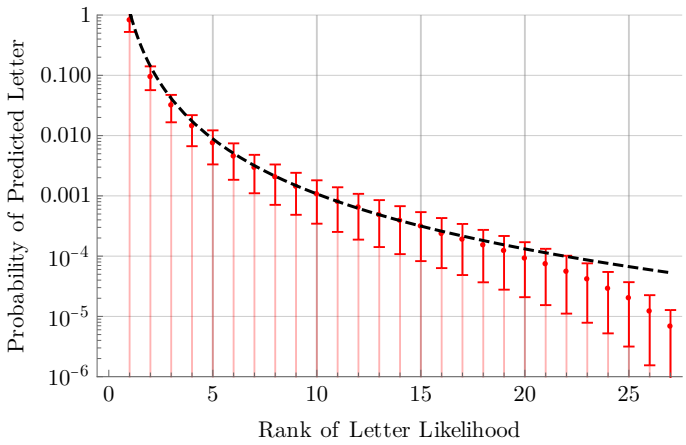


Figure 5.7: Log plot of the power law distribution of the output probabilities obtained from Mozilla's *DeepSpeech* voice recognition LSTM on the Mozilla Common Voice verified test dataset for English [Moz19a], which consists of 3995 audio samples of about ten seconds each of spoken test sentences. The dashed line is a fitted power law ar^{-b} with parameters $a = 1.2 \pm 0.1$ and $b = 3.03 \pm 0.03$. We individually process each audio file, and capture the output after the final `Softmax` layer (`logits:0`), but before it is processed further by the greedy connectionist temporal classification (CTC beam search) implemented by *DeepSpeech*.

As an example we consider an input sequence of length 500; with the above parameters and a splitting exponential $f_{\text{split}} = n^{-1/2}$ (resp. $= n^{-3}$) we can search over $N_{\text{hyp}} \approx 10^{60}$ (resp. $\approx 10^{18}$) hypotheses, with a runtime $\approx 10^{30}$ (resp. $\approx 10^{9}$). Similarly, when capping the beam width at $N_{\text{hyp}} \leq 10^{6}$, we asymptotically require $\approx 10^{3}$ iterations of the beam search decoder (which includes the post-amplification rounds); for shorter sequences, a super-Grover speedup as present in full QuantumSearchDecode is achieved.

5.7 Discussion

In summary, we have presented a quantum algorithm that is modelled on and extends the capabilities of beam search decoding for generative models. Studies of context sensitivity of language models have shown that state-of-the-art LSTM models are able to use about 200 tokens of context on average while working with standard datasets (WikiText2, Penn Treebank), but sharply distinguish nearby context (roughly 50 tokens) from distant history [KHQ+18]. The performance of an efficient classical beam search decoder using such an LSTM depends heavily on the context sensitivity of the underlying language model.

On the other hand, our quantum search decoding method is guaranteed to find—with high constant success probability—the global optimum in expected runtime that is always more than quadratically faster than possible classically (neglecting the sampling cost), and with a

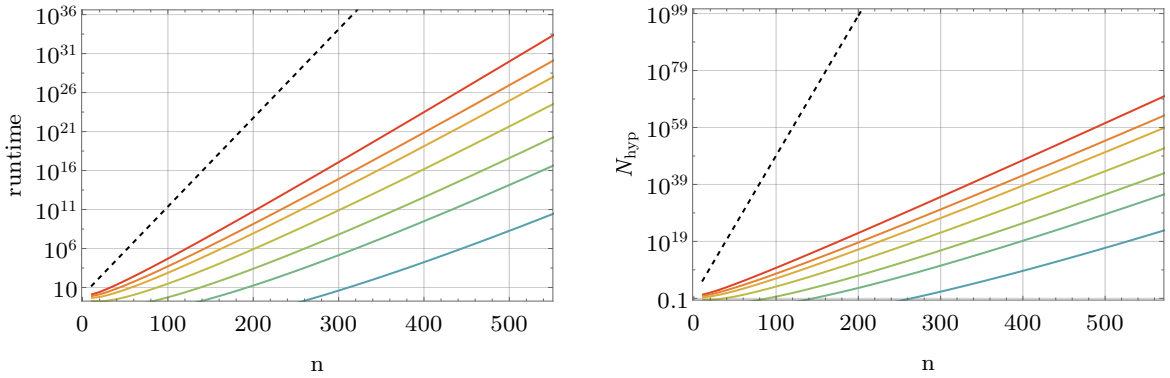


Figure 5.8: Number of iterations (eq. (5.16)) and number of hypotheses (eq. (5.15)) of quantum beam search decoding the output of Mozilla's *DeepSpeech* LSTM with a grammar, assuming an average branching ratio of $R = 3$, a token power law distribution with exponent $k = 2.91$, and post-amplification of the quantum search decoder with a retained fraction of hypotheses $C_0 = C_0(n) \in \{n^{-1/2}, n^{-2/3}, n^{-1}, n^{-3/2}, n^{-2}, n^{-3}\}$ as defined in eq. (5.14), which is plotted in rainbow colours from red to blue, top to bottom. The dashed line is the full quantum search runtime and number of hypotheses from eq. (5.11).

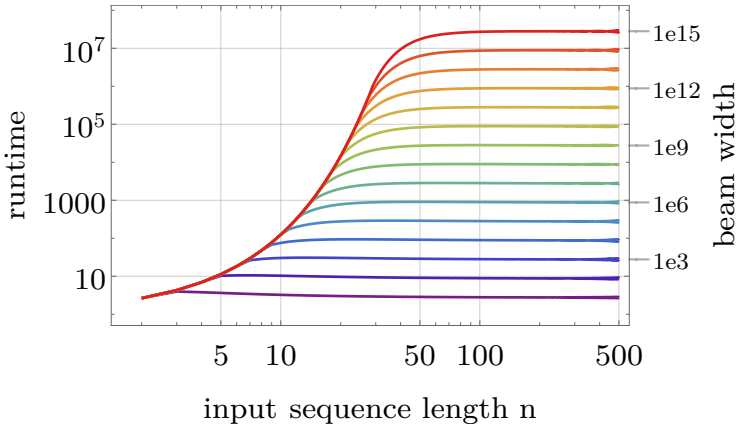


Figure 5.9: Runtime of quantum beam search decoding the output of Mozilla's *DeepSpeech* LSTM with a grammar, assuming an average branching ratio of $R = 5$, a token power law distribution with exponent $k = 2.91$, and post-amplification of the quantum search decoder with a constant number of retained hypotheses $N_{\text{hyp}} \in \{10^1, \ldots, 10^{15}\}$, plotted in rainbow colours from purple to red, bottom to top. As expected, the super-Grover speedup is achieved in the regime where full QuantumSearchDecoding happens; once the beam width saturates, the speedup asymptotically approaches a quadratic advantage as compared to classical beam search.

Grover exponent that shrinks exponentially quickly as the power law exponent k grows, thus surpassing plain Grover search for any $k > 0$.

We have further shown that neural networks used in the real world—concretely *DeepSpeech*—indeed exhibit a strong power law distribution on their outputs, which in turn supports the premise of our algorithm.

There are many extensions possible for the type of search we have described in this chapter, and we hope to make a series of improvements. Foremost, it would be interesting to study how much non-independence of the input random variables affects the runtime—either in the negative, or potentially yielding an even positive effect. Quantifying dependence for input models will likely require analysing a specific problem setup.

A fully error-corrected quantum computer will remain outside of the realm of the possible for the foreseeable future; yet we hope that our hitherto theoretical proposal of a quantum search decoder demonstrates that natural language processing is one of the areas where a potential quantum advantage can be obtained.

Appendix A Amplitude Amplification 2.0

We have already encountered the concept of amplitude amplification in Chapter 1. In this appendix, we review some of the material that we saw there once more, and also discuss some of the more recent variants of this technique that have become indispensable tools in quantum algorithm design.

Grover's quantum algorithm for searching an unstructured list [Gro97] was a major development in the theory of quantum algorithms, achieving a provable quadratic speedup over the best *possible* classical algorithm in the oracle access input model. The underlying idea is to start with the set of database indices loaded in uniform quantum superposition, and by iteratively applying two cleverly defined reflection operators the probability associated with observing the marked element can be made arbitrarily close to unity. The key principle is that the reflections, which are about the initial state and the target state respectively, multiply to a give a rotation operator that preserves the 2D subspace spanned by the target state and its orthogonal complement.

Considered from the point of view of generic quantum algorithms, Grover search can be generalised to what is known as the amplitude amplification algorithm [BHM+02]. The input is a (potentially black-box) unitary $\mathbf{U}$ that prepares from an input state $|\text{in}\rangle$ a target state $|\text{tar}\rangle$ in superposition with some unwanted orthogonal state, i.e. $\mathbf{U}|\text{in}\rangle = \alpha|\text{tar}\rangle + \beta|\perp\rangle$. Henceforth we assume the input state to be $|0\rangle$ without loss of generality. We would like to increase the amplitude of the target state and suppress the amplitude of the orthogonal state, thus boosting the probability of observing the target state on performing a suitably defined projective measurement.

Let $\Pi_\psi = |\psi\rangle\langle\psi|$ represent the projector onto the state $|\psi\rangle$. Further, let $\mathbf{R}_\psi^\phi = \mathbb{1} - (1 - e^{i\phi})\Pi_\psi$ denote a selective phase shift of $e^{i\phi}$ for the state $|\psi\rangle$. For the special case of $\phi = \pi$, we denote by $\mathbf{R}_\psi = \mathbb{1} - 2\Pi_\psi$ the reflection about the state $|\psi\rangle$.

In standard amplitude amplification, applying the Grover iterate $G = -\mathbf{R}_\psi \mathbf{R}_{\text{tar}} = -\mathbf{R}_0 \mathbf{U} \mathbf{R}_0 \mathbf{U}^\dagger$ times to the initial state $|\psi\rangle = \mathbf{U}|\text{in}\rangle$ results in a state

$$G^k|\psi\rangle = \sin[(2k+1)\theta]|\text{tar}\rangle + \cos[(2k+1)\theta]|\perp\rangle,$$

where $\beta = \cos\theta$, and θ is the angle between the undesired orthogonal state $|\perp\rangle$ and the initial state $|\psi\rangle$. From this we deduce that choosing $k \approx \frac{\pi}{4\theta}$ results in $|\langle\text{tar}|G^k\psi|| \approx 1$, whenever $k \geq 1$, i.e. $|\alpha| \leq 1/\sqrt{2}$. In the 2D subspace $\mathcal{B} = \text{span}_{\mathbb{C}}\{|\text{tar}\}, |\perp\rangle\rangle$, the iterate G has the form

$$G\big|_{\mathcal{B}} = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{pmatrix} \tag{5A.1}$$

For the search problem, the above is usually phrased in the language of a list of N elements of which $M < N/2$ are marked through some Boolean function $f : [N] \to \{0, 1\}$. In this case, the initial state in the simplest version of AA is the uniform superposition over all list indices, and the target and orthogonal states are corresponding uniform superpositions over marked and unmarked indices, viz $|\text{tar}\rangle = \frac{1}{\sqrt{M}} \sum_{i:f(i)=1} |i\rangle$ etc. Here $\alpha = \sin\theta = \sqrt{M/N}$. The Grover iterate can be written in terms of M, N by substituting for the angles in Eq. (5A.1) as

$$\begin{pmatrix} 1 - \frac{2M}{N} & \frac{2}{N}\sqrt{N-M} \\ -\frac{2}{N}\sqrt{N-M} & 1 - \frac{2M}{N} \end{pmatrix}. \tag{5A.2}$$

Grover search with a non-uniform initial state

Now consider the case where we are given classical 'advice' in the form of a probability mass function $\mu = (p_i) : [N] \to [0, 1]$ which indicates by p_i the probability that element indexed by i is marked. This can be turned into quantum advice by providing a black-box unitary $\mathbf{U}_\mu$ which prepares a state of the form

$$\mathbf{U}_\mu |0^n\rangle = |\text{prior}\rangle \equiv |\psi_\mu\rangle = \sum_{i=0}^{N-1} \sqrt{p_i}\, |i\rangle\,,$$

on $n = \lceil \log N \rceil$ qubits. Let the target state be

$$|\text{tar}_\mu\rangle = \frac{1}{p} \sum_{i:f(i)=1} \sqrt{p_i}\, |i\rangle\,, \quad p := \sum_{i:f(i)=1} p_i,$$

where we denote by p the sum of the probabilities assigned to marked elements (the 'prior success' probability). Assuming for the moment that we know how to reflect about the target state, we can use the usual Grover iterate $G_p = -\mathbf{R}_{\psi_\mu}\mathbf{R}_{\text{tar}_\mu} = -\mathbf{R}_0\mathbf{U}_\mu\mathbf{R}_0\mathbf{R}_{\text{tar}_\mu}$, which will have matrix entries

$$G|_{\mathcal{B}} = \begin{pmatrix} 1 - 2p & 2\sqrt{p(1-p)} \\ -2\sqrt{p(1-p)} & 1 - 2p \end{pmatrix}. \tag{5A.3}$$

The number of iterations required to bring the success probability of observing the target state on making a measurement is, as observed earlier, approximately $\frac{\pi}{4\theta}$, which now takes the form $k \approx \pi/4\sqrt{p}$, given $p < 1/2$. For the case where the prior distribution is uniform, we recover the usual $\mathcal{O}(\sqrt{N/M})$ iterations of Grover search.

Unfortunately, we will not usually know how to reflect about a target state of the form of $|\text{tar}_\mu\rangle$. If we decide to reflect about a uniform superposition of marked states instead, the state at each iteration becomes more difficult to analyse, and indeed will necessarily lead to a maximum success probability bounded away from unity due to the fact that the 2D subspace that is invariant under the modified Grover iterate no longer contains the initial state $|\psi_\mu\rangle$.

One possible solution to this issue is to use an ancillary qubit to mark the good states, instead of using a phase flip by virtue of a phase oracle, which will then allow the use of oblivious AA (described in section A). Fortunately, as we will see in Theorem 5.7, this is always possible for the application that we consider in this thesis. Gilyén et al. [GAW19] show how to transform back and forth between a phase oracle and a probability oracle in more general cases.

Now if the prior is a power law distribution Zipf(k), with $p_i \propto i^{-k}$ normalised by the generalised harmonic number

$$H_k(N) = \sum_{i=1}^{N} i^{-k},$$

then the prior success probability

$$p = \frac{1}{H_k(N)} \sum_{i=1}^{M} i^{-k} = \frac{H_k(M)}{H_k(N)}, \tag{5A.4}$$

where we assume the M marked elements are also the elements with the top M ranks in the distribution. From [Apo76], we have

$$H_k(N) = \begin{cases} \frac{N^{1-k}}{1-k} + \zeta(k) + \mathcal{O}(N^{-k}) & k > 0, k \neq 1 \\ \log N + const. + \mathcal{O}(1/N) & k = 1, \end{cases} \tag{5A.5}$$

where $\zeta(k) = \sum_{n=1}^{\infty} n^{-k}$ is the Riemann Zeta function. Indeed, Eq. (5A.5) holds in general for real values of N, and the $\mathcal{O}(N^{-k})$ term does not even arise for integer values of N. For integers $k > 1$, $\zeta(k)$ decreases monotonically, and is a small constant. Thus for the case of one marked element, $M = 1$, and with $k = 1$ we have that the number of iterations required in the search algorithm is

$$k \approx \frac{\pi\sqrt{\log N}}{4}. \tag{5A.6}$$

Now, we have to be a bit careful while analysing this power law case. When $p > 1/2$, the initial success probability is already significantly large, and standard amplitude amplification performs poorly as the target is overshot easily due to the large angle through which G rotates the initial state.

Oblivious AA

Oblivious amplitude amplification [BCC+14, Kot14] is a modification of AA which can be used when we do not know how to implement the reflection about the initial or target states.

In these cases, we assume that the initial state has an ancillary qubit with a flag that has somehow been set to indicate the target state, i.e.

$$\begin{aligned}\mathbf{U}\left|0\right\rangle\left|0\right\rangle &= \left|\psi\right\rangle \\ &= \sin\theta\left|\mathrm{tar}\right\rangle\left|0\right\rangle + \cos\theta\left|\perp\right\rangle, \end{aligned} \tag{5A.7}$$

where $\left|\perp\right\rangle$ is a state that has no component in any subspace with $\left|0\right\rangle$ in the flag register, i.e. $(\mathbb{1}\otimes\Pi_0)\left|\perp\right\rangle = 0$ Then using the reflection $\mathbf{R}^0_{\mathrm{flag}} = \mathbb{1} - 2\mathbb{1}\otimes\Pi_0$ about this flagged target subspace, we can define a Grover-like iterate $G = -\mathbf{R}_\psi\mathbf{R}^0_{\mathrm{flag}}$, which acts as a rotation in the 2D subspace span$\{\left|\perp\right\rangle, \left|\mathrm{tar}\right\rangle\left|0\right\rangle\}$. Applying this operator k times has the desired effect, viz

$$G^k\left|\psi\right\rangle = \sin\left[(2k+1)\theta\right]\left|\mathrm{tar}\right\rangle\left|0\right\rangle + \cos\left[(2k+1)\theta\right]\left|\perp\right\rangle\left|1\right\rangle.$$

---

function ObliviousAmplitudeAmplification$_m(\mathbf{U}_\mu,\, f)$
  $\left|\psi\right\rangle \leftarrow \mathbf{U}_\mu\left|0\right\rangle$
  counter $\leftarrow 0$
  repeat
    $\left|\psi\right\rangle \leftarrow -\mathbf{R}_0\mathbf{U}_\mu\mathbf{R}_0\mathbf{R}^0_{\mathrm{flag}}\left|\psi\right\rangle$
    counter $\leftarrow$ *counter* $+ 1$
  until *counter* $= m$
end function

Now suppose we have as input a unitary that performs the map $\mathbf{U}\left|0\right\rangle = \sin\theta\left|\mathrm{tar}\right\rangle + \cos\theta\left|\perp\right\rangle$ with $\sin\theta > 1/2$, and we would like to amplify the success probability of seeing the target state upon measurement. If we know the value of $\sin\theta$ beforehand, we can simply adjoin a single qubit to the system register and consider the map $\mathbf{U}\otimes\mathbf{V}$ where $\mathbf{V}$ is any single qubit unitary that performs the map

$$\mathbf{V}\left|0\right\rangle = \frac{1}{2\sin\theta}\left|0\right\rangle + \sqrt{1 - \left(\frac{1}{2\sin\theta}\right)^2}\left|1\right\rangle, \tag{5A.8}$$

which is a valid unitary since $\sin^2\theta > 1/4$. Now we have

$$\mathbf{U}\otimes\mathbf{V}\left|\mathrm{in}\right\rangle\left|0\right\rangle = \frac{1}{2}\left|\mathrm{tar}\right\rangle\left|0\right\rangle + \frac{\sqrt{3}}{2}\left|\perp'\right\rangle, \tag{5A.9}$$

where $\left|\perp'\right\rangle$ is a state that has no component in any subspace with $\left|0\right\rangle$ in the flag register, i.e. $(\mathbb{1}\otimes\Pi_0)\left|\perp'\right\rangle = 0$. This is now just Grover search for one marked element in a list of 4 elements, which can be performed exactly using a single iteration of OAA.

With this in mind, let us now go back to the power law distribution and ask for what values of the tuples (M, N, k) the prior success probability p in Eq. (5A.4) is greater than a half. This entails finding the satisfying regions to the inequality $2H_k(M) < H_k(N)$.

$k \in$	expected runtime $\mathcal{O}(\cdot)$
$(-1, 0)$	$\sqrt{n^{1+k}}$
$\{-1\}$	$\sqrt{\log n}$
$(-1.64, -1)$	$\sqrt{\zeta(k) + \frac{n^{1+k}}{1+k}}$
$(-\inf, -1.64)$	1

Table 5.1: Number of iterations of Amplitude Amplification required when the subspace of marked elements is known to have the highest weight under the non-uniform input state $|\mu\rangle$, and under the promise that $\mu \sim \text{Power}(k)$, for various ranges of the power law exponent k for an input of size n.

Now of course, if this condition does not hold, we have a prior success probability greater than half, and we can always resort to single query OAA if we can estimate the value of p, or reduce the problem to one with $p' = p/2$ by doing a 'blind' rotation to halve the success probability, e.g. by choosing the map $\mathbf{V}$ used with OAA to be the Hadamard gate.

In table 5.1 we summarise the ranges of the power law exponent with the corresponding number of iterations of AA or oblivious AA required to obtain a superposition with high amplitude on the target state. The critical value of k is obtained via the above arguments regarding the harmonic numbers, and corresponds to $\frac{W(\ln 2)}{\ln 2} \approx 0.641186$, where W is the Lambert W-function, defined as the inverse to the function $f(x) = xe^x$. The constant runtime for exponents less than -1.64 is obtained by using OAA. $\zeta(k)$ is the Riemann Zeta function evaluated at $z = k$.

Appendix B Postselected Product of Powerlaws

In this appendix we answer the open question left in Section 5.4. The setup here is as follows. Let $S > 1$, and X be distributed according to a product of distributions with pdf

$$p(r_1, \ldots, r_n) := \frac{1}{H_S(k)^n} \frac{1}{(r_1 \cdots r_n)^k}$$

as in eq. (5.6), i.e. where every factor is a power law distribution over S elements. If we remove a random subset of the elements such that R^n (for some $R < S$) elements are left over, is the resulting probability distribution a product-of-powerlaws, where every factor is over R elements?

In the continuous case this can be seen as follows. If $X \sim \text{Pareto}(k, S)$ with pdf p as defined in Section 5.4, then removing a random subset of elements on the interval $[1, S+1]$ is equivalent to taking a random characteristic function χ_1 over it, with $\int_{[1,S+1]} \chi(r)\mathrm{d}r = R$, and defining X' with pdf $Sp(r)\chi(r)/R$. We define the postselected random variable Y over $[1, R]$ by relabelling the points in $\operatorname{supp}\chi$ by values in $[1, R]$ in an order-preserving fashion.

Similarly, if X_n is a product of n iid Pareto random variables with pdf p_n, then postselection means taking a random characteristic function χ_n on $[1, S]^n$ with

$$\int_{[1,S+1]^n} \chi_n(r_1, \ldots, r_n) \mathrm{d}r_1 \cdots \mathrm{d}r_n = R^n.$$

We claim that the resulting random variable X'_n with pdf $S^n p_n(r)\chi_n(r)/R^n$ then factors into a product distribution. This holds because χ_n has the property that for all $\epsilon > 0$ there exists a bijection f such that for almost all $(x_1, \ldots, x_n) \in \operatorname{supp} \chi_n$, there exists

$$(y_1, \ldots, y_n) = f(x_1, \ldots, x_n) \in (\operatorname{supp} \chi_1)^n \quad \text{s.t.} \quad \sum_{i=1}^{n} |x_i - y_i| < \epsilon,$$

for some characteristic function χ_1 defined on $[1, S]$. We refer to this property as χ_n being 'product'.

We now prove this claim by induction on n. For $n = 1$, X and χ_1 are already product, so there is nothing to show. Assume the hypothesis holds for χ_n which can be factored into a product χ_1^n for some χ_1. Take a random characteristic function χ_{n+1} over $n + 1$ dimensions. Let $\epsilon > 0$. As $\mathbb{R}$ is uncountable, we take a δ-net over the interval $I := [1, S + 1]$ for some small $\epsilon \gg \delta > 0$, which we will denote with I_δ; each $x \in I$ then has a corresponding $x' \in I_\delta$ that satisfies $|x - x'| < \delta$. In particular, I_δ is countable. In a similar fashion, for χ_{n+1} we consider its discretised variant over I_δ^n as χ'_{n+1}.

So let $(x_1, \ldots, x_n, x_{n+1}) \in \operatorname{supp} \chi'_{n+1}$, and analogously define the discretised characteristic functions χ'_n and χ'_1. A counting argument shows that within each ϵ-bin (defined over I and extended over to I_δ^n accordingly), we can map $(x_1, \ldots, x_{n+1})$ to their closest corresponding point $(y_1, \ldots, y_n, z_1) \in \chi'_n \times \chi'_1$—or if that point was previously chosen its next- and next-to-next-closest one etc., while staying within ϵ distance for each original coordinate for the majority of the points.

A limiting argument $\epsilon \longrightarrow 0$ shows that this map can be constructed for almost all points. This concludes the induction.

The question that remains is what distribution Y follows. Despite scale invariance of Pareto distributions, the resulting pdf for a surviving fraction λ of the original points looks like $\tilde{p}(r) = p(1 + (x - 1)\lambda)$, which is itself not a Pareto distribution. Yet, since we actually work with a power law distribution, we already answered in Section 5.4 what this resulting sample distribution over R looks like: it can be well-approximated by a power law with a slightly worse falloff $k' < k$ that itself can be estimated numerically in a straightforward fashion. The smaller exponent should also account for any approximation errors made by the continuous variable analysis demonstrated in this appendix.

Chapter 6

Conclusions

We have shown in the preceding chapters a modest expansion in the capabilities of quantum computers by demonstrating that they can *efficiently* solve certain classes of problems. In particular, we have reinforced the idea that quantum computers may offer performance gains in linear algebraic problems. This an expectation that has its deep roots in the very formulation of quantum mechanics as linear algebra with complex numbers, in the form of matrix mechanics.

Chapter 1 introduced a unifying framework for the probabilistic quantum implementation of functions of matrices, and discussed the exponential speedup in terms of the precision parameter $1/\epsilon$ achieved by LCU and similar methods. We saw how function approximation by Chebyshev series is useful in designing quantum algorithms, and observed the simplicity of the method for implementing matrix polynomials, which in turn can approximate a very large class of functions by means of Taylor series. Chapter 2 continued this investigation by focusing on the assumption of row sparsity needed by query model quantum algorithms, showing that spectral sparsification can yield row-sparse approximations to input matrices, which can be used with sparse Hamiltonian simulation subroutines and hence in a variety of quantum algorithms including matrix inversion. Towards the quest to find applications of theoretical primitives, and design more practical quantum algorithms, we saw in Chapter 3 a method to estimate the entropy or 'randomness' of a random variable. We implemented power functions of a density matrix input in the form of a unitary preparing its purification, and estimated their trace using a single clean (or pure) qubit in the DQC1 model of quantum computation. We studied the estimation to multiplicative precision, and also looked at an analysis of the expected runtime of our algorithm, bounding it as a function of the unknown target quantity.

In Chapter 4 we shifted into the domain of machine learning problems. Zoning in on kernel methods for supervised learning, we showed an application of matrix functions techniques to invert the square root of a regularised integral operator that has full rank and may be dense, by leveraging a result from signal processing that allowed us to break this operator down into a product of quantum Fourier transforms (QFT) and diagonal matrices constructed

from input data held in QRAM. We discussed some evidence, based on the QFT, for why our algorithm, which does not need sparsity or rank assumptions, might offer a genuine speedup that is not dequantisable. We also saw how the runtime of our method is linear in the data dimension, and can be exponentially faster than the classical state-of-the-art for some realistic choices of kernels and data. Finally, in Chapter 5, once again hand in hand with the bigger theme of complexity analysis, we pick out another machine learning problem of immediate relevance: Natural Language Processing. Studying the sequence-to-sequence decoding technique of beamsearch, we fleshed out a quantum search algorithm that can zero in on the 'optimal parse', which may be a best translation, or text-to-speech transcript for instance. In the presence of prior advice that is usually available in the NLP context, in the form of a power-law distributed random variable output by an LSTM or other neural network model, our method achieves super-quadratic speedups in the expected runtime.

We have seen in the discussion sections of each of the foregoing chapters several avenues for further exploration. These range from investigating the boundary between low-rank and sparsity as assumptions in quantum algorithms, to proving lower bounds on the sample (query) complexity of estimating Renyi entropy to multiplicative precision, in the quantum purified query access model.

As we'd mentioned at the opening of this thesis, while we have focused on algorithms and upper bounds here, lower bounds are another important way of progressing our understanding of quantum computers and their power. In particular, while the quantum query complexity model has been studied in some detail over the last three decades, circuit complexity is an area that has only recently started garnering more attention. Proving bounds on the size and depth of the smallest quantum circuits (in some gate set of interest) that can solve any of the problems we have considered in this thesis, are open questions. These questions include a very practical circuit optimisation component, as well as a more theoretical consideration of complexity class inclusions.

More generally, the solution of problems involving matrices, and their applications to machine learning, is a young and vibrant area of research. I hope, within my limited abilities and sample of research, to have passed on to the readers of this thesis a sense of this vibrancy, and some exciting new ideas to think about in their own work!

Bibliography

[Aar15] Scott Aaronson. "Read the fine print". *Nature Physics*, 11(4):291, 2015 (page 82).

[AGS18] Scott Aaronson, Daniel Grier, and Luke Schaeffer. "A Quantum Query Complexity Trichotomy for Regular Languages", 2018. arXiv: 1812.04219 (page 127).

[AIS+17] Jayadev Acharya, Ibrahim Issa, Nirmal V. Shende, and Aaron B. Wagner. "Measuring Quantum Entropy", 2017. arXiv: 1711.00814 (pages 55, 58, 64, 66).

[AZ18] Dorit Aharonov and Leo Zhou. *Hamiltonian Sparsification and Gap-Simulation*. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 2:1–2:21, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. DOI: 10.4230/LIPIcs.ITCS.2019.2 (page 38).

[AK99] Ashish Ahuja and Sanjiv Kapoor. *A quantum algorithm for finding the maximum*, 1999. arXiv: quant-ph/9911082 [quant-ph] (pages 49, 136).

[AM15] Ahmed Alaoui and Michael W Mahoney. "Fast randomized kernel ridge regression with statistical guarantees". In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 775–783. Curran Associates, Inc., 2015 (page 79).

[Amb12] Andris Ambainis. *Variable time amplitude amplification and quantum algorithms for linear algebra problems*. In Thomas Wilke Christoph Dürr, editor, *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647, Paris, France. LIPIcs, 2012. DOI: 10.4230/LIPIcs.STACS.2012.636 (page 118).

[Apo76] Tom M. Apostol. *Introduction to Analytic Number Theory*. Springer New York, 1976. DOI: 10.1007/978-1-4757-5579-4 (page 157).

[ACL+19] Srinivasan Arunachalam, Sourav Chakraborty, Troy Lee, Manaswi Paraashar, and Ronald de Wolf. *Two New Results About Quantum Exact Learning*. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 16:1–16:15, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. DOI: 10.4230/LIPIcs.ICALP.2019.16 (page 50).

[AdW17] Srinivasan Arunachalam and Ronald de Wolf. "A Survey of Quantum Learning Theory". *ACM SIGACT News*, 48(2):41–67, 2017. DOI: 10.1145/3106700.3106710. arXiv: 1701.06806 (page 92).

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. "Quantum supremacy using a programmable superconducting processor". *Nature*, 574(7779):505–510, 2019 (page 82).

[AKM+17] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. *Random Fourier features for kernel ridge regression: approximation bounds and statistical guarantees*. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 253–262, International Convention Centre, Sydney, Australia. PMLR, 2017 (page 79).

[ASY+16] Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W. Mahoney. "Quasi-monte carlo feature maps for shift-invariant kernels". *Journal of Machine Learning Research*, 17(120):1–38, 2016 (page 79).

[AK20] Koji Azuma and Go Kato. *Second law of black hole thermodynamics*, 2020. eprint: arXiv:2001.02897 (page 54).

[AS18] Koji Azuma and Sathyawageeswar Subramanian. *Do black holes store negative entropy?*, 2018. eprint: arXiv:1807.06753 (page 54).

[Bac17] Francis Bach. "On the equivalence between kernel quadrature rules and random feature expansions". *Journal of Machine Learning Research*, 18(21):1–38, 2017 (pages 77, 79, 80, 86, 89, 99, 122).

[Bac13] Francis Bach. *Sharp analysis of low-rank kernel matrix approximations*. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 185–209, Princeton, NJ, USA. PMLR, 2013 (page 79).

[BBG18] Siddharth Barman, Arnab Bhattacharyya, and Suprovat Ghoshal. *Testing sparsity over known and unknown bases*. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 491–500, Stockholmsmässan, Stockholm Sweden. PMLR, 2018 (page 47).

[BSS14] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. "Twice-ramanujan sparsifiers". *SIAM Review*, 56(2):315–334, 2014. DOI: 10.1137/130949117 (page 37).

[BSS+13] Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. "Spectral sparsification of graphs". *Communications of the ACM*, 56(8):87, 2013. DOI: 10.1145/2492007.2492029 (page 37).

[BDK+02] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. "The complexity of approximating the entropy". *Proceedings of the Annual IEEE Conference on Computational Complexity*:17, 2002. DOI: 10.1109/CCC.2002.1004329 (pages 55, 56).

[Bau18] Johannes Bausch. "Classifying data using near-term quantum devices". *International Journal of Quantum Information*, 16(08):1840001, 2018. DOI: 10.1142/S0219749918400014 (page 127).

[BL18] Johannes Bausch and Felix Leditzky. "Quantum Codes from Neural Networks", 2018. arXiv: 1806.08781 (page 127).

[BSP19] Johannes Bausch, Sathyawageeswar Subramanian, and Stephen Piddock. *A quantum search decoder for natural language processing*, 2019. arXiv: 1909.05023 [quant-ph] (page 7).

[Bek73] Jacob D. Bekenstein. "Black holes and entropy". *Physical Review D*, 7(8):2333–2346, 1973. DOI: 10.1103/physrevd.7.2333 (page 54).

[Bel19] Aleksandrs Belovs. "Quantum Algorithms for Classical Probability Distributions":1–14, 2019. arXiv: 1904.02192 (page 59).

[BG12] Olivier Bernardi and Omer Giménez. "A Linear Algorithm for the Random Sampling from Regular Languages". *Algorithmica*, 62(1-2):130–145, 2012. DOI: 10.1007/s00453-010-9446-5 (page 129).

[Ber14] Dominic W. Berry. "High-order quantum algorithm for solving linear differential equations". *Journal of Physics A: Mathematical and Theoretical*, 47(10):105301, 2014. DOI: 10.1088/1751-8113/47/10/105301 (page 28).

[BC12] Dominic W. Berry and Andrew M. Childs. "Black-box Hamiltonian Simulation and Unitary Implementation". *Quantum Information & Computation*, 12(1-2):29–62, 2012. DOI: 10.26421/QIC12.1-2 (pages 13, 26, 36).

[BCC+14] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. *Exponential improvement in precision for simulating sparse hamiltonians*. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 283–292, New York, New York. ACM, 2014. DOI: 10.1145/2591796.2591854 (pages 137, 157).

[BCC+15] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. "Simulating Hamiltonian dynamics with a truncated taylor series". *Physical Review Letters*, 114(9):1–5, 2015. DOI: 10.1103/PhysRevLett.114.090502 (pages 12, 32).

[BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. "Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters". *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2015-Decem:792–809, 2015. DOI: 10.1109/FOCS.2015.54 (pages 13, 24, 34).

[BCO+17] Dominic W. Berry, Andrew M. Childs, Aaron Ostrander, and Guoming Wang. "Quantum algorithm for linear differential equations with exponentially improved dependence on precision". *Communications in Mathematical Physics*, 356(3):1057–1081, 2017. DOI: 10.1007/s00220-017-3002-y (pages 28, 127).

[BWP+17] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. "Quantum machine learning". *Nature*, 549(7671):195, 2017 (page 77).

[BMP+16] Bernd Bohnet, Ryan McDonald, Emily Pitler, and Ji Ma. *Generalized Transition-based Dependency Parsing via Control Parameters*. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 150–160, Stroudsburg, PA, USA. Association for Computational Linguistics, 2016. DOI: 10.18653/v1/P16-1015 (page 126).

[BKL+19] Fernando G. S. L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu. *Quantum SDP Solvers: Large Speed-Ups, Optimality, and Applications to Quantum Learning*. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 27:1–27:14, Dagstuhl,

Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. DOI: 10.4230/LIPIcs.ICALP.2019.27 (pages 11, 27).

[BHM+02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum amplitude amplification and estimation*. In *Quantum Computation and Information*. Samuel J. Lomonaco, Jr., and Howard E. Brandt, editors. Volume 305. AMS Contemporary Mathematics series, 2002, pages 53–74. DOI: 10.1090/conm/305/05215 (pages 48, 56, 155).

[BJ95] Nader H. Bshouty and Jeffrey C. Jackson. *Learning DNF over the uniform distribution using a quantum example oracle*. In *Proceedings of the eighth annual conference on Computational learning theory - COLT '95*. ACM Press, 1995. DOI: 10.1145/225298.225312 (page 92).

[BBD16] Jacob Buckman, Miguel Ballesteros, and Chris Dyer. *Transition-Based Dependency Parsing with Heuristic Backtracking*. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2313–2318, Stroudsburg, PA, USA. ACL (Association for Computational Linguistics), Association for Computational Linguistics, 2016. DOI: 10.18653/v1/D16-1254 (page 126).

[BTV01] Harry Buhrman, John Tromp, and Paul Vitányi. "Time and space bounds for reversible simulation". *Journal of Physics A: Mathematical and General*, 34(35):6821–6830, 2001. DOI: 10.1088/0305-4470/34/35/308. arXiv: 0101133v2 [arXiv:quant-ph] (page 133).

[CM18] Chris Cade and Ashley Montanaro. *The Quantum Complexity of Computing Schatten p-norms*. In Stacey Jeffery, editor, *13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018)*, volume 111 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 4:1–4:20, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. DOI: 10.4230/LIPIcs.TQC.2018.4 (page 61).

[CL18] Zi Cai and Jinguo Liu. "Approximating quantum many-body wave functions using artificial neural networks". *Physical Review B*, 97(3):035116, 2018. DOI: 10.1103/PhysRevB.97.035116 (page 127).

[CCD09] Pasquale Calabrese, John Cardy, and Benjamin Doyon. "Entanglement entropy in extended quantum systems". *Journal of Physics A: Mathematical and Theoretical*, 42(50):500301, 2009. DOI: 10.1088/1751-8121/42/50/500301 (page 54).

[CPP+13] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais. "Quantum algorithm and circuit design solving the Poisson equation". *New Journal of Physics*, 15(1):013021, 2013. DOI: 10.1088/1367-2630/15/1/013021 (page 15).

[CT17] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". *Science*, 355(6325):602–606, 2017. DOI: 10.1126/science.aag2302 (page 127).

[CRR18] Luigi Carratino, Alessandro Rudi, and Lorenzo Rosasco. "Learning with sgd and random features". In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10192–10203. Curran Associates, Inc., 2018 (pages 78, 99).

[CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. *The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation*. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 33:1–33:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. DOI: 10.4230/LIPIcs.ICALP.2019.33 (pages 19, 57, 59, 118).

[CLY+17] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. *Data-driven random fourier features using stein effect*. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017. DOI: 10.24963/ijcai.2017/207 (page 79).

[Che52] Herman Chernoff. "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations". *Ann. Math. Statist.*, 23(4):493–507, 1952. DOI: 10.1214/aoms/1177729330 (page 42).

[CGL+19] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. *Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning*. 2019 (pages 81, 97).

[CKS17] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. "Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision". *SIAM Journal on Computing*, 46(6):1920–1950, 2017. DOI: 10.1137/16M1087072 (pages 11–13, 16, 18, 30, 31, 34).

[CW12] Andrew M. Childs and Nathan Wiebe. "Hamiltonian simulation using linear combinations of unitary operations". *Quantum Information & Computation*, 12(11-12):901–924, 2012 (pages 12, 16).

[CSS19] Anirban N. Chowdhury, Rolando D. Somma, and Yigit Subasi. "Computing partition functions in the one clean qubit model", 2019. arXiv: 1910.11842 [quant-ph] (pages 56, 57, 63, 69, 70).

[CS17] Anirban Narayan Chowdhury and Rolando D. Somma. "Quantum algorithms for gibbs sampling and hitting-time estimation". *Quantum Information & Computation*, 17(1-2):41–64, 2017 (page 28).

[CHI+18] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. "Quantum machine learning: a classical perspective". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018. DOI: 10.1098/rspa.2017.0551. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2017.0551 (page 77).

[CC13] Peter Clifford and Ioana Cosma. *A simple sketching algorithm for entropy estimation over streaming data*. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 196–206, Scottsdale, Arizona, USA. PMLR, 2013 (page 54).

[CS02] Felipe Cucker and Steve Smale. "On the mathematical foundations of learning". *Bulletin of the American Mathematical Society*, 39:1–49, 2002 (page 85).

[Dąb08] Ewa Dąbrowska. "Questions with long-distance dependencies: a usage-based perspective". *Cognitive Linguistics*, 19(3), 2008. DOI: 10.1515/cogl.2008.015 (page 130).

[DXH+14] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. "Scalable kernel methods via doubly stochastic gradients". In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3041–3049. Curran Associates, Inc., 2014 (page 78).

[DSC08] Animesh Datta, Anil Shaji, and Carlton M. Caves. "Quantum discord and the power of one qubit". *Phys. Rev. Lett.*, 100:050502, 5, 2008. DOI: 10.1103/PhysRevLett.100.050502 (page 60).

[DLD17] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. "Quantum Entanglement in Neural Network States". *Physical Review X*, 7(2):021021, 2017. DOI: 10.1103/PhysRevX.7.021021 (page 127).

[Den96] Alain Denise. "Génération aléatoire uniforme de mots de langages rationnels". *Theoretical Computer Science*, 159(1):43–63, 1996. DOI: 10.1016/0304-3975(95)00200-6 (page 129).

[DRT00] Alain Denise, Olivier Roques, and Michel Termier. "Random generation of words of context-free languages according to the frequencies of letters". In *Mathematics and Computer Science*, pages 113–125. Birkhäuser Basel, Basel, 2000. DOI: 10.1007/978-3-0348-8405-1_10 (page 129).

[dWol19] Ronald de Wolf. *Quantum computing: lecture notes*. 2019 (page 7).

[Din06] Irit Dinur. "The PCP theorem by gap amplification". *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2006(3):241–250, 2006. DOI: 10.1145/1236457.1236459 (page 38).

[Don16] Xi Dong. "The gravity dual of rényi entropy". *Nature Communications*, 7(1), 2016. DOI: 10.1038/ncomms12472 (page 54).

[DB18] Vedran Dunjko and Hans J Briegel. "Machine learning & artificial intelligence in the quantum domain: a review of recent progress". *Reports on Progress in Physics*, 81(7):074001, 2018. DOI: 10.1088/1361-6633/aab406 (page 77).

[DH96] Christoph Durr and Peter Hoyer. *A quantum algorithm for finding the minimum*, 1996. arXiv: quant-ph/9607014 [quant-ph] (pages 49, 136).

[DBL+15] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. *Transition-Based Dependency Parsing with Stack Long Short-Term Memory*. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Stroudsburg, PA, USA. Association for Computational Linguistics, 2015. DOI: 10.3115/v1/P15-1033 (page 126).

[Ear70] Jay Earley. "An efficient context-free parsing algorithm". *Communications of the ACM*, 13(2):94–102, 1970. DOI: 10.1145/362007.362035 (page 128).

[Egg00] Leo Egghe. "The Distribution of N-Grams". *Scientometrics*, 47(2):237–252, 2000. DOI: 10.1023/A:1005634925734 (pages 132, 137).

[FLD18] Angela Fan, Mike Lewis, and Yann Dauphin. "Hierarchical Neural Story Generation", 2018. arXiv: 1805.04833 (page 126).

[FS02] Shai Fine and Katya Scheinberg. "Efficient svm training using low-rank kernel representations". *Journal of Machine Learning Research*, 2:243–264, 2002 (page 78).

[FKM+18] Keisuke Fujii, Hirotada Kobayashi, Tomoyuki Morimae, Harumichi Nishimura, Shuhei Tamate, and Seiichiro Tani. "Impossibility of classically simulating one-clean-qubit model with multiplicative error". *Phys. Rev. Lett.*, 120:200502, 20, 2018. DOI: 10.1103/PhysRevLett.120.200502 (page 60).

[Gid18] Craig Gidney. "Halving the cost of quantum addition". *Quantum*, 2:74, 2018. DOI: 10.22331/q-2018-06-18-74 (pages 48, 137).

[GAW19] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. "Optimizing quantum optimization algorithms via faster quantum gradient computation". In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019. DOI: 10.1137/1.9781611975482.87. arXiv: 1711.00465 (pages 127, 157).

[GL19] András Gilyén and Tongyang Li. "Distributional property testing in a quantum world":1–18, 2019. arXiv: 1902.00814 (pages 52, 55, 57–59, 72, 73).

[GSL+19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. *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*, STOC 2019, pages 193–204, Phoenix, AZ, USA, 2019. DOI: 10.1145/3313276.3316366. eprint: 1806.01838 (pages 12, 13, 20, 28, 46, 57, 59, 68, 69, 81, 84, 97, 98, 111, 119).

[GLM08a] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Architectures for a quantum random access memory". *Phys. Rev. A*, 78:052310, 5, 2008. DOI: 10.1103/PhysRevA.78.052310 (page 93).

[GLM08b] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum random access memory". *Phys. Rev. Lett.*, 100:160501, 16, 2008. DOI: 10.1103/PhysRevLett.100.160501 (page 93).

[GGJ11] Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. "Producing Power-Law Distributions and Damping Word Frequencies with Two-Stage Language Models". *Journal of Machine Learning Research*, 12:2335–2382, 2011 (page 137).

[GPS01] Massimiliano Goldwurm, Beatrice Palano, and Massimo Santini. *On the circuit complexity of random generation problems for regular and context-free languages*. In Afonso Ferreira and Horst Reichel, editors, *STACS 2001*, pages 305–316, Berlin, Heidelberg. Springer Berlin Heidelberg, 2001. DOI: 10.1007/3-540-44693-1_27 (pages 129, 133).

[GOS+11] Parikshit Gopalan, Ryan O'Donnell, Rocco A. Servedio, Amir Shpilka, and Karl Wimmer. "Testing fourier dimensionality and sparsity". *SIAM J. Comput.*, 40(4):1075–1100, 2011. DOI: 10.1137/100785429 (page 47).

[GJK+97] Vivek Gore, Mark Jerrum, Sampath Kannan, Z. Sweedyk, and Steve Mahaney. "A Quasi-polynomial-time Algorithm for Sampling Words from a Context-Free Language". *Information and Computation*, 134(1):59–74, 1997. DOI: 10.1006/inco.1997.2621 (page 129).

[Gra13] Alex Graves. "Generating Sequences With Recurrent Neural Networks", 2013. arXiv: 1308.0850 (page 126).

[GR02] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions". *quant-ph/0208112*, 2002 (pages 24, 93).

[Gro97] Lov K. Grover. "Quantum mechanics helps in searching for a needle in a haystack". *Phys. Rev. Lett.*, 79:325–328, 2, 1997. DOI: 10.1103/PhysRevLett.79.325 (page 155).

[Gue19] Gian Giacomo Guerreschi. "Repeat-until-success circuits with fixed-point oblivious amplitude amplification". *Phys. Rev. A*, 99:022306, 2, 2019. DOI: 10.1103/PhysRevA.99.022306. eprint: 1808.02900 (page 24).

[HH00] Lisa Hales and Sean Hallgren. "Improved quantum fourier transform algorithm and applications":515, 2000. DOI: 10.1109/SFCS.2000.892139 (pages 82, 97, 111).

[HRS18] Thomas Häner, Martin Roetteler, and Krysta M. Svore. *Optimizing quantum circuits for arithmetic*, 2018 (page 113).

[HZZ+19] Connor T. Hann, Chang-Ling Zou, Yaxing Zhang, Yiwen Chu, Robert J. Schoelkopf, S. M. Girvin, and Liang Jiang. "Hardware-efficient quantum random access memory with hybrid quantum acoustic systems". *Phys. Rev. Lett.*, 123:250501, 25, 2019. DOI: 10.1103/PhysRevLett.123.250501 (page 93).

[HCC+14] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. "Deep Speech: Scaling up end-to-end speech recognition", 2014. arXiv: 1412.5567 (page 151).

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". *Physical Review Letters*, 103(15):150502, 2009. DOI: 10.1103/PhysRevLett.103.150502 (pages 1, 11, 12, 15, 16, 18, 59, 80–82, 127).

[HV19] David Harvey and Joris Van Der Hoeven. *Integer multiplication in time O(n log n)*. 2019 (page 111).

[HGK+10] Matthew B. Hastings, Iván González, Ann B. Kallin, and Roger G. Melko. "Measuring renyi entanglement entropy in quantum Monte Carlo simulations". *Physical Review Letters*, 104(15):157201, 2010. DOI: 10.1103/PhysRevLett.104.157201 (page 55).

[HCT+19] Vojtech Havlícek, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. "Supervised learning with quantum-enhanced feature spaces". *Nature*, 567(7747):209–212, 2019. DOI: 10.1038/s41586-019-0980-2 (page 82).

[HS19] Steven Herbert and Sathyawageeswar Subramanian. *Spectral sparsification of matrix inputs as a preprocessing step for quantum algorithms*, 2019. arXiv: 1910.02861 [quant-ph] (page 6).

[HC83] Timothy Hickey and Jacques Cohen. "Uniform Random Generation of Strings in a Context-Free Language". *SIAM Journal on Computing*, 12(4):645–655, 1983. DOI: 10.1137/0212044 (page 129).

[HBF+19] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. "The Curious Case of Neural Text Degeneration", 2019. arXiv: 1904.09751 (page 126).

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 2nd edition*, volume 32 of number 1. 2001, page 60. DOI: 10.1145/568438.568455 (page 126).

[IMP+15] Rajibul Islam, Ruichao Ma, Philipp M. Preiss, M. Eric Tai, Alexander Lukin, Matthew Rispoli, and Markus Greiner. "Measuring entanglement entropy in a quantum many-body system". *Nature*, 528(7580):77–83, 2015. DOI: 10.1038/nature15750 (page 55).

[Jäg12] Gerhard Jäger. "Power Laws And Other Heavy-Tailed Distributions In Linguistic Typology". *Advances in Complex Systems*, 15(03n04):1–21, 2012. DOI: 10.1142/S0219525911500196 (page 137).

[JNN19] Prateek Jain, Dheeraj Nagaraj, and Praneeth Netrapalli. *Making the last iterate of sgd information theoretically optimal*. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 1752–1755, Phoenix, USA. PMLR, 2019 (pages 98, 99, 120, 122).

[JLS19] Dhawal Jethwani, François Le Gall, and Sanjay K. Singh. *Quantum-inspired classical algorithms for singular value transformation*. 2019 (pages 81, 97).

[JZW+18] Zhih-Ahn Jia, Yuan-Hang Zhang, Yu-Chun Wu, Guang-Can Guo, and Guo-Ping Guo. "Efficient Machine Learning Representations of Surface Code with Boundaries, Defects, Domain Walls and Twists", 2018. arXiv: 1802.03738 (page 127).

[JPC+19] N Jiang, Y-F Pu, W Chang, C Li, S Zhang, and L-M Duan. "Experimental realization of 105-qubit random access quantum memory". *npj Quantum Information*, 5(1):28, 2019 (page 93).

[JVH+15] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. "Minimax estimation of functionals of discrete distributions". *IEEE Transactions on Information Theory*, 61(5):2835–2885, 2015. DOI: 10.1109/tit.2015.2412945 (page 55).

[Jor08] Stephen P. Jordan. *Quantum Computation Beyond the Circuit Model*. PhD thesis, 2008. arXiv: 0809.2307 (pages 60, 61).

[KP17a] Iordanis Kerenidis and Anupam Prakash. "Quantum gradient descent for linear systems and least squares", 2017 (page 20).

[KP17b] Iordanis Kerenidis and Anupam Prakash. *Quantum Recommendation Systems*. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 49:1–49:21, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. DOI: 10.4230/LIPIcs.ITCS.2017.49 (pages 3, 12, 81, 93, 127).

[KHQ+18] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. *Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context*. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Stroudsburg, PA, USA. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/P18-1027 (pages 130, 152).

[Kim18] Young-Sik Kim. "Low complexity estimation method of rényi entropy for ergodic sources". *Entropy*, 20(9):657, 2018. DOI: 10.3390/e20090657 (page 54).

[KR03] Andreas Klappenecker and Martin Rötteler. "Engineering functional quantum algorithms". *Physical Review A*, 67(1), 2003. DOI: 10.1103/PhysRevA.67.010302 (page 12).

[KL98] E. Knill and R. Laflamme. "Power of one bit of quantum information". *Physical Review Letters*, 81(25):5672–5675, 1998. DOI: 10.1103/PhysRevLett.81.5672. arXiv: 9802037 [quant-ph] (pages 57, 60).

[KRS09] Robert Konig, Renato Renner, and Christian Schaffner. "The operational meaning of min- and max-entropy". *IEEE Transactions on Information Theory*, 55(9):4337–4347, 2009. DOI: 10.1109/tit.2009.2025545 (page 54).

[Kot14] Robin Kothari. "Efficient algorithms in quantum query complexity", 2014 (pages 12, 32, 157).

[KMC+18] Ilya Kulikov, Alexander H. Miller, Kyunghyun Cho, and Jason Weston. "Importance of a Search Strategy in Neural Dialogue Modelling", 2018. arXiv: 1811.00907 (page 126).

[Laf16] Nicolas Laflorencie. "Quantum entanglement in condensed matter systems". *Physics Reports*, 646:1–59, 2016. DOI: 10.1016/j.physrep.2016.06.008 (page 54).

[LSS13] Quoc Le, Tamas Sarlos, and Alexander Smola. *Fastfood - computing hilbert space expansions in loglinear time*. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of number 3 in *Proceedings of Machine Learning Research*, pages 244–252, Atlanta, Georgia, USA. PMLR, 2013 (pages 78, 101).

[LO08] Sarah K. Leyton and Tobias J. Osborne. "A quantum algorithm to solve nonlinear differential equations". *arXiv:0812.4423*, 2008 (page 12).

[LW19] Tongyang Li and Xiaodi Wu. "Quantum Query Complexity of Entropy Estimation". *IEEE Transactions on Information Theory*, 65(5):2899–2921, 2019. DOI: 10.1109/TIT.2018.2883306 (page 55).

[LTO+19] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. *Towards a unified analysis of random Fourier features*. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3905–3914, Long Beach, California, USA. PMLR, 2019 (page 79).

[LHC+19] Fanghui Liu, Xiaolin Huang, Yudong Chen, Jie Yang, and Johan A.K. Suykens. "Random fourier features via fast surrogate leverage weighted sampling", 2019 (page 79).

[Llo96] Seth Lloyd. "Universal quantum simulators". *Science*, 273(5278):1073–1078, 1996. DOI: 10.1126/science.273.5278.1073. eprint: https://science.sciencemag.org/content/273/5278/1073.full.pdf (page 127).

[LGZ16] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. "Quantum algorithms for topological and geometric analysis of data". *Nature communications*, 7:10138, 2016 (page 81).

[LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". *Nature Physics*, 10(9):631–633, 2014. DOI: 10.1038/nphys3029 (pages 27, 81).

[LP13] William Andrew Lorenz and Yann Ponty. "Non-redundant random generation algorithms for weighted context-free grammars". *Theoretical Computer Science*, 502:177–194, 2013. DOI: 10.1016/j.tcs.2013.01.006 (page 129).

[LC17] Guang Hao Low and Isaac L Chuang. "Optimal Hamiltonian Simulation by Quantum Signal Processing". *Physical Review Letters*, 118(1):010501, 2017. DOI: 10.1103/PhysRevLett.118.010501 (pages 18, 46).

[LC16] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". *arXiv:1610.06546*, 2016 (pages 12, 13, 18, 19, 59).

[McK97] Bruce McKenzie. *Generating Strings at Random from a Context Free Grammar*. Technical report, Department of Computer Science, University of Canterbury, Engineering Reports, 1997 (page 129).

[MD19] Riccardo Mengoni and Alessandra Di Pierro. "Kernel methods in quantum machine learning". *Quantum Machine Intelligence*, 1(3):65–71, 2019. DOI: 10.1007/s42484-019-00007-4 (page 81).

[Mon11] Ashley Montanaro. "Quantum search with advice". *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6519 LNCS:77–93, 2011. DOI: 10.1007/978-3-642-18073-6_7. arXiv: arXiv:0908.3066v1 (pages 131, 138, 139).

[MdW16] Ashley Montanaro and Ronald de Wolf. "A Survey of Quantum Property Testing". *Theory of Computing*, 1(1):1–81, 2016. DOI: 10.4086/toc.gs.2016.007. arXiv: 1310.2035 (pages 52, 127).

[MFF14] Tomoyuki Morimae, Keisuke Fujii, and Joseph F. Fitzsimons. "Hardness of classically simulating the one-clean-qubit model". *Phys. Rev. Lett.*, 112:130502, 13, 2014. DOI: 10.1103/PhysRevLett.112.130502 (page 60).

[Moz19a] Mozilla. *Common Voice*, 2019 (page 152).

[Moz19b] Mozilla. *DeepSpeech v.0.41*, 2019 (page 151).

[MDS+13] Martin Müller-Lennert, Frédéric Dupuis, Oleg Szehr, Serge Fehr, and Marco Tomamichel. "On quantum rényi entropies: a new generalization and some properties". *Journal of Mathematical Physics*, 54(12):122203, 2013. DOI: 10.1063/1.4838856 (page 53).

[MC18] Kenton Murray and David Chiang. *Correcting Length Bias in Neural Machine Translation*. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Stroudsburg, PA, USA. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/W18-6322 (page 126).

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2010. DOI: 10.1017/CBO9780511976667 (pages 7, 108, 111, 116, 127, 133).

[NRM+10] Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez-Rodríguez. *Evaluation of dependency parsers on unbounded dependencies*. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 833–841, Beijing China. Association for Computational Linguistics, 2010 (page 130).

[ODG13] Johan Oudinet, Alain Denise, and Marie-Claude Gaudel. "A new dichotomic algorithm for the uniform random generation of words in regular languages". *Theoretical Computer Science*, 502:165–176, 2013. DOI: 10.1016/j.tcs.2012.07.025 (pages 129, 133).

[PP18] Apoorva Patel and Anjani Priyadarsini. "Efficient quantum algorithms for state measurement and linear algebra applications". *International Journal of Quantum Information*, 16(06):1850048, September 2018. DOI: 10.1142/s021974991850048x (pages 27, 28).

[Pia14] Steven T. Piantadosi. "Zipf's word frequency law in natural language: A critical review and future directions". *Psychonomic Bulletin & Review*, 21(5):1112–1130, 2014. DOI: 10.3758/s13423-014-0585-6 (pages 132, 137).

[Pon12] Yann Ponty. *Rule-weighted and terminal-weighted context-free grammars have identical expressivity*. Research Report, 2012 (page 129).

[Pro07] John G Proakis. *Digital signal processing: principles algorithms and applications*. Pearson Prentice Hall, Upper Saddle River, N.J, 2007 (pages 91, 102).

[RR08] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008 (pages 77–79, 85, 95).

[RR09] Ali Rahimi and Benjamin Recht. "Weighted sums of random kitchen sinks: replacing minimisation with randomization in learning". In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1313–1320. Curran Associates, Inc., 2009 (pages 78, 79).

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification". *Phys. Rev. Lett.*, 113:130503, 13, 2014. DOI: 10.1103/PhysRevLett.113.130503 (page 81).

[RPW13] Vladimir Reinharz, Yann Ponty, and Jérôme Waldispühl. "A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution". *Bioinformatics*, 29(13):i308–i315, 2013. DOI: 10.1093/bioinformatics/btt217 (page 129).

[Rén61] Alfréd Rényi. *On measures of entropy and information*. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561, Berkeley, Calif. University of California Press, 1961 (page 53).

[RBM+17] Jonathan Romero, Ryan Babbush, Jarrod R. McClean, Cornelius Hempel, Peter Love, and Alán Aspuru-Guzik. "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz". *Quantum Science and Technology*, 5(1):4–11, 2017. DOI: 10.1088/2058-9565/aad3e4. eprint: 1701.02691 (page 28).

[RCC+18] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco. "On fast leverage score sampling and optimal learning". In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5672–5682. Curran Associates, Inc., 2018 (page 79).

[RR17] Alessandro Rudi and Lorenzo Rosasco. "Generalization properties of learning with random features". In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3215–3225. Curran Associates, Inc., 2017 (pages 77–79).

[SLS+19] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. "Black-box quantum state preparation without arithmetic". *Phys. Rev. Lett.*, 122:020502, 2, 2019. DOI: 10.1103/PhysRevLett.122.020502. eprint: 1807.03206 (pages 24, 48).

[Sch14] Juergen Schmidhuber. "Deep Learning in Neural Networks: An Overview". *Neural Networks*, 61:85–117, 2014. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828 (page 126).

[SS01] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001 (pages 77, 78).

[SV99] Leonard J. Schulman and Umesh V. Vazirani. *Molecular scale heat engines and scalable quantum computation*. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 322–329, Atlanta, Georgia, USA. ACM, 1999. DOI: 10.1145/301250.301332 (page 59).

[Sch95] Benjamin Schumacher. "Quantum coding". *Phys. Rev. A*, 51:2738–2747, 4, 1995. DOI: 10.1103/PhysRevA.51.2738 (page 54).

[SR07] David Sena Oliveira and Rubens Ramos. "Quantum bit string comparator: circuits and applications". *Quantum Computers and Computing*, 7, 2007 (page 137).

[SK19] Shahin Shahrampour and Soheil Kolouri. *On sampling random features from empirical leverage scores: implementation and theoretical guarantees*. 2019 (page 79).

[Sha49] C. E. Shannon. "Communication in the presence of noise". *Proceedings of the IRE*, 37(1):10–21, 1949. DOI: 10.1109/JRPROC.1949.232969 (pages 94, 102).

[SJ08] Peter W. Shor and Stephen P. Jordan. "Estimating jones polynomials is a complete problem for one clean qubit". *Quantum Info. Comput.*, 8(8):681–714, 2008 (page 60).

[SD16] Aman Sinha and John C Duchi. "Learning kernels with random features". In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1298–1306. Curran Associates, Inc., 2016 (page 77).

[SS00] Alex J. Smola and Bernhard Schökopf. *Sparse greedy matrix approximation for machine learning*. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., 2000 (page 78).

[SRG+02] Rolando D. Somma, G. Rtiz, J. E. Gubernatis, E. Knill, and R. Laflamme. "Simulating physical phenomena by quantum networks". *Physical Review A*, 65(4):042323, 2002. DOI: 10.1103/PhysRevA.65.042323. eprint: 0108146v1 (quant-ph) (page 12).

[SS11] Daniel A. Spielman and Nikhil Srivastava. "Graph Sparsification by Effective Resistances". *SIAM Journal on Computing*, 40(6):1913–1926, 2011. DOI: 10.1137/080734029. arXiv: 0803.0929 (pages 37–40, 43, 44, 46).

[ST11] Daniel A. Spielman and Shang Hua Teng. "Spectral sparsification of graphs". *SIAM Journal on Computing*, 40(4):981–1025, 2011. DOI: 10.1137/08074489X. arXiv: arXiv:0808.4134v3 (page 38).

[ST14] Daniel A. Spielmanm and Shang Hua Teng. "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems". *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. DOI: 10.1137/090771430. arXiv: 0607105v5 [arXiv:cs] (page 38).

[STN94] Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. *Improvements in beam search*. In *Third International Conference on Spoken Language Processing*, 1994 (page 126).

[SB16] Massimo Stella and Markus Brede. "Investigating the Phonetic Organisation of the English Language via Phonological Networks, Percolation and Markov Models". In pages 219–229. 2016. DOI: 10.1007/978-3-319-29228-1_19 (page 137).

[SBJ19] Sathyawageeswar Subramanian, Stephen Brierley, and Richard Jozsa. "Implementing smooth functions of a hermitian matrix on a quantum computer". *Journal of Physics Communications*, 3(6):065002, 2019. DOI: 10.1088/2399-6528/ab25a2 (page 5).

[SH19] Sathyawageeswar Subramanian and Min-Hsiu Hsieh. *Quantum algorithm for estimating renyi entropies of quantum states*, 2019. arXiv: 1908.05251 [quant-ph] (page 6).

[SGT18] Yitong Sun, Anna Gilbert, and Ambuj Tewari. "But how does it work in theory? linear svm with random features". In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3379–3388. Curran Associates, Inc., 2018 (page 79).

[SMH11] Ilya Sutskever, James Martens, and Geoffrey Hinton. *Generating text with recurrent neural networks*. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1017–1024, Bellevue, Washington, USA. Omnipress, 2011 (page 126).

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014 (page 126).

[Tan19] Ewin Tang. *A quantum-inspired classical algorithm for recommendation systems*. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing - STOC 2019*, pages 217–228, New York, New York, USA. ACM Press, 2019. DOI: 10.1145/3313276.3316310 (pages 81, 97, 127).

[Tha64] Henry C. Thacher Jr. "Conversion of a power to a series of chebyshev polynomials". *Commun. ACM*, 7(3):181–182, 1964. DOI: 10.1145/363958.363998 (page 32).

[UMM+18] Enayat Ullah, Poorya Mianjy, Teodor Vanislavov Marinov, and Raman Arora. "Streaming kernel pca with tilde o(sqrt n) random features". In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7311–7321. Curran Associates, Inc., 2018 (page 77).

[VV11] G. Valiant and P. Valiant. *The power of linear estimators*. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 403–412, 2011. DOI: 10.1109/FOCS.2011.81 (page 55).

[Val11] Paul Valiant. "Testing symmetric properties of distributions". *SIAM Journal on Computing*, 40(6):1927–1968, 2011. DOI: 10.1137/080734066 (page 56).

[VGG+17] Joran Van Apeldoorn, András Gilyén, Sander Gribling, Ronald de Wolf, Andras Gilyen, Sander Gribling, and Ronald de Wolf. "Quantum SDP-Solvers: Better upper and lower bounds". *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, (617), 2017. DOI: 10.1109/FOCS.2017.44 (pages 11, 13, 25, 28, 127, 136, 137).

[vAG18] Joran van Apeldoorn and András Gilyén. "Improvements in Quantum SDP-Solving with Applications". *arXiv:1804.05058v1*, 2018 (page 27).

[VCS+16] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. "Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models":1–16, 2016. arXiv: 1610.02424 (page 126).

[VG18] David Vilares and Carlos Gómez-Rodríguez. "Transition-based Parsing with Lighter Feed-Forward Networks", 2018. arXiv: 1810.08997 (page 126).

[WBS+19] Nathan Wiebe, Alex Bocharov, Paul Smolensky, Matthias Troyer, and Krysta M Svore. "Quantum Language Processing", 2019. arXiv: 1902.05162 (page 127).

[WBL12] Nathan Wiebe, Daniel Braun, and Seth Lloyd. "Quantum algorithm for data fitting". *Phys. Rev. Lett.*, 109:050505, 5, 2012. DOI: 10.1103/PhysRevLett.109.050505 (page 81).

[WS01] Christopher K. I. Williams and Matthias Seeger. "Using the nyström method to speed up kernel machines". In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001 (page 78).

[WR16] Sam Wiseman and Alexander M. Rush. *Sequence-to-Sequence Learning as Beam-Search Optimization*. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Stroudsburg, PA, USA. Association for Computational Linguistics, 2016. DOI: 10.18653/v1/D16-1137 (page 126).

[WZP18] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. "Quantum linear system algorithm for dense matrices". *Phys. Rev. Lett.*, 120:050502, 5, 2018. DOI: 10.1103/PhysRevLett.120.050502 (pages 81, 127).

[WY16] Y. Wu and P. Yang. "Minimax rates of entropy estimation on large alphabets via best polynomial approximation". *IEEE Transactions on Information Theory*, 62(6):3702–3720, 2016. DOI: 10.1109/TIT.2016.2548468 (page 55).

[YSS+20] Hayata Yamasaki, Sathyawageeswar Subramanian, Sho Sonoda, and Masato Koashi. *Fast quantum algorithm for learning with optimized random features*, 2020. arXiv: 2004.10756 [quant-ph] (page 7).

[YHM18] Yilin Yang, Liang Huang, and Mingbo Ma. *Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Stroudsburg, PA, USA. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/D18-1342. arXiv: 1808.09582 (page 126).

[YSC+16] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. "Orthogonal random features". In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1975–1983. Curran Associates, Inc., 2016 (pages 78, 101).

[ZGH+18] Chao Zhang, Xianjie Gao, Min-Hsiu Hsieh, Hanyuan Hang, and Dacheng Tao. "Matrix Infinitely Divisible Series: Tail Inequalities and Applications in Optimization", 2018. eprint: 1809.00781 (page 73).

[Zha12] Shengyu Zhang. *Bqp-complete problems*. In *Handbook of Natural Computing*. Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pages 1545–1571. DOI: 10.1007/978-3-540-92910-9_46 (page 82).

[ZC08] Yue Zhang and Stephen Clark. *A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics, 2008 (page 126).

[ZN11] Yue Zhang and Joakim Nivre. *Transition-based dependency parsing with rich non-local features*. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193. Association for Computational Linguistics, 2011 (page 126).

[ZLO+07] Haiquan (Chuck) Zhao, Ashwin Lall, Mitsunori Ogihara, Oliver Spatscheck, Jia Wang, and Jun Xu. *A data streaming algorithm for estimating entropies of od flows*. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 279–290, San Diego, California, USA. Association for Computing Machinery, 2007. DOI: 10.1145/1298306.1298345 (page 54).

[ZFF19] Zhikuan Zhao, Jack K. Fitzsimons, and Joseph F. Fitzsimons. "Quantum-assisted gaussian process regression". *Phys. Rev. A*, 99:052331, 5, 2019. DOI: 10.1103/PhysRevA.99.052331 (page 81).

[ZQH15] Chenxi Zhu, Xipeng Qiu, and Xuanjing Huang. *Transition-Based Dependency Parsing with Long Distance Collocations*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 12–24, 2015. DOI: 10.1007/978-3-319-25207-0_2 (pages 126, 130, 138).