Yanlin Chen On Quantum Algorithms and Limitations for Convex Optimization and Lattice Problems

On Quantum Algorithms and Limitations for Convex Optimization and Lattice Problems

ILLC Dissertation Series DS-2024-12

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

Cover design by Tsú-hâm Tshî.
Printed and bound by Ipskamp Printing.

ISBN: 978–94–6473–607–6.

On Quantum Algorithms and Limitations for Convex Optimization and Lattice Problems

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P. P. C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op donderdag 21 november 2024, te 10.00 uur

door Yanlin Chen

geboren te Taiwan

Promotiecommissie

Promotor:	prof. dr. R.M. de Wolf	Universiteit van Amsterdam
Copromotor:	dr. M. Ozols	Universiteit van Amsterdam
Overige leden:	prof. dr. S.M. Jeffery	Universiteit van Amsterdam
	prof. dr. H.M. Buhrman	Universiteit van Amsterdam
	prof. dr. D.N. Dadush	Universiteit Utrecht
	dr. F. Speelman	Universiteit van Amsterdam
	dr. J. Zuiddam	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

件件工作

反應自我

凡經我手

必爲佳作

李德財　提於中研院資訊所

List of publications

This thesis is based on the following papers. In each work, all authors contributed equally.

[CGW24] ***A Quantum Speed-Up for Approximating the Top Eigenvectors of a Matrix***
Yanlin Chen, András Gilyén, Ronald de Wolf.
In submission. Presented as a talk at QIP'24.
Chapter 4 and Chapter 7 are based on this paper.

[CCK+23] ***QSETH Strikes Again: Finer Quantum Lower Bounds for Lattice Problem, Strong Simulation, Hitting Set Problem, and More***
Yanlin Chen, Yilei Chen, Rajendra Kumar, Subhasree Patro, Florian Speelman.
In submission.
Chapter 8 is based on this paper.

[CW23] ***Quantum Algorithms and Lower Bounds for Linear Regression with Norm Constraints***
Yanlin Chen, Ronald de Wolf.
In Proceedings of the 50th International Colloquium on Automata, Languages, and Programming. Also presented as a talk at QIP'22.
Chapter 3 and Chapter 6 are based on this paper.

[ACK+21b] ***Improved (Provable) Algorithms for the Shortest Vector Problem via Bounded Distance Decoding***
Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, Yixin Shen.
In Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science. Also presented as a talk at QIP'22.
Chapter 5 is based on the quantum part of this paper.

The author also coauthored the following papers during his PhD, which are not included in this thesis.

[BCK+23] ***Lattice Problems in General Norms: Algorithms with Explicit Constants, Dimension-Preserving Reductions, and More***
Huck Bennett, Yanlin Chen, Rajendra Kumar, Zeyong Li, Spencer Peters.
In submission.

[ACK+21a] ***Dimension-Preserving Reductions Between SVP and CVP in Different p-Norms***
Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, Zeyong Li, and Noah Stephens-Davidowitz.
In Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms.

Acknowledgements

First and foremost, I would like to thank Ronald de Wolf, who has been an incredibly thorough supervisor, mentor, promotor, and coauthor. He consistently offers valuable advice and feedback to everyone. I began my PhD at a relatively challenging and somber time, during the COVID pandemic. Life was expected to be difficult, but with Ronald's help, I smoothly integrated into the new environment and group. I have to be honest: I barely knew anything about quantum computing when I joined QuSoft as a PhD student, and since my undergraduate major was electrical engineering, I knew very little about theoretical computer science either. Still, within nine months of work, we demonstrated a provable quantum advantage for Lasso, which certainly helped me overcome my imposter syndrome. Ronald is also exceptionally skilled at organizing my racing thoughts and extracting the most useful parts from them. In many cases, I might say something random and nonsensical, yet he can distill those ideas into a concrete result. I also greatly appreciate that he trained me to write proper mathematical statements and clear English. Both skills have really helped me clarify my thoughts and extract the most succinct and valuable insights. The most important thing I learned from him is to be succinct, and I will strive to follow this principle for the rest of my academic life.

I would also like to thank my co-promotor, Māris Ozols. Though we did not collaborate on any publications, I highly appreciate your organization of QuLunch and Stage Night, as well as the many useful career discussions.

I am grateful to Stacey Jeffery, Harry Buhrman, Daniel Dadush, Florian Speelman, and Jeroen Zuiddam for agreeing to be members of my PhD committee.

Before I began my PhD, I spent three years in substitute military service in Kai-Min Chung's lab at Academia Sinica, and I would like to thank Kai-Min and all the group members there: Yu-Chi Chen, Wei-Kai Lin, Tsung-Hsuan Hung, Chi-Ning Chou, Yin-Hsun Huang, Jyun-Jie Liao, Kuan-Yi Ho, Mi-Ying Huang, Hao Chung, Chiao-Hsun Wang, Hao-Ting Wei, Chun-Hsiang Chan, Yi Lee, Yu-Ching Shen, and Yao-Ting Lin. That was a very fun and enjoyable time, where we freely chatted during my Master's studies and military service. I truly enjoyed the moments when we had random, crazy ideas, which often turned out to be impossible in the end. I would also like to thank Chi-Ning and Xun Gao for continuing discussions on quantum max-cut and quantum SOS relaxation even after I left Kai-Min's lab. I also thank Wei-Ching Chien for always assisting me with administrative matters at Academia Sinica.

I would like to thank Divesh Aggarwal for hosting me at CQT during the last year of my military service, and I am incredibly grateful that he introduced me to the research problem of studying the time-space tradeoffs for SVP. I thank Yixin Shen and Rajendra Kumar for revisiting the idea of a smooth time-space tradeoff for SVP. It was a fun time, and it was exciting to discover that our quantum algorithm accidentally outperformed the classical state-of-the-art. I also thank Rajendra Kumar for brainstorming lattice algorithms for different norms.

I am thankful to Srinivasan Arunachalam, my mentor during a summer internship at IBM. He always had plenty of research ideas, and it was a lot of fun exploring quantum lower bounds for covariance estimation. I also thank Yassine Hamoudi for countless useful discussions on quantum covariance estimation at the Simons Institute. Additionally, I greatly appreciate Ming-Ko Cho, Ting-Yuan Hsia, Cheng-You Yao, Lang-Chi Yu, and Yu-Chuan Yen for showing me around some famous sights in the Bay Area during the summer internship.

During my PhD, I learned a lot from my incredibly talented coauthors, and I would like to express my gratitude here. Most of my lattice-related results were in collaboration with Rajendra Kumar, Zeyong Li, Yixin Shen, Divesh Aggarwal, Spencer Peters, Huck Bennett, and Noah Stephens-Davidowitz. I would especially like to thank Rajendra and Yixin for always being available to discuss lattice-related algorithmic ideas, and I am very pleased that we found a way to further improve our quantum algorithm for SVP. I am also very appreciative of coauthoring with Ronald de Wolf and András Gilyén on quantum algorithms for finding the top eigenvectors, which is such a fundamental problem and was suggested by Alex Wang and Daniel Dadush. As András said, we all did our best on this paper, and the result is indeed amazing and near-optimal. I also appreciate Subhasree Patro and Florian Speelman for discussions on quantum fine-grained complexity. It was painful but rewarding to carefully write down the quantifiers for everything. I would also like to thank Rajendra Kumar and Yilei Chen for suggesting many useful lattice and optimization problems to include in our paper on quantum fine-grained complexity.

Thanks to all my not-yet-mentioned collaborators who worked with me on topics beyond the scope of this dissertation: Amin Shiraz Gilani, Stacey Jeffery, Lynn Engelberts, Amira Abbas, Tom Gur, Matthias Caro, Maya-Iggy van Hoof, Simon Apers, Jop Briët, Jonas Helsen, Davi Silva, Niels Neumann, Arjan Cornelissen, Wei-Cheng Lee, and Yi-Shan Wu. Special thanks to Yi-Shan for helping me with all sorts of ML-related problems and discussions.

I am a weak learner, and as a result, I need a group of amazing oracles to learn from. I would like to thank my friends for serving as my "oracles" and allowing me to query them all the time. My deepest thanks go to my ML oracles, Yi-Shan Wu and Po-An Wang, my statistics oracle, Hsin-Po Wang, my (quantum) crypto oracles, Kai-Min Chung, Yao-Ting Lin, Han Hsuan Lin, and Wei-Kai Lin, my optimization oracle, Harold Nieuwboer, my fine-grained complexity oracle, Subhasree Patro, my additive combinatorics oracle, Jyun-Jie Liao, and both my academic writing oracle and quantum oracle, Ronald de Wolf. I would particularly like to thank Yi-Shan Wu, Hsin-Po

Wang, Subhasree Patro, Jyun-Jie Liao, Harold Nieuwboer, and Wei-Kai Lin for being my "rubber ducks" all the time: I can always explain what I've done, and you never hesitate to offer opinions and feedback.

I would like to thank, for many inspiring and insightful discussions, Sander Gribling, Joran van Apeldoorn, Nikhil Mande, Léo Ducas, Andris Ambainis, Gilles Brassard, Daniel Dadush, Jeroen Zuiddam, John van de Wetering, Māris Ozols, Mario Szegedy, Christian Majenz, Armando Bellante, Jordi Weggemans, and Qipeng Liu. I would also like to thank Michael Walter for hosting me during my visit to Ruhr University Bochum, François Le Gall for hosting me during my visit to Nagoya University, Yassine Hamoudi for hosting me during my visit to LaBRI, and Frédéric Magniez for hosting me during my visit to IRIF.

I had an amazing time at CWI and QuSoft, and my peers there enriched my life. I would like to thank Nikhil for always being ready for foosball and always being full of energy. I also thank Lynn and Francisco for hosting junior meetings twice per month. I am grateful to have shared (or to currently share) an office with Adam, Harold, Galina, Jelena, Garazi, and Ake. It's always wonderful to discuss research and chat about daily life with all of you. I thoroughly enjoyed playing foosball with Amira, Léo, Dyon, Francisco, Randy, Llorenc, Simona, Lisa, and Shane. I also thank Doutzen and Susanne for hosting QuTea every Tuesday afternoon and for organizing social events, both during and after COVID. I also thank Koen, Peter, Joris, Freek, Philip, René, Dmitry, Bas, Poojith, Sebastian, Alex, Jana, Ido, Chris, Yaroslav, Seenivasan, John, Akshay, Victor, Yvonne, Marten, Jordi, Quinten, Sebastian, Manideep, Ailsa, Daan, Gina, Maxim, Marten, and Álvaro for making QuSoft a wonderful place to be.

Finally, I am grateful to my parents and grandmother for teaching me to always be humble, friendly, and polite to others. I am also thankful to my brother for constant support. I am very lucky to have a wonderful girlfriend like Chao-Yu, who is always considerate and understanding. Although dealing with a two-body problem is quite difficult, we have managed it well so far. Hopefully, we can solve this problem soon.

I truly enjoyed my PhD journey, and I received an immense amount of help from so many people. Even though I have tried to express my gratitude in this text, I cannot fully capture how fortunate I feel to have experienced all of this. Hopefully, I have given due credit to everyone who deserves it and have not overlooked anyone in this acknowledgment.

Amsterdam, Yanlin Chen
October 1, 2024

Abstract

Optimization is the process of selecting the best option from all possibilities, and problems related to finding the best option among many options are called optimization problems. A few examples of such problems are route picking, partial loading, crew scheduling, and portfolio optimization, and those problems naturally appear everywhere in society and in our daily lives.

Most optimization problems could be solved by listing all possibilities and then searching for the best one, but we aim to find a systematic strategy (known as an algorithm) that finds the optimal solution in the least amount of time, as brute-force searching is often too time-consuming. That is, we would like to find optimal algorithms for finding optimal solutions.

Quantum physics provides a more accurate explanation and prediction of nature in many cases than classical physics. Therefore, a quantum computer has the potential to be more powerful and useful than a classical one for solving optimization problems, though the extent of this advantage remains an active area of research. Moreover, designing algorithms for a quantum computer requires fundamentally different ideas and concepts than for a classical computer.

In this thesis, we explore the uses of quantum computers in solving several fundamental optimization problems. We specifically consider solving linear regression, finding the top eigenvectors of a matrix, and finding the shortest nonzero vector of a lattice. More precisely, we provide near-optimal quantum algorithms that are asymptotically better than the best-possible classical algorithms for linear regression with ℓ_1-norm constraint (known as Lasso) and for approximating the top eigenvectors of a matrix. Our algorithms for finding the shortest nonzero vector of a lattice are asymptotically better than the best known classical algorithms.

On the other hand, there are some problems for which no useful computational advantage is possible, even when we use quantum computers. This situation, however, can sometimes be beneficial if we do not want quantum computers to solve specific problems fast - say, problems relevant to post-quantum cryptography. In such a situation, we would like evidence demonstrating the difficulty of solving these problems on a quantum computer. To this end, we propose variants of Boolean satisfiability problems that are believed to not be more efficiently solvable on a quantum computer, and then via reductions use those (conjectured) quantum lower bounds to study the quantum limitations of several popular optimization problems like lattice problems, quantum strong simulation, and hitting set problems.

Samenvatting

Optimalisatie is het proces waarbij de beste optie uit alle mogelijkheden wordt geselecteerd, en problemen die te maken hebben met het vinden van de beste optie uit vele opties worden optimalisatieproblemen genoemd. Enkele voorbeelden van dergelijke problemen zijn routekeuze, gedeeltelijke belading, roosterplanning en portefeuille-optimalisatie. Deze problemen komen veel voor in de maatschappij en in ons dagelijks leven.

De meeste optimalisatieproblemen kunnen worden opgelost door alle mogelijkheden op te sommen en vervolgens de beste te selecteren, maar wij streven ernaar een systematische strategie te vinden (een algoritme) die de optimale oplossing in de kortst mogelijke tijd vindt, aangezien brute-force zoeken vaak te tijdrovend is. Met andere woorden, we willen optimale algoritmen vinden voor het vinden van optimale oplossingen.

Kwantumfysica biedt in veel gevallen een nauwkeurigere verklaring en voorspelling van de natuurlijke fenomenen dan de klassieke fysica. Daarom heeft een kwantumcomputer het potentieel om krachtiger en nuttiger te zijn dan een klassieke computer voor het oplossen van optimalisatieproblemen, hoewel het nog steeds een actief onderzoeksgebied is om uit te zoeken hoe groot dat voordeel is. Het ontwerpen van algoritmen voor een kwantumcomputer vereist fundamenteel andere ideeën en concepten dan voor een klassieke computer.

In dit proefschrift onderzoeken we het gebruik van kwantumcomputers bij het oplossen van verschillende fundamentele optimalisatieproblemen. We richten ons specifiek op het oplossen van lineaire regressie, het vinden van de belangrijkste eigenvectoren van een matrix, en het vinden van de kortste niet-nul vector van een lattice. Meer in het bijzonder bieden we bijna-optimale kwantum algoritmen die asymptotisch beter zijn dan de best mogelijke klassieke algoritmen voor lineaire regressie met ℓ_1-normbeperking (ook bekend als Lasso) en voor het benaderen van de belangrijkste eigenvectoren van een matrix. Onze algoritmen voor het vinden van de kortste niet-nul vector van een lattice zijn asymptotisch beter dan de best bekende klassieke algoritmes.

Aan de andere kant zijn er enkele problemen waarvoor geen nuttig computationeel voordeel mogelijk is, zelfs niet wanneer we kwantumcomputers gebruiken. Deze situatie kan soms voordelig zijn als we niet willen dat kwantumcomputers specifieke bepaalde problemen snel kunnen oplossen - bijvoorbeeld problemen die relevant zijn

voor post-kwantum cryptografie. In zo'n situatie willen we bewijs voor de moeilijkheid van het oplossen van deze problemen op een kwantumcomputer. Om deze reden stellen we varianten van Booleaanse vervulbaarheidsproblemen voor waarvan wordt aangenomen dat ze niet efficiënter kunnen worden opgelost door een kwantumcomputer, en gebruiken we deze (vermoede) kwantum ondergrenzen via reducties om de kwantumbeperkingen van enkele populaire optimalisatieproblemen te bestuderen, zoals lattice problemen, kwantum sterke-simulatie en problemen over hitting sets.

摘要

最佳化（Optimization）指從所有的可能性中選擇最好、或者盡可能好的那個。在日常生活中，我們會請導航軟體帶我們從甲地到乙地、在固定的機艙空間中塞進更多更緊急的貨品、爲醫院員工和火車安排班表、尋找低風險高收益的投資組合。這些都是最佳化演算法能派得上用場的地方。

這類的最佳化演算法儘管已經非常成熟、且每年帶來數百億美元的效益，但他們始終沒有眞正觸及最核心的問題，即，「最佳」意味著從「所有」的可能中找到最好的。但是在實際應用中，列出所有可能性的成本往往遠遠超過預期的好處，這就是爲什麼我們迫切的需要更多、更準、更便宜的數學工具來解決這些問題。又或者，我們可以利用量子力學的特性，請大自然來幫我們解決這些問題？

在許多情況下，量子力學比古典力學提供了更準確的自然解釋和預測。因此，量子電腦在解決最佳化問題時有潛力比古典電腦更快更好，儘管這一優勢仍是一個活躍的研究領域。此外，爲量子電腦設計演算法需要本質上完全不同的想法和概念。依據量子力學，我們可以考慮每個粒子狀態以及粒子與粒子間的纏結（交互作用），使得一個精心設計的量子系統可以以一種「平行化」的方式，在固定的時間內探索更大量的可能性。而爲了把最佳解決方案從這樣的量子系統中「提取」出來，我們亦需要設計相對應的「量子轉古典」的介面。

本論文意在探討如何使用量子電腦解決最佳化問題。我們設計了可以解線性回歸、找特徵根與特徵向量、找最短晶格向量的量子演算法。並且我們還證明，因爲某些數學以及量子物理上的限制，這些量子演算法已經好到幾乎沒有改進的空間。在解有ℓ_1-constraint 的線性回歸（通常稱爲Lasso）和特徵向量時，隨著參數慢慢變大，我們的量子演算法會優於任何古典演算法。這論述包含尚未被發現的古典演算法在內，因爲數學以及古典物理上的限制，任何不用到量子力學的演算法的能力有一個明確的上限。至於最短晶格向量的問題，雖然沒辦法證明我們的方法好過所有的古典演算法，但卻好過所有我們知道的古典演算法。

另一方面，即使使用量子電腦，也有一些問題無法獲得任何實質性的計算優勢。然而，在某些情況下，如果我們不希望量子電腦快速解決特定問題——例如與後量子密碼學相關的問題，這種情況可能是有利的。在這種情況下，我們希望有證據顯示解決這些問題的困難性以及在量子電腦上進行這些計算的精確複雜度。爲此，我們提出了一些布林可滿足性問題的變體問題，這些變體問題被廣泛認爲無法有效地通過量子電腦解決，並且我們使用這些（假設的）量子限制來研究一些流行的最佳化問題（如晶格問題、量子強模擬和碰撞集問題）的量子限制。

Contents

CHAPTER 1

Overview

1.1 Introduction

This thesis is dedicated to the potential use of quantum computers in fundamental optimization problems.

1.1.1 Quantum computing

Quantum computing is interdisciplinary, lying at the intersection of two fields: *quantum mechanics* and *computer science*.

Quantum mechanics The field of quantum mechanics began at the start of the 20th century. Planck proposed the idea of "quantized energy" in 1900 [Pla00a; Pla00b] to explain the observations of black-body radiation. This "quantum" idea soon inspired many scientists, to explain and predict the nature of the photoelectric effect [Ein05], the spectral lines of the hydrogen atom [Boh13], Bose-Einstein condensation [Ein24], and Pauli's exclusion principle [Pau25]. Heisenberg, Born, and Jordan [Hei25; BJ25; BHJ26] further proposed wave function and matrix mechanics notions to formalize quantum mechanics and explain the seemingly contradictory idea that a photon, the particle of light, behaves like a wave. Light indeed comes in discrete packets (as do other particles), but the position of these packets is not precisely deterministically defined. However, it can still be described by a certain probability, leading to wave-like behavior as these probabilities spread through space. The celebrated uncertainty principle of Heisenberg states that these probabilities do not originate from our lack of knowledge about the world; they are inherent to the world: until observed, there is no real position of the particle. Instead, the system is described by a set of amplitudes, one for each possible position in space, from which the corresponding probability can be derived. By the end of the 1930s, quantum mechanics had grown to become a major field of physics.

Computer science Computer science studies *computation*, *automation*, and *resources*. People thousands of years ago already cared about computing things faster and more accurately: ancient people invented the abacus to perform daily calculations. Modern computer science, on the other hand, started with Church and Turing, built on the work of Gödel, formalizing the notion of an automatic machine [Tur37; Chu36], now known as a Turing machine. An *algorithm*, which is the key to automation, is a finite sequence of rigorous instructions so precise that even an idiot or a machine can perform it. Indeed, we also care about how complex and time-consuming such instructions are, and hence, people design algorithms that not only compute the answer but use the fewest resources or, equivalently, algorithms that are the most efficient. After Edmonds introduced the idea of efficient algorithms [Edm65], Hartmanis and Stearns formalized the notion of a complexity class [HS65], and by further introducing the notion of the class of problems that are solvable by an efficient algorithm (that is, an algorithm that runs in polynomial time in the input size), they started the field of complexity theory, which classifies computational problems according to their resource usage.

Quantum computing Nature can be seen as a machine. Quantum mechanics, a more elaborate and complicated theory than classical mechanics to explain Nature, could, of course, possibly provide new insights for designing automatic machines. Wiesner introduced the idea of quantum money [Wie83] and started the whole field of quantum information.[1] The field of quantum computation, on the other hand, was initiated in the early 1980s by Deutsch, Feynman, Manin, and Benioff [Deu85; Fey82; Fey86; Man80; Man99; Ben82], when they realized an exponential increase in overhead for simulating quantum dynamics. They proposed that a quantum computer, based on the laws of quantum mechanics, could be more efficient than a classical computer for simulating quantum systems. Several quantum advantages were explored soon after the initiation of the quantum computation and quantum information fields: the perfect security of quantum key distribution by Bennett and Brassard [BB84], an efficient quantum algorithm for factoring by Shor [Sho97], and a general quadratic speedup for the unstructured search by Grover [Gro96]. Quantum computing has since blossomed into a major field at the intersection of physics, mathematics, and computer science. The past two decades have seen much research trying to understand the tasks for which quantum provides an advantage.

1.1.2 Optimization

Optimization is the process of selecting the best element/object from a set of available alternatives with regard to some criterion. Optimization problems naturally arise in all quantitative disciplines, and the development of solution methods has been of interest in mathematics for centuries. In practice, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the function's value. Depending on the

[1] The idea in [Wie83] was proposed by Wiesner already in 1970 and remained unpublished until 1983.

variables of an optimization problem, it can generally be divided into two categories: continuous optimization and discrete optimization. Here, we introduce some notable branches of these two categories.

One of the most popular branches of continuous optimization is *convex optimization*. As its name suggests, it optimizes a convex function over a convex domain. One of the fundamental problems in convex optimization is least-squares linear regression: given a bunch of data and its corresponding label, we would like to fit a line through those data (with respect to least-squares error). Linear regression and its variants have many applications in the field of machine learning and hence attract much attention. Another fundamental problem in convex optimization is finding the top eigenvalue and its corresponding eigenvector of a given matrix, which could be considered the most important information of a matrix. Many convex optimization algorithms also involve finding the top eigenvector of a matrix as a subroutine. One more important fundamental problem in convex optimization is linear programming, which optimizes a linear function over a domain described by linear constraints. Linear programming plays a major role in convex optimization because many practical problems can be expressed as linear programming problems.

Integer programming, on the other hand, plays a very important role in discrete optimization, in which the variables of the integer program are restricted to integer points or lattice points. Integer programming is a natural problem generalized from linear programming. The combined algorithmic study of these problems was called the algorithmic geometry of numbers by Kannan [Kan87b]. This type of research soon attracted the attention of mathematicians to test new combinatorial and geometric techniques on exceedingly difficult computational problems [Kan87a; LLL82; Len83]. The subject of lattice algorithms has grown exponentially due to its many new connections to cryptography and complexity theory.

1.1.3 Quantum algorithms and limitations for optimization problems

As the hardware of a quantum computer is rapidly evolving, quantum scientists are also continually improving and creating new quantum algorithms for solving optimization problems. Many quantum algorithms have been invented during this decade, including algorithms for solving semidefinite programming, least-squares fitting, gradient descent, and more. Still, these quantum algorithms are not yet enough to handle many existing optimization problems, and hence, we need to explore further the possibilities of a quantum computer.

Of course, quantum computing has its *limits*. Bennett, Bernstein, Brassard, and Vazirani showed that the quadratic speedup over the unstructured search is the best we can hope for by an argument called the "hybrid method" [BBB+97]. After that, Beals, Buhrman, Cleve, Mosca, and de Wolf gave a general tool for recognizing the limitations of quantum queries [BBC+01] by observing that the amplitudes of the final state of T-query quantum algorithms are degree-T polynomials. Ambainis further generalized the idea of Bennett, Bernstein, Brassard, and Vazirani, giving a clean combinatorial

method for proving quantum query lower bounds [Amb02]. These methods are very helpful to prevent people from wasting their time trying to invent something totally impossible.

The above-mentioned lower bound tools are for query lower bounds. Time lower bounds for a computation/optimization problem are, in general, hard to prove. An alternative way to study (both classical and quantum) time lower bounds for these problems is to assume a certain important problem (which people believe is hard) needs exponential time to solve, and then try to reduce that problem to other problems, which then must be also hard. This is called *conditional (quantum) lower bound* or *(quantum) fine-grained complexity* in the computer science field. By transferring the conjectured intractability to another problem, we can characterize the hardness of other problems. Moreover, if we find a more efficient algorithm for any of these problems, this algorithm can immediately be converted to a more efficient algorithm for solving such a certain important problem.

1.2 Contribution

In this thesis, we propose quantum algorithms and analyze their limitations for solving linear regression, top eigenvectors, and lattice problems. Parts of our quantum algorithms are created by "quantizing" existing classical algorithms. Obviously, not all "quantized" classical algorithms have a quantum speedup, and many of them have actually been shown to have no quantum speedup [GKN+21]. All classical algorithms we choose to quantize (and achieve quantum speedup) in this thesis are typically *robust* against tiny errors.

The results presented in this work are organized into three parts. In Part I, we will discuss quantum algorithms for fundamental optimization problems like linear regression, finding the top eigenvectors, and the shortest vector problem of a lattice; in Part II, we include both classical and quantum lower bounds for those problems we mentioned above, showing our quantum algorithms in Part I are nearly optimal and surely have a quantum advantage over the best possible classical ones. In Part III, we will introduce a framework on quantum conditional lower bounds and apply this framework to other optimization problems.

Parts of our quantum algorithms include the use of QRAM (also called QCRAM). In analogy with classical RAM, a QRAM is a device that stores a classical string and allows efficient access to the individual bits of the string, as well as to several bits in superposition. Note that the QRAM memory content itself is classical: it is just a classical string, not a superposition of such strings. It should be noted that QRAM is a controversial notion in some corners of quantum computing. Still, classical RAM is not considered a problematic notion and in practice is implemented in near-perfect hardware. Since classical RAM is not problematic and one has to allow quantum superposition anyway in order to do anything in quantum computing, we feel that assuming QRAM is conceptually acceptable. On the other hand, we will also include QRAM-free counterparts in this thesis, and those still show quantum advantages.

1.2.1 Overview of each chapter

In the first half of Chapter 2, we introduce some basic notations and terminology. We then briefly introduce the basics of quantum computation, followed by a list of useful quantum algorithmic techniques that will be used throughout this thesis, including quantum unstructured search, amplitude estimation, generalized quantum minimum-finding, and amplitude amplification. After that, we introduce a type of quantum-accessible classical data structure for efficient state preparation, followed by block-encoding and Hamiltonian simulation. For the second half of Chapter 2, we start with some useful notions and theorems of probability theory, concentration bounds, and Fourier analysis. Then, we finish the chapter with lattice problems and some useful high-dimensional geometry facts and tools.

In Chapter 3, we give quantum algorithms for solving linear regression (with respect to squared loss) with ℓ_1-norm constraint, or *Lasso*. Our quantum algorithms provide a quadratic quantum speedup in terms of the dimension by speeding up the cost per iteration of the *Frank-Wolfe* algorithm. We use a variant of a QRAM data structure to store the nonzero entries of our candidate solution in each iteration in such a way that we can (1) quickly prepare this candidate as a quantum state, and (2) quickly incorporate the change of this candidate incurred by a Frank-Wolfe iteration. The Frank-Wolfe algorithm ensures that the candidate is always sparse, and hence we will not need too many QRAM bits. In fact, we can still have a quadratic quantum speedup in terms of dimension for Lasso by paying an extra sparsity-factor cost, without using QRAM.

In Chapter 4, we give two different quantum algorithms that, given query access to the entries of a $d \times d$ Hermitian matrix A and assuming a constant eigenvalue gap, output a classical description of a good approximation of the top eigenvector: one algorithm with time complexity $\tilde{\mathcal{O}}(d^{1.75})$ and one with time complexity $d^{1.5+o(1)}$. We extend this to a quantum algorithm that outputs a classical description of the subspace spanned by the top-q eigenvectors in time $qd^{1.5+o(1)}$. Our quantum algorithms run a version of the classical power method that is *robust to certain benign kinds of errors*, where we implement each *matrix-vector multiplication* with small and well-behaved error on a quantum computer, in different ways for the two algorithms. We also develop an almost optimal time-efficient process tomography algorithm for reflections around bounded-rank subspaces, providing the basis for our top-eigensubspace estimation algorithm, and in turn providing a pure-state tomography algorithm that only requires a reflection about the state rather than a state preparation unitary as input.

In Chapter 5, we give two quantum algorithms for solving the *shortest vector problem* (SVP). SVP is one of the most fundamental lattice problems, and faster algorithms for SVP threaten the leading candidates for post-quantum cryptography. Given a basis of a lattice with rank n, we can solve SVP in $2^{0.9497n+o(n)}$ time, which improves over the best known classical algorithm by Aggarwal, Dadush, Regev, and Stephens-Davidowitz [ADR+15]. If we are allowed to use QRAM memory, then the time complexity can be further improved to $2^{0.8345n+o(n)}$ time. Our algorithms make exponentially many calls to a bounded distance decoding oracle (BDD), and we use quantum

minimum-finding to reduce the number of queries we make to that oracle. We also show how to improve the time complexity of extra calls to BDD oracles by storing many discrete Gaussian samples in QRAM memory. The running time of our quantum algorithms is obtained using a known upper bound on a quantity related to a geometric object called the lattice kissing number, which is $2^{0.402n}$. For most lattices, this quantity is likely to be subexponential, and in such a situation, our quantum algorithm without using QRAM runs in time $2^{0.75n+o(n)}$ and the one using QRAM runs in time $2^{0.667n+o(n)}$.

In Chapter 6, we show quantum query lower bounds for linear regression with ℓ_1- and ℓ_2-norm constraints (which are called Lasso and Ridge, respectively). We prove such lower bounds by introducing a problem called the *hidden set-finding problem*: in this problem, we receive an $N \times d$ ± 1-matrix X that hides a subset of the columns by letting those columns have slightly more $+1$s than -1 (say, the probability of $+1$ is $0.5 + p$ for some $p > 0$), and the rest of the columns have the same amount of $+1$ and -1. We use the composition property of the quantum adversary lower bound to show one needs $\sim \sqrt{dw/p}$ quantum queries to entries of X to recover this hidden set with size w. Together with a worst-case to average-case reduction, we show a Lasso/Ridge solver can solve this hidden set problem, and hence it obtains a quantum query lower bound for Lasso and Ridge. Interestingly, because of some ℓ_2-norm properties, we can choose a bigger subset for Ridge than the one for Lasso, implying a better quantum lower bound for Ridge. Unfortunately, our quantum lower bound shows that there is *no quantum speedup for Ridge* in terms of dimension. We further extend this result to prove the first classical lower bound for Lasso that is tight up to polylog factors.

In Chapter 7, we give an $\Omega(d^2)$ classical lower bound and an $\tilde{\Omega}(d^{1.5})$ quantum lower bound for approximating the top eigenvector. This classical lower bound shows that both our quantum algorithms in Chapter 4 for approximating the top eigenvector provide a polynomial speedup over the best possible classical algorithm, and our quantum lower bound shows that our $d^{1.5+o(1)}$-time quantum algorithm is *nearly optimal*. We show both our classical and quantum lower bounds by analyzing a hard instance $A = \frac{1}{d}uu^T + N$, which hides a vector $u \in \{-1, 1\}^d$ using a $d \times d$ matrix N with i.i.d. Gaussian entries of mean 0 and standard deviation $\sim 1/\sqrt{d}$. By using random matrix theory, we know that with overwhelming probability, the top eigenvector of A is very close to $u/\sqrt{d}$ and hence, a (quantum) lower bound for approximating u is also a (quantum) lower bound for approximating the top eigenvector. To learn one entry of u, it requires $\sim d$ classical queries (or $\sim \sqrt{d}$ quantum queries). Hence it requires $\sim d^2$ classical queries (or $\sim d^{1.5}$ quantum queries) for approximating the whole u.

In Chapter 8, we will discuss a framework called *quantum strong exponential-time hypothesis*, or *QSETH*. Inspired by Buhrman, Patro, and Speelman's QSETH framework [BPS21], we give quantum (conditional) time lower bounds for a few natural variants of CNFSAT, such as parity-CNFSAT, counting-CNFSAT, and approximate-counting-CNFSAT. We further use those quantum time lower bounds for variants of CNFSAT to study the quantum fine-grained complexity for (the corresponding variants of) lattice problems, strong simulation and hitting set problem.

CHAPTER 2

Preliminaries

2.1 Notations

Throughout the thesis, log without a base means the binary logarithm, $\ln = \log_e$ is the natural logarithm, and $\exp(f) = e^f$. We let $[d]$ denote the set $\{1,\dots,d\}$ and $[d]-1$ denote the set $\{0,\dots,d-1\}$. For each integer $q \geq 2$, we write $\mathbb{Z}_q$ for the cyclic group $\{0,\dots,q-1\}$ with addition modulo q. $\mathcal{U}_N = \mathcal{U}\{0,\dots,N-1\}$ is the discrete uniform distribution over integers $0,1,2,\dots,N-1$.

It will be convenient for us to index entries of vectors starting from 0, so the entries x_i of a d-dimensional vector $\boldsymbol{x}$ are indexed by $i \in \{0,\dots,d-1\} = [d]-1$. For $1 \leq p < \infty$ the ℓ_p-norm $\|\boldsymbol{x}\|_p$ of any vector $\boldsymbol{x} \in \mathbb{C}^d$ is defined by

$$\|\boldsymbol{x}\|_p := \left(\sum_{i=0}^{d-1} |x_i|^p\right)^{1/p}.$$

Additionally, the ℓ_∞-norm of such a vector $\boldsymbol{x}$ is defined as $\|\boldsymbol{x}\|_\infty := \max_{i\in[d]-1} |x_i|$. The unit ℓ_p-ball $B_p^d \subset \mathbb{R}^d$ is defined by $\{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x}\|_p \leq 1\}$. For $r > 0$, we also use $B_p^d(r) := r \cdot B_p^d$ to denote the ℓ_p unit ball with radius r.

For $A \in \mathbb{C}^{d\times d'}$, we define the spectral norm $\|A\| = \max_{v\in\mathbb{C}^{d'}} \frac{\|Av\|}{\|v\|}$. For a set S, we define the indicator function $\mathbb{1}_S$ as

$$\mathbb{1}_S(x) = \begin{cases} 1 \text{ if } x \in S, \\ 0 \text{ otherwise.} \end{cases}$$

The *total variation distance* between probability distributions P and Q is defined as $d_{TV}(P,Q) = \sup_A P(A) - Q(A)$, where the supremum is over events A. In particular, for discrete distributions we have $d_{TV}(P,Q) = \frac{1}{2}\sum_x |P(x) - Q(x)|$. We say that two random variables are δ*-close* to each other if the total variation distance between their distributions is at most δ. One can also see that if $d_{TV}(P,Q) \leq \varepsilon$, then for every function f,

$d_{TV}(f(P), f(Q)) \leq \varepsilon$. For two distributions P, Q over the same space, the *relative entropy* $D_{KL}(P \| Q)$ (also called *Kullback-Leibler divergence* or *KL-divergence*) from P to Q is defined as

$$D_{KL}(P \| Q) = \int p(x) \cdot \ln \frac{p(x)}{q(x)} \mathrm{d}x = \mathbb{E}_p \left[\ln \frac{p(x)}{q(x)} \right],$$

where $p(x), q(x)$ are the probability density functions (pdf) of P and Q, respectively. In case P is not absolutely continuous with respect to Q we define $D_{KL}(P \| Q) = \infty$.

For us a projector is always a matrix Π which is idempotent ($\Pi^2 = \Pi$) and Hermitian ($\Pi^\dagger = \Pi$). This is sometimes called an "orthogonal projector" in the literature, but we drop the adjective "orthogonal" in order to avoid confusion with orthogonality between a pair of projectors. For a subspace S we denote the unique (orthogonal) projector to S by Π_S.

We use big-Oh notation. When we say $T(n) = \mathcal{O}(f(n))$, it means there exist constants c, $n' \geq 0$ s.t. for all integers $n \geq n'$, we have $T(n) \leq c \cdot f(n)$. We similarly use big-Omega notation for lower bounds. When we say $T(n) = \Omega(f(n))$, it means there exist constants c, $n' \geq 0$ s.t. $T(n) \geq c \cdot f(n)$ for all $n \geq n'$. We denote $T(n) = \Theta(f(n))$ if both $T(n) = \mathcal{O}(f(n))$ and $T(n) = \Omega(f(n))$ hold. We also use $\tilde{\mathcal{O}}$, $\tilde{\Omega}$, and $\tilde{\Theta}$ to "ignore" polylogarithmic relevant factors. We further use the little-oh notation. When we say $T(n) = o(f(n))$, it means that $\lim_{n \to \infty} T(n)/f(n) = 0$.

2.2 Computational model

Our computational model is a classical computer (a classical random-access machine) that can invoke a quantum computer as a subroutine. In Chapters 3 and 4, the input is stored in quantum-readable read-only memory (a QROM), whose bits can be queried. The classical computer can also write bits to a quantum-readable classical-writable classical memory (a QRAM). In Section 2.5 and Chapter 4, we also consider the special case where the input matrix A is s-sparse, meaning that each of its rows and columns has at most s-nonzero entries, we additionally assume we can also query the location of the ℓth nonzero entry in the jth column. This is called "sparse-query-access to A", and is a common assumption for quantum algorithms working on sparse matrices (for instance in Hamiltonian simulation). This corresponds to storing the matrix A using an "adjacency list", i.e., the locations and values of the nonzero entries for each row and column in, a QROM. In Chapter 5, the input is a classical description of vectors. The classical computer can send a description of a quantum circuit to the quantum computer; the quantum computer runs the circuit (which may include queries to the input bits stored in QROM and to the bits stored by the computer itself in the QRAM), measures the full final state in the computational basis, and returns the measurement outcome to the classical computer. In this model, an algorithm has time complexity T if it uses at most T elementary classical operations and quantum gates, quantum queries to the input bits stored in QROM, and quantum queries to the QRAM. The query complexity of an algorithm only measures the number of queries to the input

stored in QROM. We call a (quantum) algorithm *bounded-error* if (for every possible input) it returns a correct output with probability at least 2/3 (standard methods allow us to change this 2/3 to any constant in $(1/2,1)$ by only changing the complexity by a constant factor).

We will represent real numbers (and complexity numbers by pairs of real numbers) in computer memory using a number of bits of precision that is polylogarithmic in the relevant parameters (i.e., $\tilde{\mathcal{O}}(1)$ bits). This ensures all numbers are represented throughout our algorithms with negligible approximation error and we will ignore those errors later on for ease of presentation. For all Boolean functions $f : \{0,1\}^N \to \{0,1\}$ stored in a quantum oracle $O_f : |r,a\rangle \to |r, a \oplus f(r)\rangle$ for all $r \in \{0,1\}^N$, we define the quantum query complexity $Q_\delta(f)$ as the minimum number of queries (to O_f) required to determine $f(x)$ with failure error probability no more than δ.

2.3 Useful quantum algorithmic techniques

Below we state some important quantum algorithms that we will use as subroutines, starting with (an exact version of) Grover search and amplitude estimation.

Theorem 2.1 ([Gro96; BHT98]). *Let $f : [d]-1 \to \{0,1\}$ be a function that marks a set of elements $F = \{j \in [d]-1 : f(j) = 1\}$ of known size $|F|$. Suppose that we have a quantum oracle O_f such that $O_f : |j\rangle|b\rangle \to |j\rangle|b \oplus f(j)\rangle$. Then there exists a quantum algorithm that finds an index $j \in F$ with probability 1, using $\left\lceil \frac{\pi}{4}\sqrt{\frac{d}{|F|}} \right\rceil$ queries to O_f.*

Note that we can use the above "exact Grover" repeatedly to find all elements of F with probability 1, removing in each search the elements of F already found in earlier searches. This even works if we only know an upper bound on $|F|$.

Corollary 2.2 ([BCW+99]). *Let $f : [d]-1 \to \{0,1\}$ be a function that marks a set of elements $F = \{j \in [d]-1 : f(j) = 1\}$. Suppose we know an upper bound u on the size of F and we have a quantum oracle O_f such that $O_f : |j\rangle|b\rangle \to |j\rangle|b \oplus f(j)\rangle$. Then there exists a quantum algorithm that finds F with probability 1, using $\frac{\pi}{2}\sqrt{du} + u$ queries to O_f.*

Proof. Use the following algorithm:

1. Set $S = \emptyset$

2. For $k = u$ downto 1 do:
   use Theorem 2.1 on a modification g of f, where $g(j) = 0$ for all $j \in S$, assuming $|F| = k$;
   check that the returned value j satisfies $f(j) = 1$ by one more query; if so, add j to S.

Since we don't know $|F|$ exactly at the start, we are not guaranteed that each run of Grover finds another solution. However, k will always be an upper bound on the number of not-yet-found elements of F: either we found a new solution j and we can reduce k by 1 for that reason, or we did not find a new solution and then we know (by the correctness of the algorithm of Theorem 2.1) that the actual number of not-yet-found solutions was $< k$ and we are justified in reducing k by 1. Hence at the end of the algorithm all elements of F were found ($S = F$) with probability 1. The total number of queries is $\sum_{k=1}^{u} (\frac{\pi}{4}\sqrt{d/k}+1) \le \frac{\pi}{4}\sqrt{d}\int_0^u \frac{1}{\sqrt{x}}dx + u = \frac{\pi}{2}\sqrt{du}+u.$ □

Theorem 2.3 (follows from Section 4 of [BHM+02], amplitude estimation)**.** *Let $\delta \in (0,1)$. Given a natural number M and access to an $(n+1)$-qubit unitary U satisfying*

$$U|0^n\rangle|0\rangle = \sqrt{a}|\phi_0\rangle|0\rangle + \sqrt{1-a}|\phi_1\rangle|1\rangle,$$

where $|\phi_0\rangle$ and $|\phi_1\rangle$ are arbitrary n-qubit states and $a \in [0,1]$, there exists a quantum algorithm that uses $\mathcal{O}(M\log(1/\delta))$ applications of U and $U^\dagger$ and $\tilde{\mathcal{O}}(M\log(1/\delta))$ elementary gates, and outputs an estimator λ such that, with probability $\ge 1-\delta$,

$$|\sqrt{a}-\lambda| \le \frac{1}{M}.$$

Theorem 2.5 below is a modified version of quantum minimum-finding, which in its basic form is due to Høyer and Dürr [DH96]. Our proof of Theorem 2.5 relies on the following result.

Theorem 2.4 ([AGG+20], Theorem 49)**.** *Let $m \in \mathbb{R}$ and $\delta_1 \in (0,1)$. Suppose we have a unitary U that maps $|0\rangle \to \sum_{\ell\in[M]-1} \sqrt{p_\ell}|\psi_\ell\rangle|x_\ell\rangle$, where the $|\psi_\ell\rangle$ are normalized states and the x_ℓ are real numbers satisfying $x_0 < x_1 < \cdots < x_{M-1}$, and define X a random variable with $\Pr[X = x_\ell] = p_\ell$. Let K be a natural number $\ge \frac{1000}{\sqrt{\Pr[X\le m]}}\cdot\log(1/\delta_1)$. Then there exists a quantum algorithm that outputs a state $|\psi_i\rangle|x_i\rangle$ where $x_i \le m$ with probability $\ge 1-\delta_1$, using K applications of U and $U^\dagger$, and $\tilde{\mathcal{O}}(K)$ elementary gates.*

Theorem 2.5 (min-finding with an approximate unitary)**.** *Let $\delta_1, \delta_2, \varepsilon \in (0,1)$, $v_0, \ldots, v_{d-1} \in \mathbb{R}$. Suppose we have a unitary $\tilde{A}$ that maps $|j\rangle|0\rangle \to |j\rangle|\Lambda_j\rangle$ such that for every $j \in [d]-1$, after measuring the state $|\Lambda_j\rangle$, with probability $\ge 1-\delta_2$ the first register λ of the measurement outcome satisfies $|\lambda - v_j| \le \varepsilon$. There exists a quantum algorithm that finds an index j such that $v_j \le \min_{k\in[d]-1} v_k + 2\varepsilon$ with probability $\ge 1-\delta_1-1000\log(1/\delta_1)\cdot\sqrt{2d\delta_2}$, using $1000\sqrt{d}\cdot\log(1/\delta_1)$ applications of $\tilde{A}$ and $\tilde{A}^\dagger$, and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates. In particular, if $\delta_2 \le \delta_1^2/(2000000d\log(1/\delta_1))$, then the above algorithm finds such a j with probability $\ge 1-2\delta_1$.*

Proof. Without loss of generality, we assume $\log d$ is a natural number. Let $v^* = \min_{k\in[d]-1} v_k$. For every $j \in [d]-1$, we let $|\Lambda_j\rangle = \sqrt{p_j^\varepsilon}|\Lambda_j^\varepsilon\rangle + \sqrt{1-p_j^\varepsilon}|\Lambda_j^{\varepsilon^\perp}\rangle$, where $|\Lambda_j^\varepsilon\rangle$ is the superposition over numbers that are ε-approximations of v_j, $|\Lambda_j^{\varepsilon^\perp}\rangle$ is the superposition over

numbers that are not ε-approximations of v_j, and $p_j^\varepsilon \geq 1-\delta_2$ for every $j \in [d]-1$. Suppose we have a unitary A that maps $|j\rangle|0\rangle \to |j\rangle|\Lambda_j^\varepsilon\rangle$ and let $U = A(H^{\otimes \log d} \otimes I)$. Then one can see that if we apply the algorithm of Theorem 2.4 with the unitary U, then after using $K = 1000\sqrt{d}\cdot\log(1/\delta_1) \geq 1000/\sqrt{\Pr[X \leq v^* + \varepsilon]}\cdot\log(1/\delta_1)$ applications of A and $A^\dagger$, and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates, with probability$\geq 1-\delta_1$, the first outcome λ of the second register satisfies $\lambda \leq v^* + \varepsilon$. Note that if $\lambda \leq v^* + \varepsilon$, then the corresponding state $|\Phi\rangle$ satisfies that after measuring in the computational basis, the outcome j satisfies $v_j \leq v^* + 2\varepsilon$. Therefore one can find a j such that $v_j \leq v^* + 2\varepsilon$.

By the deferred measurement principle, one can consider the algorithm above as applying the unitary $\mathcal{A} = U_0 E_0 U_1 E_2 \cdots U_{K-1} E_{K-1}$ to the state $|0\rangle|0\rangle$ and measuring in the computational basis to get an outcome, where $U_i \in \{U, U^\dagger\}$ and E_i is a circuit of elementary gates. Let us consider $\tilde{U} = \tilde{A}(H^{\otimes \log d} \otimes I)$, and let

$$|\tilde{\psi}\rangle = \tilde{U}|0\rangle|0\rangle = \frac{1}{\sqrt{d}}\sum_{j\in[d]-1}|j\rangle|\Lambda_j\rangle \text{ and } |\psi\rangle = \frac{1}{\sqrt{d}}\sum_{j\in[d]-1}|j\rangle|\Lambda_j^\varepsilon\rangle = \alpha|\tilde{\psi}\rangle + \beta|\tilde{\psi}^\perp\rangle.$$

Where $\alpha \geq \sqrt{1-\delta_2}$ because $p_j^\varepsilon \geq 1-\delta_2$ for every j, and $\beta = \sqrt{1-\alpha^2}$.

Claim 2.1. There exists a unitary U such that $U|0\rangle|0\rangle = |\psi\rangle$ and $\|U - \tilde{U}\| \leq \sqrt{2\delta_2}$.

Proof: Define a unitary V such that

- $V|\tilde{\psi}\rangle = |\psi\rangle$.
- $V|\tilde{\psi}^\perp\rangle = -\beta|\tilde{\psi}\rangle + \alpha|\tilde{\psi}^\perp\rangle$.
- For every $|\phi\rangle$ orthogonal to span$\{|\psi\rangle, |\tilde{\psi}\rangle\}$, $V|\phi\rangle = |\phi\rangle$.

Let $U = V\tilde{U}$. One can see that $U|0\rangle|0\rangle = V\tilde{U}|0\rangle|0\rangle = V|\tilde{\psi}\rangle = |\psi\rangle$. Also, if we consider orthonormal basis $\{|\tilde{\psi}\rangle, |\tilde{\psi^\perp}\rangle, |\phi_2\rangle, |\phi_3\rangle, \ldots, |\phi_{d-1}\rangle\}$, then V will be $\begin{pmatrix}\alpha & -\beta \\ \beta & \alpha\end{pmatrix} \oplus I_{d-2}$, and hence

$$\begin{aligned}\|U-\tilde{U}\| = \|I - V\| &= \left\|\begin{pmatrix}1-\alpha & \beta \\ -\beta & 1-\alpha\end{pmatrix}\right\| \\ &= \max_{\substack{a,b\in\mathbb{C},\\ \text{s.t. } |a|^2+|b|^2=1}} \sqrt{|a(1-\alpha)+b\beta|^2 + |b(1-\alpha)-a\beta|^2} \\ &= \sqrt{|1-\alpha|^2+|\beta|^2} \leq \sqrt{2\delta_2}.\end{aligned}$$

This proves the claim. ■

Now let us consider the unitary $\tilde{\mathcal{A}} = \tilde{U}_0 E_0 \tilde{U}_1 E_2 \cdots \tilde{U}_{K-1} E_{K-1}$ applying to the state $|0\rangle|0\rangle$ and measuring in the computational basis to get an outcome, where $\tilde{U}_i \in \{\tilde{U}, \tilde{U}^\dagger\}$. Because $\|\tilde{\mathcal{A}} - \mathcal{A}\| \leq K\cdot\|U-\tilde{U}\| \leq 1000\log(1/\delta_1)\sqrt{2d\delta_2}$, with probability$\geq 1-\delta_1-1000\log(1/\delta_1)\sqrt{2d\delta_2}$, the first outcome λ of the second register of $\tilde{\mathcal{A}}|0\rangle|0\rangle$ also satisfies $\lambda \leq v^* + \varepsilon$, and hence we also find a j such that $v_j \leq v^* + 2\varepsilon$ by the same arguments as above. □

Theorem 2.6 ([GSL+19; YLC14], fixed-point amplitude amplification)**.** *Let $a,\delta > 0$, U be a unitary that maps $|0\rangle \to |\psi\rangle$ and $R_{\mathcal{A}}, R_{|0\rangle}$ be quantum circuits that reflect through subspaces $\mathcal{A}$ and (the span of) $|0\rangle$ respectively. Suppose $\|\Pi_{\mathcal{A}}|\psi\rangle\| \geq a$. There is a quantum algorithm that prepares $|\psi'\rangle$ satisfying $\||\psi'\rangle - \frac{\Pi_{\mathcal{A}}|\psi\rangle}{\|\Pi_{\mathcal{A}}|\psi\rangle\|}\| \leq \delta$ using a total number of $\mathcal{O}(\log(1/\delta)/a)$ applications of U, U^{-1}, controlled $R_{\mathcal{A}}, R_{|0\rangle}$ and additional single-qubit gates.*

We also use the following theorem to help us find all marked items in a d-element search space with high probability. The theorem was implicit in [Gro96; BBH+98]. For more details and for a better d,δ-dependency, see [AGN24, Section 3]. Note that Corollary 2.2 is mainly about query complexity, while the theorem below is about time complexity.

Theorem 2.7. *Let $f : [d]-1 \to \{0,1\}$ be a function that marks a set of elements $F = \{j \in [d]-1 : f(j) = 1\}$, and $\delta \in (0,1)$. Suppose we know an upper bound u on the size of F and we have a quantum oracle O_f such that $O_f : |j\rangle|b\rangle \to |j\rangle|b \oplus f(j)\rangle$. Then there exists a quantum algorithm that finds F with probability at least $1-\delta$, using $\mathcal{O}(\sqrt{du}\cdot \mathit{poly}\log(d/\delta))$ time.*

2.4 KP-tree: a data structure for efficient state preparation

Kerenidis and Prakash [Pra14; KP17] gave a quantum-accessible classical data structure to store a vector θ with support t (i.e., t nonzero entries) to enable efficient quantum state preparation. In this section, we modify their data structure such that for arbitrary $a, b \in \mathbb{R}$ and $j \in [d]-1$, we can efficiently update a data structure for the vector θ to a data structure for the vector $a\theta + be_j$, without having to individually update all nonzero entries of the vector. We call this data structure a "KP-tree" (or KP_θ if we're storing vector θ) in order to credit Kerenidis and Prakash. We will include two types of KP-trees here with respect to different kinds of norms.

2.4.1 KP-tree with respect to ℓ_1-norm

First we explain how to modify the data structure by Kerenidis and Prakash to efficiently prepare the following state:

$$|\theta\rangle = \sum_{j\in[d]-1} \sqrt{\frac{|\theta_j|}{\|\theta\|_1}}\,|j\rangle\,|\mathrm{phase}(\theta_j)\rangle\,.$$

Note that here the amplitude in front of $|j\rangle$ is $\sqrt{\frac{|\theta_j|}{\|\theta\|_1}}$ in stead of $\frac{|\theta_j|}{\|\theta\|_2}$. This modification allows us to prepare a quantum state θ with respect to its ℓ_1-norm. We also store the phase of θ_j in the second register. If $\theta \in \mathbb{R}^d$, then $\mathrm{phase}(\theta_j) = \mathrm{sign}(\theta_j)$.

Definition 2.8 (KP-tree w.r.t. ℓ_1-norm)**.** Let $\theta \in \mathbb{C}^d$ have support t. We define a KP-tree KP_θ of θ w.r.t. ℓ_1-norm as follows:

- KP_θ is a rooted binary tree with depth $\lceil \log d \rceil$ and with $\mathcal{O}(t \log d)$ vertices.
- The root stores a scalar $A \in \mathbb{R} \setminus \{0\}$ and the support t of θ.
- Each edge of the tree is labelled by a bit.
- For each $j \in \mathrm{supp}(\theta)$, there is one corresponding leaf storing $\frac{\theta_j}{A}$. The number of leaves is t.
- The bits on the edges of the path from the root to the leaf corresponding to the j^{th} entry of θ, form the binary description of j.
- Each intermediate node stores the sum of its children's absolute values.

For $\ell \in [\lceil \log d \rceil] - 1$ and $j \in [2^\ell] - 1$, we define $KP_\theta(\ell, j)$ as the value of the j^{th} node in the ℓ^{th} layer, i.e., the value stored in the node that we can reach by the path according to the binary representation of j from the root. Also, we let $KP_\theta(0,0)$ be the sum of all absolute values stored in the leaves. If there is no corresponding j^{th} node in the ℓ^{th} layer (that is, we cannot reach a node by the path according to the binary representation of j from the root), then $KP_\theta(\ell, j)$ is defined as 0. Note that both the numbering of the layer and the numbering of nodes start from 0. In the special case where θ is the all-0 vector, the corresponding tree will just have a root node with $t = 0$.

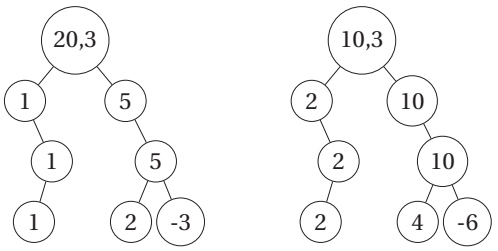

Figure 2.1: Each of the above two binary trees represents the vector $\theta = 20e_2 + 40e_6 - 60e_7$. If we see the second layer of KP_θ on the right-hand side, $KP_\theta(2,0) = 0$, $KP_\theta(2,1) = 2$, $KP_\theta(2,2) = 0$, and $KP_\theta(2,3) = 10$.

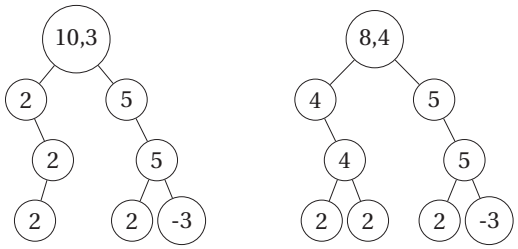

Figure 2.2: The update rule: we update the vector $\theta = 20e_2 + 20e_6 - 30e_7$ to the new vector $\frac{4}{5}\theta + 16e_4$ by updating the scalar in the root to $\frac{4}{5} \cdot 10 = 8$, adding a new leaf with value $16/8 = 2$, recomputing the values of the intermediate nodes between the root and the leaf, and updating the support number to 4.

Theorem 2.9. *For each $j \in [d] - 1$, one can read the number θ_j by reading at most poly$\log d$ nodes of KP_θ and by using poly$\log d$ many (classical) elementary operations.*

Proof. Read the scalar A stored in the root. Choose the path according to the binary representation of j. If the chosen path reaches a leaf, then read the value v at that leaf and output vA. If it does not reach a leaf, output 0. The total cost is at most $\mathrm{polylog}\, d$ because the depth of KP_θ is $\lceil \log d \rceil$.

From the fourth bullet of Definition 2.8, if $j \in \mathrm{supp}(\theta)$, then the corresponding leaf j stores θ_j / A and hence the output is $(\theta_j / A) \cdot A = \theta_j$. On the other hand, if $j \notin \mathrm{supp}(\theta)$, then we do not reach a leaf and know $\theta_j = 0$. □

Theorem 2.10. *Given a KP-tree KP_θ, $j \in [d]-1$, and numbers $a \in \mathbb{R} \setminus \{0\}$ and $b \in \mathbb{R}$, we can update KP_θ to $KP_{a\theta+be_j}$ by using $\mathrm{polylog}\, d$ elementary operations and by modifying $\mathrm{polylog}\, d$ many values stored in the nodes of KP_θ.*

Proof. Read the scalar A and support t stored in the root.

If there does not exist a leaf for the entry j, then add a new leaf for the entry j and a path according to its binary representation. Now update the stored value in the leaf j to $b/(aA)$. After that, update the stored values for all nodes on the path from the root to the leaf for the entry j. Update the scalar in the root to aA, and if $b \neq 0$, update the support value in the root to $t+1$.

If, instead, there already existed a leaf for the entry j, then read the value v stored in the leaf j, update the value stored in the leaf j to $v' = v + b/(aA)$, and then update the stored values for all nodes on the path from the root to the leaf j for the entry j, and update the scalar to aA. After that, check the value v' stored in the leaf for the entry j; if $v' = 0$, then remove all nodes storing the value 0 from the leaf j to the root, and update the support value at the root to $t-1$. □

Theorem 2.11. *Suppose we have a KP-tree KP_θ of vector θ, and suppose we can make quantum queries to a unitary O_{KP_θ} that maps $|\ell,k\rangle|0\rangle \to |\ell,k\rangle|KP_\theta(\ell,k)\rangle$. Then one can prepare the state $|\theta\rangle = \sum_{j\in[d]-1} \sqrt{\frac{|\theta_j|}{\|\theta\|_1}}\,|j\rangle\,|\mathit{phase}(\theta_j)\rangle$ up to negligible error[1] by using $\mathrm{polylog}\, d$ queries to O_{KP_θ} and $O_{KP_\theta}^\dagger$, and $\tilde{\mathcal{O}}(1)$ elementary gates.*

Proof. For simplicity and without loss of generality, we assume $\log d$ is a natural number. Define the two-controlled rotation unitary as for each $a, b \in \mathbb{C}$

$$U_{2CR} : |a\rangle|b\rangle|0\rangle \to \begin{cases} |a\rangle|b\rangle\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right), & \text{if } a = b = 0, \\ |a\rangle|b\rangle\left(\sqrt{\frac{|a|}{|a|+|b|}}|0\rangle + \sqrt{\frac{|b|}{|a|+|b|}}|1\rangle\right), & \text{otherwise,} \end{cases}$$

which can be implemented up to negligibly small error by $\tilde{\mathcal{O}}(1)$ elementary gates. Also, define the children-reading gate as $U_C : |\ell\rangle|k\rangle|0\rangle^{\otimes 2} \to |\ell\rangle|k\rangle|l_{\ell,k}\rangle|r_{\ell,k}\rangle$, where the left child $l_{\ell,k} = KP_\theta(\ell+1, 2k)$ and the right child $r_{\ell,k} = KP_\theta(\ell+1, 2k+1)$; this can be implemented by using two queries to O_{KP_θ} and $\tilde{\mathcal{O}}(1)$ elementary gates. Last, define the phase

[1] By this we mean an error smaller than the inverse of an arbitrary polynomial in the input length.

gate $U_p : |j\rangle|0\rangle \to |j\rangle|\text{phase}(\theta_j)\rangle$, which can be implemented by using two queries to O_{KP_θ}, $O_{KP_\theta}^\dagger$, and $\tilde{\mathcal{O}}(1)$ elementary gates.

To prepare $|\theta\rangle$, we first prepare the state $|KP_\theta^0\rangle = |0\rangle$, and for the purpose of induction, suppose we can prepare the state

$$|KP_\theta^\ell\rangle = \frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{2^\ell-1} \sqrt{|KP_\theta(\ell,k)|}\,|k\rangle,$$

where $KP_\theta(0,0)$ is the sum of all absolute values stored in the leaves and hence $KP_\theta(0,0) = \sum_{k=0}^{2^\ell-1} |KP_\theta(\ell,k)|$. We prepare the state $|\ell\rangle|KP_\theta^\ell\rangle|0\rangle^{\otimes 2}|0\rangle$, apply U_C on the first four registers, and apply U_{2CR} on the last three registers to get

$$\begin{aligned}
&|\ell\rangle \frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{2^\ell-1} \sqrt{|KP_\theta(\ell,k)|}\,|k\rangle\,|l_{\ell,k}\rangle\,|r_{\ell,k}\rangle \Big(\frac{\sqrt{|l_{\ell,k}|}}{\sqrt{|l_{\ell,k}|+|r_{\ell,k}|}}|0\rangle + \frac{\sqrt{|r_{\ell,k}|}}{\sqrt{|r_{\ell,k}|+|r_{\ell,k}|}}|1\rangle\Big) \\
=&|\ell\rangle \frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{2^\ell-1} |k\rangle\,|l_{\ell,k}\rangle\,|r_{\ell,k}\rangle \Big(\sqrt{|l_{\ell,k}|}|0\rangle + \sqrt{|r_{\ell,k}|}|1\rangle\Big),
\end{aligned}$$

where the equation holds because $|KP_\theta(\ell,k)| = |KP_\theta(\ell+1,2k)| + |KP_\theta(\ell+1,2k+1)| = |l_{\ell,k}| + |r_{\ell,k}|$, from the sixth bullet of Definition 2.8. Uncomputing the third and fourth registers, and discarding the first, third, and fourth registers, we get

$$\begin{aligned}
&\frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{2^\ell-1} |k\rangle \Big(\sqrt{|l_{\ell,k}|}|0\rangle + \sqrt{|r_{\ell,k}|}|1\rangle\Big) \\
=&\frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{2^\ell-1} \Big(\sqrt{|l_{\ell,k}|}|k\rangle|0\rangle + \sqrt{|r_{\ell,k}|}|k\rangle|1\rangle\Big) \\
=&\frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{2^{\ell+1}-1} \sqrt{|KP_\theta(\ell+1,k)|}\,|k\rangle = |KP_\theta^{\ell+1}\rangle.
\end{aligned}$$

Therefore, iterating the above process for $\log d$ times, we can prepare the state

$$|KP_\theta^{\log d}\rangle = \frac{1}{\sqrt{KP_\theta(0,0)}} \sum_{k=0}^{d-1} \sqrt{|KP_\theta(\log d,k)|}\,|k\rangle = \sum_{j\in[d]-1} \sqrt{\frac{|\theta_j|}{\|\theta\|_1}}\,|j\rangle,$$

where the last equation follows from the fourth and sixth bullets of Definition 2.8. To obtain $|\theta\rangle$, we prepare $\sum_{j\in[d]-1} \sqrt{\frac{|\theta_j|}{\|\theta\|_1}}\,|j\rangle|0\rangle$ and apply U_p.

There are $\log d$ layers, and each layer only uses $\mathcal{O}(1)$ queries to O_{KP_θ}, $O_{KP_\theta}^\dagger$ and $\tilde{\mathcal{O}}(1)$ elementary gates. Hence $\mathcal{O}(\log d)$ queries to O_{KP_θ}, $O_{KP_\theta}^\dagger$, and $\tilde{\mathcal{O}}(1)$ other gates suffice to prepare $|\theta\rangle$. □

To implement O_{KP_θ} and $O^\dagger_{KP_\theta}$ in the above theorem, we use QRAM to store KP_θ, then we can make quantum queries to the bits of the data structure directly. Or, if we want to avoid QRAM altogether, then we can use the following theorem with $\tilde{\mathcal{O}}(s)$ extra cost in circuit size for each query, where s is the sparsity of the bitstring that represents KP_θ. From Definition 2.8 we can see that the number of bits is $s = \tilde{\mathcal{O}}(t \log d)$, where t is the sparsity of θ.

Theorem 2.12. *Suppose $p, s \in \mathbb{N}$ and $D \in \{0,1\}^p$ is a bit string with sparsity s (i.e., the number of 1s in D is $\leq s$), then for each $b \in \{0,1\}$ and $k \in [p]-1$, we can implement the unitary $U_D : |k,b\rangle \to |k, b \oplus D_k\rangle$ using $\mathcal{O}(s \log p)$ elementary gates.*

Proof. For every $i \in [p]-1$, we define the controlled bit-reading unitary U_i as for each $b \in \{0,1\}$ and $k \in \{0,1\}^p$

$$U_i : |k\rangle|b\rangle \to \begin{cases} |k\rangle|b \oplus 1\rangle, & \text{if } k = i, \\ |k\rangle|b\rangle, & \text{otherwise,} \end{cases}$$

which can be implemented using $\mathcal{O}(\log p)$ elementary gates. Observing that $U_D = \Pi_{i:D_i=1} U_i$, we can therefore implement U_D using $\mathcal{O}(s \cdot \log p)$ elementary gates. □

2.4.2 KP-tree with respect to ℓ_2-norm

Here we will introduce another variant of KP-tree which can help us efficiently prepare $|\theta\rangle$ with respect to ℓ_2-norm. Precisely, we would like to have a data structure to efficiently prepare the following state:

$$|\theta\rangle = \sum_{j \in [d]-1} \text{phase}(\theta_j) \cdot \frac{|\theta_j|}{\|\theta\|_2} |j\rangle = \sum_{j \in [d]-1} \frac{\theta_j}{\|\theta\|_2} |j\rangle.$$

This can be done by using a data structure similar to the KP-tree with respect to ℓ_1-norm (Definition 2.8), as follows.

Definition 2.13 (KP-tree w.r.t. ℓ_2-norm)**.** Let $v \in \mathbb{C}^d$. We define a KP-tree KP_v of v with respect to ℓ_2-norm as follows:

- The root stores the scalar $\|v\|$ and the size of the support $t = |\text{supp}(v)|$ of v.
- KP_v is a binary tree on $\mathcal{O}(t \log d)$ vertices with depth $\lceil \log d \rceil$.
- The number of leaves is t; for each $j \in \text{supp}(v)$ there is one corresponding leaf storing v_j.
- Each edge of the tree is labeled by a bit; the bits on the edges of the path from the root to the leaf corresponding to the j^{th} entry of v form the binary description of j.
- Intermediate nodes store the square root of the sum of their children's squared absolute values.

Suppose we have a classical vector $\theta \in \mathbb{C}^d$, we can use $\widetilde{\mathcal{O}}(d)$ time and QRAM bits to build a KP-tree (w.r.t. ℓ_2-norm) for θ. Given a KP-tree KP_θ, we are allowed to query entries of θ and prepare the quantum state $\sum_{j\in[d]-1} \frac{\theta_j}{\|\theta\|_2} |j\rangle$ efficiently by using a proof similar to Theorem 2.11.

Theorem 2.14. *Suppose we have a KP-tree KP_θ of vector θ, and suppose we can apply a unitary O_{KP_θ} that maps $|\ell,k\rangle|0\rangle \to |\ell,k\rangle|KP_\theta(\ell,k)\rangle$. Then one can implement a unitary U_θ that maps $|0\rangle$ to $|\frac{\theta}{\|\theta\|_2}\rangle = \sum_{j\in[d]-1} \frac{\theta_j}{\|\theta\|_2}|j\rangle$ up to negligible error by using $\mathrm{poly}\log d$ applications of O_{KP_θ} and $O^\dagger_{KP_\theta}$, and $\tilde{\mathcal{O}}(1)$ elementary gates.*

Note that if $\|\theta\|_2 < 1$, then we can also prepare a quantum state $|\theta\rangle = |0\rangle \sum_{j\in[d]-1} \theta_j |j\rangle + |1\rangle|\Phi\rangle$ for some $|\Phi\rangle$ using just $\mathrm{poly}\log d$ time and queries to KP_θ.

2.5 Block-encoding and Hamiltonian simulation

Block-encoding embeds a scaled version of a (possibly non-unitary) matrix A in the upper-left corner of a bigger unitary matrix U.

Definition 2.15. Suppose that A is a 2^w-dimensional matrix, $\alpha, \varepsilon > 0$, and $a \in \mathbb{N}$. We call an $(a+w)$-qubit unitary U an (α, a, ε)-block-encoding of A if

$$\|A - \alpha\langle 0^a| \otimes I_{2^w}) U (|0^a\rangle \otimes I_{2^w})\| \le \varepsilon.$$

Theorem 2.16 ([LC19])**.** *Suppose that U is an $(\alpha, a, \varepsilon/|2t|)$-block-encoding of the Hamiltonian H. Then we can implement an ε-precise Hamiltonian-simulation unitary V which is a $(1, a+2, \varepsilon)$-block-encoding of e^{itH}, with $\mathcal{O}(|\alpha t| + \log(1/\varepsilon))$ uses of controlled-U and its inverse, and with $\mathcal{O}(a|\alpha t| + a\log(1/\varepsilon))$ additional elementary gates.*

Theorem 2.17 ([Low19, Theorem 2])**.** *Let A be a $d \times d$ Hermitian matrix with operator norm ≤ 1, and $t > 0$. Suppose A has sparsity s and we have sparse-query-access to A. Then we can implement a unitary U such that $\|U - \exp(iAt)\| \le \varepsilon$ using $\widetilde{\mathcal{O}}\left(t\sqrt{s}(t\sqrt{s}/\varepsilon)^{o(1)}\right)$ time and queries.*

If we do not have a sparse oracle or if A is dense, then the time complexity of the above theorem simply becomes $\tilde{\mathcal{O}}((\sqrt{d}\,t)^{1+o(1)}/\varepsilon^{o(1)})$ by setting $s = d$.

Theorem 2.18 ([GSL+19, Corollary 71])**.** *Let $\varepsilon \in (0, 1/2)$, A be a $d \times d$ Hermitian matrix with operator norm $\le 1/2$, and $U = \exp(iA)$. Then we can implement a $(2/\pi, 2, \varepsilon)$-block-encoding of A, using $\mathcal{O}(\log(1/\varepsilon))$ applications of controlled-U, controlled-U inverse, $\mathcal{O}(\log(1/\varepsilon))$ time, and one auxiliary qubit.*

Combining the above two theorems, we have the following theorem.

Theorem 2.19. *Let A be a $d \times d$ Hermitian matrix with operator norm ≤ 1. Suppose A has sparsity s and we have sparse-query-access to A. Then we can implement a unitary U which is a $(4/\pi, 2, \varepsilon)$-block-encoding of A with $\widetilde{\mathcal{O}}\left(\sqrt{s}(s/\varepsilon)^{o(1)}\right)$ time and queries.*

Theorem 2.20 ([ACG+23], Lemma 6)**.** *Let $U = \sum_x U_x \otimes |x\rangle\langle x|$ and $V = \sum_x V_x \otimes |x\rangle\langle x|$ be controlled (by the second register) state-preparation unitaries, where $U_x : |0\rangle|0^{\otimes a}\rangle \to |0\rangle|\psi_x\rangle + |1\rangle|\tilde{\psi}_x\rangle$ and $V_x : |0\rangle|0^{\otimes a}\rangle \to |0\rangle|\phi_x\rangle + |1\rangle|\tilde{\phi}_x\rangle$ are $(a+1)$-qubit state-preparation unitaries for some (sub-normalized) a-qubit quantum states $|\psi_x\rangle, |\phi_x\rangle$. Then $(I \otimes V^\dagger)(SWAP \otimes I_{2^{a+1}})(I \otimes U)$ is a $(1, a+2, 0)$-block-encoding of the diagonal matrix $diag(\{\langle\psi_x|\phi_x\rangle\})$, where the SWAP gate acts on the first and second qubits.*

2.6 Singular-value and singular-vector-perturbation bounds

We invoke some tight bounds on the perturbation of singular values and singular vectors of a matrix. In the following we order the singular values $\varsigma_1(A), \varsigma_2(A), \ldots, \varsigma_n(A)$ of a matrix A in decreasing order, such that $i < j \Rightarrow \varsigma_i(A) \geq \varsigma_j(A)$.

Theorem 2.21 (Weyl's singular value perturbation bound [Bha97, Corollary III.2.6, Problem III.6.13])**.** *Let $A, B \in \mathbb{C}^{n \times m}$ be any matrices, then for all $i \in [n]$ we have*

$$|\varsigma_i(A) - \varsigma_i(B)| \leq \|A - B\|.$$

In order to state the following perturbation bound we define Π_S^X to be the projector onto the subspace spanned by the left-singular vectors of X having singular values in S.

Theorem 2.22 (Wedin-Davis-Kahan $\sin(\theta)$ theorem [Wed72])**.** *Let $A, B \in \mathbb{C}^{n \times m}$ be any matrices, and $\alpha, \delta \geq 0$, then*[2]

$$\|(I - \Pi_{>\alpha}^A)\Pi_{\geq \alpha+\delta}^B\| \leq \frac{\|A - B\|}{\delta}.$$

Lemma 2.23 (Operator norm equivalence to $\sin(\theta)$ between subspaces [Bha97, after Exercise VII.1.11])**.** *Let $P, Q \in \mathbb{C}^{n \times n}$ be projectors with equal rank, then $\|P - Q\| = \|P(I - Q)\| = \|(I - P)Q\|$.*

[2] This bound is tight for any rank-r projectors A, B, when $\alpha = 0$ and $\delta = 1$ due to Lemma 2.23. Wedin's paper proves the statement for singular-vector subspaces that we use here, and the analogous statement for normal matrices is proven in Bhatia's book [Bha97, Theorem VII.3.1], which actually also implies Theorem 2.22 with a bit of work.

2.7 Probability distributions and concentration inequalities

2.7.1 Gaussian, Sub-Gaussian, and discrete Gaussian distributions

A random variable X over $\mathbb{R}$ has Gaussian distribution with mean $\mu = \mathbb{E}[X]$ and variance $\sigma^2 = \text{Var}(X)$, denoted $X \sim N(\mu, \sigma^2)$, if its probability density function is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big), \text{ for } x \in \mathbb{R}.$$

A well-known property of a Gaussian is that its tail decays rapidly, which can be quantified as:

- If $X \sim N(\mu, \sigma^2)$, then for any $t > 0$, it holds that

$$\Pr[X - \mu > t] \le \frac{1}{\sqrt{2\pi}\, t} \exp(-t^2/(2\sigma^2)).$$

A random variable X over $\mathbb{R}$ is called τ-sub-Gaussian with parameter $\alpha > 0$, denoted in short by $X \sim \tau\text{-subG}(\alpha^2)$, if its moment-generating function satisfies $\mathbb{E}[\exp(tX)] \le \exp(\tau)\exp(\alpha^2 t^2/2)$ for all $t \in \mathbb{R}$. We list and prove a few useful properties of sub-Gaussian distributions below.

- The tails of $X \sim \tau\text{-subG}(\alpha^2)$ are dominated by a Gaussian with parameter α, i.e.,

$$\Pr[|X| > t] \le 2\exp(\tau)\exp(-t^2/(2\alpha^2)) \quad \forall\, t > 0. \tag{2.1}$$

- If $X_i \sim \tau\text{-subG}(\alpha_i^2)$ are independent, then for any $a = (a_1, \ldots, a_d)^T \in \mathbb{R}^d$, the weighted sum $\sum_{i\in[d]} a_i X_i$ is $(\sum_{i\in[d]} \tau_i)$-sub-Gaussian with parameter $\tilde{\alpha} = \sqrt{\sum_{i=1}^{d} a_i^2 \alpha_i^2}$.

To prove (2.1), we first use Markov's inequality to obtain that for all $s > 0$

$$\Pr[X \ge t] = \Pr[\exp(sX) \ge \exp(st)] \le \mathbb{E}[\exp(sX)]/\exp(st) \le \exp(\tau)\exp(\alpha^2 s^2/2 - st).$$

Since the above inequality holds for every $s > 0$, we have $\Pr[X \ge t] \le \exp(-t^2/(2\alpha^2))$ because $\min_{s>0}(\alpha^2 s^2/2 - st) = -t^2/(2\alpha^2)$. The same argument applied to $-X$ gives the bound on $\Pr[X \le -t]$.

The second property can be easily derived using the independence of the X_i's:

$$\mathbb{E}[\exp(t\sum_{i=1}^{d} a_i X_i)] = \prod_{i=1}^{d} \mathbb{E}[\exp(t a_i X_i)] \le \prod_{i=1}^{d} \exp(\tau_i)\exp(\alpha_i^2 a_i^2 t^2/2) = \exp(\sum_{i=1}^{d} \tau_i)\exp(\sum_{i=1}^{d} \alpha_i^2 a_i^2 t^2/2).$$

Next, we explain what a *discrete* Gaussian is. For any $s > 0$, we define the function $\rho_s \colon \mathbb{R} \to \mathbb{R}$ as $\rho_s(x) = \exp(-\pi x^2/s^2)$. When $s = 1$, we simply write $\rho(x)$. For a countable

set S we define $\rho_s(S) = \sum_{a\in S} \rho_s(a)$; if $\rho_s(S) < \infty$, we define $\mathcal{D}_{S,s} = \frac{1}{\rho_s(S)}\rho_s$ to be the discrete probability distribution over S such that the probability of drawing $x \in S$ is proportional to $\rho_s(x)$. We call this the discrete Gaussian distribution over S with parameter s. The following theorem states that such distributions over $\mathbb{Z}$ are sub-Gaussian with parameter s:

Theorem 2.24 ([MP12, Lemma 2.8])**.** *For any $\lambda, \nu > 0$ and $\tau \in (0, 0.1]$, if $s \geq \lambda\sqrt{\log(12/\tau)/\pi}$,*[3] *then $\mathcal{D}_{\lambda\mathbb{Z}+\nu,s}$ is τ-sub-Gaussian with parameter s. Moreover, for every $s > 0$, $\mathcal{D}_{\lambda\mathbb{Z},s}$ is 0-sub-Gaussian with parameter s.*

We also consider truncations of the above infinite discrete Gaussian distributions, and define $\mathcal{D}_{\mathbb{Z},s}^{[-L,R]}$ as $\mathcal{D}_{\mathbb{Z}\cap[-L,R],s}$. For every $\delta \in (0,1]$, if $L, R \geq s\sqrt{2\ln(1/\delta)}$, then by Corollary 2.35 we have $d_{TV}(\mathcal{D}_{\mathbb{Z},s}, \mathcal{D}_{\mathbb{Z},s}^{[-L,R]}) \leq 2\delta$, because the tail of $\mathcal{D}_{\mathbb{Z},s}$ can be bounded by 2δ using (2.1) due to Theorem 2.24. We also define $\mathcal{D}_{\mathbb{Z},s}^{\bmod N}$ as a modular version of the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},s}$, which has probability $\mathcal{D}_{\mathbb{Z},s}^{\bmod N}(k) = \mathcal{D}_{\mathbb{Z},s}(N\cdot\mathbb{Z} + k)$ for every $k \in \{-\lfloor N/2\rfloor, \ldots, \lceil N/2\rceil - 1\}$, and probability 0 for all other $k \in \mathbb{Z}$. Again, by using Theorem 2.24 and (2.1), we have that for every $\delta \in (0, 1/2)$, if $N \geq 2s\sqrt{2\ln(1/\delta)}$, then $d_{TV}(\mathcal{D}_{\mathbb{Z},s}, \mathcal{D}_{\mathbb{Z},s}^{\bmod N}) \leq 2\delta$. Combining these we get $d_{TV}(\mathcal{D}_{\mathbb{Z},s}^{\bmod N}, \mathcal{D}_{\mathbb{Z},s}^{[-L,R]}) \leq 4\delta$ if $N, L, R \geq 2s\sqrt{2\ln(1/\delta)}$. Finally, these arguments can also be applied to $\mathcal{D}_{\mathbb{Z}+c,s}$ with large enough s.

Corollary 2.25. *Let $\delta \in (0,1]$. For any $c > 0$ and $\tau \in (0, 0.1]$, if $s \geq \sqrt{\log(12/\tau)/\pi}$ and $N, L, R \geq 10s\sqrt{2\ln(2/\delta)}$, then $\mathcal{D}_{\mathbb{Z}+c,s}$, $\mathcal{D}_{\mathbb{Z}+c,s}^{[-L,R]}$, and $\mathcal{D}_{\mathbb{Z}+c,s}^{\bmod N}$ are $4\delta\exp(\tau)$-close to each other in total variation distance.*

2.7.2 Concentration inequalities

Repeated sampling is very important for our quantum tomography algorithms, and here we describe some of the tail bounds we need.

Proposition 2.26 (Bennett-Bernstein Bound [BLM13, Theorem 2.9 & Eqn. 2.10])**.** *Let $X^{(i)} : i \in [n]$ be independent random variables with finite variance such that, for each i, $X^{(i)} \leq b$ for some $b > 0$ almost surely (i.e., this event has probability measure 1). Let*

$$S = \sum_{i=1}^{n} X^{(i)} - \mathbb{E}[X^{(i)}], \qquad v = \sum_{i=1}^{n} \mathbb{E}[(X^{(i)})^2],$$

then for any $t > 0$,

$$\Pr[S \geq t] \leq \exp\left(-\frac{v}{b^2} h\left(\frac{bt}{v}\right)\right) \leq \exp\left(-\frac{t^2}{2v + \frac{2}{3}bt}\right),$$

[3] Here the lattice we consider is the one-dimensional lattice $\lambda\mathbb{Z}$, so the length of the shortest nonzero vector $\lambda_1(\lambda\mathbb{Z})$ is just λ. Also, by using the equation between the smoothing parameter and the length of the shortest vector ([MR07, Lemma 3.3], or see Lemma 2.49 in this thesis), we have $\eta_\tau(\lambda\mathbb{Z}) \leq \sqrt{\log(2+2/\tau)/\pi}\cdot\lambda_1(\lambda\mathbb{Z}) \leq \lambda\sqrt{\log(3/\tau)/\pi}$. Therefore, if $s \geq \lambda\sqrt{\log(3/\tau)/\pi}$, then $s \geq \eta_\tau(\lambda\mathbb{Z})$ and hence $\mathcal{D}_{\lambda\mathbb{Z}+\nu,s}$ is $\log((1+\tau)/(1-\tau))$-sub-Gaussian. By the fact $\log((1+\tau)/(1-\tau)) \leq 4\tau$ and changing $\tau \to \tau/4$, we get the theorem as stated here.

where $h(x) = (1+x)\ln(1+x) - x$.

Proposition 2.27 (Chernoff-Hoeffding Bound [Che52], [Hoe63, Theorem 1], [BLM13, Section 2.6])**.** *Let* $0 \le X \le 1$ *be a bounded random variable and* $p := \mathbb{E}[X]$. *Suppose we take n i.i.d. samples* $X^{(i)}$ *of X and denote the normalized outcome by* $s = \frac{X^{(1)}+X^{(2)}+\cdots+X^{(n)}}{n}$. *Then we have for all* $\varepsilon > 0$

$$\Pr[s \ge p+\varepsilon] \le e^{-D_{KL}(p+\varepsilon\|p)n} \le \exp\left(-\frac{\varepsilon^2}{2(p+\varepsilon)}n\right), \tag{2.2}$$

$$\Pr[s \le p-\varepsilon] \le e^{-D_{KL}(p-\varepsilon\|p)n} \le \exp\left(-\frac{\varepsilon^2}{2p}n\right), \tag{2.3}$$

where $D_{KL}(x \parallel p) = x\ln\frac{x}{p} + (1-x)\ln\left(\frac{1-x}{1-p}\right)$ *is the Kullback–Leibler divergence between Bernoulli random variables with mean x and p respectively.*

Proof. The first inequality is [Hoe63, Theorem 1], while (2.3) follows from (2.2) by considering $1 - X^{(i)}$. The rightmost inequalities come from the observation that $\forall x, y \ge 0$: $D(x \parallel y) \ge \frac{(x-y)^2}{2\max\{x,y\}}$. □

Corollary 2.28 (Okamoto-Hoeffding Bound)**.** *Let X, s be as in Proposition 2.27, then we have*

$$\Pr[\sqrt{s} \ge \sqrt{p}+\varepsilon)] \le \exp\left(-2\varepsilon^2 n\right),$$
$$\Pr[\sqrt{s} \le \sqrt{p}-\varepsilon] \le \exp\left(-\varepsilon^2 n\right).$$

Proof. This directly follows from (2.2)-(2.3) using the observation [Oka59] that

$$D_{KL}(x \parallel p) \ge 2\left(\sqrt{x}-\sqrt{p}\right)^2 \qquad \forall\, 0 \le p \le x \le 1,$$
$$D_{KL}(x \parallel p) \ge \left(\sqrt{p}-\sqrt{x}\right)^2 \qquad \forall\, 0 \le x \le p \le 1.$$ □

2.7.3 Matrix concentration inequalities

We state some random matrix concentration results for tall matrices (i.e., matrices having more rows than columns), but in our case we mostly apply them to flat matrices (having more columns than rows), thus we effectively apply the statements to $G^\dagger$. We will use the following non-asymptotic bounds.

Theorem 2.29 (Well-conditioned tall Gaussian matrices [Ver12, Theorem 5.39 & Footnote 25])**.** *There exist absolute constants*[4] $c, C \ge 1$ *such that the following holds for all*

[4] In the real case [DS01, Theorem II.13] gives $c, C = 1$. While [Ver12, Footnote 25] asserts that the statement can be adapted to the complex case, there are no specifics provided, and the proof of [Ver12, Corollary 5.35] is borrowed from [DS01, Theorem II.13], where the adaptation of the statement to the complex case is presented as an open question. It is tempting to try and adapt the proof of the real case using the observation that $\varsigma_{\min}(G) = \min_{\|u\|=1}\max_{\|v\|=1}\Re\langle u^*, Gv\rangle$ together with the Slepian-Gordon lemma [Gor85, Theorem 1.4], however this approach seems to irrecoverably fail due to a banal issue: while $\||u\rangle|v\rangle - |u'\rangle|v'\rangle\|^2 \le \|u-u'\|^2 + \|v-v'\|^2$ holds for real unit vectors, for complex unit vectors only the weaker $\||u\rangle|v\rangle - |u'\rangle|v'\rangle\|^2 \le 2\|u-u'\|^2 + 2\|v-v'\|^2$ holds in general. Indeed, for any $\alpha \in (0, 2\pi)$ consider the complex numbers $u = 1, v = i\exp(i\alpha), u' = \exp(-i\alpha), v' = i$, then we have $|uv - u'v'|^2/(|u-u'|^2 + |v-v'|^2) = 1 + \cos(\alpha)$.

$N \geq n$: if $G \in \mathbb{C}^{N\times n}$ is a random matrix whose matrix elements have i.i.d. real or complex standard normal distribution,[5] *then its smallest singular value $\varsigma_{\min}(G)$ and largest singular value $\varsigma_{\max}(G)$ satisfy, for all $t \geq 0$*

$$\Pr[\sqrt{N} - C\sqrt{n} - t < \varsigma_{\min}(G) \leq \varsigma_{\max}(G) < \sqrt{N} + C\sqrt{n} + t] > 1 - 2\exp(-t^2/(2c)).$$

Corollary 2.30. *In the setting of Theorem 2.29, if $N \geq 16C^2 n$, we have*

$$\Pr\left[\frac{1}{4}\sqrt{N} < \varsigma_{\min}(G) \leq \varsigma_{\max}(G) < \frac{7}{4}\sqrt{N}\right] > 1 - 2\exp(-N/(8c)).$$

Proof. Apply Theorem 2.29 with $t = \sqrt{N}/2$. □

The following slightly tighter bound can be used for bounding the norm of individual columns or rows of Gaussian random matrices.

Proposition 2.31. Let v be an n-dimensional random vector whose coordinates have i.i.d. real or complex standard normal distribution, then $\mathbb{E}[\|v\|] \leq \sqrt{n}$ and $\Pr[\|v\| \geq \sqrt{n} + t] \leq \exp(-\frac{t^2}{2})\ \forall t \geq 0$.

Proof. We have $\mathbb{E}[\|v\|] \leq \sqrt{\mathbb{E}[\|v\|^2]} = \sqrt{n}$ by Jensen's inequality. The function $v \mapsto \|v\|$ is 1-Lipschitz due to the triangle inequality, and hence by the concentration of Lipschitz functions on vectors with the canonical Gaussian measure (Proposition 2.18 & Equation (2.35) of [Led01]) we have $\Pr[\|v\| \geq \sqrt{n} + t] \leq \exp(-t^2/2)$.[6] □

Next, we invoke a concentration bound for the operator norm of a random matrix with independent bounded rows.

Theorem 2.32 (Independent bounded rows [Ver12, Theorem 5.44 & Remark 5.49])**.** *There exists an absolute constant $c' > 0$ such that the following holds. Let $G \in \mathbb{C}^{N\times n}$ be a random matrix whose rows G_i are independent, each having mean 0 and a covariance matrix*[7] *with operator norm at most S^2, and almost surely $\|G_i\|_2 \leq B$ for all $i \in [N]$. Then for every $t \geq 0$, with probability at least $1 - 2n\exp(-c't^2)$ one has*

$$\|G\| \leq 2|S|\sqrt{N} + tB. \tag{2.4}$$

In case G is a real-symmetric Gaussian matrix, we have the following non-asymptotic bound.

[5] A complex standard normal random variable has independent real and imaginary parts each having centered normal distribution with variance $\frac{1}{2}$.

[6] Actually, in the complex case the upper bound is even stronger: $\exp(-t^2)$.

[7] If $\psi \in \mathbb{C}^n$ is a mean-0 random vector and $C = \mathbb{E}[\psi^\dagger \psi]$ is its covariance matrix, then the covariance matrix of the complex conjugate random variable ψ^* is $\mathbb{E}[\psi^T \psi^*] = C^*$. On the other hand we have $\mathrm{Cov}(\Re(\psi)) + \mathrm{Cov}(\Im(\psi)) = \frac{C+C^*}{2}$, and therefore $\|\mathrm{Cov}(\Re(\psi)) + \mathrm{Cov}(\Im(\psi))\| \leq \|C\|$. Thus we can apply [Ver12, Theorem 5.44 & Remark 5.49] separately to the real and imaginary parts of the random vectors G_i, whence the extra (possibly sub-optimal) factor of 2 in (2.4).

Theorem 2.33 (Symmetric Gaussian matrix [BH16, Corollary 3.9 with $\varepsilon = 0.25$])**.** *Let $G \in \mathbb{R}^{d\times d}$ be a symmetric matrix with $G_{ij} = b_{ij} \cdot g_{ij}$, where the random variables $\{g_{ij} : i \geq j\}$ are i.i.d. $\sim N(0,1)$ and the $\{b_{ij} : i \geq j\}$ are arbitrary real scalars. Denote $b_{max} = \max_i \sqrt{\sum_j b_{ij}^2}$ and $b^*_{max} = \max_{ij} |b_{ij}|$. Then for every $t \geq 0$,*

$$\Pr\left[\|G\| \geq 2.5 \cdot b_{max} + \frac{7.5}{\ln(1.25)} b^*_{max}\sqrt{\ln d} + t\right] \leq \exp(-t^2/(4{b^*}^2_{max})).$$

2.7.4 Bounds on random variables with adaptive dependency structure

Here we present some useful bounds on random variables, where the dependency structure follows some martingale-like structure.

The first bound gives an intuitive total variation distance bound for "adaptive" processes. The main idea is to couple two adaptive random processes that are step-wise similar. We use the following folklore [AS19, p. 1] observation:

Theorem 2.34 (Total variation distance and optimal coupling)**.** *Let X, Y be random variables. Then $d_{TV}(X,Y) \leq \varepsilon$ iff there exist ε-coupled random variables $\widetilde{X}$ and $\widetilde{Y}$ (possibly dependent) with the same distribution as X and Y, respectively, such that $\Pr[\widetilde{X} \neq \widetilde{Y}] \leq \varepsilon$.*

Corollary 2.35. *Let X be a random variable and A an event of the underlying probability space such that $\Pr[A] > 0$. Then for $Y' = X|A$, the conditioned version of X (i.e., $\Pr[Y' \in S] = \Pr[A \,\&\, X \in S]/\Pr[A]$ for all measurable sets S), we have that $d_{TV}(X,Y') \leq 1 - \Pr[A]$.*

Proof. Let Y be a random variable that is independent of A, but its distribution is identical to that of Y', i.e., $\Pr[Y' \in S] = \Pr[Y \in S] = \Pr[A \,\&\, X \in S]/\Pr[A]$ for all measurable sets S. In Theorem 2.34 take $\widetilde{X} := X$, and $\widetilde{Y} := X$ on A and $\widetilde{Y} := Y$ on the complement of A; by construction we have $\Pr[\widetilde{X} \neq \widetilde{Y}] \leq 1 - \Pr[A]$. Finally, observe that for all measurable sets S we have

$$\begin{aligned}\Pr[\widetilde{Y} \in S] &= \Pr[A \,\&\, X \in S] + \Pr[\bar{A} \,\&\, Y \in S] = \Pr[A \,\&\, X \in S] + \Pr[\bar{A}] \cdot \Pr[Y \in S] \\ &= \left(1 + \frac{\Pr[\bar{A}]}{\Pr[A]}\right)\Pr[A \,\&\, X \in S] = \frac{\Pr[A \,\&\, X \in S]}{\Pr[A]} = \Pr[X \in S \mid A] = \Pr[Y' \in S]. \qquad \square\end{aligned}$$

Lemma 2.36 (Conditional total variation distance based bound)**.** *Let $X = (X_1, X_2)$ and $X' = (X'_1, X'_2)$ be two discrete random variables, and let $X_{2x} := X_2|X_1 = x$, $X'_{2x} := X'_2|X'_1 = x$. Suppose that $d_{TV}(X_1, X'_1) \leq \varepsilon_1$ and $d_{TV}(X_{2x}, X'_{2x}) \leq \varepsilon_2$ for all x such that $\Pr[X_1 = x]\Pr[X'_1 = x] > 0$, then $d_{TV}(X,X') \leq \varepsilon_1 + (1-\varepsilon_1)\varepsilon_2$.*

Proof. Due to Theorem 2.34 we can find ε_1-coupled random variables $\widetilde{X}_1$, $\widetilde{X}'_1$, and similarly ε_2-coupled $\widetilde{X}_{2x}$, $\widetilde{X}'_{2x}$ for all x in the range of X_1, X'_1 respectively. We can assume without loss of generality that $\widetilde{X}_1$ and $\widetilde{X}_{2x}$ are mutually independent for all x in the range of X_1, X'_1 and likewise are $\widetilde{X}_1$ and $\widetilde{X}_{2x}$. We then define $\widetilde{X} = (\widetilde{X}_1, \widetilde{X}_{2\widetilde{X}_1})$ and $\widetilde{X}' = (\widetilde{X}'_1, \widetilde{X}'_{2\widetilde{X}'_1})$, so that clearly $\Pr[X = (x_1,x_2)] = \Pr[\widetilde{X} = (x_1,x_2)]$ and $\Pr[X' = (x_1,x_2)] =$

$\Pr[\widetilde{X}' = (x_1, x_2)]$. On the other hand due to the tight coupling of Theorem 2.34 we also get

$$\begin{aligned} d_{TV}(X, X') &\le \Pr[\widetilde{X} \neq \widetilde{X}'] \\ &= \Pr[\widetilde{X}_1 \neq \widetilde{X}'_1] + \sum_x \Pr[\widetilde{X}_1 = \widetilde{X}'_1 = x \,\&\, \widetilde{X}_{2x} \neq \widetilde{X}'_{2x}] \\ &= \Pr[\widetilde{X}_1 \neq \widetilde{X}'_1] + \sum_x \Pr[\widetilde{X}_1 = \widetilde{X}'_1 = x] \Pr[\widetilde{X}_{2x} \neq \widetilde{X}'_{2x}] \\ &\le \varepsilon_1 + \sum_x \Pr[\widetilde{X}_1 = \widetilde{X}'_1 = x]\varepsilon_2 \\ &= \varepsilon_1 + (1 - \varepsilon_1)\varepsilon_2. \end{aligned}$$

□

The following is essentially a martingale property, which could be stated more generally, but here we prove a simple version for completeness.

Lemma 2.37 (Martingale-like covariance sum)**.** *If $X, Y, Z \in \mathbb{C}^d$ are vector-valued discrete random variables such that $\mathbb{E}[Z|(X, Y)] = 0$ (i.e., $\mathbb{E}[Z|(X = x, Y = y)] = 0$ for all x, y), then $\mathrm{Cov}(X + Z) = \mathrm{Cov}(X) + \mathrm{Cov}(Z)$.*

Proof. It is easy to see that $\mathbb{E}[Z] = 0$, and we can assume without loss of generality that $\mathbb{E}[X] = 0$.

$$\begin{aligned} \mathrm{Cov}(X + Z) &= \mathbb{E}[|X + Z\rangle\langle X + Z|] \\ &= \sum_{\substack{(x,y):\\ \Pr[(X,Y)=(x,y)]>0}} \Pr[(X, Y) = (x, y)]\mathbb{E}[|x + Z\rangle\langle x + Z| \,|\, (X, Y) = (x, y)] \\ &= \sum_{\substack{(x,y):\\ \Pr[(X,Y)=(x,y)]>0}} \Pr[(X, Y) = (x, y)] \left(|x\rangle\langle x| + \mathbb{E}[|Z\rangle\langle Z| \,|\, (X, Y) = (x, y)]\right) \\ &= \sum_x \Pr[X = x]|x\rangle\langle x| + \sum_{\substack{(x,y):\\ \Pr[(X,Y)=(x,y)]>0}} \Pr[(X, Y) = (x, y)]\mathbb{E}[|Z\rangle\langle Z| \,|\, (X, Y) = (x, y)] \\ &= \mathrm{Cov}(X) + \mathrm{Cov}(Z). \end{aligned}$$

□

2.8 Fourier transform and analysis

2.8.1 Fourier transform

The Fourier transform $\hat{h} : \mathbb{R} \to \mathbb{C}$ of a function $h : \mathbb{R} \to \mathbb{C}$ is defined as

$$\hat{h}(\omega) = \int_{-\infty}^{\infty} h(x) \exp(-2\pi i x\omega)\, dx.$$

The next facts follow easily from the above definition. If h is defined as $h(\omega) = g(\omega + \nu)$ for some function g and value ν, then we have

$$\hat{g}(\omega) = \hat{h}(\omega) \exp(2\pi i \nu\omega).$$

On the other hand, if $h(x) = g(x)\exp(2\pi i x v)$, then

$$\hat{h}(\omega) = \hat{g}(\omega - v).$$

Another important fact is that the Fourier transform of ρ_s is $s \cdot \rho_{1/s}$ for all $s > 0$. Also, the sum of $\rho_s(x)$ over $C \cdot \mathbb{Z}$ satisfies the Poisson summation formula [Reg09, Lemma 2.14]:

Theorem 2.38. *For any scalar $C > 0$ and any Schwartz function $f : \mathbb{R} \to \mathbb{C}$ (i.e., f and each of its derivatives go to 0 faster than every inverse polynomial as the absolute value of the argument goes to infinity),*

$$\sum_{j \in C \cdot \mathbb{Z}} f(j) = C^{-1} \sum_{j \in C^{-1} \cdot \mathbb{Z}} \hat{f}(j).$$

2.9 Lattice, lattice problems, and some useful properties

For any set of n linearly independent vectors $\mathbf{B} = \{\boldsymbol{b_1}, \ldots, \boldsymbol{b_n}\}$ from $\mathbb{R}^d$, the lattice $\mathcal{L}$ generated by basis $\mathbf{B}$ is

$$\mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} z_i \boldsymbol{b_i} : z_i \in \mathbb{Z} \right\}.$$

We call n the *rank* of the lattice $\mathcal{L}$ and d the *dimension*. The vectors $\mathbf{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ form a *basis* of the lattice. Given a basis $\mathbf{B}$, we use $\mathcal{L}(\mathbf{B})$ to denote the lattice generated by $\mathbf{B}$. For a rank n lattice $\mathcal{L} \subset \mathbb{R}^d$, the *dual lattice*, denoted $\mathcal{L}^*$, is defined as the set of all points in $\mathsf{span}(\mathcal{L})$ that have integer inner products with all lattice points,

$$\mathcal{L}^* = \{\boldsymbol{w} \in \mathsf{span}(\mathcal{L}) : \forall \boldsymbol{y} \in \mathcal{L}, \langle \boldsymbol{w}, \boldsymbol{y} \rangle \in \mathbb{Z}\}.$$

A basis matrix $\mathbf{B}^* \in \mathbb{R}^{n \times d}$ for the dual lattice (i.e. the columns form a basis of $\mathcal{L}^*$) can obtained from a basis $\mathbf{B}$ of $\mathcal{L}$, by letting $\mathbf{B}^* = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$. Note that if $\mathcal{L}$ is n-dimensional, then the expression for $\mathbf{B}^*$ simplifies to $\mathbf{B}^{-T}$.

2.9.1 Lattice problems

Definition 2.39. For any $1 \le p \le \infty$, the Shortest Vector Problem SVP_p is defined as follows: The input is a basis $\mathbf{B} \in \mathbb{R}^{d \times n}$ for a lattice $\mathcal{L}$. The goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y}\|_p = \min_{x \in \mathcal{L} \setminus \{0\}} \|x\|_p$.

We also use $\lambda_1^{(p)}(\mathcal{L})$ to denote the length of the shortest nonzero lattice vector of $\mathcal{L}$ w.r.t. ℓ_p-norm. This definition can be generalized to define the ith successive minimum w.r.t. ℓ_p-norm as the smallest $\lambda_i^{(p)}$ such that $B_p^d(\lambda_i)$ contains i linearly independent lattice points:

$$\lambda_i^{(p)} = \min\{r : \dim(\mathrm{span}(\mathcal{L} \cap B_p^d(\lambda_i))) \ge i\}.$$

When we say the *shortest vector* of a lattice, we always mean the shortest nonzero lattice vector. One can use the well-known Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm to estimate $\lambda_1^{(p)}$ up to 2^n multiplicative factor.

Theorem 2.40 ([LLL82])**.** *Let $\mathbf{B} \in \mathbb{R}^{d\times n}$ be a basis of lattice $\mathcal{L}(\mathbf{B}) \subset \mathbb{R}^d$. For every $p \in [1,\infty)$, there exists a poly(n)-time algorithm that with probability at least $1-2^{-\Omega(n)}$, outputs a vector $\boldsymbol{v} \in \mathcal{L}$ such that $\lambda_1^{(p)}(\mathcal{L}) \le \|\boldsymbol{v}\|_p \le 2^n \lambda_1^{(p)}(\mathcal{L})$.*

Definition 2.41. For any $1 \le p \le \infty$, the Closest Vector Problem CVP_p is the search problem defined as follows: The input is a basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ for a lattice $\mathcal{L}$ and a target vector $\boldsymbol{t}$. The goal is to output a vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v}-\boldsymbol{t}\|_p = \min_{x\in\mathcal{L}} \|\boldsymbol{x}-\boldsymbol{t}\|_p$.

Definition 2.42. For any $1 \le p \le \infty$ and $\alpha = \alpha(n) < 1/2$, the Bounded Distance Decoding problem with decoding distance α α-BDD_p is defined as follows: The input is a basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ for a lattice $\mathcal{L}$ and a target vector $\boldsymbol{t} \in \mathbb{R}^d$ with $\min_{x\in\mathcal{L}} \|\boldsymbol{t}-x\|_p \le \alpha \cdot \lambda_1^{(p)}(\mathcal{L})$. The goal is to output a vector $\boldsymbol{y} \in \mathcal{L}$ with $\|\boldsymbol{y}-\boldsymbol{t}\|_p = \min_{x\in\mathcal{L}} \|\boldsymbol{t}-x\|_p$.

Note that α-BDD_p becomes more difficult as α gets larger. For CVP and BDD, we can also define a preprocessing version, as follows.

Definition 2.43. For any $1 \le p \le \infty$ and $\alpha = \alpha(n) < 1/2$, the search problems CVPP_p and α-BDDP are the preprocessing analogues of CVP and α-BDD_p respectively. The input for preprocessing is a basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ of lattice $\mathcal{L}$. Given the advice from the preprocessing algorithm and the target vector $\boldsymbol{t} \in \mathbb{R}^d$. The goal is to return solution of CVP and α-BDD respectively. The preprocessing algorithm is allowed to take arbitrary time.

The following theorem shows how to sample a randomly chosen "sparsified" sublattice of a lattice using a CVP_p oracle.

Theorem 2.44 ([Ste16], modified Theorem 3.2)**.** *Let $\mathbf{B} \in \mathbb{R}^{d\times n}$ be a basis of a lattice $\mathcal{L}(\mathbf{B})$ and Q be a prime number. Consider the following sparsification process: input any two vectors $\boldsymbol{z}, \boldsymbol{c} \in \mathbb{Z}_Q^n$, the sparsification process $Spar(\mathbf{B}, Q, \boldsymbol{z}, \boldsymbol{c})$ outputs a basis $\mathbf{B}_{Q,\boldsymbol{z}}$ of the sublattice $\mathcal{L}_{\boldsymbol{z}} \subset \mathcal{L} = \{\boldsymbol{u} \in \mathcal{L} : \langle \mathbf{B}\boldsymbol{z}, \boldsymbol{u}\rangle = 0 \mod Q\}$ and $\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}} = \mathbf{B}\boldsymbol{c}$. Then for every $\boldsymbol{t} \in \mathbb{R}^n$, $\boldsymbol{x} \in \mathcal{L}$ with $N = |(\mathcal{L}-\boldsymbol{t}) \cap \|\boldsymbol{x}-\boldsymbol{t}\| \cdot B_p^d| \le Q$, and CVP_p oracle, we have*

$$\frac{1}{Q} - \frac{N}{Q^2} - \frac{N}{Q^{n-1}} \le \Pr_{\boldsymbol{z},\boldsymbol{c}\in\mathbb{Z}_Q^n}[CVP_p(\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}, \mathcal{L}_{\boldsymbol{z}}) = \boldsymbol{x}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}] \le \frac{1}{Q} + \frac{1}{Q^n},$$

and in particular,

$$\frac{N}{Q} - \frac{N^2}{Q^2} - \frac{N^2}{Q^{n-1}} \le \Pr_{\boldsymbol{z},\boldsymbol{c}\in\mathbb{Z}_Q^n}[\min_{\boldsymbol{u}\in\mathcal{L}_z} \|(\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}-\boldsymbol{u}\|_p) \le \|\boldsymbol{x}-\boldsymbol{t}\|_p] \le \frac{N}{Q} + \frac{N}{Q^n}.$$

The following theorem shows how to solve SVP_p by an exponential number of calls to α-BDD_p oracle.

Theorem 2.45 ([CCL18, modified Theorem 8])**.** *Given a basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ of lattice $\mathcal{L}(\mathbf{B}) \subset \mathbb{R}^d$, a target vector $\boldsymbol{t} \in \mathbb{R}^d$, an α-BDD_p oracle BDD_α with $\alpha < 0.5$, and an integer scalar $q > 0$. Let $f_q^\alpha : \mathbb{Z}_q^n \to \mathbb{R}^n$ be $f_q^\alpha(\boldsymbol{s}) = -q \cdot BDD_\alpha(\mathcal{L}, (\mathbf{B}\boldsymbol{s}-\boldsymbol{t})/q) + \mathbf{B}\boldsymbol{s}$, then the list $m = \{f_q^\alpha(\boldsymbol{s}) \mid \boldsymbol{s} \in \mathbb{Z}_q^n\}$ contains all lattice points within distance $q\alpha\lambda_1^{(p)}(\mathcal{L})$ to $\boldsymbol{t}$.*

2.9.2 Lattice on Euclidean norm

Here we introduce some tools and properties for lattices under the Euclidean norm. We first introduce some basic facts of the lattice problems in ℓ_2-norm. For arbitrary basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ of $\mathcal{L}$, one can always find a unitary U such that $U\mathbf{B} = \begin{bmatrix} \mathbf{B}' \\ 0 \end{bmatrix}$ where $\mathbf{B}' \in \mathbb{R}^{n\times n}$. Finding such a U and applying U on $\mathbf{B}$ takes only poly(d) time. For SVP, if v' is a shortest vector of $\mathcal{L}(\mathbf{B}')$, then $U^{-1}\begin{bmatrix} v' \\ 0 \end{bmatrix}$ will be a shortest vector of $\mathcal{L}(\mathbf{B})$. As for CVP, if $x \in \mathcal{L}(\mathbf{B})$ is the closest lattice vector, then $x' \in \mathcal{L}(\mathbf{B}')$ will be the closest vector to Ut', where $\begin{bmatrix} x' \\ 0 \end{bmatrix} = Ux$ and $t' = \mathrm{Proj}_{\mathsf{span}(\mathbf{B})}(t)$. As a result, in the case of ℓ_2-norm, one can always, without loss of generality, assume $d = n$. We assume this is the case below when we discuss about the lattice problems w.r.t. ℓ_2-norm.

For simplicity, lattice problems without a subscript are always lattice problems with respect to ℓ_2-norm (for example, CVP means CVP_2) and $\lambda_1(\mathcal{L})$ simply denotes $\lambda_1^{(2)}(\mathcal{L})$.

Discrete Gaussian Distribution.

For any $s > 0$, define $\rho_s(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}\|^2/s^2)$ for all $\boldsymbol{x} \in \mathbb{R}^n$. We write ρ for ρ_1. Here we overload the notation ρ_s introduced in Section 2.7.1. Similarly, for a countable set S, we extend ρ to sets by $\rho_s(S) = \sum_{\boldsymbol{x}\in S}\rho_s(\boldsymbol{x})$. Given a lattice $\mathcal{L}$, the *discrete Gaussian* $\mathcal{D}_{\mathcal{L},s}$ is the distribution over $\mathcal{L}$ such that the probability of a vector $\boldsymbol{y} \in \mathcal{L}$ is proportional to $\rho_s(\boldsymbol{y})$:

$$\Pr_{X\sim\mathcal{D}_{\mathcal{L},s}}[X = \boldsymbol{y}] = \frac{\rho_s(\boldsymbol{y})}{\rho_s(\mathcal{L})}.$$

We also call s the *width* of $\mathcal{D}_{\mathcal{L},s}$. For $s = 1$, we simply denote $\mathcal{D}_{\mathcal{L},}$.

The following problems play a central role in Chapter 5. For convenience, when we discuss the running time of algorithms solving the problems below, we ignore polynomial factors in the bit-length of the individual input basis vectors (i.e. we assume that the input basis has bit-size polynomial in the ambient dimension n).

Definition 2.46. For $\delta = \delta(n) \geq 0$, σ a function that maps lattices to non-negative real numbers, and $m = m(n) \in \mathbb{N}$, δ-DGS_σ^m (the Discrete Gaussian Sampling problem) is defined as follows: The input is a basis $\mathbf{B}$ of a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a parameter $s > \sigma(\mathcal{L})$. The goal is to output a sequence of m vectors whose joint distribution is δ-close to m independent samples from $\mathcal{D}_{\mathcal{L},s}$.

We omit the parameter δ if $\delta = 0$, and the parameter m if $m = 1$. We stress that δ bounds the total variation distance between the *joint* distribution of the output vectors and m independent samples from $D_{\mathcal{L},s}$.

We will use the following lemma which is initially proved in [Ban93].

Lemma 2.47 ([DRS14, Lemma 2.13])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$,*

$$\Pr_{\boldsymbol{y}\sim\mathcal{D}_{\mathcal{L}}}\left[\|\boldsymbol{y}\| \geq t\sqrt{\frac{n}{2\pi}}\right] \leq e^{-\frac{n}{2}(t-1)^2}.$$

For a lattice $\mathcal{L}$ and $\varepsilon > 0$, the *smoothing parameter* $\eta_\varepsilon(\mathcal{L})$ is the smallest s such that $\rho_{1/s}(\mathcal{L}^*) = 1+\varepsilon$. Recall that if $\mathcal{L}$ is a lattice and $\boldsymbol{v} \in \mathcal{L}$ then $\rho_s(\mathcal{L}+\boldsymbol{v}) = \rho_s(\mathcal{L})$ for all s. Note that the smoothing parameter is a monotone-decreasing function of ε. That is, if $\varepsilon_1 > \varepsilon_2$, then $\eta_{\varepsilon_1} < \eta_{\varepsilon_2}$ The smoothing parameter has the following well-known properties.

Lemma 2.48 ([Reg09, Claim 3.8])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, $\varepsilon > 0$, and $s \geq \eta_\varepsilon(\mathcal{L})$,*

$$\frac{1-\varepsilon}{1+\varepsilon} \leq \frac{\rho_s(\mathcal{L}+\boldsymbol{c})}{\rho_s(\mathcal{L})} \leq 1\,.$$

One can consider the smoothing parameter to be the smallest "width" such that the discrete Gaussian still behaves like a continuous Gaussian. Micciancio and Regev related the smoothing parameter to $\lambda_n(\mathcal{L})$.

Lemma 2.49 ([MR07, Lemma 3.3])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon \in (0,1)$,*

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}}\lambda_n(\mathcal{L}).$$

Another useful fact is that when we add a continuous Gaussian to a discrete Gaussian, the resulting distribution of the vector will be close to a continuous Gaussian, if both widths of the discrete Gaussian and the continuous Gaussian are sufficiently larger than the smoothing parameter.

Lemma 2.50 ([Reg09, Claim 3.9])**.** *Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice, $\varepsilon \in (0, 0.5)$, widths $s, t > 0$, and $\nu_t := \rho_t / t^n$ be an n-dimensional continuous Gaussian probability density function with width t.*[8] *Suppose $1/(\sqrt{1/s^2 + 1/t^2}) \geq \eta_\varepsilon(\mathcal{L})$. Consider the continuous distribution Y on $\mathbb{R}^n$ obtained by sampling from $\mathcal{D}_{\mathcal{L},s}$ and then adding a noise vector taken from ν_t. Then the total variation distance between Y and $\nu_{\sqrt{s^2+t^2}}$ is at most 4ε.*

The following lemma gives a bound on the smoothing parameter.

Lemma 2.51 ([ADR+15, Lemma 2.7])**.** *For any lattice $\mathcal{L} \subset \mathbb{R}^n, \varepsilon \in (0,1)$ and $k > 1$, we have $k\eta_\varepsilon(\mathcal{L}) > \eta_{\varepsilon^{k^2}}(\mathcal{L})$*

Here we define the *honest* Discrete Gaussian Sampling problem.

Definition 2.52 ([ADR+15, Definition 5.1])**.** For $\varepsilon \geqslant 0$, σ a function that maps lattices to non-negative real numbers, and $m \in \mathbb{N}$, the *honest* Discrete Gaussian Sampling problem ε-hDGS$^m_\sigma$ is defined as follows: the input is a basis $\mathbf{B}$ for a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a parameter $s > 0$. The goal is to output a sequence of m' vectors whose joint distribution is ε-close to $\mathcal{D}^{m'}_{\mathcal{L},s}$ for some independent random variable $m' \geqslant 0$. If $s > \sigma(\mathcal{L})$ then m' must be equal to m.

[8] Note that $\int_{\boldsymbol{x}\in\mathbb{R}^n} \rho_t(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = t^n$.

The idea of "honest" is that when we ask a sampler to produce samples with a too-small parameter, the output should still consist of discrete Gaussian samples of the desired parameter, but potentially less of them (or even none at all). The authors in [ADR+15] also show how to construct a honest discrete Gaussian sampler.

Theorem 2.53 ([ADR+15, Theorem 5.11])**.** *Let σ be the function that maps a lattice $\mathcal{L}$ to $\sqrt{2}\eta_{1/2}(\mathcal{L})$. There is an algorithm that solves $\exp(-\Omega(\kappa))$-hDGS$_\sigma^{2^{n/2}}$ in time $2^{n/2+\mathrm{polylog}(\kappa)+o(n)}$ for an $\kappa \geqslant \Omega(n)$.*

We also use the lemma below to find a super lattice with a smaller smoothing parameter, where a super lattice $\mathcal{L}'$ of $\mathcal{L}$ is a superset of $\mathcal{L}$.

Lemma 2.54 ([ADR+15, Lemma 5.12])**.** *There is a polynomial-time algorithm that takes as input a basis $\mathbf{B}$ of a lattice $\mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$ of rank n and an integer a with $n/2 \leq a < n$ and returns a super lattice $\mathcal{L}' \supset \mathcal{L}$ of index 2^a with $\mathcal{L}' \subseteq \mathcal{L}/2$ such that for any $\varepsilon \in (0,1)$, we have $\eta_{\varepsilon'}(\mathcal{L}') \leq \eta_\varepsilon(\mathcal{L})/\sqrt{2}$ with probability at least $1/2$ where $\varepsilon' := 2\varepsilon^2 + 2^{(n/2)+1-a}(1+\varepsilon)$.*

Combining the above lemmas and theorems, we can sample a discrete Gaussian sample at the smoothing parameter in $\sim 2^{0.5n}$ time.

Lemma 2.55. *There is a probabilistic algorithm that, given a lattice $\mathcal{L} \subset \mathbb{R}^n$, $m \in \mathbb{Z}_+$ and $s \geq \eta_{1/3}(\mathcal{L})$ as input, outputs m samples from a distribution $(m \cdot 2^{-\Omega(n^2)})$-close to $D_{\mathcal{L},s}$ in expected time $m \cdot 2^{n/2+o(n)}$ and space $(m + 2^{n/2}) \cdot 2^{o(n)}$.*

Proof. Let $a = \frac{n}{2} + 4$. We repeat the following until we output m vectors. We use the algorithm in Lemma 2.54 to obtain a lattice $\mathcal{L}' \supset \mathcal{L}$ of index 2^a. We then run the algorithm from Theorem 2.53 with input $(\mathcal{L}', s)$ to obtain a list of vectors from $\mathcal{L}'$. We output the vectors in this list that belong to $\mathcal{L}$. The correctness of the algorithm, assuming it outputs anything, is clear as long as the samples obtained from Theorem 2.53 are (sufficiently) independent, which we will prove below.

By Theorem 2.53, we obtain, in time and space $2^{(n/2)+o(n)}$, $M \leqslant 2^{n/2}$ vectors whose joint distribution is $2^{-\Omega(n^2)}$-close to $\mathcal{D}^M_{\mathcal{L}',s}$. The theorem guarantees that $M = 2^{n/2}$ if $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$. Also, by Lemma 2.54, with probability at least $1/2$, we have $s \geq \eta_{1/3}(\mathcal{L}) \geq \sqrt{2}\eta_{1/2}(\mathcal{L}')$. Note that when $s < \sqrt{2}\eta_{1/2}(\mathcal{L}')$, the samples obtained from Theorem 2.53 are still $2^{-\Omega(n^2)}$-close to M vectors independently sampled from $D_{\mathcal{L}',s}$ but M could be much lower than $2^{n/2}$ or even 0. On the other hand, if $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$ then $M = 2^{n/2}$.

Assume that $s \geqslant \sqrt{2}\eta_{1/2}(\mathcal{L}')$, which happens with probability at least $1/2$. From these $M = 2^{n/2}$ vectors, we will reject the vectors which are not in the lattice $\mathcal{L}$. It is easy to see that the probability that a vector sampled from the distribution $D_{\mathcal{L}',s}$ is in $\mathcal{L}$ is at least $\rho_s(\mathcal{L})/\rho_s(\mathcal{L}') \geq \frac{1}{2^a}$ by Lemma 2.48. Thus, the probability that we obtain at least one vector from $\mathcal{L}$ (which is distributed as $D_{\mathcal{L},s}$) is at least

$$\frac{1}{2}\left(1-(1-1/2^a)^{2^{n/2}}\right) = \frac{1}{2}\left(1-(1-1/2^{n/2+4})^{2^{n/2}}\right) \geq \frac{1}{2}\cdot\left(1-e^{-2^{n/2}/2^{n/2+4}}\right) = \frac{1}{2}(1-e^{-1/16}).$$

It implies that after rejection of vectors, with constant probability we will get at least one vector from $D_{\mathcal{L},s}$. Thus, the expected number of times we need to repeat the algorithm is $\mathcal{O}(m)$ until we obtain vectors $\boldsymbol{y_1},\ldots,\boldsymbol{y_m}$ whose joint distribution is statistically close to being independently distributed from $\mathcal{D}^m_{\mathcal{L},s}$. The time and space complexity is clear from the algorithm.

□

2.10 High-dimensional geometry

2.10.1 Area of hyperspherical cap

The surface area of $B_2^d(r)$ is well-known to be $\frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}r^{d-1}$ [Li11, p. 66], where Γ is the Gamma function. Let $A_d^r(\phi)$ be the surface area of a hyperspherical cap in $B_2^d(r)$ with spherical angle ϕ. The area of this hyperspherical cap can be calculated by integrating the surface area of a $(d-1)$-dimensional sphere with radius $r\sin\theta$ [Li11, p. 67]:

$$A_d^r(\phi)=\int_0^{\phi}2A_{d-1}^{r\sin\theta}(\pi/2)r\,\mathrm{d}\theta=\frac{2\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d-1}{2})}r^{d-1}\cdot\int_0^{\phi}\sin^{d-2}\theta\mathrm{d}\theta.$$

We abbreviate $A_d(\phi):=A_d^1(\phi)$ for simplicity. Note that by some fundamental calculations, $A_d(\phi)/A_d(\pi/2)$ is $\mathrm{poly}(d)\cdot(\sin\phi)^d$ [BDG+16, Lemma 2.1].

Following Ravsky's computation in his reply to a question on StackExchange [Rav21], we now use the area of the hyperspherical cap to upper bound the probability of the event that a uniformly random vector u on S^{d-1} only has a small overlap with another (fixed) unit vector v.

Theorem 2.56. *Let $d\geq 3$ be and integer, $v\in\mathbb{R}^d$ be a unit vector, and $a\in[0,1]$. Then we have*

$$\Pr_{u\sim S^{d-1}}\left[|\langle v,u\rangle|<a\right]\leq\frac{2}{\sqrt{\pi}}\cdot\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})}\cdot a.$$

Proof. Let $\phi\in[0,\pi/2]$ such that $\cos\phi=a$. We can see that if $|\langle v,u\rangle|\geq a$, then u will be in the hyperspherical cap (whose center is v) with spherical angle ϕ, and the probability that a uniformly-random u lands in that hyperspherical cap is $\frac{A_d(\phi)}{A_d(\pi/2)}$. Therefore,

$$\begin{aligned}\Pr_{u\sim S^{d-1}}\left[|\langle v,u\rangle|<a\right]&=\frac{A_d(\pi/2)-A_d(\phi)}{A_d(\pi/2)}=\left(\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}\right)^{-1}\cdot\left(\frac{2\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d-1}{2})}\right)\int_{\phi}^{\pi/2}\sin\theta^{d-2}\mathrm{d}\theta\\&\leq\left(\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}\right)^{-1}\cdot\left(\frac{2\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d-1}{2})}\right)\int_{\phi}^{\pi/2}\sin\theta\mathrm{d}\theta=\frac{2\Gamma(\frac{d}{2})}{\sqrt{\pi}\Gamma(\frac{d-1}{2})}\cdot(-\cos\frac{\pi}{2}+\cos\phi)\\&=\frac{2}{\sqrt{\pi}}\cdot\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})}\cdot a.\end{aligned}$$

□

By using Legendre's duplication formula $\Gamma(\frac{d}{2})\Gamma(\frac{d-1}{2}) = \frac{\sqrt{\pi}}{2^{d-2}}\Gamma(d-1)$ [Rud76, Chap. 8.21, Eq. 102] and the fact $\Gamma(d) = (d-1)!$, we obtain

$$\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} = \begin{cases} \frac{\Gamma(\frac{d}{2})^2}{\Gamma(\frac{d}{2})\Gamma(\frac{d-1}{2})} = \frac{2^{d-2}((\frac{d}{2}-1)!)^2}{\sqrt{\pi}(d-2)!} = \frac{2^{d-2}}{\sqrt{\pi}} \cdot \binom{d-2}{\frac{d-2}{2}}^{-1}, & \text{if } d \text{ is even,} \\ \frac{\Gamma(\frac{d}{2})\Gamma(\frac{d-1}{2})}{\Gamma(\frac{d-1}{2})^2} = \frac{\sqrt{\pi}(d-2)!}{2^{d-2}(\frac{d-3}{2}!)^2} = \frac{\sqrt{\pi}(d-2)}{2^{d-2}} \cdot \binom{d-3}{\frac{d-3}{2}}, & \text{if } d \text{ is odd.} \end{cases}$$

Plugging the above into Theorem 2.56, we have the following corollary.

Corollary 2.57. *Let $d \geq 4$ be an integer, $v \in \mathbb{R}^d$ be a unit vector, and $c \geq 1$. Then we have*

$$\Pr_{u \sim S^{d-1}}\left[|\langle v, u\rangle| < \frac{1}{c\sqrt{d}}\right] < \frac{1}{c}.$$

Proof. By Theorem 2.56, it suffices to show $\frac{2}{\sqrt{\pi}} \cdot \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} \cdot \frac{1}{\sqrt{d}} < 1$ for every $d \geq 4$. When d is even, by using Robbins' bound $\frac{4^m}{\sqrt{\pi m}}\exp(-\frac{1}{6m}) \leq \binom{2m}{m} \leq \frac{4^m}{\sqrt{\pi m}}$ [Rob55, consequence of Eq. 1], we have

$$\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} = \frac{2^{d-2}}{\sqrt{\pi}} \cdot \binom{d-2}{\frac{d-2}{2}}^{-1} \leq \sqrt{\frac{d-2}{2}}\exp(\frac{1}{3d-6}) \leq \sqrt{\frac{d}{2}}\exp(\frac{1}{6}),$$

implying that $\frac{2}{\sqrt{\pi}} \cdot \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} \cdot \frac{1}{\sqrt{d}} \leq \sqrt{\frac{2}{\pi}} \cdot \exp(\frac{1}{6}) < 1$. Similarly, when d is odd, we have

$$\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} = \frac{\sqrt{\pi}(d-2)}{2^{d-2}} \cdot \binom{d-3}{\frac{d-3}{2}} \leq \frac{d-2}{\sqrt{2(d-3)}},$$

implying that $\frac{2}{\sqrt{\pi}} \cdot \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} \cdot \frac{1}{\sqrt{d}} \leq \sqrt{\frac{2}{\pi}} \cdot \frac{d-2}{\sqrt{d(d-3)}} < 1$. □

2.10.2 Lattice kissing number (w.r.t. ℓ_2-norm) and related quantities

Here we introduce the kissing number of the lattice with respect to ℓ_2-norm, and by the discussion at the beginning of Section 2.9.2, we here again assume the lattice is full rank and $d = n$. For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and $r > 0$, let $N(\mathcal{L}, r)$ denote the number of nonzero lattice vectors of length (ℓ_2-norm distance) at most r. A natural question is to bound this quantity in terms of r. When $r < \lambda_1(\mathcal{L})$, only the origin lies inside the ball so $N(\mathcal{L}, r) = 0$. When $r = \lambda_1(\mathcal{L})$, this quantity is known as the kissing number $\tau(\mathcal{L})$ of the lattice:

$$\tau(\mathcal{L}) = |\{\boldsymbol{x} \in \mathcal{L} : \|\boldsymbol{x}\| = \lambda_1(\mathcal{L})\}|.$$

Finally when $r \to \infty$, $N(\mathcal{L}, r) = \frac{r^n \cdot \mathrm{Vol}(B_2^n)}{\det(\mathcal{L})} + o(r^n)$ by the geometric interpretation of the determinant of a lattice. The precise behavior for intermediate values of r, however, is unclear and for that reason we introduce the quantity

$$\gamma(\mathcal{L}) = \inf\{\gamma : \forall r \geq 1,\ N(\mathcal{L}, r\lambda_1(\mathcal{L})) \leq \gamma \cdot r^n\}. \tag{2.5}$$

It is clear by the definition that $\gamma(\mathcal{L}) \geq \tau(\mathcal{L})$. The best known upper bound on this quantity comes from the breakthrough work of Kabatyanskii and Levenshtein [KL78]:

$$\gamma(\mathcal{L}) \leqslant 2^{0.401n+o(n)}. \tag{2.6}$$

Recently, Serge Vlăduţ [Vlă19] gave a construction of an infinite set of lattices whose kissing number is greater than $2^{0.0338n+o(n)}$. This is the first (and only known) construction of a family of lattices with kissing number $2^{\Omega(n)}$. Given this result (and the fact that it was hard to find such a family of lattices), one might conjecture that $\tau(\mathcal{L}) \leq c^{n+o(n)}$ for some constant c much smaller than $2^{0.401}$. Moreover, in practice most lattices have a much smaller kissing number, which is often $2^{o(n)}$. Given the close connection between $\tau(\mathcal{L})$ and $\gamma(\mathcal{L})$, it is not unreasonable to conjecture that $\gamma(\mathcal{L})$ is also $2^{o(n)}$ for most lattices. In view of the fact that $\gamma(\mathcal{L})$ can be anywhere between 2 and $2^{0.401n+o(n)}$, we will study the dependence of the time complexity of our algorithms for SVP in Chapter 5 on $\gamma(\mathcal{L})$ by introducing

$$\beta(\mathcal{L}) = \gamma(\mathcal{L})^{1/n}. \tag{2.7}$$

The upper bound above can then be reformulated as $\beta(\mathcal{L}) \leqslant 2^{0.401+o(1)}$ for any lattice $\mathcal{L}$ and 1 is the trivial lower bound.

The following lemma connects $\lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*)$ to $\beta(\mathcal{L})$.

Lemma 2.58 (Variant of [ADR+15, Lemma 6.1])**.** *For every lattice $\mathcal{L} \subset \mathbb{R}^n$ and $\varepsilon \in (0,1)$,*

$$\sqrt{\frac{\ln(1/\varepsilon)}{\pi}} < \lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\beta(\mathcal{L})^2 n}{2\pi e}} \cdot \varepsilon^{-1/n} \cdot (1+o(1)). \tag{2.8}$$

Moreover, if $\varepsilon \leq (e/\beta(\mathcal{L})^2 + o(1))^{-\frac{n}{2}}$, we have

$$\sqrt{\frac{\ln(1/\varepsilon)}{\pi}} < \lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\ln(1/\varepsilon) + n\ln\beta(\mathcal{L}) + o(n)}{\pi}}. \tag{2.9}$$

We also prove the following two lemmas related to the kissing number.

Lemma 2.59. *For every lattice $\mathcal{L} \subset \mathbb{R}^n$, $s > 0$, and $r \geqslant s\sqrt{n}/\lambda_1(\mathcal{L})$,*

$$\rho_s(\mathcal{L} \setminus B_2^n(r\lambda_1(\mathcal{L})) \leqslant 2er^n\beta(\mathcal{L})^n \rho_s(\mathcal{L}\setminus\{0\})^{r^2}.$$

Proof. Let $t = 1+1/n$, $R = r\lambda_1(\mathcal{L}) \geq s\sqrt{n}$, $r_i = Rt^i$ and $T_i = B_2^n(r_{i+1}) \setminus B_2^n(r_i)$ for all $i \in \mathbb{N}$. By the definition of $\beta(\mathcal{L})$,

$$|\mathcal{L} \cap T_i| \leqslant |\mathcal{L} \cap B_2^n(r_{i+1})| \leqslant \beta(\mathcal{L})^n (rt^{i+1})^n.$$

It then follows that

$$\rho_s(\mathcal{L}\setminus B_2^n(R)) = \sum_{i=0}^{\infty} \rho_s(\mathcal{L}\cap T_i) \leqslant \sum_{i=0}^{\infty} |\mathcal{L}\cap T_i| e^{-\pi\frac{r_i^2}{s^2}} \leqslant \beta(\mathcal{L})^n \sum_{i=0}^{\infty} \underbrace{(rt^{i+1})^n e^{-\pi\frac{R^2}{s^2}t^{2i}}}_{:=f(i)}.$$

Note that for all $i \in \mathbb{N}$,

$$\frac{f(i+1)}{f(i)} = t^n e^{-\pi \frac{R^2}{s^2} t^{2i}(t^2-1)} \leqslant e^{1-3\pi \frac{R^2}{ns^2}} \leqslant e^{1-3\pi} < 1/2,$$

where we have used that $t^n \leqslant e$, $t^{2i} \geqslant 1$, and $t^2 - 1 \leqslant \frac{3}{n}$ in the second equality and $R \geqslant s\sqrt{n}$ in the third one. It follows that

$$\rho_s(\mathcal{L} \setminus B_2^n(R)) \leqslant \beta(\mathcal{L})^n \cdot 2 \cdot f(0) \leqslant \beta(\mathcal{L})^n \cdot 2(rt)^n e^{-\pi \frac{R^2}{s^2}} \leqslant 2e \cdot r^n \beta(\mathcal{L})^n \cdot e^{-\pi \frac{r^2 \lambda_1(\mathcal{L})^2}{s^2}}.$$

On the other hand,

$$\rho_s(\mathcal{L} \setminus \{0\}) \geqslant \rho_s(\boldsymbol{v}) = e^{-\pi \frac{\lambda_1(\mathcal{L})^2}{s^2}},$$

where $\boldsymbol{v}$ is the shortest vector, so the result follows immediately. □

Lemma 2.60. *For all constant $c > 1$, lattice $\mathcal{L} \subset \mathbb{R}^n$, and $\varepsilon \in (0, 1/e)$, we have*

$$t\eta_\varepsilon(\mathcal{L}) \leqslant \eta_{\varepsilon^{t^2} f(c,n,\beta)}(\mathcal{L}),$$

where $t = 1 + \frac{1}{n^c}$ and $f(c,n,\beta) = e^{-3n^{1-c}(\ln \beta(\mathcal{L}^*) + c \ln n)}(1 - e^{(c \cdot n \ln(n) + n \ln \beta(\mathcal{L}^*) + 2 + \ln 2) - n^{2c}})t^2$.[9]

Proof. Let $r = n^c$ and $t = 1 + \frac{1}{r}$ with $c > 1$. Since $\varepsilon \leqslant 1/e$ and $n \geq 4$, by Lemma 2.58 we obtain

$$\frac{\sqrt{n}}{\eta_\varepsilon(\mathcal{L})\lambda_1(\mathcal{L}^*)} \leqslant \sqrt{\frac{n\pi}{\ln(1/\varepsilon)}} \leqslant r. \tag{2.10}$$

Let $\Gamma = (\mathcal{L}^* \setminus \{0\}) \cap B_2^n(r\lambda_1(\mathcal{L}^*))$. By the definition of $\beta(\mathcal{L}^*)$ we have

$$|\Gamma| \leqslant \beta(\mathcal{L}^*)^n r^n.$$

Now observe that

$$\begin{aligned}
\rho_{1/(t\cdot\eta_\varepsilon(\mathcal{L}))}(\mathcal{L}^* \setminus \{0\}) &= \sum_{x \in \mathcal{L}^* \setminus \{0\}} \rho_{1/(t\cdot\eta_\varepsilon(\mathcal{L}))}(x) = \sum_{x \in \mathcal{L}^* \setminus \{0\}} \rho_{1/\eta_\varepsilon(\mathcal{L})}(x)^{t^2} \\
&\geqslant \sum_{x \in \Gamma} \rho_{1/\eta_\varepsilon(\mathcal{L})}(x)^{t^2} \geqslant |\Gamma| \left(\frac{1}{|\Gamma|} \sum_{x \in \Gamma} \rho_{1/\eta_\varepsilon(\mathcal{L})}(x) \right)^{t^2} && \text{(by Jensen's inequality)} \\
&= |\Gamma|^{1-t^2} \rho_{1/\eta_\varepsilon(\mathcal{L})}(\Gamma)^{t^2} \\
&= |\Gamma|^{1-t^2} \left(\rho_{1/\eta_\varepsilon(\mathcal{L})}(\mathcal{L}^* \setminus \{0\}) - \rho_{1/\eta_\varepsilon(\mathcal{L})}(\mathcal{L}^* \setminus B_2^n(r\lambda_1(\mathcal{L}^*))) \right)^{t^2} \\
&\geqslant \left(\beta(\mathcal{L}^*)^n r^n \right)^{1-t^2} \left(\varepsilon - 2e\beta(\mathcal{L}^*)^n r^n \varepsilon^{r^2} \right)^{t^2}
\end{aligned}$$

where the last equality holds by $\rho_{1/\eta_\varepsilon(\mathcal{L})}(\mathcal{L}^* \setminus \{0\}) = \varepsilon$ and Lemma 2.59 (with Eq (2.10)). Now observe that $1 - t^2 \geqslant -3/r$ and

$$\left(\beta(\mathcal{L}^*)^n r^n \right)^{-3/r} = e^{-3n^{1-c}(\ln \beta(\mathcal{L}^*) + c \ln n)}$$

[9] $f(c,n,\beta)$ is actually $e^{o(1)}$ since $\lim_{n\to\infty} f(c,n,\beta) = 1 + o(1)$.

since $c > 1$. We also have that

$$\begin{aligned} 2e\beta(\mathcal{L}^*)^n r^n \varepsilon^{r^2} &= e^{1+\ln 2+n\ln\beta(\mathcal{L}^*)+nc\ln(n)-(r^2-1)\ln\frac{1}{\varepsilon}}\varepsilon \\ &\leqslant e^{(c\cdot n\ln(n)+n\ln\beta(\mathcal{L}^*)+2+\ln 2)-n^{2c}}\varepsilon \qquad \text{(since } \varepsilon \leqslant 1/e\text{)}. \end{aligned}$$

Then it follows that

$$\begin{aligned} \rho_{1/(t\cdot\eta_\varepsilon(\mathcal{L}))}(\mathcal{L}^* \setminus \{0\}) &\geqslant e^{-3n^{1-c}(\ln\beta(\mathcal{L}^*)+c\ln n)}(\varepsilon)^{t^2}(1-e^{(c\cdot n\ln(n)+n\ln\beta(\mathcal{L}^*)+2+\ln 2)-n^{2c}})^{t^2} \\ &= \varepsilon^{t^2} f(c,n,\beta). \end{aligned}$$

Therefore we must have $\eta_{\varepsilon^{t^2} f(c,n,\beta)}(\mathcal{L}) \geqslant t\cdot\eta_\varepsilon(\mathcal{L})$. □

Part I

Quantum algorithms

CHAPTER 3

Quantum algorithms for Lasso

3.1 Introduction

One of the simplest, most useful and best-studied problems in machine learning and statistics is *linear regression*. We are given N data points $\{(x_i, y_i)\}_{i=0}^{N-1}$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, and want to fit a line through these points that has small error. In other words, we want to find a vector $\theta \in \mathbb{R}^d$ of coefficients such that the inner product $\langle \theta, x \rangle = \sum_{j=1}^{d} \theta_j x_j$ is a good predictor for the y-variable. There are different ways to quantify the error ("loss") of such a θ-vector. One of the most common is the squared error $(\langle \theta, x \rangle - y)^2$, averaged over the N data points (or over an underlying distribution $\mathcal{D}$ that generated the data). If we let X be the $N \times d$ matrix whose N rows are the x-vectors of the data, then the linear regression problem wants us to find a $\theta \in \mathbb{R}^d$ that minimizes $\|X\theta - y\|_2^2$. This minimization problem has a well-known closed-form solution: $\theta = (X^T X)^+ X^T y$, where the superscript '+' indicates the Moore-Penrose pseudoinverse.

In practice, unconstrained least-squares regression sometimes has problems with overfitting. Precisely, the closed-form solution ($(X^T X)^+ X^T y$) often yields solutions θ where all entries are non-zero, even when only a few of the d coordinates in the x-vector really matter and one would really hope for a sparse vector θ [SB14, see Chapters 2 and 13]. This situation may be improved by "regularizing" θ via additional constraints. The most common constraints are to require that the ℓ_1-norm or ℓ_2-norm of θ is at most some bound B.[1] Linear regression with an ℓ_1-constraint is called *Lasso* (due to Tibshirani [Tib96]), while with an ℓ_2-constraint it is called *Ridge* (due to Hoerl and Kennard [HK70]).[2]

[1] For ease of presentation we will set $B = 1$. However, one can also set B differently or even do a binary search over its values, finding a good θ for each of those values and selecting the best one at the end. Instead of putting a hard upper bound B on the norm, one may also include it as a penalty term in the objective function itself, by just minimizing the function $\|X\theta - y\|_2^2 + \lambda \|\theta\|$, where λ is a Lagrange multiplier and the norm of θ could be ℓ_1 or ℓ_2 (and could also be squared). This amounts to basically the same thing as our setup.

[2] Another popular choice for the constraints is ℓ_0-norm constraint, which is called best subset selection. This problem, however, is not convex and it is proven to be in general NP-hard [DK08].

Lasso

The *least absolute shrinkage and selection operator*, or *Lasso*, is a special case of linear regression restricting solutions to the unit ℓ_1-ball. Introducing the ℓ_1-penalty in linear regression offers several advantages. Firstly, it effectively handles scenarios where the original data is sparse by encouraging sparsity in the model itself (i.e., the θ-vector), aligning with the inherent sparsity of the data. Moreover, the ℓ_1-penalty enhances the robustness of the model against noise, as it prioritizes the selection of relevant features while disregarding noise or irrelevant variables [ZY06].

One other notable benefit is the guarantee of always having an ε-minimizer (i.e., a vector θ whose loss is only an additive ε worse than the minimal-achievable loss) with sparsity proportional to $\mathcal{O}(1/\varepsilon)$ (This statement actually holds for every convex objective function with constant curvature constant, see Section 3.2.3 and [Jag11, Chapter 3.2]). This property ensures that even in the presence of noise or outliers, the model can still identify a solution that effectively represents the underlying patterns in the data. Additionally, the computational efficiency of linear regression with an ℓ_1-penalty is significantly improved because every candidate in the iterative optimization algorithms is also sparse. By promoting sparsity, the computation is streamlined, resulting in faster model training and inference, which is advantageous, particularly for large datasets or real-time applications.

While Lasso (ℓ_1-regularization) offers various advantages, it also presents certain drawbacks. The most significant one is the absence of a closed-form solution. Unlike ordinary linear regression or Ridge (see below), which can be solved analytically through matrix operations, the Lasso problem requires iterative optimization techniques due to its non-differentiable penalty term (the $\|\theta\|_1$ term). Finding the exact minimizer, therefore, becomes computationally more demanding.

Ridge

The Ridge regression is a special case of linear regression restricting solutions to the unit ℓ_2-ball. Introducing the ℓ_2-penalty in linear regression has several advantages. Firstly, it stabilizes all convex Lipschitz or convex smooth learning processes. By incorporating the ℓ_2-penalty, the optimization becomes more well-behaved in terms of convergence. Moreover, while the closed-form loss function might only exhibit Lipschitz continuity without ℓ_2-regularization, the addition of ℓ_2-penalty (which is strongly convex) transforms the loss function into a strongly convex form. This ensures some favorable properties for optimization algorithms, like faster convergence rates. Meanwhile, by promoting strong convexity through the ℓ_2-penalty, the model becomes more robust and less likely to overfit. This property is particularly useful in the scenarios where the dataset is noisy, as the regularization provided by the ℓ_2-penalty helps lessen the effects of such disturbances, leading to more reliable and interpretable models [SB14, Chapters 13]. One other notable advantage of Ridge regression is its closed-form solution. Unlike Lasso or some other regularization techniques that require iterative optimization algorithms, Ridge regression can be solved analytically, which can be com-

putationally efficient and straightforward to implement.

Ridge regression also presents certain drawbacks. The most well-known one is that its output tends to be dense. Unlike Lasso, which encourages sparsity by driving certain coefficients to zero, Ridge regression tends to retain all features in the model, albeit with reduced weights. This denseness in the output can lead to increased complexity and computational overhead, especially when dealing with high-dimensional datasets. Furthermore, in situations where feature sparsity is desirable, the dense output of Ridge regression may not be ideal. Sparse solutions are often preferred for computational efficiency and model interpretability, making Ridge regression less suitable for such scenarios compared to other regularization techniques like Lasso.

Both Lasso and Ridge are widely used for robust regression and sparse estimation in ML problems and elsewhere [Vin78; BG11]. Consequently, there has been great interest in finding the fastest-possible algorithms for them. For reasons of efficiency, algorithms typically aim at finding not the exactly optimal solution but an ε-minimizer (a vector θ whose loss is only an additive ε worse than the minimal-achievable loss). The best known results on the time complexity of classical algorithms for Lasso are an upper bound of $\tilde{\mathcal{O}}(d/\varepsilon^2)$ [HK12] and a lower bound of $\Omega(d/\varepsilon)$ [CSS11] (which we actually improve to a tight lower bound, see Chapter 6); for Ridge the best bound is $\tilde{\Theta}(d/\varepsilon^2)$ [HK12], which is tight up to logarithmic factors.[3]

3.1.1 Main results and high-level intuition

In this chapter, we will talk about a quantum algorithm that finds an ε-minimizer for Lasso in time $\tilde{\mathcal{O}}(\sqrt{d}/\varepsilon^2)$. This gives a quadratic quantum speedup over the best-possible classical algorithm in terms of d, while the ε-dependence remains the same as in the best known classical algorithm. We will also discuss some possible ways to achieve a quantum speedup in terms of the ε-dependency at the end of this chapter.

Our quantum algorithm is based on the Frank-Wolfe algorithm, a well-known iterative convex optimization method [FW56]. Frank-Wolfe, when applied to a Lasso instance, starts at the all-zero vector θ and updates this in $\mathcal{O}(1/\varepsilon)$ iterations to find an ε-minimizer. Each iteration looks at the gradient of the loss function at the current point θ and selects the best among $2d$ directions for changing θ (each of the d coordinates can change positively or negatively, whence $2d$ directions). The new θ will be a convex combination of the previous θ and this optimal direction of change. Note that Frank-Wolfe automatically generates *sparse* solutions: only one coordinate of θ can change from zero to nonzero in one iteration, so the number of nonzero entries in the final θ is at most the number of iterations, which is $\mathcal{O}(1/\varepsilon)$.

Our quantum version of Frank-Wolfe does not reduce the number of iterations, which remains $\mathcal{O}(1/\varepsilon)$, but it does reduce the cost per iteration. In each iteration it

[3] For such bounds involving additive error ε to be meaningful, one has to put certain normalization assumptions on X and y, which are given in the body of this chapter. It is known that $N = \mathcal{O}((\log d)/\varepsilon^2)$ data points suffice for finding an ε-minimizer, which explains the absence of N as a separate variable in these bounds.

selects the best among the $2d$ possible directions for changing θ by using a version of quantum minimum-finding on top of a quantum approximation algorithm for entries of the gradient (which in turn uses amplitude estimation). Both this minimum-finding and our approximation of entries of the gradient will result in approximation errors throughout. Fortunately Frank-Wolfe is a very robust method which still converges if we carefully ensure those quantum-induced approximation errors are sufficiently small.

Our quantum algorithm assumes coherent quantum query access to the entries of the data points (x_i, y_i), as well as a relatively small QRAM (quantum-readable classical-writable classical memory). We use a variant of a QRAM data structure developed by Prakash and Kerenidis [Pra14; KP17], to store the nonzero entries of our current solution θ in such a way that we can (1) quickly generate θ as a quantum state, and (2) quickly incorporate the change of θ incurred by a Frank-Wolfe iteration.[4] Because our θ is $\mathcal{O}(1/\varepsilon)$-sparse throughout the algorithm, we only need $\tilde{\mathcal{O}}(1/\varepsilon)$ bits of QRAM.

Roadmap

We include classical preliminaries for Lasso and Ridge in Section 3.2. In Section 3.2.1 we will introduce basic notations and definitions for the expected loss and empirical loss, as well as theorems that connect those two losses. In Section 3.2.3 we will introduce the Frank-Wolfe algorithm and explation how many iterations suffice for finding a good ε-approximation solution. In Section 3.3 we show how to approximate entries of gradient efficiently, and by plugging the result into the Frank-Wolfe algorithm we obtain the results in Section 3.4.

Remark

In the whole Chapter 3, when we use the notation for KP-tree, it always means the KP-tree with respect to ℓ_1-norm (see Section 2.4.1).

3.2 Linear regression problems and the Frank-Wolfe algorithm

We will introduce some basic notations and definitions for linear regression with norm constraints in this section. At the end of this section, we will introduce the classical Frank-Wolfe algorithm, and its converge rate.

[4] Each iteration will actually change all nonzero entries of θ because the new θ is a convex combination of the old θ and a vector with one nonzero entry. Our data structure keeps track of a global scalar, which saves us the cost of separately adjusting all nonzero entries of θ in the data structure in each iteration.

3.2.1 Expected and empirical loss

Let sample set $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be a set of i.i.d. samples from $\mathbb{R}^d \times \mathbb{R}$, drawn according to an unknown distribution $\mathcal{D}$. A *hypothesis* is a function $h : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{H}$ denotes a set of hypotheses. To measure the performance of the prediction, we use a convex loss function $\ell : \mathbb{R}^2 \to \mathbb{R}$. The *expected loss* of h with respect to $\mathcal{D}$ is denoted by $L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$, and the *empirical loss* of h with respect to S is denoted by $L_S(h) = \frac{1}{N}\sum_{i\in[N]-1} \ell(h(x_i), y_i)$.

Definition 3.1. Let $\varepsilon > 0$. An $h \in \mathcal{H}$ is an ε-minimizer over $\mathcal{H}$ with respect to distribution $\mathcal{D}$ if

$$L_{\mathcal{D}}(h) - \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') \leq \varepsilon.$$

Definition 3.2. Let $\varepsilon > 0$. An $h \in \mathcal{H}$ is an ε-minimizer over $\mathcal{H}$ with respect to sample set S if

$$L_S(h) - \min_{h'\in\mathcal{H}} L_S(h') \leq \varepsilon.$$

3.2.2 Linear regression problems and their classical and quantum setup

In linear regression problems, the hypothesis class is the set of linear functions on $\mathbb{R}^d$. The goal is to find a vector θ for which the corresponding hypothesis $\langle\theta, x\rangle$ provides a good prediction of the target y. One of the most natural choices for regression problems is the squared loss

$$\ell(\hat{y}, y) = (\hat{y} - y)^2.$$

We can instantiate the expected and empirical losses as a function of θ using the squared loss:

$$L_{\mathcal{D}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(\langle\theta, x\rangle, y)] = \mathbb{E}_{(x,y)\sim\mathcal{D}}[(\langle\theta, x\rangle - y)^2],$$

$$L_S(\theta) = \frac{1}{N}\sum_{i\in[N]-1} \ell(\langle\theta, x\rangle, y_i) = \frac{1}{N}\sum_{i\in[N]-1} (\langle\theta, x\rangle - y_i)^2.$$

We also write the empirical loss as $L_S(\theta) = \frac{1}{N}\|X\theta - y\|_2^2$, where matrix entry X_{ij} is the jth entry of the vector x_i, and y is the N-dimensional vector with entries y_i. As we will see below, if the instances in the sample set are chosen i.i.d. according to $\mathcal{D}$, and N is sufficiently large, then $L_S(\theta)$ and $L_{\mathcal{D}}(\theta)$ are typically close by the law of large numbers.

In the quantum case, we assume the sample set S is stored in a QROM, which we can access by means of queries to the oracles $O_X : |i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|X_{ij}\rangle$ and $O_y : |i\rangle|0\rangle \to |i\rangle|y_i\rangle$.

Lasso

The *least absolute shrinkage and selection operator*, or *Lasso*, is a special case of linear regression with a norm constraint on the vector θ: it restricts solutions to the unit ℓ_1-ball, which we denote by B_1^d. For the purpose of normalization, we require that every

sample (x, y) satisfies $\|x\|_\infty \leq 1$ and $|y| \leq 1$.[5] The goal is to find a $\theta \in B_1^d$ that (approximately) minimizes the expected loss. Since the expected loss is not directly accessible, we instead find an approximate minimizer of the empirical loss. Mohri, Rostamizadeh, and Talwalkar [MRT18] showed that with high probability, an approximate minimizer for *empirical* loss is also a good approximate minimizer for *expected* loss.

Theorem 3.3 ([MRT18], Theorem 11.16)**.** *Let $\mathcal{D}$ be an unknown distribution over $[-1,1]^d \times [-1,1]$ and $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be a sample set containing N i.i.d. samples from $\mathcal{D}$. Then, for each $\delta > 0$, with probability $\geq 1-\delta$ over the choice of S, the following holds for all $\theta \in B_1^d$:*

$$L_{\mathcal{D}}(\theta) - L_S(\theta) \leq 4\sqrt{\frac{2\log(2d)}{N}} + 4\sqrt{\frac{\log(1/\delta)}{2N}}.$$

This theorem implies that if $N = c\log(d/\delta)/\varepsilon^2$ for sufficiently large constant c, then finding (with error probability $\leq \delta$) an ε-minimizer for the *empirical* loss L_S, implies finding (with error probability $\leq 2\delta$ taken both over the randomness of the algorithm and the choice of the sample S) a 2ε-minimizer for the *expected* loss $L_{\mathcal{D}}$.

Ridge

Another special case of linear regression with a norm constraint is *Ridge*, which restricts solutions to the unit ℓ_2-ball B_2^d. For the purpose of normalization, we now require that every sample (x, y) satisfies $\|x\|_2 \leq 1$ and $|y| \leq 1$. Similarly to the Lasso case, Mohri, Rostamizadeh, and Talwalkar [MRT18] showed that with high probability, an approximate minimizer for the empirical loss is also a good approximate minimizer for the expected loss.

Theorem 3.4 ([MRT18], Theorem 11.11)**.** *Let $\mathcal{D}$ be an unknown distribution over $B_2^d \times [-1,1]$ and $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be a sample set containing N i.i.d. samples from $\mathcal{D}$. Then, for each $\delta > 0$, with probability $\geq 1-\delta$ over the choice of S, the following holds for all $\theta \in B_2^d$:*

$$L_{\mathcal{D}}(\theta) - L_S(\theta) \leq 8\sqrt{\frac{1}{N}} + 4\sqrt{\frac{\log(1/\delta)}{2N}}.$$

3.2.3 The classical Frank-Wolfe algorithm

Below is a description of the Frank-Wolfe algorithm with approximate linear solvers. For now this is for an arbitrary convex objective function L and arbitrary compact convex domain $\mathcal{X}$ of feasible solutions; for Lasso we will later instantiate these to the quadratic loss function and ℓ_1-ball, respectively. Frank-Wolfe finds an ε-approximate solution to a convex optimization problem, using $\mathcal{O}(1/\varepsilon)$ iterations. It is a first-order

[5] Note that if $\theta \in B_1^d$ and $\|x\|_\infty \leq 1$, then $|\langle \theta, x\rangle| \leq 1$ by Hölder's inequality.

method: each iteration assumes access to the gradient of the objective function at the current point. The algorithm considers the linearization of the objective function, and moves towards a minimizer of this linear function without ever leaving the domain $\mathcal{X}$ (in contrast to for instance projected gradient descent).

input : number of iterations $T > 0$; convex differentiable function L; compact convex domain $\mathcal{X}$;
Let C_L be the curvature constant of L;
Let θ^0 be an arbitrary point in $\mathcal{X}$;
for $t \leftarrow 0$ **to** T **do**
  $\tau_t = \frac{2}{t+2}$;
  find $s \in \mathcal{X}$ such that $\langle s, \nabla L(\theta^t)\rangle \leq \min_{s' \in \mathcal{X}} \langle s', \nabla L(\theta^t)\rangle + \frac{\tau_t C_L}{4}$;
  $\theta^{t+1} = (1-\tau_t)\theta^t + \tau_t s$;
end
output: θ^T;

Algorithm 1: The Frank-Wolfe algorithm with approximate linear subproblems

The convergence rate of the Frank-Wolfe algorithm is affected by the "non-linearity" of the objective function L, as measured by the curvature constant C_L:

Definition 3.5. The curvature constant C_L of a convex and differentiable function $L : \mathbb{R}^d \to \mathbb{R}$ with respect to a convex domain $\mathcal{X}$ is defined as

$$C_L \equiv \sup_{\substack{x,s \in \mathcal{X}, \gamma \in [0,1], \\ y = x + \gamma(s-x)}} \frac{2}{\gamma^2}(L(y) - L(x) - \langle \nabla L(x), (y-x)\rangle).$$

Next we give an upper bound for the curvature constant of the empirical loss function for Lasso.

Theorem 3.6. *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ with all entries of x_i and y_i in $[-1,1]$. Then the curvature constant C_{L_S} of L_S with respect to B_1^d is ≤ 8.*

Proof. We know

$$L_S(\theta) = \frac{1}{N}\|X\theta - y\|_2^2 = \frac{(X\theta - y)^T(X\theta - y)}{N} = \frac{\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y}{N},$$

which implies the Hessian of L_S is $\nabla^2 L_S(z) = \frac{2X^T X}{N}$, independent of z. By replacing sup by max because the domain is compact, we have

$$\begin{aligned} C_{L_S} &= \max_{\substack{x,s \in \mathcal{X}, \gamma \in [0,1], \\ y = x + \gamma(s-x)}} \frac{2}{\gamma^2}(L_S(y) - L_S(x) - \langle \nabla L_S(x), (y-x)\rangle) \\ &= \max_{x,s \in \mathcal{X}, \gamma \in [0,1]} \langle (s-x), \nabla^2 L_S \cdot (s-x)\rangle = \max_{x,s \in \mathcal{X}} \frac{2}{N}\|X(s-x)\|_2^2. \end{aligned}$$

Each coefficient of X is at most 1 in absolute value, and $s-x \in 2B_1^d$, hence each entry of the vector $X(s-x)$ has magnitude at most 2. Therefore $\max_{x,y\in B_1^d} \frac{2}{N}\|X(s-x)\|_2^2$ is at most 8. □

The original Frank-Wolfe algorithm [FW56] assumed that the minimization to determine the direction-of-change s was done exactly, without the additive error term $\frac{\tau_t C_{L_S}}{4}$ that we wrote in Algorithm 1. However, the following theorem, due to Jaggi [Jag13], shows that solving approximate linear subproblems is sufficient for the Frank-Wolfe algorithm to converge at an $\mathcal{O}(C_{L_S}/T)$ rate, which means one can find an ε-approximate solution with $T = \mathcal{O}(C_{L_S}/\varepsilon)$ iterations.

Theorem 3.7 ([Jag13], Theorem 1)**.** *For each iteration $t \geq 1$, the corresponding θ^t of Algorithm 1 satisfies*

$$L_S(\theta^t) - \min_{\theta' \in B_1^d} L_S(\theta') \leq \frac{3C_{L_S}}{t+2}.$$

3.3 Approximating the quadratic loss function and entries of its gradient

In this section, we give a quantum algorithm to estimate the quadratic loss function $L_S(\theta)$ and entries of its gradient, given query access to entries of the vectors in $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ and given a KP-tree for $\theta \in B_1^d$. One can estimate these numbers with additive error β in time roughly $1/\beta$.

We start with estimating entries of the gradient of the loss function at a given θ:

Theorem 3.8. *Let $\theta \in B_1^d$, and $\beta, \delta > 0$. Suppose we have a KP-tree KP_θ of vector θ and can make quantum queries to $O_{KP_\theta} : |\ell, k\rangle|0\rangle \to |\ell, k\rangle|KP_\theta(\ell, k)\rangle$. One can implement $\tilde{U}_{\nabla L_S} : |j\rangle|0\rangle \to |j\rangle|\Lambda\rangle$ such that for all $j \in [d]-1$, after measuring the state $|\Lambda\rangle$, with probability $\geq 1-\delta$ the first register λ of the outcome will satisfy $|\lambda - \nabla_j L_S(\theta)| \leq \beta$, by using $\tilde{\mathcal{O}}(\frac{\log(1/\delta)}{\beta})$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates.*

Proof. Fix j in the following proof. Note that

$$\nabla_j L_S(\theta) = \frac{2}{N}(X^T(X\theta - y))_j = \frac{2}{N}\sum_{i\in[N]-1} X_{ij}\cdot(X\theta - y)_i = \frac{2}{N}\sum_{i\in[N]-1}\sum_{k\in[d]-1} X_{ij}X_{ik}\theta_k - \frac{2}{N}\sum_{i\in[N]-1} X_{ij}y_i. \tag{3.1}$$

We will show how to estimate both terms of the right-hand side of Equation (3.1). Define the positive-controlled rotation such that for each $a \in \mathbb{R}$

$$U_{CR^+} : |a\rangle|0\rangle \to \begin{cases} |a\rangle(\sqrt{a}|1\rangle + \sqrt{1-a}|0\rangle), & \text{if } a \in (0,1] \\ |a\rangle|0\rangle, & \text{otherwise.} \end{cases}$$

This can be implemented up to negligibly small error by $\tilde{\mathcal{O}}(1)$ elementary gates. Also, by Theorem 2.11, we can implement $U_\theta : |0\rangle|0\rangle \to |\theta\rangle$, where $|\theta\rangle = \sum_{j\in[d]-1} \frac{\sqrt{|\theta_j|}}{\sqrt{\|\theta\|_1}} |j\rangle|\mathrm{sign}(\theta_j)\rangle$, using $\tilde{\mathcal{O}}(1)$ queries to O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates. We also use $U_u : |i\rangle|j\rangle|k\rangle|s\rangle|0\rangle \to |i\rangle|j\rangle|k\rangle|s\rangle|s\cdot X_{ij}\cdot X_{ik}\rangle$, where the last register is the product of three numbers; this U_u can be implemented (with negligible error) via $\mathcal{O}(1)$ queries to O_X, $O_X^\dagger$, and $\tilde{\mathcal{O}}(1)$ elementary gates.

First we estimate the first term $\frac{2}{N}\sum_{i\in[N]-1}\sum_{k\in[d]-1} X_{ij}X_{ik}\theta_k$ of Equation (3.1) with additive error $\beta/2$. Generating the state

$$\frac{1}{\sqrt{N}}\sum_{i\in[N]-1}|i\rangle|j\rangle|0\rangle|0\rangle|0\rangle|0\rangle,$$

and applying U_θ on the third and fourth registers, U_u on the first five registers, and U_{CR^+} on the last two registers, with $s_k = \mathrm{sign}(\theta_k)$, we get

$$\begin{aligned}
&\frac{1}{\sqrt{N\|\theta\|_1}}\Bigg(\sum_{\substack{i\in[N]-1,k\in[d]-1\\ s_kX_{ij}X_{ik}>0}}\sqrt{|\theta_k|}\,|i,j,k,s_k,s_kX_{ij}X_{ik}\rangle\left(\sqrt{s_kX_{ij}X_{ik}}\,|1\rangle+\sqrt{1-s_kX_{ij}X_{ik}}\,|0\rangle\right)\\
&+\sum_{\substack{i\in[N]-1,k\in[d]-1\\ s_kX_{ij}X_{ik}\le 0}}\sqrt{|\theta_k|}\,|i,j,k,s_k,s_kX_{ij}X_{ik}\rangle\,|0\rangle\Bigg)\\
=&\frac{1}{\sqrt{N\|\theta\|_1}}\Bigg(\sum_{\substack{i\in[N]-1,k\in[d]-1\\ \theta_kX_{ij}X_{ik}>0}}\sqrt{\theta_kX_{ij}X_{ik}}\,|i,j,k,s_k,s_kX_{ij}X_{ik}\rangle\,|1\rangle\\
&+\Bigg(\sum_{\substack{i\in[N]-1,k\in[d]-1\\ \theta_kX_{ij}X_{ik}>0}}\sqrt{|\theta_k|(1-s_kX_{ij}X_{ik})}\,|i,j,k,s_k,s_kX_{ij}X_{ik}\rangle+\sum_{\substack{i\in[N]-1,k\in[d]-1\\ \theta_kX_{ij}X_{ik}\le 0}}\sqrt{|\theta_k|}\,|i,j,k,s_k,s_kX_{ij}X_{ik}\rangle\Bigg)|0\rangle\Bigg)\\
=&\sqrt{a_+}\,|\phi_1\rangle|1\rangle+\sqrt{1-a_+}\,|\phi_0\rangle|0\rangle, \quad \text{for } a_+=\sum_{\substack{i\in[N]-1,k\in[d]-1\\ \theta_kX_{ij}X_{ik}>0}}\theta_kX_{ij}X_{ik}/(N\|\theta\|_1).
\end{aligned}$$

By Theorem 2.3, with failure probability at most 1/1000, we can estimate a_+ with additive error $\beta/(8\|\theta\|_1)$ using $\mathcal{O}(\|\theta\|_1/\beta)$ applications of O_X, $O_X^\dagger$, U_θ, $U_\theta^\dagger$, and $\tilde{\mathcal{O}}(\|\theta\|_1/\beta)$ elementary gates. Note that our algorithm knows $\|\theta\|_1$ because it is stored in the root of KP_θ. We similarly estimate

$$a_- = -\sum_{\substack{i\in[N]-1,k\in[d]-1\\ \theta_kX_{ij}X_{ik}<0}}\theta_kX_{ij}X_{ik}/(N\|\theta\|_1)$$

with additive error $\beta/(8\|\theta\|_1)$. Hence we estimate $\frac{2}{N}\sum_{i\in[N]-1} X_{ij}\cdot(X\theta)_i = 2\|\theta\|_1\cdot(a_+-a_-)$ with additive error $\beta/2$.

For the second term of the right-hand side of Equation (3.1) we use a very similar strategy: we separately estimate its positive term and its negative term, each with additive error $\beta/4$, using $\mathcal{O}(1/\beta)$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, and $\tilde{\mathcal{O}}(1/\beta)$ elementary gates, respectively. Therefore, we can estimate $\frac{1}{N}\sum_{i\in[N]-1} X_{ij}y_i$ with additive error $\beta/2$.

Combining the previous estimations, with failure probability at most 1/100, we estimate $\nabla_j L_S(\theta)$ with additive error β. Since $\|\theta\|_1 \le 1$, we use $\tilde{\mathcal{O}}(1/\beta)$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates. By repeating the procedure $\mathcal{O}(\log(1/\delta))$ times and taking the median of the outputs, we can decrease the failure probability from at most 1/100 to at most δ. □

Next we show how to estimate the value of the loss function itself at a given θ:

Theorem 3.9. *Let $\theta \in B_1^d$, and $\beta, \delta > 0$. Suppose we have a KP-tree KP_θ of vector θ and can make quantum queries to $O_{KP_\theta} : |\ell, k\rangle|0\rangle \to |\ell, k\rangle|KP_\theta(\ell, k)\rangle$. Then we can implement $\tilde{U}_{L_S} : |0\rangle \to |\Lambda\rangle$ such that after measuring the state $|\Lambda\rangle$, with probability $\ge 1-\delta$ the first register λ of the outcome will satisfy $|\lambda - L_S(\theta)| \le \beta$, by using $\tilde{\mathcal{O}}(\frac{\log(1/\delta)}{\beta})$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates.*

Proof. Recall

$$L_S(\theta) = \frac{1}{N}\sum_{i\in[N]-1} |\langle x_i, \theta\rangle - y_i|^2 = \frac{1}{N}\sum_{i\in[N]-1} \langle x_i, \theta\rangle^2 - \frac{2}{N}\sum_{i\in[N]-1} y_i\langle x_i, \theta\rangle + \frac{1}{N}\sum_{i\in[N]-1} y_i^2. \tag{3.2}$$

We use the positive controlled rotation gate defined in the proof of Theorem 3.8. By Theorem 2.11, we can implement $U_\theta : |0\rangle|0\rangle \to |\theta\rangle$, where $|\theta\rangle = \frac{1}{\sqrt{\|\theta\|_1}} \sum_{j\in[d]-1} \sqrt{|\theta_j|}\,|j\rangle|\mathrm{sign}(\theta_j)\rangle$, using $\tilde{\mathcal{O}}(1)$ queries to O_{KP_θ} and elementary gates.

We start by estimating (with additive error $\beta/4$) the first term on the right-hand side of Equation (3.2), which is

$$\frac{1}{N}\sum_{i\in[N]-1} \langle x_i, \theta\rangle^2 = \frac{1}{N}\sum_{i\in[N]-1}\sum_{k_1,k_2\in[d]-1} \theta_{k_1}\theta_{k_2} X_{ik_1} X_{ik_2}.$$

Let $U_u : |i\rangle|k_1\rangle|s_1\rangle|k_2\rangle|s_2\rangle|0\rangle \to |i\rangle|k_1\rangle|s_1\rangle|k_2\rangle|s_2\rangle|s_1 s_2 X_{ik_1} X_{ik_2}\rangle$, where the last register is the product of four numbers. This can be implemented (with negligible error) via $\mathcal{O}(1)$ queries to O_X, $O_X^\dagger$, and $\tilde{\mathcal{O}}(1)$ elementary gates.

We generate $\frac{1}{\sqrt{N}}\sum_{i\in[N]-1} |i\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$, and apply U_θ twice to obtain the state $\frac{1}{\sqrt{N}}\sum_{i\in[N]-1} |i\rangle|\theta\rangle^{\otimes 2}|0\rangle|0\rangle$. Applying U_u on the second to seventh registers and applying

U_{CR^+} on the last two, we obtain

$$\begin{aligned}
&\frac{1}{\sqrt{N\|\theta\|_1^2}}\Big(\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ s_{k_1}s_{k_2}X_{ik_1}X_{ik_2}>0}}\sqrt{|\theta_{k_1}\theta_{k_2}|}\,|Z_{i,k_1,k_2}\rangle\,(\sqrt{s_{k_1}s_{k_2}X_{ik_1}X_{ik_2}}\,|1\rangle+\sqrt{1-s_{k_1}s_{k_2}X_{ik_1}X_{ik_2}}\,|0\rangle)\\
&+\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ s_{k_1}s_{k_2}X_{ik_1}X_{ik_2}\le 0}}\sqrt{|\theta_{k_1}\theta_{k_2}|}\,|Z_{i,k_1,k_2}\rangle\,|0\rangle\Big)\\
=&\frac{1}{\sqrt{N\|\theta\|_1^2}}\Big(\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ \theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}>0}}\sqrt{\theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}}\,|Z_{i,k_1,k_2}\rangle\,|1\rangle\\
&+\Big(\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ \theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}>0}}\sqrt{|\theta_{k_1}\theta_{k_2}|(1-s_{k_1}s_{k_2}X_{ik_1}X_{ik_2})}\,|Z_{i,k_1,k_2}\rangle+\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ \theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}\le 0}}\sqrt{|\theta_{k_1}\theta_{k_2}|}\,|Z_{i,k_1,k_2}\rangle\Big)\,|0\rangle\Big)\\
=&\sqrt{a_+}\,|\phi_1\rangle\,|1\rangle+\sqrt{1-a_+}\,|\phi_0\rangle\,|0\rangle,\quad \text{for } a_+=\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ \theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}>0}}\frac{\theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}}{N\|\theta\|_1^2},
\end{aligned}$$

where $s_{k_1}=\operatorname{sign}(\theta_{k_1})$, $s_{k_2}=\operatorname{sign}(\theta_{k_2})$, $Z_{i,k_1,k_2}=(i,k_1,s_{k_1},k_2,s_{k_2},s_{k_1}s_{k_2}X_{ik_1}X_{ik_2})$.

By applying Theorem 2.3, with failure probability at most 1/1000, we can estimate a_+ with additive error $\beta/(6\|\theta\|_1^2)$ using $\mathcal{O}(\|\theta\|_1^2/\beta)$ applications of O_X, $O_X^\dagger$, U_θ, $U_\theta^\dagger$, and $\tilde{\mathcal{O}}(\|\theta\|_1^2/\beta)$ elementary gates. Note that our algorithm knows $\|\theta\|_1$ because it is stored in the root of KP_θ. Similarly we estimate

$$a_-=-\sum_{\substack{i\in[N]-1,k_1,k_2\in[d]-1\\ \theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}<0}}\frac{\theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}}{N\|\theta\|_1^2}$$

with the same additive error. Hence we can estimate

$$\frac{1}{N}\sum_{i\in[N]-1,k_1,k_2\in[d]-1}\theta_{k_1}\theta_{k_2}X_{ik_1}X_{ik_2}=\|\theta\|_1^2\cdot(a_+-a_-)$$

with additive error $\beta/3$.

For the second and third terms of the right-hand side of Equation (3.2), we use a similar strategy to estimate each with additive error $\beta/3$, using $\mathcal{O}(1/\beta)$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, U_θ, $U_\theta^\dagger$, and $\tilde{\mathcal{O}}(1/\beta)$ elementary gates.

Combining the previous estimations and the fact that we can implement U_θ by $\tilde{\mathcal{O}}(1)$ queries to O_{KP_θ}, $O_{KP_\theta}^\dagger$ and $\|\theta\|_1\le 1$, with failure probability at most 1/100, we can estimate $L_S(\theta)$ with additive error β by using $\tilde{\mathcal{O}}(1/\beta)$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates. By repeating the procedure $\Theta(\log(1/\delta))$ times and taking the median among the outputs, we can decrease the failure probability from at most 1/100 to at most δ. □

If we have multiple vectors $\theta^0,\dots,\theta^{m-1}$, then we can apply the previous theorem conditioned on the index of the vector we care about:

Corollary 3.10. *Let $\theta^0,\theta^1,\dots,\theta^{m-1}\in B_1^d$, and $\beta,\delta>0$. Suppose for all $h\in[m]-1$, we have a KP-tree KP_{θ^h} of vector θ^h and can make quantum queries to $O_{KP_\theta}:|h,\ell,k\rangle\,|0\rangle\to$*

$|h,\ell,k\rangle|KP_{\theta^h}(\ell,k)\rangle$. Then we can implement $\tilde{U}_{L_S}:|h\rangle|0\rangle\to|h\rangle|\Lambda\rangle$ such that for all $h\in[m]-1$, after measuring the state $|\Lambda\rangle$, with probability $\geq 1-\delta$ the first register λ of the outcome will satisfy $|\lambda-L_S(\theta^h)|\leq\beta$, by using $\tilde{\mathcal{O}}(\frac{\log(1/\delta)}{\beta})$ applications of O_X, $O_X^\dagger$, O_y, $O_y^\dagger$, O_{KP_θ}, $O_{KP_\theta}^\dagger$, and elementary gates.

3.4 Quantum algorithms for Lasso

3.4.1 Quantum algorithms for Lasso with respect to S

In this subsection, we will show how to find an approximate minimizer for Lasso with respect to a given sample set S. The following algorithm simply applies the Frank-Wolfe algorithm to find an ε-minimizer for Lasso with respect to the sample set S given C, a guess for the curvature constant C_{L_S} (which our algorithm does not know in advance). Note that to find an $s\in B_1^d$ such that $\langle s,\nabla L_S(\theta^t)\rangle\leq\min_{s'\in\mathcal{X}}\langle s',\nabla L_S(\theta^t)\rangle+\tau_t C_{L_S}/4$, it suffices to only check $s\in\{\pm e_0,\ldots,\pm e_{d-1}\}$ because the domain is B_1^d and ∇L_S is a linear function in θ. Also, by Theorem 3.6, the curvature constant C_{L_S} of loss function L_S is at most 8 because (x_i,y_i) is in $[-1,1]^d\times[-1,1]$ for all $i\in[N]-1$.

input : a positive value C; additive error ε;
Let θ^0 be the d-dimensional all-zero vector;
Let $T=6\cdot\lceil\frac{C}{\varepsilon}\rceil$;
for $t\leftarrow 0$ **to** T **do**
  $\tau_t=\frac{2}{t+2}$;
  Let $s\in\{\pm e_0,\ldots,\pm e_{d-1}\}$ be such that
  $\langle\nabla L_S(\theta^t),s\rangle\leq\min_{j'\in[d]-1}-|\nabla_{j'}L_S(\theta^t)|+\frac{C}{8t+16}$;
  $\theta^{t+1}=(1-\tau_t)\theta^t+\tau_t s$;
end
output: θ^T;

Algorithm 2: The algorithm for Lasso with a guess C for the value of the curvature constant

It is worth mentioning that Algorithm 2 also outputs an ε-minimizer if its input C equals the curvature constant C_{L_S} approximately instead of exactly. For example, suppose we only know that the curvature constant C_{L_S} is between C and $2C$, where C is the input in Algorithm 2. Then the output of Algorithm 2 is still an ε-minimizer. We can see this by first observing that the error we are allowed to make for the linear subproblem in iteration t is $\frac{C_{L_S}}{4t+8}\geq\frac{C}{8t+16}$, and hence by Theorem 3.7, after $T=6\cdot\lceil\frac{C}{\varepsilon}\rceil$ iterations, the output θ^T is a $\frac{3C}{(T+2)}=\frac{3C}{6\cdot\lceil C/\varepsilon\rceil+2}$-minimizer for L_S. Because $\frac{3C}{6\cdot\lceil C/\varepsilon\rceil+2}\leq\varepsilon$, the output θ^T is therefore an ε-minimizer.

In the Lasso case, we do not know how to find a positive number C such that $C_{L_S}\in[C,2C]$, but we know $C_{L_S}\leq 8$ by Theorem 3.6. Hence we can try different intervals of

possible values for C_{L_S}: we apply Algorithm 2 with different input $C = 8, 4, 2, \ldots, 2^{-\lceil \log(1/\varepsilon) \rceil}$, and then we collect all outputs of Algorithm 2 with those different inputs, as candidates. After that, we compute the objective values of all those candidates, and output the one with minimum objective value. If $C_{L_S} \in (\varepsilon, 8]$, then at least one of the values we tried for C will be within a factor of 2 of the actual curvature constant C_{L_S}. Hence one of our candidates is an ε-minimizer.

However, we also need to deal with the case that $C_{L_S} \leq \varepsilon$. In this case, we consider the "one-step" version of the Frank-Wolfe algorithm, where the number of iterations is 1. But now we do not estimate $\langle \nabla L_S(\theta^t), s \rangle$ anymore (i.e., we do not solve linear subproblems anymore). We find that the only possible directions are the vertices of the ℓ_1-ball, and θ^0 is the all-zero vector, implying that θ^1, the output of one-step Frank-Wolfe, must be in $I = \{\pm e_0/3, \ldots, \pm e_{d-1}/3\}$ by the update rule of Frank-Wolfe. Besides, $C_{L_S} \leq \varepsilon$ implies that θ^1 is a $\frac{3C_{L_S}}{1+2} \leq \varepsilon$-minimizer for Lasso. Hence we simply output a $v = \arg\min_{v' \in I} L_S(v')$ if $C_{L_S} \leq \varepsilon$.

Combining the above arguments gives the following algorithm:

input : ε;
Let $v \in \{\pm e_0/3, \ldots, \pm e_{d-1}/3\}$ be such that $L_S(v) - \min_{j \in [d]-1} L_S(\pm e_j/3) \leq \varepsilon/10$;
Let candidate set $A = \{v\}$;
for $C \leftarrow 8, 4, 2, 1, \frac{1}{2}, \ldots, 2^{-\lceil \log(1/\varepsilon) \rceil - 1}$ **do**
  RUN Algorithm 2 with inputs C and $\varepsilon/10$;
  ADD the output of Algorithm 2 to A;
end
output: $\arg\min_{w \in A} L_S(w)$;

Algorithm 3: The algorithm for Lasso

Theorem 3.11. *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be the given sample set stored in QROM. For each $\varepsilon \in (0, 0.5)$, there exists a bounded-error quantum algorithm that finds an ε-minimizer for Lasso with respect to sample set S using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ QRAM and classical space.*

Proof. We will implement Algorithm 3 in $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ QRAM space. Below we analyze its different components.

Analysis of Algorithm 2. We first show that we can implement Algorithm 2 in $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time. Because $C_{L_S} \leq 8$ (Theorem 3.6), the number of iterations for Algorithm 2 with input $C = C_{L_S}$ is at most $6 \cdot \lceil \frac{8}{\varepsilon} \rceil$. However, as we mentioned above, we don't know how large C_{L_S} is exactly, so we try all possible inputs (of Algorithm 2) in Algorithm 3. Note that for every input $C \in \{8, 4, 2, 1, \frac{1}{2}, \ldots, 2^{-\lceil \log(1/\varepsilon) \rceil - 1}\}$ and for every number of iterations $t \in \{1, \ldots, 6 \cdot \lceil \frac{C}{\varepsilon} \rceil\}$, $\frac{C}{4t+8}$ is at least $\frac{\varepsilon}{10}$, so it suffices to ensure that in each iteration in each of our runs of Algorithm 2, the additive error for the approximate linear subproblem is $\leq \frac{\varepsilon}{10}$.

Suppose we have KP_{θ^t} for each iteration t of Algorithm 2, and suppose we can make queries to $O_{KP_{\theta^t}}$, then by Theorem 3.8, one can implement $\tilde{U}_{\nabla L_S} : |j\rangle|0\rangle \to |j\rangle|\Lambda\rangle$ such that for all $j \in [d]-1$, after measuring the state $|\Lambda\rangle$, with probability $\geq 1-\frac{\varepsilon^2}{2d\cdot 10^{20}\cdot \log^6(1/\varepsilon)}$ the first register λ of the measurement outcome will satisfy $|\lambda - \nabla_j L_S(\theta)| \leq \frac{\varepsilon}{20}$, by using $\tilde{\mathcal{O}}(\frac{\log(d/\varepsilon)}{\varepsilon})$ time and queries to $O_{KP_{\theta^t}}$, $O^\dagger_{KP_{\theta^t}}$. Then by Theorem 2.5, with failure probability at most $\frac{\varepsilon}{10000\log(1/\varepsilon)}$, one can find $s \in \{\pm e_0, \ldots, \pm e_{d-1}\}$ such that $\langle \nabla L_S(\theta^t), s\rangle \leq \min_{j'\in[d]-1} -|\nabla_{j'} L_S(\theta^t)| + 2\cdot\frac{\varepsilon}{20}$, by using $\tilde{\mathcal{O}}(\sqrt{d}\cdot\log(1/\varepsilon))$ applications of $\tilde{U}_{\nabla L_S}$ and $\tilde{U}^\dagger_{\nabla L_S}$, and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates.

For each iteration t in Algorithm 2, we also maintain KP_{θ^t} and hence we can make quantum queries to $O_{KP_{\theta^t}}$. The cost for constructing KP_{θ^0} and the cost for updating KP_{θ^t} to $KP_{\theta^{t+1}}$ is $\tilde{\mathcal{O}}(1)$ for both time and space by Theorem 2.10. Moreover, the total number of iterations T is at most $6\cdot\lceil\frac{8}{\varepsilon}\rceil$ in Algorithm 2 because $C_{L_S} \leq 8$, and hence the space cost for maintaining KP_{θ^t} and implementing $O_{KP_{\theta^t}}$ is $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ bits. Hence we can implement Algorithm 2 with failure probability at most $\lceil\frac{8}{\varepsilon}\rceil\cdot\frac{6\varepsilon}{10000\log(1/\varepsilon)}$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ bits of QRAM and classical space.

Analysis of Algorithm 3. Now we show how to implement Algorithm 3 with failure probability at most $1/10$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time. By Corollary 3.10, one can implement $\tilde{U}_{L_S} : |j\rangle|0\rangle \to |j\rangle|\Lambda\rangle$ such that for all $j \in [d]-1$, after measuring the state $|\Lambda\rangle$, with failure probability at most $\frac{1}{2d\cdot 10^{16}}$ the first register λ of the outcome will satisfy $|\lambda - L_S(e_j/3)| \leq \varepsilon/20$ using $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ time. Then by Theorem 2.5, with failure probability at most $0.0001 + 1000\cdot\log(1000)\sqrt{\frac{2d}{2d\cdot 10^{16}}} \leq \frac{2}{1000}$ we can find $v \in \{\pm e_0/3, \ldots, \pm e_{d-1}/3\}$ such that $L_S(v) - \min_{j\in[d]-1} L_S(\pm e_j/3) \leq 2\cdot\varepsilon/20 = \varepsilon/10$ by using $\tilde{\mathcal{O}}(\sqrt{d})$ applications of $\tilde{U}_{L_S}$ and $\tilde{U}^\dagger_{L_S}$ and $\tilde{\mathcal{O}}(\sqrt{d})$ elementary gates, and hence $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon})$ time.

Because Algorithm 3 runs Algorithm 2 $\lceil\log(1/\varepsilon)\rceil$ times and each run fails with probability at most $\lceil\frac{8}{\varepsilon}\rceil\cdot\frac{6\varepsilon}{10000\log(1/\varepsilon)}$, the candidate set A, with failure probability $\lceil\frac{8}{\varepsilon}\rceil\cdot\frac{6\varepsilon}{10000\log(1/\varepsilon)}\cdot\lceil\log(1/\varepsilon)\rceil + \frac{2}{1000} \leq \frac{1}{20}$, contains an $\frac{\varepsilon}{10}$-minimizer. To output $\arg\min_{w\in A} L_S(w)$, we use Theorem 3.9 to evaluate $L_S(w)$ for all $w \in A$ with additive error $\frac{\varepsilon}{10}$ with failure probability at most $\frac{1}{40\log(1/\varepsilon)}$, and hence we find an $\varepsilon/10$-minimizer among A with probability at least $1-1/20-\lceil\log(1/\varepsilon)\rceil\cdot\frac{1}{40\log(1/\varepsilon)} \geq 0.9$. Because the candidate set A contains an $\frac{\varepsilon}{10}$-minimizer for Lasso, the $\frac{\varepsilon}{10}$-minimizer among A is therefore an ε-minimizer for Lasso. The QRAM and classical space cost for each run is at most $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ because the space cost for Algorithm 2 is $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$. Hence the total cost for implementing Algorithm 3 is $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ time and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ bits of QRAM and classical space. □

3.4.2 Quantum algorithms for Lasso with respect to $\mathcal{D}$

In the previous subsection, we showed that we can find an ε-minimizer for Lasso with respect to sample set S. Here we show how we can find an ε-minimizer for Lasso with

respect to distribution $\mathcal{D}$. First sample a set S of $N = \tilde{\mathcal{O}}((\log d)/\varepsilon^2)$ i.i.d. samples from $\mathcal{D}$, which is the input that will be stored in QROM, and then find an $\varepsilon/2$-minimizer for Lasso with respect to S by Theorem 3.11. By Theorem 3.3, with high probability, an $\varepsilon/2$-minimizer for Lasso with respect to S will be an ε-minimizer for Lasso with respect to distribution $\mathcal{D}$. Hence we obtain the following corollary:

Corollary 3.12. *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be the given sample set, sampled i.i.d. from $\mathcal{D}$. For arbitrary $\varepsilon > 0$, if $N = \tilde{\mathcal{O}}(\frac{\log d}{\varepsilon^2})$, then there exists a bounded-error quantum algorithm that finds an ε-minimizer for Lasso with respect to distribution $\mathcal{D}$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^2})$ queries to O_X, O_y and elementary gates, and using $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ space (QRAM and classical bits).*

We can also use Theorem 2.12 to avoid the usage of QRAM in the above corollary with $\tilde{\mathcal{O}}(1/\varepsilon)$ extra overhead.

Corollary 3.13. *Let $S = \{(x_i, y_i)\}_{i=0}^{N-1}$ be the given sample set, sampled i.i.d. from $\mathcal{D}$. For arbitrary $\varepsilon > 0$, if $N = \tilde{\mathcal{O}}(\frac{\log d}{\varepsilon^2})$, then there exists a bounded-error quantum algorithm that finds an ε-minimizer for Lasso with respect to distribution $\mathcal{D}$ using $\tilde{\mathcal{O}}(\frac{\sqrt{d}}{\varepsilon^3})$ queries to O_X, O_y and elementary gates, and using $\tilde{\mathcal{O}}(\frac{1}{\varepsilon})$ classical bits.*

3.5 Open problems

We mention a few directions for future work:

- While we gave a quantum speedup for Lasso with respect to d-dependence (we also show this dimension-dependency is tight in Chapter 6), the ε-dependence still remains the same as classical. So a natural question is if we can also show a quantum speedup in terms of ε-dependency. At the very beginning we felt this might be possible by using a version of accelerated gradient descent [Nes83] with $\mathcal{O}(1/\sqrt{\varepsilon})$ iterations instead of Frank-Wolfe's $\mathcal{O}(1/\varepsilon)$ iterations. However, we soon realized that to leverage accelerated gradient descent effectively, precise gradient calculations are necessary (thus negating any quantum speedup). Otherwise, the accumulation of errors would result in a convergence rate of $\mathcal{O}(1/T)$ rather than $\mathcal{O}(1/T^2)$ (where T represents the number of iterations), implying that the algorithm still requires $\mathcal{O}(1/\varepsilon)$ iterations, similar to the Frank-Wolfe approach.

- Similar question for Ridge: can we provide a quantum algorithm with better ε-dependence while the d-dependency remains the same? If we don't ask for the linear dimension-dependency, we can actually achieve a quantum algorithm for Ridge with ε-dependency $\tilde{\mathcal{O}}(1/\varepsilon)$ by combining the state-tomography result (Corollary 4.7 in Chapter 4) and the result by Chakraborty, Morolia, and Peduri [CMP23]. However, the running time of the algorithm stated above will end up being $\tilde{\mathcal{O}}(d^{1.5}/\varepsilon)$ (assuming the condition number of the matrix X is constant) whose d-dependency is significantly worse than the best known classical algorithm. The most interesting outcome would be a quantum algorithm for Ridge with better ε-dependence

than the optimal classical complexity of $\tilde{\Theta}(d/\varepsilon^2)$ and with the same d-dependency; currently we do not know of any reasonable quantum speedup for Ridge.

- Can we speed up some other methods for (smooth) convex optimization? In particular, can we find a classical iterative method where quantum algorithms can significantly reduce the number of iterations, rather than just the cost per iteration as we did here?

- There are many connections between Lasso and Support Vector Machines [Jag14], and there are recent quantum algorithms for optimizing SVMs [RML14; SK19; SA19; AH20; SPA21]. We would like to understand this connection better, especially about how an ε-minimizer for Lasso can be converted to an ε-optimizer for SVM.

CHAPTER 4

Quantum algorithms for approximating the top eigenvectors

4.1 Introduction

Arguably, the most important property of a diagonalizable $d \times d$ matrix A is its largest eigenvalue λ_1, with an associated eigenvector v_1. One can consider the top eigenvector v_1 as the most important "direction" in which the matrix A operates. The ability to efficiently find v_1 is an important tool in many applications, for instance, in the PageRank algorithm of Google's search engine, as a starting point for Principal Component Analysis (for clustering or dimensionality-reduction), for Fisher discriminant analysis, or in continuous optimization problems where sometimes the best thing to do is to move the current point in the direction of the top eigenvector of an associated matrix.

One way to find the top eigenvector of A is to diagonalize the whole matrix. Theoretically this takes matrix multiplication time: $\mathcal{O}(d^\omega)$ where $\omega \in [2, 2.37\ldots)$ is the still-unknown matrix multiplication exponent. In practice often Gaussian elimination (which takes time $\mathcal{O}(d^3)$) is actually faster, unless d is enormous. Diagonalization gives us not only the top eigenvector but a complete orthonormal set of d eigenvectors. However, this is overkilled if we only care about finding the top eigenvector, or the top-q eigenvectors for some $q \ll d$, and better methods exist in this case (see e.g. [Par87] for a whole book about this).

The power method for approximating a top eigenvector A quite efficient method for (approximately) finding the top eigenvector is the iterative "power method". This uses simple matrix-vector multiplications instead of any kind of matrix decompositions, and works as follows. We start with a random unit vector w_0 (say with i.i.d. Gaussian entries with variance $1/d$). This is a linear combination $\sum_{i=1}^{d} \alpha_i v_i$ of the d unit eigenvectors $v_1, \ldots, v_d$ of the Hermitian matrix A, with coefficients of magnitude typically around $1/\sqrt{d}$. Then we apply A to this vector some K times, computing $w_1 = Aw_0$, $w_2 = Aw_1$, etc., up to $w_K = A^K w_0$. This has the effect of multiplying each coefficient α_i

with the powered eigenvalue λ_i^K. If there is some "gap" between the first two eigenvalues (say $|\lambda_1|-|\lambda_2| \geq \gamma > 0$, or $|\lambda_1/\lambda_2| > 1$), then the relative weight of the coefficient of v_1 will start to dominate all the other coefficients even already for small K, and the renormalization of the final vector $w_K = A^K w_0$ will be close to v_1, up to global phase. Specifically, if A has bounded operator norm and eigenvalue gap γ, then $K = \mathcal{O}\left(\log(d)/\gamma\right)$ iterations suffice to approximate v_1 up to $1/\mathrm{poly}(d)$ ℓ_2-error (see e.g. [GL13, Section 8.2.1] for details).

The cost of this algorithm is dominated by the K matrix-vector multiplications, each of which costs $\widetilde{\mathcal{O}}\left(d^2\right)$ time classically.[1] Hence if the eigenvalue gap γ is not too small, say constant or at least $1/\mathrm{polylog}(d)$, then the power method takes $\tilde{\mathcal{O}}(d^2)$ time to approximate a top eigenvector.Unsurprisingly, as we show in Section 7.3 in Chapter 7, $\Omega(d^2)$ queries to the entries of A are also *necessary* for classical algorithms for this task.

4.1.1 Main results and high-level intuition

Our main results in this chapter are two faster quantum algorithms for (approximately) finding the top eigenvector of a Hermitian matrix A.[2] Both quantum algorithms run a version of the classical power method (also known as the noisy power method) that is robust to certain benign kinds of errors, where we implement each matrix-vector multiplication with small and well-behaved (sub-Gaussian) error in different ways for the two algorithms.

Approximate matrix-vector multiplication by Gaussian phase estimation. Our first algorithm uses that each entry of the vector Aw is an inner product between a row of A and the column-vector w. Because such an inner product is the sum of d numbers, we may hope to approximate it well via some version of amplitude estimation or quantum counting, using roughly $\sqrt{d}$ time per entry and $d^{1.5}$ time for all d entries of Aw together. This approach is easier said than done, because basic quantum-counting subroutines produce small errors in the approximation of each entry, and those errors might add up to a large ℓ_2-error in the d-dimensional vector Aw as a whole. To mitigate this issue we develop a "Gaussian phase estimation" procedure that can estimate one entry of Aw with a complexity that is similar to standard phase estimation, but with well-behaved *sub-Gaussian* error. These well-behaved errors in individual entries typically still add up to a large ℓ_2-error for the vector Aw as a whole. However, with very high probability the error remains small in one or a few fixed directions—including the direction of the unknown top eigenvector. To speed up the computation of each entry, we split the rows into "small" and "large" entries, and handle them separately.

[1] It costs less if A is sparse: if A has s nonzero entries, given in some easily-accessible way like lists of nonzero entries for each row, then it costs $\widetilde{\mathcal{O}}(s)$ time.

[2] If the matrix A is non-Hermitian, then we could instead use the Hermitian matrix $A' = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$ for finding the largest (left or right) singular value of A, replacing the eigenvalue gap with the singular value gap.

This divide-and-conquer approach uses $\widetilde{\mathcal{O}}\left(d^{1.75}\right)$ time in total for the d entries of Aw (Theorem 4.10). This is asymptotically worse than our second algorithm (described below), but we still feel it merits inclusion because it uses an intuitive entry-by-entry approach, it has a slightly better dependence on the precision than our second algorithm, and most importantly our new technique of Gaussian phase estimation may find applications elsewhere.

Approximate matrix-vector multiplication by unbiased pure state tomography. Our second algorithm is faster and more "holistic"; it does not approximate the matrix-vector product Aw entry-by-entry. Instead it implements a block-encoding of the matrix A, uses that to (approximately) produce Aw as a $\log(d)$-qubit state, and then applies a subtle tomography procedure to obtain a classical estimate of the vector Aw with small ℓ_2-error.[3] The introduced new tomography procedure incorporates ideas from [KP20] and [ACG+23]. The procedure first estimates the magnitudes of an unknown quantum state $|\psi\rangle$'s entries to get a reference state $|\bar{\psi}\rangle$ whose entries are just (approximate) magnitudes of $|\psi\rangle$. After that, it prepares and measures $(|+\rangle|\psi\rangle+|-\rangle|\bar{\psi}\rangle)/\sqrt{2}$ in the computational basis. By $\mathcal{O}(d)$ measurement outcomes, we can recover the unknown state $|\psi\rangle$ with both bounded ℓ_2-norm error and bounded variance.[4] This procedure matches the essentially optimal query complexity stated in [ACG+23], but improves upon both their time complexities. Doing tomography at the end of each iteration of the power method is somewhat expensive (since we need $d^{0.5+o(1)}$ time to produce $|Aw\rangle$, and extra $\mathcal{O}(d)$ overhead for state tomography), but still leaves us with a time complexity of only $d^{1.5+o(1)}$ (Corollary 4.16 with $q=1$ and $s=d$), which turns out to be a near-optimal quantum speed-up over classical, as our lower bounds (discussed in Chapter 7) imply.

Quantum noisy power method. In both of our algorithms, the vector resulting in each iteration from our approximate matrix-vector multiplication will have small errors compared to the perfect matrix-vector product. In the basic power method we cannot tolerate small errors in adversarial directions: if w_0 has roughly $1/\sqrt{d}$ overlap with the top eigenvector v_1, and we compute Aw_0 with ℓ_2-error $> \lambda_1/\sqrt{d}$, then our approximation to the vector Aw_0 may have *no overlap with* v_1 *at all anymore*! If this happens, if we lose the initially-small overlap with the top eigenvector, then the power method fails to converge to v_1 even if all later matrix-vector multiplications are implemented perfectly. Fortunately, Hardt and Price [HP14] have already showed that the power method is robust against errors in the matrix-vector computations if they

[3] In fact our second algorithm first implements a block-encoding of the rank-1 projector $\Pi = v_1 v_1^\dagger$ using $\widetilde{\mathcal{O}}\left(1/\gamma\right)$ applications of an approximate block-encoding of the matrix A (by applying QSVT [GSL+19, Theorem 31]), and then applies our state tomography algorithm to obtain a classical estimate of the vector Πw, which is proportional to v_1. This increases the spectral gap from γ to $\Theta(1)$ and hence further improves our γ-dependency.

[4] The measurement outcome of $(|+\rangle|\psi\rangle+|-\rangle|\bar{\psi}\rangle)/\sqrt{2}$ only gives us the information of the real part of the unknown state, but we can similarly prepare and measure the state $(|+\rangle|\psi\rangle+i|-\rangle|\bar{\psi}\rangle)/\sqrt{2}$ in the computational basis to read out the information of the imaginary part. See Section 4.3.2 for more details.

are sufficiently well-behaved (in particular, entrywise sub-Gaussian errors with moderate variance suffice). Much of the technical effort in our quantum subroutines for matrix-vector multiplication is to ensure that the errors in the resulting vector are indeed sufficiently well-behaved not to break the noisy power method.[5]

Finding the top-q eigenvectors. Going beyond just the top eigenvector, the ability to find the top-q eigenvectors (with $q \ll d$) is crucial for many applications in machine learning and data analysis, such as spectral clustering, Principal Component Analysis, low-rank approximation of A, and dimensionality reduction of d-dimensional data vectors w (e.g., projecting the data vectors onto the span of the top-q eigenvectors of the covariance matrix $A = \mathbb{E}[ww^T]$). In most cases it suffices to (approximately) find the subspace spanned by the top-q eigenvectors of A rather than the individual top-q eigenvectors $v_1, \ldots, v_q$, which is fortunate because distinguishing $v_1, \ldots, v_q$ can be quite expensive if the corresponding eigenvalues $\lambda_1, \ldots, \lambda_q$ are close together.

The noisy power method can also (approximately) find the subspace spanned by the top-q eigenvectors of A, assuming some known gap γ between the qth and $(q+1)$th eigenvalue. Using our knowledge of the gap, we give an algorithm to approximate λ_q (Corollary 4.14). Knowing λ_q and this gap (at least approximately), we then show how a block-encoding of A can be efficiently converted using quantum singular-value transformation [GSL+19] into a block-encoding of the projector Π that projects onto the subspace spanned by the top-q eigenvectors. In Section 4.4.3 we give a new almost optimal process-tomography algorithm for recovering the projector Π, assuming only the ability to apply (controlled) reflections $2\Pi - I$ about the rank-q subspace that we are trying to recover (Theorem 4.11). This algorithm, applied to Π, gives us the subspace corresponding to the top-q eigenvectors of A. For constant eigenvalue gap and desired precision, it uses time $qd^{1.5+o(1)}$. In fact, what we above called our "second algorithm" for finding the top eigenvector is just the special case $q = 1$. In the case where A is s-sparse (meaning each row and column of A has $\leq s$ nonzero entries) and we have sparse-query-access to it, the time complexity becomes $q\sqrt{s}d^{1+o(1)}$ (Corollary 4.16; this result implies the claim of the previous sentence by setting $s = d$). If the pairwise spacing between the first q eigenvalues is at least $\Omega(1/q)$, then we can also (approximately) *find* each of the eigenvectors $v_1, \ldots, v_q$ individually, at the expensive of poly(q) more time.

[5] For simplicity we assume here that γ (or some sufficiently good approximation of it) is known to our algorithm. However, because we can efficiently approximate $|\lambda_1|$ (by [AGG+20, Lemma 50], one can estimate λ_1 with additive error $\gamma/4$ using $\tilde{O}(d^{1.5}/\gamma)$ time) and verify whether the output of our algorithm is approximately an eigenvector for this eigenvalue by one approximate matrix-vector computation, we can actually try exponentially decreasing guesses for γ until the algorithm returns an approximate top eigenvector.

It should be noted that our algorithm has polynomial dependence on the precision ε, namely linear in $1/\varepsilon$, which is worse than the $\log(1/\varepsilon)$ dependence of the classical power method with perfect matrix-vector calculations. This is the price we pay for our polynomial speed-up in terms of the dimension. For many applications, the precision need not be extremely small and there our polynomial dependence on ε would be an acceptable price to pay.

As a byproduct of this algorithm we also obtain a qualitatively improved tomography procedure that works assuming the ability to reflect around the state that we want to estimate, but does not need the stronger assumption of being able to prepare that state.

Roadmap In Section 4.2 we introduce a new method to do the phase estimation with the guarantee that the error is sub-Gaussian and in Section 4.3, we introduce a new time-efficient quantum state tomography procedure. Both above-mentioned tools can be used to compute approximate matrix-vector products, which will be elaborated in Section 4.4. In Section 4.4 we further explain how to use approximate matrix-vector products to approximate the top eigenvector.

Remark In the whole Chapter 4, when we use the notation for KP-tree, it always means the KP-tree with respect to ℓ_2-norm (see Section 2.4.2). In Section 4.2, when we use the notation ρ_s and when we mention discrete Gaussian distribution, it is always over $\mathbb{R}$.

4.2 Quantum Gaussian phase estimator

Before we explain our quantum noisy power method, we introduce another tool which we call the "quantum Gaussian phase estimator". Its aim is to do phase estimation with (approximately) Gaussian error on the estimate. The high-level idea of this estimator is to replace the initial uniform superposition in the algorithm of phase estimation [Kit95; CEM+98] by a discrete Gaussian quantum state, with standard deviation s; then the distribution of the error $\tilde{a} - a$ between the amplitude a and the estimator $\tilde{a}$ produced by the quantum Gaussian amplitude estimator, is also a discrete Gaussian distribution, now with standard deviation $1/s$. Since the latter distribution is sub-Gaussian with parameter $1/s$, with probability at least $1-\delta$ the output is at most $\sqrt{\log(1/\delta)}/s$ away from a.

Recall that ρ_s is the pdf for the Gaussian with standard deviation s, defined in Section 2.7.1. We first use Corollary 2.25 and the discussion above Corollary 2.25 to show that the truncated discrete Gaussian state is close to the modular discrete Gaussian state when N, s are both reasonably large.

Theorem 4.1. *Let $\delta \in (0, 0.1]$, $s \geq 8\sqrt{2\log(\frac{1}{\delta})}$, $N \geq 16s\sqrt{2\ln(\frac{1}{\delta})}$ even number, and $t \in [-\frac{N}{8}, \frac{N}{8}]$. Let $f : \mathbb{R} \to \mathbb{C}$ be an arbitrary phase function such that $|f(x)| = 1$ for every $x \in \mathbb{R}$*

and

$$|G\rangle = \frac{1}{\sqrt{G}} \sum_{x\in\mathbb{Z}} f(x+t)\rho_s(x+t)\,|x\rangle\,,$$
$$|G^{tr}\rangle = \frac{1}{\sqrt{G_{tr}}} \sum_{x\in\{-\frac{N}{2}\ldots,0,\ldots,\frac{N}{2}-1\}} f(x+t)\rho_s(x+t)\,|x\rangle\,,$$
$$|G^{mod}\rangle = \frac{1}{\sqrt{G_{mod}}} \sum_{x\in\mathbb{Z}} f(x+t)\rho_s(x+t)\,|(x+\frac{N}{2}\ mod\ N)-\frac{N}{2}\rangle\,,$$

where G, G_{mod}, G_{tr} are normalizing factors. Then $|G^{tr}\rangle, |G^{mod}\rangle, |G\rangle$ are 9δ-close to each other.[6]

Proof. Let $[[\pm a]]$ denote the set $\{-\lceil a\rceil\ldots,0,\ldots,\lceil a\rceil-1\}$. We first show $|G^{tr}\rangle$ is close to $|G\rangle$. Define $|\widetilde{G^{tr}}\rangle = \sqrt{\frac{G_{tr}}{G}}\,|G^{tr}\rangle$, which has ℓ_2-norm $\sqrt{\frac{G_{tr}}{G}} \le 1$. We can see

$$\| \,|\widetilde{G^{tr}}\rangle - |G\rangle\,\|^2 = \frac{1}{G} \sum_{x\in\mathbb{Z}\setminus[[\pm\frac{N}{2}]]} \rho_s^2(x+t) \le \frac{1}{G} \sum_{x\in\mathbb{Z}\setminus\{x:|x+t|\le N/4\}} \rho_s^2(x+t) \le \delta^2,$$

where the last equality holds because $\rho_s^2 = \rho_{s/\sqrt{2}}$ and $\mathcal{D}_{\mathbb{Z}+t,s/\sqrt{2}}$ is δ^2-sub-Gaussian with parameter $s/\sqrt{2}$ by Theorem 2.24 (note $s/\sqrt{2} \ge 8\sqrt{\log(1/\delta)} \ge \sqrt{\log(12/\delta^2)/\pi}$) and because of the first property of sub-Gaussians: $\Pr[|x+t| > N/4] \le 2\exp(\delta^2)\exp(-(N/4)^2/(2\cdot s^2/2)) \le \delta^2$. Note that $1 \ge \| \,|\widetilde{G^{tr}}\rangle\,\| \ge \| \,|G\rangle\,\| - \| \,|\widetilde{G^{tr}}\rangle - |G\rangle\,\| \ge 1-\delta$ and hence $\| \,|\widetilde{G^{tr}}\rangle - |G^{tr}\rangle\,\| = \|(\sqrt{\frac{G_{tr}}{G}}-1)\,|G^{tr}\rangle\,\| \le \delta$. Therefore, we obtain

$$\| \,|G^{tr}\rangle - |G\rangle\,\| \le \| \,|\widetilde{G^{tr}}\rangle - |G^{tr}\rangle\,\| + \| \,|\widetilde{G^{tr}}\rangle - |G\rangle\,\| \le 2\delta.$$

To show $|G^{mod}\rangle$ is close to $|G\rangle$, let us similarly define $|\widetilde{G^{mod}}\rangle = \sqrt{\frac{G_{mod}}{G}}\,|G^{mod}\rangle$. We can see

$$\| \,|\widetilde{G^{mod}}\rangle - |G\rangle\,\|^2 \le \frac{1}{G} \sum_{x\in[[\pm\frac{N}{2}]]} \Big(\sum_{y\in\mathbb{Z}\setminus\{0\}} \rho_s(x+t+Ny)\Big)^2 + \frac{1}{G} \sum_{x\in\mathbb{Z}\setminus[[\pm\frac{N}{2}]]} \Big(\rho_s(x+t)\Big)^2.$$

To upper bound the first term in the RHS above, we split the domain of x into three disjoint parts: $D_1 = \{x \in [[\pm\frac{N}{2}]] : |x+t| \le N/4\}$, $D_2 = \{x \in [[\pm\frac{N}{2}]] : N/2 \ge |x+t| > N/4\}$, and $D_3 = \{x \in [[\pm\frac{N}{2}]] : |x+t| > N/2\}$.

When $x \in D_1$, we can see $\sum_{y\in\mathbb{Z}\setminus\{0\}} \rho_s(x+t+Ny) \le \rho_s(x+t)\cdot 2\sum_{y=1}^{\infty} \delta^{2y} = \rho_s(x+t)\cdot 2\delta^2/(1-\delta^2)$, because for every $y \in \mathbb{N}\cup\{0\}$ and $x \in D_1$ both

$$\rho_s(x+t+(y+1)N)/\rho_s(x+t+yN) = \exp(-\frac{\pi}{s^2}\cdot N(2(x+t)+(2y+1)N)) \le \delta^2 \qquad (4.1)$$

[6] Equivalently, if $t \in [-5N/8, -3N/8]$ (and the constraints for N, s remain the same), then $|G\rangle$, $|(G^{tr})'\rangle = \frac{1}{\sqrt{G_{tr}}} \sum_{x\in[N]} f(x+t)\rho_s(x+t)\,|x\rangle$, $|(G^{mod})'\rangle = \frac{1}{\sqrt{G_{mod}}} \sum_{x\in\mathbb{Z}} f(x+t)\rho_s(x+t)\,|x \bmod N\rangle$ are 9δ-close to each other.

and

$$\rho_s(x+t-(y+1)N)/\rho_s(x+t-yN) = \exp(-\frac{\pi}{s^2}\cdot N(-2(x+t)+(2y+1)N)) \le \delta^2 \quad (4.2)$$

hold (because $\exp(-\frac{\pi}{s^2}\cdot N(N\pm 2(x+t)) \le \exp(-\frac{\pi}{s^2}\cdot\frac{N^2}{2}) \le \delta^2$ for every $x\in D_1$).

When $x\in D_2$ we use a similar argument, the only difference is that Equations 4.1 and 4.2 now hold for every $y\in\mathbb{N}$ (excluding 0), so

$$\sum_{y\in\mathbb{Z}\setminus\{0\}} \rho_s(x+t+Ny) \le \rho_s(x+t)\cdot 2\sum_{y=0}^{\infty}\delta^{2y} = \rho_s(x+t)\cdot 2/(1-\delta^2).$$

When $x\in D_3$, we can see that either $x+N$ or $x-N$ is $N/2$-close to (but $3N/8$-far from) $-t$, and without loss of generality and for simplicity we can assume $|x+N-(-t)|\in[3N/8,N/2]$. Then using a similar argument as for the $x\in D_2$, we can show that for every $x\in D_3$,

$$\sum_{y\in\mathbb{Z}\setminus\{0\}} \rho_s(x+t+Ny) \le \rho_s(x+N+t)\cdot 2/(1-\delta^2).$$

Since $\rho_s^2 = \rho_{s/\sqrt{2}}$ and $\mathcal{D}_{\mathbb{Z}+t,s/\sqrt{2}}$ is δ^2-sub-Gaussian with parameter $s/\sqrt{2}$, we have

$$\| |\widetilde{G^{mod}}\rangle - |G\rangle \|^2 \le \frac{1}{G}\sum_{x\in D_1\cup D_2\cup D_3}\Big(\sum_{y\in\mathbb{Z}\setminus\{0\}}\rho_s(x+t+Ny)\Big)^2 + \frac{1}{G}\sum_{x\in\mathbb{Z}\setminus[\![\pm\frac{N}{2}]\!]}\Big(\rho_s(x+t)\Big)^2$$

$$\le \frac{1}{G}\Big((\frac{2\delta^2}{1-\delta^2})^2\sum_{x\in D_1}\rho_{\frac{s}{\sqrt{2}}}(x+t) + (\frac{2}{1-\delta^2})^2\sum_{x\in D_2}\rho_{\frac{s}{\sqrt{2}}}(x+t) + (\frac{2}{1-\delta^2})^2\sum_{x\in D_3}\rho_{\frac{s}{\sqrt{2}}}(x+N+t) + \delta^2 G\Big)$$

$$\le \frac{1}{G}\Big(\frac{4\delta^4}{(1-\delta^2)^2}G + \frac{8}{(1-\delta^2)^2}(\delta^2 G) + \delta^2 G\Big) = \frac{4\delta^4+8\delta^2}{1-\delta^2} + \delta^2 \le 10\delta^2,$$

where the second equality holds because

$$\sum_{x\in D_2}\rho_{s/\sqrt{2}}(x+t) + \sum_{x\in D_3}\rho_{s/\sqrt{2}}(x+N+t)$$
$$\le \sum_{x\in\mathbb{Z}\setminus\{x:|x+t|\le N/8\}}\rho_{s/\sqrt{2}}(x+t) \le G\cdot 2\exp(\delta^2)\exp(-(N/8)^2/(2\cdot s^2/2)) \le \delta^2 G.$$

Note that $\| |\widetilde{G^{mod}}\rangle \| \in \| |G\rangle \| \pm \| |\widetilde{G^{mod}}\rangle - |G\rangle \|$ (implying $\| |\widetilde{G^{mod}}\rangle \| \in (1\pm\sqrt{10}\delta)$) and hence $\| |\widetilde{G^{mod}}\rangle - |G^{mod}\rangle \| = \|(\sqrt{\frac{G_{mod}}{G}}-1)|G^{mod}\rangle\| \le \sqrt{10}\delta$. As a result, we obtain $\| |G^{mod}\rangle - |G\rangle \| \le \| |\widetilde{G^{mod}}\rangle - |G\rangle \| + \| |\widetilde{G^{mod}}\rangle - |G^{mod}\rangle \| \le 2\sqrt{10}\delta < 7\delta$. And by triangle inequality, we obtain $\| |G^{mod}\rangle - |G^{tr}\rangle \| \le \| |G^{mod}\rangle - |G\rangle \| + \| |G\rangle - |G^{tr}\rangle \| \le 9\delta$. □

Using the above theorem, we now show the correctness of our quantum Gaussian phase estimator.

Theorem 4.2. *Let $\delta\in(0,0.1]$, $s\ge 20\sqrt{2\log(1/\delta)}$, $a\in[0,1]$, $N = 200\cdot\lceil s\sqrt{\log(100/\delta)}\rceil$, U be a unitary, $|\psi\rangle$ be an eigenvector of U such that $U|\psi\rangle = \exp(\pi i a/4)|\psi\rangle$. There exists a*

quantum algorithm that for every such U and a, given one copy of $|\psi\rangle$, outputs an estimator $\tilde{a}$ satisfying that $a-\tilde{a}$ distributes δ-close to $\mathcal{D}_{\frac{8}{N}\cdot\mathbb{Z}-\frac{8}{N}\nu,\frac{8}{\sqrt{2}s}}$ for some $\nu\in[0,1)$, using $\mathcal{O}(s\cdot\mathrm{polylog}(s/\delta))$ applications of controlled-U, controlled-U^{-1} and $\tilde{\mathcal{O}}(s\cdot\mathrm{polylog}(s/\delta))$ time.

Proof. We first explain the algorithm of our quantum Gaussian phase estimator. Let $|\tilde{\rho}_s\rangle=\frac{1}{\sqrt{\tilde{G}}}\sum_{z\in\{-N/2,\ldots,N/2-1\}}\rho_s(z)|z\rangle$, where $\tilde{G}$ is a normalizing constant. Let U_z be a unitary that maps $|z\rangle|\psi\rangle\to|z\rangle U^z|\psi\rangle$ for $z\in\{-N/2,\ldots,N/2-1\}$, and U_π be a unitary that maps $|z\rangle\to(-1)^z|z\rangle$ (this U_π is basically a Z-gate on the least-significant bit of z).

The algorithm is as follows. We first prepare the state $|\tilde{\rho}_s\rangle|w'\rangle$. We then apply U_z to this state, apply U_π, apply QFT_N^{-1} on the first register, and then measure the first register in the computational basis, divide the outcome value by $N/8$, subtract 4 from it, and output this value.

We first explain the time complexity of the above algorithm. To prepare $|\tilde{\rho}_s\rangle$, it suffices to compute all values of $\rho_s(z)$ for $z\in\{-N/2,\ldots,N/2-1\}$,[7] and computing all those values $(\rho_s(-N/2),\ldots,\rho_s(N/2-1))$ and constructing a KP-tree with those values stored in its leaves takes $\mathcal{O}(N\cdot\mathrm{polylog}N)$ time. To construct the unitary U_z, it suffices to use $\mathcal{O}(N\mathrm{polylog}N)$ applications of controlled-U, controlled-U^{-1} and time. Also, QFT_N can be implemented using $\mathcal{O}(\log^2 N)$ elementary gates. As a result, the total cost here is $\mathcal{O}(N\log N\cdot\log(1/\delta)+\log^2 N)=\mathcal{O}(s\cdot\mathrm{polylog}(s/\delta))$ time and $\mathcal{O}(N\cdot\mathrm{polylog}N)=\mathcal{O}(s\cdot\mathrm{polylog}(s/\delta))$ applications of controlled-U, controlled-U^{-1}.

Now we show the correctness of the above algorithm. Applying $U_\pi U_z$ to $|\tilde{\rho}_s\rangle|w'\rangle$ gives the state $\frac{1}{\sqrt{G'}}\sum_{z\in\{-N/2,\ldots,N/2-1\}}\rho_s(z)(-1)^z\exp(\pi iaz/4)|z\rangle|w'\rangle$. If we discard the second register, which is in tensor product with the rest of the state, then the remaining state is also 9δ-close to

$$|\Psi\rangle=\frac{1}{\sqrt{G}}\sum_{z\in\mathbb{Z}}\rho_s(z)\exp(2\pi i(a/8+1/2)z)|(z+N/2\bmod N)-N/2\rangle$$

because $s\geq 8\sqrt{2\log(1/\delta)}$, $N>16s\sqrt{2\ln(1/\delta)}$, and by Theorem 4.1, where G is a normalizing constant. Therefore, the distribution of the outcome of the quantum Gaussian phase estimator is 9δ-close to the distribution obtained by measuring the follow-

[7] Once we have those values, we can do the controlled-rotation tricks similar to how the KP-tree routine produces the quantum state. Since we only need to prepare $|\tilde{\rho}_s\rangle$ once, it is fine for us to prepare the state using $N\log N$ time. This procedure does not require the use of QRAM.

ing state (using $a' = a/8 + 1/2$)

$$\begin{aligned}
\mathrm{QFT}_N^{-1}|\Psi\rangle &= \frac{1}{\sqrt{NG}} \sum_{y\in[N]} \sum_{z\in\mathbb{Z}} \rho_s(z) \exp(2\pi i a' z) \exp(-2\pi i y((z+N/2 \bmod N) - N/2)/N)\,|y\rangle \\
&= \frac{1}{\sqrt{NG}} \sum_{y\in[N]} \sum_{z\in\mathbb{Z}} \rho_s(z) \exp(2\pi i z(Na'-y)/N)\,|y\rangle && (e^{-2\pi i y N z/N} = 1) \\
&= \frac{1}{\sqrt{NG}} \sum_{y\in[N]} \left\{ \sum_{z\in\frac{1}{N}\cdot\mathbb{Z}} \rho_{s/N}(z) \exp(2\pi i z(Na'-y)) \right\} |y\rangle && (z \leftarrow z/N) \\
&= \frac{1}{\sqrt{NG}} \sum_{y\in[N]} \left\{ N \cdot \sum_{x\in N\cdot\mathbb{Z}} \widehat{\rho_{s/N} \exp(2\pi i z(Na'-y))}(x) \right\} |y\rangle && \text{(by Theorem 2.38)} \\
&= \frac{1}{\sqrt{NG}} \sum_{y\in[N]} \left\{ \frac{s}{N} \cdot N \cdot \sum_{x\in N\cdot\mathbb{Z}} \rho_{N/s}(x - Na' + y) \right\} |y\rangle && (\widehat{\rho_{s/N}} = \frac{s}{N} \cdot \rho_{N/s}) \\
&= \frac{s}{\sqrt{NG}} \sum_{y\in[N]} \rho_{N/s}(N\cdot\mathbb{Z} - Na' + y)\,|y\rangle \\
&= \frac{s}{\sqrt{NG}} \sum_{y\in\mathbb{Z}} \rho_{N/s}(-Na' + y)\,|y \mod N\rangle\,.
\end{aligned}$$

By Theorem 4.1 again, because $a' \in [1/2, 5/8]$, $N/s \geq 8\sqrt{2\log(1/\delta)}$, and $N \geq 16(N/s)\sqrt{2\ln(1/\delta)}$, we know $\mathrm{QFT}_N^{-1}|\Psi\rangle$ is 9δ-close to $\frac{1}{\sqrt{G''}} \sum_{y\in[N]} \rho_{N/s}(-Na'+y)\,|y\rangle$ where G'' is a normalizing constant. Therefore, the probability distribution of $y - Na'$ (letting y be the measurement outcome) is 9δ-close to $\mathcal{D}^{[-N/2, N/2-1]}_{\mathbb{Z}-Na', \frac{N}{\sqrt{2}s}}$. Moreover, since $\sqrt{\log(12/\delta)/\pi} \leq \frac{N}{\sqrt{2}s}$ and $10\frac{N}{\sqrt{2}s}\sqrt{2\ln(1/\delta)} \leq N/2$, by Corollary 2.25 we know $y - Na'$ is also $9\delta + 4\delta\exp(\delta)$-close to $\mathcal{D}_{\mathbb{Z}-Na', \frac{N}{\sqrt{2}s}} = \mathcal{D}_{\mathbb{Z}-\nu, \frac{N}{\sqrt{2}s}}$ for some $\nu \in [0,1)$, implying that the distribution of $8y/N - 4 - a$ is $9\delta + 4\delta\exp(\delta)$-close to $\mathcal{D}_{\frac{8}{N}\cdot\mathbb{Z} - \frac{8}{N}\nu, \frac{8}{\sqrt{2}s}}$. As a result, the output of the algorithm in the second paragraph is $9\delta + 9\delta + 4\delta\exp(\delta)$-close to $\mathcal{D}_{\frac{8}{N}\cdot\mathbb{Z} - \frac{8}{N}\nu, \frac{8}{\sqrt{2}s}}$. Rescaling δ by a multiplicative constant, we finish the proof. □

Using Theorem 2.24, since $\frac{4\sqrt{2}}{s} \geq 8\sqrt{\log(12/\delta)/\pi}/N$ by the choice of N, we can see $\mathcal{D}_{\frac{8}{N}\cdot\mathbb{Z} - 8\nu, \frac{4\sqrt{2}}{s}}$ is δ-sub-Gaussian with parameter $\frac{4\sqrt{2}}{s}$. By letting $s = \frac{4\sqrt{2}}{\varepsilon}$, we have the following corollary.

Corollary 4.3 (Sub-Gaussian phase estimator, subGPE(U, ε, τ))**.** *Let $\varepsilon, \tau \in (0, 0.1]$, $a \in [0,1]$, U be a unitary, $|\psi\rangle$ be an eigenvector of U such that $U|\psi\rangle = \exp(\pi i a/4)|\psi\rangle$. There exists a quantum algorithm that, given one copy of $|\psi\rangle$, outputs an estimator $\tilde{a}$ satisfying that $a - \tilde{a}$ is τ-close to τ-sub-Gaussian with parameter ε using $\tilde{\mathcal{O}}(\mathrm{poly}\log(1/\tau)/\varepsilon)$ applications of controlled-U, controlled-U^{-1} and elementary gates.*

4.3 Time-efficient unbiased pure-state tomography

In this section we design efficient methods for obtaining a good classical description of a pure quantum state (i.e., tomography), by manipulating and measuring multiple

copies of that state.

4.3.1 Pure-state tomography by computational-basis measurements

A direct corollary of Corollary 2.28 as observed in [ACG+23] is that computational-basis measurements yield a good approximation of the absolute values of the amplitudes of a (sub)normalized quantum state vector.

Corollary 4.4. *Suppose that $\varepsilon, \delta \in (0,1]$, $\psi \in \mathbb{C}^d$ has ℓ_2-norm at most 1, and we are given $n \geq \frac{1}{\varepsilon^2} \ln\left(\frac{2d}{\delta}\right)$ copies of the pure quantum state $|\varphi\rangle := |\bar{0}\rangle|\psi\rangle + |\bar{0}^\perp\rangle$, where $(|\bar{0}\rangle\langle\bar{0}| \otimes I)|\bar{0}^\perp\rangle = 0$. If we measure each copy in the computational basis and denote by s_i the normalized number (i.e., frequency) of outcomes $|\bar{0}\rangle|i\rangle$ then the vector*

$$\bar{\psi}_i := \sqrt{s_i}$$

with probability at least $1-\delta$ gives an ε-ℓ_∞ approximation of $|\psi|$. Moreover, $\|\bar{\psi}\|_2 \leq 1$ with certainty and if $\|\psi\|_2 = 1$, then also $\|\bar{\psi}\|_2 = 1$, and in general $\left|\|\bar{\psi}\|_2 - \|\psi\|_2\right| \geq \varepsilon$ holds with probability $\leq \frac{\delta}{d}$.

Proof. By Corollary 2.28 we have that

$$\Pr\left[\left|\sqrt{s_i} - |\psi_i|\right| \geq \varepsilon\right] \leq \frac{\delta}{d},$$

and similarly

$$\Pr\left[\left|\|\bar{\psi}\|_2 - \|\psi\|_2\right| \geq \varepsilon\right] = \Pr\left[\left|\sqrt{\sum_{i=0}^{d-1} s_i} - \sqrt{\sum_{i=0}^{d-1} |\psi_i|^2}\right| \geq \varepsilon\right] \leq \frac{\delta}{d}.$$

Finally, $\|\bar{\psi}\|_2^2 = \sum_{i=0}^{d-1} s_i \leq 1$, where the last inequality is an equality if $\|\psi\|_2 = 1$. □

4.3.2 Pure-state tomography using conditional samples

Now we show how to produce an unbiased estimator of ψ itself (not just of the magnitudes of its entries) with bounded variance using computational-basis measurements with the help of a reference state $\bar{\psi}$. Our approach is inspired by [KP20] but improves over their biased estimator by making it unbiased.

Lemma 4.5. *Suppose that $\psi \in \mathbb{C}^d$ has $\|\psi\|_2 \leq 1$, and we are given a copy of the state $|\varphi'\rangle := \Big(|+\rangle\big(|\bar{0}\rangle|\psi\rangle + |\bar{0}^\perp\rangle\big) + |-\rangle\big(|\bar{0}\rangle|\bar{\psi}\rangle + |\bar{0}'^\perp\rangle\big)\Big)/\sqrt{2}$, where $|\bar{0}\rangle = |0^a\rangle$ for some $a \in \mathbb{N}$, $(|\bar{0}\rangle\langle\bar{0}| \otimes I)|\bar{0}^\perp\rangle = 0$, and $(|\bar{0}\rangle\langle\bar{0}| \otimes I)|\bar{0}'^\perp\rangle = 0$. If we measure $|\varphi'\rangle$ in the computational basis and denote by $X \in \{0,1\}^{2d}$ the indicator of the measurement outcomes $|b\rangle|\bar{0}\rangle|i\rangle$ (this X is a weight-1 Boolean vector indexed by (b,i) where $b \in \{0,1\}$ and $i \in [d]-1$), then the random vector $\psi' \in \mathbb{C}^d$ with coordinates*

$$\psi'_i := \frac{X_{0,i} - X_{1,i}}{|\bar{\psi}_i|}$$

is an unbiased estimator of $\psi_i^{\Re} := \mathrm{Re}\left(\psi_i \frac{\bar{\psi}_i^*}{|\bar{\psi}_i|}\right)$, *with* $\|\psi'\|_2 \leq \frac{1}{\min\{|\bar{\psi}_i|\colon i\in[d]-1\}}$ *with certainty, and covariance matrix* $\mathrm{Cov}(\psi') = \frac{I}{2} + \mathrm{diag}\left(\frac{|\psi_i|^2}{2|\bar{\psi}_i|^2}\right) - |\psi^{\Re}\rangle\langle\psi^{\Re}|$.

Proof. The probabilities of getting measurement outcomes $|0\rangle|\bar{0}\rangle|i\rangle$ and $|1\rangle|\bar{0}\rangle|i\rangle$ are

$$p_{0,i} := |\langle 0|\langle\bar{0}|\langle i||\varphi'\rangle|^2 = \left|\frac{\psi_i + \bar{\psi}_i}{2}\right|^2 = \frac{|\psi_i|^2 + \psi_i\bar{\psi}_i^* + \psi_i^*\bar{\psi}_i + |\bar{\psi}_i|^2}{4} = \mathbb{E}[X_{0,i}],$$
$$p_{1,i} := |\langle 1|\langle\bar{0}|\langle i||\varphi'\rangle|^2 = \left|\frac{\psi_i - \bar{\psi}_i}{2}\right|^2 = \frac{|\psi_i|^2 - \psi_i\bar{\psi}_i^* - \psi_i^*\bar{\psi}_i + |\bar{\psi}_i|^2}{4} = \mathbb{E}[X_{1,i}],$$

and therefore

$$\mathbb{E}[X_{0,i} - X_{1,i}] = p_{0,i} - p_{1,i} = \frac{\psi_i\bar{\psi}_i^* + \psi_i^*\bar{\psi}_i}{2} = |\bar{\psi}_i|\frac{\psi_i\frac{\bar{\psi}_i^*}{|\bar{\psi}_i|} + \psi_i^*\frac{\bar{\psi}_i}{|\bar{\psi}_i|}}{2} = |\bar{\psi}_i|\psi_i^{\Re}.$$

For the norm bound observe that

$$\|\psi'\|_2 \leq \sum_{i=0}^{d-1}|\psi'|_i \leq \sum_{i=0}^{d-1}\frac{X_{0,i}+X_{1,i}}{|\bar{\psi}|_i} \leq \sum_{i=0}^{d-1}\frac{X_{0,i}+X_{1,i}}{\min\{|\bar{\psi}_i|\colon i\in[d]-1\}} \leq \frac{1}{\min\{|\bar{\psi}_i|\colon i\in[d]-1\}}.$$

We can compute the covariance matrix directly as follows

$$\mathrm{Cov}(\psi')_{ij} = \mathbb{E}[\psi'_i\psi'_j] - \mathbb{E}[\psi'_i]\mathbb{E}[\psi'_j] = \delta_{ij}\frac{p_{0,i}+p_{1,i}}{|\bar{\psi}_i|^2} - \psi_i^{\Re}\psi_j^{\Re} = \delta_{ij}\frac{|\psi_i|^2/|\bar{\psi}_i|^2+1}{2} - \psi_i^{\Re}\psi_j^{\Re},$$

where the second equality uses that $X_{a,i}X_{b,j} = \delta_{ab}\cdot\delta_{ij}\cdot X_{a,i}$ because X is a weight-1 Boolean vector. □

By an analogous argument as in the proof of Lemma 4.5, we can obtain an unbiased estimator of the imaginary parts $\psi_j^{\Im} := \mathrm{Im}\left(\psi_j\frac{\bar{\psi}_j^*}{|\bar{\psi}_j|}\right)$ (with the same ℓ_2-norm and covariance matrix gaurantee) by measuring $|\varphi''\rangle := \left(|+\rangle\left(|\bar{0}\rangle|\psi\rangle + |\bar{0}^{\perp}\rangle\right) + i|-\rangle\left(|\bar{0}\rangle|\bar{\psi}\rangle + |\bar{0}'^{\perp}\rangle\right)\right)/\sqrt{2}$ in the computational basis.

We now give a procedure (the first part of Theorem 4.6) to find an unbiased estimator $\tilde{\psi}$ of ψ that simultaneously has a good bound on the error of the estimator (with overwhelming probability) in some k fixed directions using Lemma 4.5. The second part of Theorem 4.6 shows that the output $\tilde{\psi}$ will be close (in total variation distance) to an "almost ideal" unbiased estimator $\breve{\psi}$ that simultaneously has a good bound on the error of the estimator *with certainty* in some k fixed directions. This will be used later when estimating a matrix-vector product Aw in order to avoid an estimation-error that has too much overlap with k of the eigenvectors of A.

Theorem 4.6. *Let* $\psi\in\mathbb{C}^d$ *such that* $\|\psi\|_2\leq 1$, $\varepsilon,\delta\in(0,1]$, $\eta\in\mathbb{R}_+$, $k\in\mathbb{N}$, $n\geq\frac{4d}{\varepsilon^2}\left(\frac{4}{3}+\frac{1}{\eta}\right)\ln\left(\frac{8k}{\delta}\right)$. *Suppose there exists a "reference state" (not necessarily known to the algorithm)* $\bar{\psi}\in\mathbb{C}^d$

such that $|\bar{\psi}_j|^2 \geq \max\{\frac{\varepsilon^2}{d}, \eta|\psi_j|^2\}\ \forall j \in [d]-1$, *and* $\|\bar{\psi}\| \leq 1$. *Given n copies of the pure quantum states*

$$|\varphi'\rangle := \left(|+\rangle\left(|\bar{0}\rangle|\psi\rangle + |\bar{0}^{\perp}\rangle\right) + |-\rangle\left(|\bar{0}\rangle|\bar{\psi}\rangle + |\bar{0}'^{\perp}\rangle\right)\right)/\sqrt{2},$$
$$|\varphi''\rangle := \left(|+\rangle\left(|\bar{0}\rangle|\psi\rangle + |\bar{0}^{\perp}\rangle\right) + i|-\rangle\left(|\bar{0}\rangle|\bar{\psi}\rangle + |\bar{0}'^{\perp}\rangle\right)\right)/\sqrt{2},$$

where $|\bar{0}\rangle = |0^a\rangle$ *for some* $a \in \mathbb{N}$, $(|\bar{0}\rangle\langle\bar{0}|\otimes I)\,|\bar{0}^{\perp}\rangle = 0$, *and* $(|\bar{0}\rangle\langle\bar{0}|\otimes I)\,|\bar{0}'^{\perp}\rangle = 0$, *if we measure each copy in the computational basis and denote by* $s'_{b,j}$, $s''_{b,j}$ *the normalized number of measurement outcomes* $|b\rangle|\bar{0}\rangle|j\rangle$ *from measuring the states* $|\varphi'\rangle$ *and* $|\varphi''\rangle$ *respectively, then the random vector* $\tilde{\psi} \in \mathbb{C}^d$ *with coordinates*

$$\tilde{\psi}_j := \left(s'_{0,j} - s'_{1,j} + i\,s''_{0,j} - i\,s''_{1,j}\right)\frac{\bar{\psi}_j}{|\bar{\psi}_j|^2}$$

is an unbiased estimator of ψ. *Moreover, for every set* $V = \{v^{(j)} : j \in [k]\} \subset \mathbb{C}^d$ *of vectors we have*

$$\Pr\left[\forall v \in V : |\langle\tilde{\psi} - \psi|v\rangle| < \frac{\varepsilon}{\sqrt{d}}\|v\|_2\right] \geq 1-\delta. \tag{4.3}$$

In particular if $k \geq d$, *then* $\Pr\left[\|\tilde{\psi} - \psi\|_2 < \varepsilon\right] \geq 1-\delta$.

Finally, let $\{v^{(j)} : j \in [k]\}$ *be a fixed set of orthonormal vectors and* Π_k *be the projector to their span. Let A be the event that* $\exists j \in [k] : |\langle\tilde{\psi} - \psi|v^{(j)}\rangle| > \frac{\varepsilon}{\sqrt{d}}$, $\bar{A}$ *be the complement of A, and* X_ζ *be an independent Bernoulli random variable such that* $\Pr[X_\zeta = 0] = \zeta := \frac{\delta - p}{1-p}$ *for* $p := \Pr[A]$. *Define* $\breve{\psi} \in \mathbb{C}^d$ *as follows*[8]

$$\breve{\psi} = \begin{cases} \tilde{\psi} & \text{on } \bar{A} \cap (X_\zeta = 1) \\ (I - \Pi_k)\tilde{\psi} + \sum\limits_{j\in[k]} |v^{(j)}\rangle\,\mathbb{E}[\langle v^{(j)}|\tilde{\psi}\rangle\,|\,A \cup (X_\zeta = 0)] & \text{on } A \cup (X_\zeta = 0). \end{cases}$$

Then $\mathbb{E}[\breve{\psi}] = \psi$, $\Pr[\forall j \in [k] : |\langle\breve{\psi} - \psi|v^{(j)}\rangle| \leq \frac{k+3}{k}\frac{\varepsilon}{\sqrt{d}}] = 1$ *(which is why we call* $\breve{\psi}$ *an "almost ideal" unbiased estimator), the total variation distance between* $\tilde{\psi}$, *and* $\breve{\psi}$ *is at most* δ, *and* $\left\|\mathrm{Cov}(\Pi_k\breve{\psi})\right\| \leq \left\|\mathrm{Cov}(\Pi_k\tilde{\psi})\right\| + 25\delta\varepsilon^2\frac{k}{d} \leq \left(\frac{1}{4\ln\left(\frac{8k}{\delta}\right)} + 25\delta k\right)\frac{\varepsilon^2}{d}$.

Proof. We prove the first part of Theorem 4.6 first. Let us define the random vectors $\psi', \psi'' \in \mathbb{C}^d$ with coordinates

$$\psi'_j := \frac{X'_{0,j} - X'_{1,j}}{|\bar{\psi}_j|}, \qquad \psi''_j := \frac{X''_{0,j} - X''_{1,j}}{|\bar{\psi}_j|},$$

where $X', X'' \in \{0,1\}^{2d}$ denote the indicator of the measurements outcomes $|b\rangle|\bar{0}\rangle|j\rangle$ for the states $|\varphi'\rangle$ and $|\varphi''\rangle$, respectively. Then by Lemma 4.5 and the discussion after

[8] Here we introduce X_ζ to ensure that $\Pr[A \cup (X_\zeta = 0)]$ exactly equals δ, which is helpful because we use both upper and lower bounds on this probability in the proof.

the proof of Lemma 4.5, ψ', ψ'' are unbiased estimators of $\psi_j^{\Re} := \mathrm{Re}\left(\psi_j \frac{\bar{\psi}_j^*}{|\bar{\psi}_j|}\right)$, and $\psi_j^{\Im} := \mathrm{Im}\left(\psi_j \frac{\bar{\psi}_j^*}{|\bar{\psi}_j|}\right)$ respectively, such that $\|\psi'\|_2, \|\psi''\|_2 \le \sqrt{d}/\varepsilon$ with certainty and

$$\mathrm{Cov}(\psi') + \mathbb{E}[\psi']\mathbb{E}[\psi'^T] \preceq \left(\frac{1}{2} + \frac{1}{2\eta}\right) I, \quad \mathrm{Cov}(\psi'') + \mathbb{E}[\psi'']\mathbb{E}[\psi''^T] \preceq \left(\frac{1}{2} + \frac{1}{2\eta}\right) I, \tag{4.4}$$

where for the latter psd inequalities we used that $|\bar{\psi}_j|^2 \ge \eta |\psi_j|^2$ and hence $\mathrm{diag}\left(\frac{|\psi_i|^2}{2|\bar{\psi}_i|^2}\right) \preceq \frac{1}{2\eta} I$.

Let $w \in \mathbb{R}^d$, then the random variables $\psi'_w := \langle \psi' | w \rangle, \psi''_w := \langle \psi'' | w \rangle$ satisfy $|\psi'_w| \le \|\psi'\|_2 \|w\|_2 \le \sqrt{d}\|w\|_2/\varepsilon$, $|\psi''_w| \le \|\psi''\|_2 \|w\|_2 \le \sqrt{d}\|w\|_2/\varepsilon$ with certainty. Also, $\mathbb{E}[|\psi'_w|^2]$ and $\mathbb{E}[|\psi''_w|^2]$ are both $\le \left(\frac{1}{2} + \frac{1}{2\eta}\right)\|w\|_2^2$, because for both $\phi = \psi'$ and $\phi = \psi''$, we have

$$\mathbb{E}[|\langle \phi | w \rangle|^2] = \langle w | \mathbb{E}[\phi\phi^T] | w \rangle = \langle w | \left(\mathrm{Cov}(\phi) + \mathbb{E}[\phi]\mathbb{E}[\phi^T]\right) | w \rangle \le \|\mathrm{Cov}(\phi) + \mathbb{E}[\phi]\mathbb{E}[\phi^T]\| \cdot \|w\|_2^2.$$

Let $\Psi', \Psi'' \in \mathbb{R}^d$ be the sum of n i.i.d. copies of ψ', ψ'', respectively, obtained from the measurement outcomes of the n copies of $|\varphi'\rangle$ and $|\varphi''\rangle$, so that

$$\tilde{\psi}_j = (\Psi'_j + i\Psi''_j)\frac{\bar{\psi}_j}{n|\bar{\psi}_j|}. \tag{4.5}$$

Let us analogously define $\Psi'_w := \langle \Psi' | w \rangle$, and $\Psi''_w := \langle \Psi'' | w \rangle$. Then clearly $\mathbb{E}[\Psi'_w] = n\langle \psi^{\Re} | w \rangle$, and $\mathbb{E}[\Psi''_w] = n\langle \psi^{\Im} | w \rangle$. For all $\tau \ge 1$, the Bennett-Bernstein tail bound (Proposition 2.26) implies

$$\begin{aligned}\Pr\left[|\Psi'_w - n\langle \psi^{\Re} | w \rangle| \ge \tau \frac{\varepsilon n \|w\|_2}{2\sqrt{d}}\right] &\le 2\exp\left(-\frac{\tau^2\varepsilon^2 n^2 \|w\|_2^2/(4d)}{\left(1+\frac{1}{\eta}\right)\|w\|_2^2 n + \frac{\tau}{3}\|w\|_2^2 n}\right) \\ &\le 2\exp\left(-\tau\frac{\varepsilon^2 n/(4d)}{\frac{4}{3}+\frac{1}{\eta}}\right) \le 2\left(\frac{\delta}{8k}\right)^{\tau},\end{aligned} \tag{4.6}$$

and similarly $\Pr[|\Psi''_w - n\langle \psi^{\Im} | w \rangle| \ge \tau\varepsilon n\|w\|_2/\sqrt{4d}] \le 2\left(\frac{\delta}{8k}\right)^{\tau}$.

Let $\tilde{v}_j := v_j \frac{\bar{\psi}_j^*}{|\bar{\psi}_j|}$, $\tilde{v}_j^{\Re} := \mathrm{Re}(\tilde{v}_j)$, $\tilde{v}_j^{\Im} := \mathrm{Im}(\tilde{v}_j)$, and observe that for any $v \in V$

$$\begin{aligned}\langle \tilde{\psi} | v \rangle - \langle \psi | v \rangle &= \frac{1}{n}\langle \Psi' + i\Psi'' | \tilde{v} \rangle - \langle \psi^{\Re} + i\psi^{\Im} | \tilde{v} \rangle \\ &= \Bigg(\underbrace{\frac{\langle \Psi' | \tilde{v}^{\Re} \rangle}{n} - \langle \psi^{\Re} | \tilde{v}^{\Re} \rangle}_{:= a\|\tilde{v}^{\Re}\|_2} \underbrace{- \frac{\langle \Psi'' | \tilde{v}^{\Im} \rangle}{n} + \langle \psi^{\Im} | \tilde{v}^{\Im} \rangle}_{:= b\|\tilde{v}^{\Im}\|_2}\Bigg) + i\Bigg(\underbrace{\frac{\langle \Psi' | \tilde{v}^{\Im} \rangle}{n} - \langle \psi^{\Re} | \tilde{v}^{\Im} \rangle}_{:= c\|\tilde{v}^{\Re}\|_2} \underbrace{- \frac{\langle \Psi'' | \tilde{v}^{\Re} \rangle}{n} + \langle \psi^{\Im} | \tilde{v}^{\Re} \rangle}_{:= d\|\tilde{v}^{\Im}\|_2}\Bigg),\end{aligned}$$

so

$$\begin{aligned}|\langle\tilde{\psi}|v\rangle - \langle\psi|v\rangle| &\le \sqrt{2}\max\left\{\left|\mathrm{Re}(\langle\tilde{\psi}|v\rangle - \langle\psi|v\rangle)\right|, \left|\mathrm{Im}(\langle\tilde{\psi}|v\rangle - \langle\psi|v\rangle)\right|\right\} \\ &= \sqrt{2}\max\left\{\left|a\|\tilde{v}^{\Re}\|_2 + b\|\tilde{v}^{\Im}\|_2\right|, \left|c\|\tilde{v}^{\Re}\|_2 + d\|\tilde{v}^{\Im}\|_2\right|\right\} \\ &\le \sqrt{2}\max\{|a|,|b|,|c|,|d|\}(\|\tilde{v}^{\Re}\|_2 + \|\tilde{v}^{\Im}\|_2) \\ &\le 2\max\{|a|,|b|,|c|,|d|\}\|v\|_2,\end{aligned}$$

where the last step uses that $|\tilde{v}_j| = |v_j|$ for all j, and Cauchy-Schwarz. Using Eq. (4.6) four times with different choices of w , and the union bound over 4 events, we have $\max\{|a|,|b|,|c|,|d|\} < \frac{\tau\varepsilon}{2\sqrt{d}}$ except with probability $\le 8\left(\frac{\delta}{8k}\right)^{\tau}$. Eq. (4.3) now follows by choosing $\tau = 1$ and taking the union bound over all k vectors $v \in V$. If $k \ge d$ then we can apply the statement for a set V containing the computational basis, and then $\|\tilde{\psi} - \psi\|_\infty \le \frac{\varepsilon}{\sqrt{d}}$ implies $\|\tilde{\psi} - \psi\|_2 \le \varepsilon$ by Cauchy-Schwarz.

Now we prove the second part of Theorem 4.6, where V is an orthonormal set (the "Finally" part). Note that the previous paragraph already proved that for every $v \in \mathbb{C}^d$ and for all $\tau \ge 1$,

$$\Pr[|\langle v|\tilde{\psi} - \psi\rangle| > \tau\frac{\varepsilon}{\sqrt{d}}\|v\|_2] \le 8\left(\frac{\delta}{8k}\right)^{\tau}. \tag{4.7}$$

Defining $\mathring{\varepsilon} := \varepsilon/\sqrt{d}$ and $\mathring{\psi}^j := \langle v^{(j)}|\tilde{\psi} - \psi\rangle$, we bound

$$\begin{aligned}\mathbb{E}\left[|\mathring{\psi}^j|\mathbb{1}_{(\mathring{\varepsilon},\infty)}(|\mathring{\psi}^j|)\right] &= \sum_{\ell=0}^{\infty}\mathbb{E}\left[|\mathring{\psi}^j|\mathbb{1}_{(2^\ell\mathring{\varepsilon},2^{\ell+1}\mathring{\varepsilon}]}(|\mathring{\psi}^j|)\right] \le \sum_{\ell=0}^{\infty}2^{\ell+1}\mathring{\varepsilon}\Pr\left[|\mathring{\psi}^j| \in (2^\ell\mathring{\varepsilon}, 2^{\ell+1}\mathring{\varepsilon}]\right] \\ &\le \sum_{\ell=0}^{\infty}2^{\ell+1}\mathring{\varepsilon}\Pr\left[|\mathring{\psi}^j| > 2^\ell\mathring{\varepsilon}\right] \le \sum_{\ell=0}^{\infty}2^{\ell+1}\mathring{\varepsilon}\,8\left(\frac{\delta}{8k}\right)^{2^\ell} \\ &\qquad \left(\Pr[|\mathring{\psi}^j| > \tau\mathring{\varepsilon}] \le 8\left(\tfrac{\delta}{8k}\right)^{\tau}\right) \\ &\le \sum_{\ell=0}^{\infty}2^{\ell+1}\mathring{\varepsilon}\,8\left(\frac{\delta}{8k}\right)^{\ell+1} \qquad \text{(since } \delta/k \le 1\text{, and } 2^\ell \ge \ell+1\text{)} \\ &= 2\frac{\delta\varepsilon}{k\sqrt{d}}\sum_{\ell=0}^{\infty}\left(\frac{\delta}{4k}\right)^{\ell} \le 2\frac{\delta\varepsilon}{k\sqrt{d}}\sum_{\ell=0}^{\infty}\left(\frac{1}{4}\right)^{\ell} \qquad \text{(since } \delta/k \le 1\text{)} \\ &< \frac{3\delta}{k}\frac{\varepsilon}{\sqrt{d}}.\end{aligned} \tag{4.8}$$

We have already proven Eq. (4.3), implying that $p = \Pr[A] \le \delta$. Observe that

$$\mathbb{E}[\langle v^{(j)}|\tilde{\psi}\rangle\,|A \cup (X_\zeta = 0)] = \langle v^{(j)}|\psi\rangle + \mathbb{E}[\mathring{\psi}^j|A \cup (X_\zeta = 0)],$$

and hence

$$\mathbb{E}[\langle v^{(j)}|\tilde{\psi} - \psi\rangle\,|A \cup (X_\zeta = 0)] = \mathbb{E}[\mathring{\psi}^j|A \cup (X_\zeta = 0)].$$

Since $\Pr[A \cup (X_\zeta = 0)] = 1 - \Pr[\bar{A} \cap (X_\zeta = 1)] = \delta$, using (4.8) we have

$$|\mathbb{E}[\mathring{\psi}^j|A \cup (X_\zeta = 0)]| \le \mathbb{E}[|\mathring{\psi}^j|\,|\,A \cup (X_\zeta = 0)] = \frac{\mathbb{E}[|\mathring{\psi}^j|\mathbb{1}_{A\cup(X_\zeta=0)}]}{\delta} \le \frac{\mathring{\varepsilon}\delta + \frac{3\delta}{k}\frac{\varepsilon}{\sqrt{d}}}{\delta} = \left(1 + \frac{3}{k}\right)\frac{\varepsilon}{\sqrt{d}}. \tag{4.9}$$

Since we modified $\tilde{\psi}$ on an event of probability δ to get $\breve{\psi}$, the total variation distance between the distributions of the random variables $\breve{\psi}$ and $\tilde{\psi}$ is at most δ. The boundedness of $\breve{\psi}$ is by construction and the unbiasedness is inherited from that of $\tilde{\psi}$, as follows: abbreviating the event $\bar{A} \cap (X_\zeta = 1)$ to B, we have

$$\begin{aligned}
\mathbb{E}[\breve{\psi}] &= \Pr[B] \cdot \mathbb{E}[\breve{\psi} \mid B] + \Pr[\bar{B}] \cdot \mathbb{E}[\breve{\psi} \mid \bar{B}] \\
&= \Pr[B] \cdot \mathbb{E}[\tilde{\psi} \mid B] + \Pr[\bar{B}] \cdot \mathbb{E}[(I - \Pi_k)\tilde{\psi} + \sum_{j \in [k]} |v^{(j)}\rangle \mathbb{E}[\langle v^{(j)} | \tilde{\psi}\rangle] \mid \bar{B}] \\
&= \Pr[B] \cdot \mathbb{E}[\tilde{\psi} \mid B] + \Pr[\bar{B}] \cdot \mathbb{E}[(I - \Pi_k)\tilde{\psi} + \sum_{j \in [k]} |v^{(j)}\rangle \langle v^{(j)}| \cdot \tilde{\psi} \mid \bar{B}] \\
&= \Pr[B] \cdot \mathbb{E}[\tilde{\psi} \mid B] + \Pr[\bar{B}] \cdot \mathbb{E}[(I - \Pi_k)\tilde{\psi} + \Pi_k \tilde{\psi} \mid \bar{B}] \\
&= \Pr[B] \cdot \mathbb{E}[\tilde{\psi} \mid B] + \Pr[\bar{B}] \cdot \mathbb{E}[\tilde{\psi} \mid \bar{B}] = \mathbb{E}[\tilde{\psi}] = \psi.
\end{aligned}$$

Finally, defining $\bar{\varepsilon} := \varepsilon\sqrt{\frac{k}{d}}$ and $\mathring{\psi} := \Pi_k(\tilde{\psi} - \psi) = \sum_{j \in [k]} \mathring{\psi}^j v^{(j)}$ we bound

$$\begin{aligned}
\left\| \mathbb{E}\left[|\mathring{\psi}\rangle\langle\mathring{\psi}| \mathbb{1}_{(\bar{\varepsilon},\infty)}(\|\mathring{\psi}\|_2) \right] \right\| &\le \mathbb{E}\left[\|\mathring{\psi}\|_2^2 \mathbb{1}_{(\bar{\varepsilon},\infty)}(\|\mathring{\psi}\|_2) \right] = \sum_{\ell=0}^{\infty} \mathbb{E}\left[\|\mathring{\psi}\|_2^2 \mathbb{1}_{(2^\ell\bar{\varepsilon}, 2^{\ell+1}\bar{\varepsilon}]}(\|\mathring{\psi}\|_2) \right] \\
&\le \sum_{\ell=0}^{\infty} 4^{\ell+1}\bar{\varepsilon}^2 \Pr\left[\|\mathring{\psi}\|_2 \in (2^\ell\bar{\varepsilon}, 2^{\ell+1}\bar{\varepsilon}] \right] \le \sum_{\ell=0}^{\infty} 4^{\ell+1}\bar{\varepsilon}^2 \Pr\left[\|\mathring{\psi}\|_2 > 2^\ell\bar{\varepsilon} \right] \\
&\le \sum_{\ell=0}^{\infty} 4^{\ell+1}\bar{\varepsilon}^2 8k \left(\frac{\delta}{8k}\right)^{2^\ell} \qquad \left(\Pr[\|\mathring{\psi}\|_2 > \tau\bar{\varepsilon}] \le 8k\left(\tfrac{\delta}{8k}\right)^\tau \right) \\
&\le \sum_{\ell=0}^{\infty} 4^{\ell+1}\bar{\varepsilon}^2 8k \left(\frac{\delta}{8k}\right)^{\ell+1} \qquad (\text{since } \delta/k \le 1, \text{ and } 2^\ell \ge \ell+1) \\
&= 4\delta\varepsilon^2 \frac{k}{d} \sum_{\ell=0}^{\infty} \left(\frac{\delta}{2k}\right)^\ell \le 4\delta\varepsilon^2 \frac{k}{d} \sum_{\ell=0}^{\infty} \left(\frac{1}{2}\right)^\ell \qquad (\text{since } \delta/k \le 1) \\
&= 8\delta\varepsilon^2 \frac{k}{d}. \qquad (4.10)
\end{aligned}$$

This then implies that

$$\begin{aligned}
\left\| \mathrm{Cov}(\Pi_k\tilde{\psi}) - \mathrm{Cov}(\Pi_k\breve{\psi}) \right\| &= \left\| \mathbb{E}\left[|\mathring{\psi}\rangle\langle\mathring{\psi}| - \Pi_k |\breve{\psi} - \psi\rangle\langle\breve{\psi} - \psi| \Pi_k \right] \right\| \\
&= \left\| \mathbb{E}\left[\left(|\mathring{\psi}\rangle\langle\mathring{\psi}| - \Pi_k |\breve{\psi} - \psi\rangle\langle\breve{\psi} - \psi| \Pi_k \right) \mathbb{1}_{A \cup (X_\zeta = 0)} \right] \right\| \\
&\le \left\| \mathbb{E}\left[|\mathring{\psi}\rangle\langle\mathring{\psi}| \mathbb{1}_{A \cup (X_\zeta = 0)} \right] \right\| + \left\| \mathbb{E}\left[\Pi_k |\breve{\psi} - \psi\rangle\langle\breve{\psi} - \psi| \Pi_k \cdot \mathbb{1}_{A \cup (X_\zeta = 0)} \right] \right\|, \qquad (4.11)
\end{aligned}$$

where the second equality is because $\tilde{\psi} = \breve{\psi}$ on the complement of the event $A \cup (X_\zeta = 0)$. Using the definition of $\breve{\psi}$, and the fact that $\Pi_k(I - \Pi_k) = 0$, we can see that $\Pi_k\breve{\psi}$ conditioned on $A \cup (X_\zeta = 0)$ is actually a fixed vector $\mathbb{E}[\Pi_k\tilde{\psi} \mid A \cup (X_\zeta = 0)]$, not a random variable anymore. We now have

$$\left\| \mathbb{E}\left[\Pi_k |\breve{\psi} - \psi\rangle\langle\breve{\psi} - \psi| \Pi_k \cdot \mathbb{1}_{A \cup (X_\zeta = 0)} \right] \right\| = \Pr[A \cup (X_\zeta = 0)] \cdot \left\| \mathbb{E}\left[\Pi_k |\tilde{\psi} - \psi\rangle \mid A \cup (X_\zeta = 0) \right] \right\|_2^2.$$

Continuing with Eq. (4.11), we have

$$\begin{aligned}
\left\|\mathrm{Cov}(\Pi_k\tilde{\psi}) - \mathrm{Cov}(\Pi_k\breve{\psi})\right\| &\le \left\|\mathbb{E}\left[|\mathring{\psi}\rangle\langle\mathring{\psi}|\mathbb{1}_{A\cup(X_\zeta=0)}\right]\right\| + \delta\|\mathbb{E}[\mathring{\psi} \mid A\cup(X_\zeta=0)]\|_2^2 \\
&\le \mathbb{E}\left[\|\mathring{\psi}\|_2^2\mathbb{1}_{A\cup(X_\zeta=0)}\right] + 16\delta\varepsilon^2\frac{k}{d} \qquad \text{(by (4.9) and } (1+3/k\le 4)) \\
&= \mathbb{E}\left[\|\mathring{\psi}\|_2^2\mathbb{1}_{A\cup(X_\zeta=0)}\left(\mathbb{1}_{[0,\bar{\varepsilon}]}(\|\mathring{\psi}\|_2) + \mathbb{1}_{(\bar{\varepsilon},\infty)}(\|\mathring{\psi}\|_2)\right)\right] + 16\delta\varepsilon^2\frac{k}{d} \\
&\le \delta\bar{\varepsilon}^2 + 8\delta\varepsilon^2\frac{k}{d} + 16\delta\varepsilon^2\frac{k}{d} = 25\delta\varepsilon^2\frac{k}{d}. \\
&\qquad\qquad \text{(by (4.10) and } \Pr[A\cup(X_\zeta=0)] = \delta)
\end{aligned}$$

We obtain

$$\left\|\mathrm{Cov}(\Pi_k\breve{\psi})\right\| \le \left\|\mathrm{Cov}(\Pi_k\tilde{\psi}) - \mathrm{Cov}(\Pi_k\breve{\psi})\right\| + \left\|\mathrm{Cov}(\Pi_k\tilde{\psi})\right\| \le 25\delta\varepsilon^2\frac{k}{d} + \frac{\varepsilon^2}{4d\ln\left(\frac{8k}{\delta}\right)}$$

because $\left\|\mathrm{Cov}(\Pi_k\tilde{\psi})\right\| \le \left\|\mathrm{Cov}(\tilde{\psi})\right\| = \frac{1}{n}\left\|\mathrm{Cov}(\psi') + \mathrm{Cov}(\psi'')\right\|$, and the matrix inside the latter norm can be upper bounded by $2(\frac{1}{2}+\frac{1}{2\eta})I$ using Eq. (4.4). □

If we have n conditional samples $|\varphi\rangle := (|0\rangle\left(|\bar{0}\rangle|\psi\rangle + |\bar{0}^\perp\rangle\right) + |1\rangle|\bar{0}\rangle|0\rangle)/\sqrt{2}$, then we can first use Corollary 4.4 to produce (with success probability $\ge 1-\frac{\delta}{2}$) a $\frac{1}{\sqrt{d}}$-ℓ_∞ approximation ψ' of the vector $|\psi|$ of the magnitudes of entries, which has $\|\psi'\|_2 \le 1$. Setting $\bar{\psi}_i := \frac{|\psi'_i| + \frac{1}{\sqrt{d}}}{2}$, and building a KP-tree for $\bar{\psi}$ to be able to efficiently prepare a state that is coordinate-wise $\frac{1}{4\sqrt{d}}$-close to $|\bar{\psi}\rangle/\|\bar{\psi}\|_2$, we can transform the conditional copies $|\varphi\rangle$ to the form required by Theorem 4.6 using $\mathcal{O}\left(n\log^2(d)\right)$ classical operations, ordinary quantum gates and QRAM read-out calls. Since $\eta = \Omega(1)$, we get a time-efficient unbiased tomography algorithm using $\mathcal{O}\left(\frac{d}{\varepsilon^2}\ln\left(\frac{2d}{\delta}\right)\right)$ conditional samples.

4.3.3 Improved pure-state tomography using state-preparation oracles

If we have a state-preparation oracle available, rather than copies of the state, then the precision-dependence can be quadratically improved using iterative refinement [Gil23]:

Corollary 4.7. *Let $\psi \in \mathbb{C}^d$ such that $\|\psi\| \le 1$, and $\varepsilon, \delta \in (0, \frac{1}{2}]$. Suppose we have access to a controlled unitary U (and its inverse) that prepares the state $U|0^{\otimes a'}\rangle = |0^{\otimes a}\rangle|\psi\rangle + |0^{\otimes a}{}^\perp\rangle$, where $a', a = \mathcal{O}\left(\mathrm{poly}\log(d/(\delta\varepsilon))\right)$. There is a quantum algorithm that outputs a random vector $\tilde{\psi} \in \mathbb{C}^d$ such that, for every set $V = \{v^{(1)}, v^{(2)}, \ldots, v^{(k)}\}$ of unit vectors, with probability at least $1-\delta$, $|\langle\psi - \tilde{\psi}|v\rangle| \le \varepsilon/\sqrt{d}$ for all $v \in V$, using $\mathcal{O}(\frac{d}{\varepsilon}\mathrm{poly}\log(kd/(\varepsilon\delta)))$ applications of controlled U, $U^\dagger$, two-qubit quantum gates, read-outs of a QRAM of size $\mathcal{O}(d\cdot\mathrm{poly}\log(kd/(\varepsilon\delta)))$, and classical computation.*

If $k = d$ and V is an orthonormal set, then $\tilde{\psi}$ is δ-close in total variation distance to an "almost ideal" discrete random variable $\breve{\psi} \in \mathbb{C}^d$ such that $\mathbb{E}[\breve{\psi}] = \psi$, $\Pr[\forall v \in V : |\langle\breve{\psi} - \psi|v\rangle| \le \frac{\varepsilon}{\sqrt{d}}] = 1$, and $\|\mathrm{Cov}(\breve{\psi})\| \le \frac{\varepsilon^2}{d}$.

Proof. The idea is to use the tomography algorithm of [Gil23] to get an estimator ψ' with ℓ_2-error ε, with success probability $\geq 1-\frac{\delta}{4}$, using $\mathcal{O}\left(\frac{d}{\varepsilon}\log(d/\delta)\right)$ queries in time $\mathcal{O}(\frac{d}{\varepsilon}\log(1/\delta)\cdot \mathrm{polylog}(d/\varepsilon))$. In case of failure we set $\breve{\psi}=\psi$.

We first build the KP-tree for ψ' in QRAM. We can now prepare a state $|0\rangle|\psi'\rangle + |1\rangle|.\rangle$, and thus also the state $|00\rangle|(\psi-\psi')/2\rangle + |1\rangle|..\rangle$, and using linearized amplitude amplification [GSL+19, Theorem 30], we can also prepare a subnormalized state ϕ such that $\|\phi-(\psi-\psi')/(2\varepsilon)\| \leq \frac{\delta}{16\sqrt{d}}$ with $\mathcal{O}\left(\log(d/\delta)/\varepsilon\right)$ (controlled) uses of U and $U^\dagger$.

As discussed at the start of this subsection, by Corollary 4.4 using $d\ln(\frac{6d}{\delta})$ copies of $|\phi\rangle$ we can output a vector $\vec{\mu}\in[0,1]^d$ such that with probability at least $1-\frac{\delta}{4}$, $|\vec{\mu}_j - |\phi_j|| \leq \frac{1}{\sqrt{d}}$ for every $j\in[d]$. (In case of failure we once again set $\breve{\psi}=\psi$.) Upon success, the vector $\vec{\mu}' := \frac{1}{2}\vec{\mu} + \frac{1}{2\sqrt{d}}\mathbf{1}_d$ where $\mathbf{1}_d$ is the d-dimensional all-1 vector, satisfies $|\vec{\mu}'_j|^2 \geq \frac{1}{4}\max\{|\phi_j|^2, \frac{1}{d}\}$ for every $j\in[d]$. Also, by using $\tilde{\mathcal{O}}(d)$ time and QRAM bits, we can construct a KP-tree $\mathrm{KP}_{\vec{\mu}'}$ for $\vec{\mu}'$. Thus, by using one query to $\mathrm{KP}_{\vec{\mu}'}$ and $\tilde{\mathcal{O}}(1)$ time, we can prepare a state $|\bar{0}\rangle|\vec{\mu}'\rangle + |\bar{0}'^{\perp}\rangle$, where $|\vec{\mu}'\rangle = \sum_{j\in[d]} \vec{\mu}'_j|j\rangle$.

By Theorem 4.6 we can output an unbiased estimator $\tilde{\phi}$ of ϕ such that $\Pr[\forall v\in V: |\langle\tilde{\phi}-\phi|v\rangle| \leq \frac{1}{32\sqrt{d}}] \geq 1-\frac{\delta}{4}$. Defining $\tilde{\psi} := \psi' + 2\varepsilon\tilde{\phi}$ we then have $\Pr[\forall v\in V: |\langle\tilde{\psi}-\psi|v\rangle| \leq \frac{\varepsilon}{8\sqrt{d}}] \geq 1-\frac{\delta}{4}$ since $\|\phi-(\psi-\psi')/(2\varepsilon)\| \leq \frac{\delta}{16\sqrt{d}}$. If V is an orthonormal basis, then furthermore $\tilde{\phi}$ is $\delta/4$-close to an "ideal" (though not error-free) unbiased estimator ϕ' of ϕ such that $\Pr[\forall v\in V: |\langle\phi'-\phi|v\rangle| \leq \frac{1}{8\sqrt{d}}] = 1$. Since $\|\phi-(\psi-\psi')/(2\varepsilon)\| \leq \frac{\delta}{16\sqrt{d}}$ there is another discrete-valued estimator $\breve{\phi}$ within total variation distance $\frac{\delta}{4}$ to ϕ' that satisfies $\mathbb{E}[\breve{\phi}] = (\psi-\psi')/(2\varepsilon)$ and $\Pr[\|\breve{\phi}-\phi'\| > \frac{1}{4\sqrt{d}}] = 0$ in turn implying $\Pr[\forall v\in V: |\langle\breve{\phi}-(\psi-\psi')/(2\varepsilon)|v\rangle| \leq \frac{1}{2\sqrt{d}}] = 1$. We then set $\breve{\psi} := \psi' + 2\varepsilon\breve{\phi}$ (in case no failure happened). We have $\|\mathrm{Cov}(\breve{\phi})\| \leq 2\|\mathrm{Cov}(\breve{\phi}-\phi')\| + 2\|\mathrm{Cov}(\phi')\|$, because for all vectors a,b, the matrix $2aa^\dagger + 2bb^\dagger - (a+b)(a+b)^\dagger = (a-b)(a-b)^\dagger$ is psd. Therefore the claimed properties of $\tilde{\psi},\breve{\psi}$ follow from those of $\tilde{\phi},\phi'$ as guaranteed by Theorem 4.6, assuming without loss of generality that $\delta\leq\frac{1}{k}$. □

4.4 Quantum noisy power method

In this section we introduce quantum algorithms for approximating the top eigenvector or top-q eigenvectors by using the two tools we introduced in the previous two sections. For simplicity, we assume the input matrix A is real and Hermitian, and has operator norm $\|A\|\leq 1$ (which implies all entries are in $[-1,1]$). Since A is Hermitian, its eigenvalues $\lambda_1,\ldots,\lambda_d$ are real, and we assume them to be ordered in descending order according to their absolute value.[9] Since the entries of A are real, there is always an

[9] Sometimes the eigenvalues are ordered $1\geq\lambda_1\geq\cdots\geq\lambda_d\geq -1$ according to their value (instead of their absolute value). To find the v_1 associated with λ_1 in this situation, one can just let $A' = A/3 + 2I/3$. Then the eigenvectors of A and A' are the same, and the eigenvalues of A' now are all between $1/3$

associated orthonormal basis of *real* eigenvectors $v_1, \ldots, v_d$. For simplicity and without loss of generality, when we mention the qth eigenvector of A, we mean v_q in this basis. The goal of the algorithms in this section is to find a unit vector w which has large overlap with v_1 in the sense that $|\langle w, v_1\rangle| \geq 1 - \varepsilon^2/2$. Note that this is equivalent to finding a unit vector w satisfying that either $\|w - v_1\|_2$ or $\|w + v_1\|_2$ is small (at most ε), and hence we say w approximates v_1 with small ℓ_2-error.

4.4.1 Classical noisy power method for approximating the top eigenvector

For the sake of completeness and pedagogy, we start with the noisy power method of Hardt and Price [HP14], given in Algorithm 4 below. Like the usual power method, it works by starting with a random vector and applying A some K times to it; the resulting vector will converge to the top eigenvector after relatively small K, assuming some gap between the first and second eigenvalues of A. We include a short proof explaining how the noisy power method can approximate the top eigenvector of A even if there is a small noise vector G_k in the kth matrix-vector computation that does not have too much overlap with v_1.

input : a Hermitian matrix $A \in [-1,1]^{d\times d}$ with operator norm $\|A\| \leq 1$;
Let w_0 be a unit vector randomly chosen from S^{d-1};
for $k \leftarrow 0$ **to** $K-1$ **do**
  $y_k = Aw_k + G_k$;
  $w_{k+1} = y_k/\|y_k\|_2$;
end
output: w_K;

Algorithm 4: Noisy power method (NPM) for approximating the top eigenvector of A

Theorem 4.8. *Let A be a $d\times d$ Hermitian matrix with top eigenvector v_1, first and second eigenvalues λ_1 and λ_2, and $\gamma = |\lambda_1| - |\lambda_2|$. Let $\varepsilon \in (0, 0.5)$ and $K = \frac{10|\lambda_1|}{\gamma}\log(20d/\varepsilon)$.*[10] *Suppose G_k satisfies $|\langle G_k, v_1\rangle| \leq \gamma/(50\sqrt{d})$ and $\|G_k\|_2 \leq \varepsilon\gamma/5$ for all $k \in [K]-1$. Then the unit vector w_K in Algorithm 4 satisfies $|\langle w_K, v_1\rangle| \geq 1 - \varepsilon^2/2$ with probability ≥ 0.9.*

Proof. Let $w_0 = \sum_{i\in[d]} \alpha_i^{(0)} v_i$. Because w_0 is chosen uniformly at random over the unit sphere, by Corollary 2.57 (without loss of generality assuming $d \geq 4$), with probability ≥ 0.9, we have $|\alpha_1^{(0)}| \geq 1/(10\sqrt{d})$ and hence we assume $|\alpha_1^{(0)}| \geq 1/(10\sqrt{d})$ below for simplicity. Suppose $w_k = \sum_{i\in[d]} \alpha_i^{(k)} v_i$. We define the tangent angle of w_k as $\tan\theta_k =$

and 1 (and hence one can use Algorithm 4 to find v_1). This trick can also be used to find v_d by simply considering $A'' = -A/3 + 2I/3$. Note that the gap between the top and the second eigenvalues of A' might be different from the gap between the top and the second eigenvalues of A''.

[10] If $K \geq \frac{10|\lambda_1|}{\gamma}\log(20d/\varepsilon)$, then the theorem still holds.

$\frac{\sin\theta_k}{\cos\theta_k} = \frac{\sqrt{\sum_{i=2}^d (\alpha_i^{(k)})^2}}{|\alpha_1^{(k)}|}$, and hence $\tan\theta_0 \le 10\sqrt{d}$. It suffices to show that $\tan\theta_K \le \varepsilon/2$, because that implies $|\langle w_K, v_1\rangle| = \cos\theta_K \ge 1-\varepsilon^2/4$.

Since $w_k = \sum_{i\in[d]} \alpha_i^{(k)} v_i$, we have $Aw_k = \sum_{i\in[d]} \alpha_i^{(k)} \lambda_i v_i$. Also because G_k satisfies $|\langle G_k, v_1\rangle| \le \gamma/(50\sqrt{d})$ and $\|G_k\|_2 \le \varepsilon\gamma/50$, we can give an upper bound for $\tan\theta_{k+1}$ as follows:

$$\begin{aligned}
\tan\theta_{k+1} &\le \frac{\sqrt{\sum_{i=2}^d (\lambda_i^2\alpha_i^{(k)})^2} + \|G_k\|_2}{|\lambda_1|\cdot|\alpha_1^{(k)}| - |\langle G_k, v_1\rangle|} \le \frac{|\lambda_2|\sqrt{\sum_{i=2}^d (\alpha_i^{(k)})^2} + \varepsilon\gamma/5}{|\lambda_1|\cdot|\alpha_1^{(k)}| - \gamma/(50\sqrt{d})} \le \frac{1}{\cos\theta_k}\cdot\frac{|\lambda_1|\sin\theta_k + \varepsilon\gamma/5}{|\lambda_1| - \gamma/5} \\
&= \frac{1}{\cos\theta_k}\cdot\frac{|\lambda_1|\sin\theta_k + \varepsilon\gamma/5}{|\lambda_2| + 4\gamma/5} = \frac{\sin\theta_k}{\cos\theta_k}\cdot\frac{|\lambda_1|}{|\lambda_2|+4\gamma/5} + \frac{1}{\cos\theta_k}\cdot\frac{\varepsilon\gamma/5}{|\lambda_2|+4\gamma/5} \\
&\le \tan\theta_k\cdot\frac{|\lambda_1|}{|\lambda_2|+4\gamma/5} + (1+\tan\theta_k)\cdot\frac{\varepsilon\gamma/5}{|\lambda_2|+4\gamma/5} \\
&= \Big(1 - \frac{\gamma/5}{|\lambda_2|+\gamma/5}\Big)\frac{|\lambda_2|+\varepsilon\gamma/5}{|\lambda_2|+3\gamma/5}\tan\theta_k + \frac{\gamma/5}{|\lambda_2|+4\gamma/5}\varepsilon \le \max\{\varepsilon, \frac{|\lambda_2|+\varepsilon\gamma/5}{|\lambda_2|+3\gamma/5}\tan\theta_k\}.
\end{aligned}$$

Note that $\frac{|\lambda_2|+\varepsilon\gamma/5}{|\lambda_2|+3\gamma/5} \le \max\{\varepsilon, \frac{|\lambda_2|}{|\lambda_2|+2\gamma/5}\}$ because the left-hand side is a weighted mean of the components on the right ($\frac{|\lambda_2|+\varepsilon\gamma/5}{|\lambda_2|+3\gamma/5} = \varepsilon\cdot\frac{\gamma/5}{|\lambda_2|+3\gamma/5} + \frac{|\lambda_2|}{|\lambda_2|+2\gamma/5}\cdot\frac{|\lambda_2|+2\gamma/5}{|\lambda_2|+3\gamma/5}$). Also, $\frac{|\lambda_2|}{|\lambda_2|+2\gamma/5} \le (\frac{|\lambda_2|}{|\lambda_2|+5\gamma/5})^{2/5} = (\frac{|\lambda_2|}{|\lambda_1|})^{2/5}$, so we have $\tan\theta_{k+1} \le \max\{\varepsilon, \tan\theta_k\cdot\max\{\varepsilon, (|\frac{\lambda_2}{\lambda_1}|)^{2/5}\}\}$. By letting $K = \frac{10|\lambda_1|}{\gamma}\log(20d/\varepsilon)$, we obtain $\tan\theta_K \le \varepsilon/2$, which concludes the proof. □

4.4.2 Quantum noisy power method using Gaussian phase estimator

In this subsection we combine the noisy power method and the quantum Gaussian phase estimator (introduced in the previous subsection) to get a quantum version of the noisy power method. It approximates the top eigenvector of a given matrix A with additive ℓ_2-error ε in $\tilde{\mathcal{O}}(d^{1.75}/(\gamma^2\varepsilon))$ time, which is a factor $d^{0.25}$ faster in its d-dependence than the best-possible classical algorithm (see Section 7.3 for the $\Omega(d^2)$ classical lower bound).

We first prove the following theorem, which helps us estimate an individual entry of a matrix-vector product Aw (the vector u would be one of the rows of A).

Theorem 4.9 (Inner product estimator, IPE$(\tau,\delta,\varepsilon)$)**.** *Let $\tau,\delta,\varepsilon \in (0, 0.1]$, and $u, w \in B_2^d$ s.t. $\|w\|_2 = 1$. Suppose we can access a KP-tree KP_w of w and have quantum query access to entries of u by a unitary O_u. There is a quantum algorithm that with probability at least $1-\delta$, outputs an estimator $\tilde{\mu}$ satisfying that $\tilde{\mu} - \langle u, w\rangle$ is τ-close to τ-subG(ε^2), using time $\tilde{\mathcal{O}}(d^{0.75}\,poly\log(d/\delta) + d^{0.25}\,poly\log(1/\tau)/\varepsilon)$.*

Proof. In this proof, we index entries of vectors starting from 0. Let $I_1 = [-d^{-0.25}, d^{-0.25}]$ and $I_2 = [-1,1]\setminus I_1$. Let $u = u_1 + u_2$, where $(u_1)_j = u_j \mathbb{1}_{I_1}(u_j)$ and $(u_2)_j = u_j\mathbb{1}_{I_2}(u_j)$ for every $j \in [d]-1$. Informally, u_1 is the vector with smallish entries, and u_2 is the vector with largish entries. We separately estimate $\langle u_1, w\rangle$ and $\langle u_2, w\rangle$.

Finding u_2 and computing $\langle u_2, w\rangle$. The number of nonzero entries of u_2 is at most $\sqrt{d}$ because $\|u\|_2 \le 1$. We first find (with success probability at least $1-\delta/2$) all the nonzero entries of u_2 in $\mathcal{O}(\sqrt{d\cdot\sqrt{d}}\cdot \mathrm{polylog}(d/\delta)) = \mathcal{O}(d^{0.75}\cdot\mathrm{polylog}(d/\delta))$ time using Theorem 2.7. Since we can query the entries of w through its KP-tree, we can now compute $\langle u_2, w\rangle$ in time $\tilde{\mathcal{O}}(\sqrt{d})$.

Estimating $\langle u_1, w\rangle$. Our goal below is first to show how to prepare (in time $\tilde{\mathcal{O}}(1)$) a superposition corresponding to the vector $d^{-0.25}u_1$, using the fact that all entries of u_1 are small; and then to use this to estimate $\langle u_1, w\rangle$ with Gaussian error in time $\tilde{\mathcal{O}}(d^{0.25}/\varepsilon)$.

We can implement O_{u_1} using 2 queries to O_u and $\tilde{\mathcal{O}}(1)$ elementary gates: query O_u, and then apply O_u^{-1} conditional on the magnitude of the value being $> d^{-0.25}$ to set the value back to 0 for entries that are in the support of u_2 rather than u_1. Let CR be a controlled rotation such that for every $a \in [-d^{-0.25}, d^{-0.25}]$

$$CR\,|a\rangle\,|0\rangle = |a\rangle\,(a\cdot d^{0.25}\,|0\rangle + \sqrt{1-a^2\cdot d^{0.5}}\,|1\rangle).$$

This can be implemented up to negligibly small error by $\tilde{\mathcal{O}}(1)$ elementary gates. Using one application each of O_{u_1}, $O_{u_1}^{-1}$, and CR, and $\tilde{\mathcal{O}}(1)$ elementary gates, we can map

$$\begin{aligned}
|0^{\otimes \log d}\rangle\,|0\rangle\,|0\rangle &\xrightarrow{H^{\otimes\log d}\otimes I} \sum_{j\in[d]-1} d^{-0.5}\,|j\rangle\,|0\rangle\,|0\rangle \xrightarrow{O_{u_1}\otimes I} \sum_{j\in[d]-1} d^{-0.5}\,|j\rangle\,|(u_1)_j\rangle\,|0\rangle \\
&\xrightarrow{I_d\otimes CR} \sum_{j\in[d]-1} (d^{-0.25}(u_1)_j\,|j\rangle\,|(u_1)_j\rangle\,|0\rangle + (\sqrt{d^{-1}-(u_1)_j^2\cdot d^{-0.5}}\,|j\rangle\,|(u_1)_j\rangle\,|1\rangle) \\
&\xrightarrow{O_{u_1}^{-1}\otimes I} \sum_{j\in[d]-1} (d^{-0.25}(u_1)_j\,|j\rangle\,|0\rangle\,|0\rangle + (\sqrt{d^{-1}-(u_1)_j^2\cdot d^{-0.5}}\,|j\rangle\,|0\rangle\,|1\rangle).
\end{aligned}$$

Swapping the second register to the front of the state, we showed how to implement the state-preparation unitary $U_{d^{-0.25}u_1}$ that maps

$$U_{d^{-0.25}u_1} : |0\rangle\,|0^{\otimes\log d}\rangle \to |0\rangle \sum_{j\in[d]-1} (d^{-0.25}(u_1)_j)\,|j\rangle + |1\rangle\,|\Phi\rangle\,,$$

for some arbitrary unnormalized state $|\Phi\rangle$.

Now we show how to estimate $\langle u_1, w\rangle$. Since we have a KP-tree of w, we can implement the state-preparation unitary U_w that maps $|0^{\otimes\log d}\rangle \to \sum_{j\in[d]-1} w_j\,|j\rangle$ using $\tilde{\mathcal{O}}(1)$ time by Theorem 2.11. Let $W = I\otimes U_w\otimes Z$ and $V = U_{d^{-0.25}u_1}\otimes I$. Using Theorem 2.20 (with x ranging over 2 cases), using $\tilde{\mathcal{O}}(1)$ time we can implement a $(1, \log d+2, 0)$ block-encoding of $\mathrm{diag}(\{d^{-0.25}\langle u_1, w\rangle, -d^{-0.25}\langle u_1, w\rangle\})$. In order to be able to use our Gaussian phase estimator, we want to convert $\langle u_1, w\rangle$ into an eigenphase. To that end, by Theorem 2.16, we implement a unitary U_{exp} which is a $(1, \log d+2, 0)$-block-encoding of $W = \exp(i\pi(\langle u_1, w\rangle/4)Z)$ using $\tilde{\mathcal{O}}(d^{0.25})$ time. Since $|0\rangle$ is an eigenvector of W with eigenvalue $\exp(i\pi\langle u_1, w\rangle/4)$, our Gaussian phase estimator (Corollary 4.3) can output (with success probability $\ge 1-\delta/2$) an estimator $\tilde{\eta}$ such that $\tilde{\eta} - \langle u_1, w\rangle$ is τ-close to

τ-subG(ε^2) using $\tilde{\mathcal{O}}(\text{polylog}(1/\tau)/\varepsilon)$ applications of controlled-U_{exp}, controlled-U_{exp}^{-1}, and time.

Because $\langle u, w\rangle = \langle u_1, w\rangle + \langle u_2, w\rangle$, with probability at least $1-\delta$ we obtain an estimator $\tilde{\mu} = \tilde{\eta} + \langle u_2, w\rangle$ such that $\tilde{\mu} - \langle u, w\rangle = \tilde{\eta} - \langle u_1, w\rangle$ is τ-close to τ-subG(ε^2). □

Note that the two terms in the time complexity of the above theorem are roughly equal if ε is a small constant times $1/\sqrt{d}$. Such a small error-per-coordinate translates into a small overall ℓ_2-error for a d-dimensional vector. Accordingly, such a setting of ε is what we use in our quantum noisy power method for estimating the top eigenvector.

input : a Hermitian matrix $A \in [-1,1]^{d\times d}$ with operator norm at most 1;
Let $\gamma = |\lambda_1(A)| - |\lambda_2(A)|$; $\varepsilon \in (0,1)$; $K = \frac{10|\lambda_1|}{\gamma}\log(20d/\varepsilon)$;
Let $\delta = 1/(1000Kd)$; $\tau = \delta/(1000Kd^2)$; $\zeta = \frac{\varepsilon\gamma}{100d^{0.5}\sqrt{\log(1000Kd/\delta)}}$;
Let w_0 be a unit vector randomly chosen from S^{d-1};
for $k \leftarrow 0$ **to** $K-1$ **do**
  Prepare a KP-tree for w_k;
  For every $j \in [d]$, compute an estimator $(y_k)_j$ of $\langle A_j, w_k\rangle$ using IPE(τ,δ,ζ) of Theorem 4.9 (A_j is the jth row of A);
  $w_{k+1} = y_k/\|y_k\|_2$;
end
output: w_K;

Algorithm 5: Quantum noisy power method using Gaussian phase estimator

Theorem 4.10 (Quantum noisy power method using Gaussian phase estimator, Algorithm 5)**.** *Let $A \in [-1,1]^{d\times d}$ be a symmetric matrix with operator norm at most 1, first and second eigenvalues $\lambda_1(A)$ and $\lambda_2(A)$, $\gamma = |\lambda_1(A)| - |\lambda_2(A)|$, $v_1 = v_1(A)$ be the top eigenvector of A, and $\varepsilon \in (0,1]$. Suppose we have quantum query access to entries of A. There exists a quantum algorithm (namely Algorithm 5) that with probability at least 0.89, outputs a d-dimensional vector w such that $|\langle w, v_1\rangle| \geq 1-\varepsilon^2/2$, using $\tilde{\mathcal{O}}(d^{1.75}/(\gamma^2\varepsilon))$ time and $\tilde{\mathcal{O}}(d)$ QRAM bits.*

Proof. Each iteration of Algorithm 5 uses $\tilde{\mathcal{O}}(d)$ time and QRAM bits to build the KP-tree for w_k, and for every $j \in [d]$, we use $\tilde{\mathcal{O}}(d^{0.75}\text{polylog}(d/\delta) + d^{0.25}\text{polylog}(1/\tau)/\zeta)$ time and $\tilde{\mathcal{O}}(\sqrt{d})$ QRAM bits for estimating $\langle A_j, w_\ell\rangle$ by IPE(τ,δ,ζ) in Theorem 4.9. Hence the total number of elementary gates we used and queries to entries of A is $\tilde{\mathcal{O}}(d\cdot(d^{0.75} + d^{0.25}/\zeta)\cdot K) = \tilde{\mathcal{O}}(d^{1.75}/(\gamma^2\varepsilon))$.

Now we are ready to show the correctness of Algorithm 5. By Theorem 4.8 and the union bound, it suffices to show that for each $k \in [K]-1$, both $\|y_k - Aw_k\|_2 \leq \gamma\varepsilon/5$ and $|\langle y_k - Aw_k, v_1\rangle| \leq \gamma/(50\sqrt{d})$ hold with probability $\geq 1-1/(100K)$. Fix k. By Theorem 4.9, we know that for every $j \in [d]$, with probability at least $1-\delta$, $(y_k - Aw_k)_j$ is τ-close to τ-subG(ζ^2) for every $j \in [d]$. Let $e = y_k - Aw_k$. There are three different kinds of bad events, whose probabilities we now analyze. Firstly, we can see that for every $j \in [d]$,

with probability at least $1-\delta$,

$$\Pr\left[|e_j| > \frac{\gamma\varepsilon}{5\sqrt{d}}\right] \le 2\exp(\tau)\cdot\exp\left(-\frac{(\frac{\gamma\varepsilon}{5\sqrt{d}})^2}{2\zeta^2}\right)+\tau = 2\exp(\tau-200\log(1000Kd/\delta))+\tau \le \frac{\delta}{100Kd}.$$

Therefore, with probability at least $1-d\delta-\delta/(100K)$, $|e_j| \le \frac{\gamma\varepsilon}{5\sqrt{d}}$ for every $j\in[d]$, implying that $\|e\|_2 \le \frac{\gamma\varepsilon}{5}$. Secondly, by the properties of sub-Gaussians from Section 2.7.1, we know that with probability at least $1-d\delta$, $\langle e, v_1\rangle$ is $d\tau$-close to $d\tau$-subG($\sum_{j\in[d]}(v_1)_j^2\zeta^2$). Thirdly, since v_1 is a unit vector, we have that (with probability at least $1-d\delta$) $\langle e, v_1\rangle$ is $d\tau$-close to $d\tau$-subG(ζ^2) and hence

$$\begin{aligned}\Pr\left[|\langle e, v_1\rangle| > \frac{\gamma}{50\sqrt{d}}\right] &\le 2\exp(d\tau)\cdot\exp\left(-\frac{(\frac{\gamma}{50\sqrt{d}})^2}{2\zeta^2}\right)+d\tau \\ &= 2\exp(d\tau)\cdot\exp\left(-2\frac{\log(1000Kd/\delta)}{\varepsilon^2}\right)+d\tau \le \frac{\delta}{100Kd}.\end{aligned}$$

As a result, by the union bound over the three kinds of error probabilities, for each $k\in[K]\cup\{0\}$, with probability $1-2d\delta-\delta/(50K)\ge 1-1/(100K)$, we have both $\|y_k - Aw_k\|_2 \le \gamma\varepsilon/5$ and $|\langle y_k - Aw_k, v_1\rangle| \le \gamma/(50\sqrt{d})$. This proves correctness of Algorithm 5. □

4.4.3 Almost optimal process-tomography of "low-rank" reflections

In this subsection we describe an essentially optimal algorithm for the "tomography" of projectors Π of rank at most q (or of the corresponding reflection $2\Pi - I$, which is a unitary).[11] We will use this in the next subsection to approximate the eigensubspace spanned by the top-q eigenvectors. For generality, we will from here on allow our matrices to have complex entries, not just real entries like in the earlier subsections.

Our algorithm is inspired by the noisy power method and has query complexity $\widetilde{\mathcal{O}}(dq/\varepsilon)$ and time complexity $\widetilde{\mathcal{O}}(dq/\varepsilon + dq^2)$ (using QRAM). When $q \ll d$ this gives a better complexity than the optimal unitary process-tomography algorithm of Haah, Kothari, O'Donnell, and Tang [HKO+23]. Also in the special case when $q=1$ this gives a qualitative improvement over prior pure-state tomography algorithms [KP20; ACG+23] which required a state-preparation unitary, while for us it suffices to have a reflection about the state, which is a strictly weaker input model.[12] Surprisingly, it turns out that this weaker input essentially does not affect the query and time complexity.

[11] Having access to a controlled reflection $2\Pi - I$ is equivalent up to constant factors to having access to controlled $U_\Pi^{\pm 1}$ (i.e., controlled-U_Π and its inverse) for a block-encoding U_Π of the projector Π, as follows from the QSVT framework [GSL+19].

[12] Indeed, we can implement a reflection about a state $|\psi\rangle$ by a state-preparation unitary and its inverse as in amplitude amplification. However, if we only have access to a reflection about an unknown classical basis state $|i\rangle$ for $i\in[d]$, then we need to use this reflection $\Omega(\sqrt{d})$ times to find i (because of the optimality of Grover search) showing that the reflection input is substantially weaker than the state-preparation-unitary input.

Observe that if we have two projectors Π, Π' of rank r, then

$$\|\Pi - \Pi'\|_1/(2r) \le \|\Pi - \Pi'\| \le \|\Pi - \Pi'\|_1/2. \tag{4.12}$$

This implies that the ε-precise estimation of a rank-1 projector Π (i.e., the density matrix corresponding to a pure state) is equivalent to ε-precise approximation of the quantum state Π. Since the complexity of pure-state tomography using state-preparation unitaries is known to be $\widetilde{\Theta}(d/\varepsilon)$ [ACG+23], it follows that our algorithm is optimal up to log factors in the $q = 1$ case.

Similarly, the query complexity of our algorithm is optimal for $\varepsilon = \frac{1}{6}$ up to log factors, as can be seen by an information-theoretic argument using ε-nets, since by Lemma 4.12 there is an ensemble of $\exp(\Omega(dq))$ rank-q' projectors for $q' = \min(q, \lfloor \frac{d}{2} \rfloor)$ such that each pair of distinct projectors is more than $\frac{1}{3}$ apart in operator norm. We conjecture that the query complexity of the task is actually $\widetilde{\Omega}(dq/\varepsilon)$, meaning that our algorithm has essentially optimal query complexity for all $\varepsilon \in (0, \frac{1}{6}]$.

Now we briefly explain our algorithm for finding an orthonormal basis of the image of Π, assuming for ease of exposition that its rank is exactly q. We want to find a $d \times q$ isometry W such that $\|WW^\dagger - \Pi\|$ is small. Our algorithm (Algorithm 6 below) can be seen as a variant of the noisy power method: We start with generating m Gaussian vectors $g_1, \ldots, g_m$ ($m = \widetilde{\Theta}(q)$ will be slightly bigger than q), with i.i.d. (complex) normal entries each having standard deviation $\sim 1/\sqrt{d}$. Subsequently, we repeat the following process $K \approx \log \sqrt{d/m}$ times: we estimate Πg_i for every $i \in [m]$ using our quantum state tomography algorithm (Theorem 4.6 and Corollary 4.7), and multiply the outcome by 2, resulting in m new vectors (these factors of 2 allows our analysis to treat all the errors together in one geometric series later). Finally, let V be the $d \times m$ matrix whose columns are the versions of those m vectors after the last iteration. The algorithm classically computes the singular value decomposition (SVD) of this V, and outputs those left-singular vectors $w_1, w_2, \ldots, w_q$ that have singular value greater than the threshold of $\frac{1}{14}$.

Note that the span of $g_1, \ldots, g_m$ includes the q-dimensional image of Π almost surely. Hence if no error occurs in the tomography step, then $V = 2^K \Pi[g_1, \ldots, g_m]$, and the image of V is the image of Π (almost surely). Thus, if $\sum_{i=1}^{q} \varsigma_i w_i u_i^\dagger$ is the SVD of V, then $\sum_{i \in [q]} w_i w_i^\dagger = \Pi$. That is, we get the desired output W by rounding the singular values of V appropriately: the first q singular values are rounded to 1, and the others are rounded to 0. To show the stability of this approach it remains to show that the singular values $\varsigma_1, \ldots, \varsigma_q$ of the final V in the non-error-free case are still large (≈ 1), the other $d - q$ singular values are still essentially 0, and the error incurred by the quantum state tomography is small. Our analysis relies on the concentration of the singular values of sufficiently random matrices, see Section 2.7.3.

To bound the effect of errors induced by tomography we combine the operator norm bound on random matrices of Theorem 2.32 with our unbiased tomography algorithm (Theorem 4.6). The key observation is that Corollary 4.7 gives an estimator $\tilde{\psi}$ that is δ-close in total variation distance to an "ideal" (though not error-free) estimator $\breve{\psi}$ that satisfies $\mathbb{E}[\breve{\psi}] = \psi$, whose covariance matrix has operator norm at most $S^2 \le \frac{\varepsilon^2}{d}$,

and $\|\Pi\breve{\psi} - \Pi\psi\|_2 \le \varepsilon\sqrt{\frac{q}{d}}$ with certainty. For the sake of analysis we can assume that we work with $\breve{\psi}$, because we only notice the difference between $\breve{\psi}$ and $\tilde{\psi}$ with probability at most δ, which can be made negligibly small at a logarithmic cost in Corollary 4.7. Finally, we use perturbation bounds on singular values and vectors from Section 2.6.

Input : Dimension d, failure probability $\delta' \in (0, \frac{1}{2}]$, target precision $\varepsilon \in (0, \frac{1}{2}]$, $\widetilde{\mathcal{O}}\left(\frac{\varepsilon\delta'}{dq}\right)$-approximate block-encoding U_Π of a projector Π of rank $\le q$

Output: ε-approximate orthonormal basis W of the image of Π, i.e., $\left\|WW^\dagger - \Pi\right\| \le \varepsilon$

Init: $m := \min\left(\left\lceil \max\left(16C^2 q, 8c\ln\left(\frac{10}{\delta'}\right)\right)\right\rceil, d\right)$, $K := \left\lceil \log_2\left(\sqrt{\frac{d}{m}} + 1\right)\right\rceil$, $\varepsilon' := \frac{\varepsilon}{65+98\sqrt{\ln\left(\frac{10d}{\delta'}\right)/c'}}$

// the constants c, C come from Corollary 2.30 and in the real case[4] $c, C = 1$

// the constant c' comes from Theorem 2.32

1.) **if** $m = d$ **then** set $g_j = |j\rangle/4$, **else** generate m random vectors g_j with i.i.d. complex (or real if $\Pi \in \mathbb{R}^{d\times d}$) standard normal entries multiplied by $\eta := \frac{4\cdot 2^{-K}}{7\sqrt{m}}$

// note $\eta < \frac{4}{7\sqrt{d}}$

2.) **for** $j = 1$ **to** m **do**

  Set $g_j^{(0)} = g_j$; **if** $\|g_j\| > 2$ **then ABORT**

  for $k = 0$ **to** $K - 1$ **do**

    Classically compute $\|g_j^{(k)}\|$ and store $g_j^{(k)}/\|g_j^{(k)}\|$ in a KP-tree

    (we can now unitarily prepare $|\psi\rangle = \Pi g_j^{(k)}/\|g_j^{(k)}\|$ using this KP-tree and U_Π)

    Obtain $y_j^{(k+1)}$ via $\varepsilon'/\|g_j^{(k)}\|$-precise tomography (Corollary 4.7) on $|\psi\rangle$ setting $\delta \leftarrow \delta'/(5mK)$ // query complexity is $\widetilde{\mathcal{O}}(d/\varepsilon)$

    Set $g_j^{(k+1)} \leftarrow 2\|g_j^{(k)}\| y_j^{(k+1)}$; **if** $\|g_j^{(k+1)}\| > 2$ **then ABORT**

  endfor

endfor

3.) Output the left-singular vectors of $V = [g_1^{(K)}, \ldots, g_m^{(K)}]$ with singular value above $\frac{1}{14}$.

// Classical complexity is $\widetilde{\mathcal{O}}\left(dq^2\right)$ via diagonalization of $V^\dagger V$

// The output is correct with probability at least $1 - \delta'$

// The total U_Π-query and quantum time complexity is $\widetilde{\mathcal{O}}\left(dq/\varepsilon\right)$

Algorithm 6: Time-efficient approximation of the top-q eigensubspace

We say that U_Π is an ε-approximate block-encoding of Π if $\|U_\Pi - U\| \le \varepsilon$ for some U satisfying $\Pi = (\langle 0^a| \otimes I)U(|0^a\rangle \otimes I)$.[13]

[13] The condition $\|(\langle 0^a| \otimes I)U_\Pi(|0^a\rangle \otimes I) - \Pi\| \le \varepsilon'$ appears similar, however is in some sense quadratically

Theorem 4.11 (Correctness of Algorithm 6)**.** *Let $\Pi \in \mathbb{C}^{d\times d}$ be an orthonormal projector of rank at most q, given via an $\widetilde{\mathcal{O}}\left(\frac{\varepsilon\delta'}{dq}\right)$-approximate block-encoding U_Π. Algorithm 6 outputs an isometry W such that, with probability at least $1-\delta'$, $\|WW^\dagger - \Pi\| \le \varepsilon$, using $\widetilde{\mathcal{O}}\left(\frac{dq}{\varepsilon}\right)$ controlled U_Π, $U_\Pi^\dagger$, two-qubit quantum gates, read-outs of a QRAM of size $\tilde{\mathcal{O}}(d)$, and $\widetilde{\mathcal{O}}\left(dq^2\right)$ classical computation.*

Proof. First consider the case when there is no error in tomography and in the implementation of Π. Then we end up with $g_j^{(K)} = 2^K \Pi g_j^{(0)}$, and the corresponding matrix $V_{ideal} = 2^K[\Pi g_1^{(0)}, \ldots, \Pi g_m^{(0)}]$ has almost surely rank(Π) nonzero singular values with associated left-singular vectors lying in the image of Π. The $m = d$ case is trivial. If $m < d$, then due to Corollary 2.30 all corresponding singular values are in $(\frac{1}{7}, 1)$ with probability at least $1 - \frac{\delta'}{5}$.[14] By Proposition 2.31 (and a union bound over all $j \in [m]$) we know that with probability at least $1 - \frac{\delta'}{10}$ we have for all $j \in [m]$ that[15] $\|g_j\| < \eta(\sqrt{d} + \sqrt{2\ln(10m/\delta')}) < \frac{6}{7}$. Thereby with probability at least $1 - \frac{3\delta'}{10}$ Algorithm 6 does not abort at the intialization of $g_j^{(0)}$, and the left-singular vectors of V_{ideal} with singular value at least $1/7$ form the columns of the desired matrix W such that $WW^\dagger = \Pi$; in the remainder of the proof we assume this is the case.

Second, we consider what happens when the tomography has error, but we can implement Π exactly. Let $\tilde{e}_j^{(k,\ell)} := g_j^{(k)} - 2^{k-\ell}\Pi g_j^{(\ell)}$ be the aggregate tomography error that occurred from the ℓ-th iteration to the k-th iteration where $k \in [K]$, $\ell \in [k]-1$, and observe that $\tilde{e}_j^{(k,\ell)} = \tilde{e}_j^{(k,k-1)} + \sum_{i=\ell+1}^{k-1} 2^{k-i} \cdot \Pi \tilde{e}_j^{(i,i-1)}$. For each iteration, we do the tomography with precision $\varepsilon'/\|g_j^{(k)}\|$ on $\Pi g_j^{(k)}/\|g_j^{(k)}\|$ via Corollary 4.7, guaranteeing that the random variable $\tilde{e}_j^{(k,k-1)}$ is δ-close in total variation distance to an "ideal" random variable $e_j^{(k,k-1)}$ such that $\|e_j^{(k,k-1)}\| \le \varepsilon'$ and $\|\Pi e_j^{(k,k-1)}\| \le \varepsilon'\sqrt{q/d}$ almost surely, and $\mathbb{E}[e_j^{(k,k-1)}] = 0$, $\|\mathrm{Cov}(e_j^{(k,k-1)})\| \le \frac{\varepsilon'}{d^2}$ (to see this, choose V to be an orthonormal basis whose first q elements span the image of Π). We define analogously $e_j^{(k,\ell)} := e_j^{(k,k-1)} +$

---

weaker. Consider, e.g., $\Pi = 1$, $a = 1$, and $U_\Pi = \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix}$, then $1 - \cos(x) = x^2/2 + \mathcal{O}(x^4)$, but for any unitary $U = |0\rangle\langle 0| + z|1\rangle\langle 1|$ we have $\|U_\Pi - U\| \ge |\sin(x)| = |x| + \mathcal{O}(|x|^3)$. Nevertheless, because Π is a projector, we can remedy this in general by converting U_Π to an (approximate) block-encoding of $\Pi/2$ via linear combination of unitaries, and then applying quantum singular value transformation with the polynomial $-T_3(x) = 3x - 4x^3$; the resulting unitary is then indeed $\mathcal{O}(\varepsilon')$-close to a perfect block-encoding of Π, see for example the proof of [GSL+19, Lemma 23].

[14] The matrix $\frac{1}{\eta}[g_1^{(0)}, \ldots, g_m^{(0)}]$ is a $d \times m$ random matrix with i.i.d. (complex) standard normal entries. After multiplying by Π this effectively (up to a rotation) becomes a $q \times m$ random matrix with i.i.d. (complex) standard normal entries. We apply Corollary 2.30 (with $N = m$) to the latter matrix, obtaining the interval $[\frac{1}{4}\sqrt{m}, \frac{7}{4}\sqrt{m}]$ for its singular values. Multiplying by $\eta 2^K = 4/(7\sqrt{m})$ we get the interval $[\frac{1}{7}, 1]$ for the singular values of V_{ideal}.

[15] We have $\eta(\sqrt{d} + \sqrt{2\ln(10m/\delta')}) \le \eta\sqrt{d} + \frac{2}{7\sqrt{m}}\sqrt{2\ln(10m/\delta')} < \frac{4}{7} + \frac{2}{7}\sqrt{\frac{2\ln(10m/\delta')}{m}} = \frac{4}{7} + \frac{2}{7}\sqrt{\frac{2\ln(m)+2\ln(10/\delta')}{m}} < \frac{6}{7}$, because $m \ge 8\ln(10/\delta')$ and $\frac{2\ln(x)}{x}$ takes its maximum at $x = e$, where it is less than $\frac{3}{4}$.

$\sum_{i=\ell+1}^{k-1} 2^{k-i} \cdot \Pi e_j^{(i,i-1)}$. Note that the distribution of $\tilde{e}_j^{(k,k-1)}$ (and $e_j^{(k,k-1)}$) can depend on $\tilde{e}_j^{(i,i-1)}$ (and $e_{j'}^{(i,i-1)}$, respectively) only if $j = j'$ and $i \le k$. Let $\vec{\tilde{e}}_j := (\tilde{e}_j^{(1,0)}, \tilde{e}_j^{(2,1)}, \ldots, \tilde{e}_j^{(K,K-1)})$, and define $\vec{e}_j$ analogously. We can apply Lemma 2.36 recursively to show that $d_{TV}(\vec{\tilde{e}}_j, \vec{e}_j) \le K\delta$. Since the $\vec{\tilde{e}}_j$ are independent from each other, we can assume without loss of generality that so are the $\vec{e}_j$. Once again by Lemma 2.36 we get that when comparing the two sequences of m random variables, we have $d_{TV}((\vec{\tilde{e}}_j : j \in m), (\vec{e}_j : j \in m)) \le mK\delta = \frac{\delta'}{5}$. From now on we replace $(\vec{\tilde{e}}_j : j \in m)$ by $(\vec{e}_j : j \in m)$ throughout the analysis, which can therefore hide an additional failure probability of at most $\frac{\delta'}{5}$.

By the triangle inequality we have that

$$\begin{aligned}\|e_j^{(k,0)}\| &\le \|e_j^{(k,k-1)}\| + \sum_{i=1}^{k-1} 2^{k-i}\|\Pi e_j^{(i,i-1)}\| \le \varepsilon' + \sum_{i=1}^{k-1} 2^{k-i}\varepsilon'\sqrt{q/d} \\ &\le (1 + 2^k\sqrt{q/d})\varepsilon' < (1 + 2(\sqrt{d/m}+1)\sqrt{q/d})\varepsilon' \le 5\varepsilon'\end{aligned}$$

for every $k \in [K]$. This also implies that for every $k \in [K]$ we have $\|g_j^{(k)}\| \le \|2^k \Pi g_j^{(0)}\| + \|e_j^{(k,0)}\| \le \|2^K \Pi g_j^{(0)}\| + 5\varepsilon' \le 1 + 5\varepsilon' \le 2$, and therefore Algorithm 6 also does not abort in the for-loop.

Let us define $E_{tomo} := [e_1^{(K,0)}, \ldots, e_m^{(K,0)}] = V - V_{ideal}$ as the matrix of accumulated tomography errors. We can apply Lemma 2.37 recursively with $X = \Pi e_j^{(k-1,0)}$, $Y = (I - \Pi)e_j^{(k-1,0)}$, $Z = e_j^{(k,k-1)}$ to show that for all $k \in [K]$ we have

$$\begin{aligned}\|\mathrm{Cov}(e_j^{(k,0)})\| &\le \|\mathrm{Cov}(e_j^{(k,k-1)})\| + \sum_{i=1}^{k-1} 4^{k-i}\|\mathrm{Cov}(\Pi e_j^{(i,i-1)})\| \\ &\le \|\mathrm{Cov}(e_j^{(k,k-1)})\| + \sum_{i=1}^{k-1} 4^{k-i}\|\mathrm{Cov}(e_j^{(i,i-1)})\| \\ &\le \sum_{i=1}^{k} 4^{k-i}\frac{\varepsilon'^2}{d} \le \frac{4^K\varepsilon'^2}{3d} \le \frac{4(\sqrt{d/m}+1)^2\varepsilon'^2}{3d} \le \frac{16\varepsilon'^2}{3m}.\end{aligned}$$

Applying Theorem 2.32 for $t = \sqrt{\ln(10d/\delta')/c'}$ gives that with probability at least $1 - \frac{\delta'}{5}$ we have

$$\|E_{tomo}\| \le \frac{8\varepsilon'}{\sqrt{3}} + 7\varepsilon'\sqrt{\ln(10d/\delta')/c'} \le \frac{\varepsilon}{14}. \tag{4.13}$$

Since $\|E_{tomo}\| \le \frac{\varepsilon}{14}$, using the notation of Theorem 2.22 we have that $\Pi = \Pi_{>0}^{V_{ideal}}$ and the rank of $\Pi_{>\frac{1}{14}}^{V}$ is $\mathrm{rank}(\Pi) \le q$ due to Weyl's bound (Theorem 2.21). Therefore, by Theorem 2.22 and Lemma 2.23 we have $\|\Pi - \Pi_{>\frac{1}{14}}^{V}\| \le 14\|V - V_{ideal}\| = 14\|E_{tomo}\| \le \varepsilon$ as desired.

Finally, let us analyze the effect of implementation errors in U_Π. We perform tomography mK times via Corollary 4.7, each time using $T = \mathcal{O}(\frac{d}{\varepsilon}\mathrm{polylog}(d/(\varepsilon\delta))$ appli-

cations of $U_\Pi^{\pm 1}$, therefore the induced total variation distance[16] in the output distribution is at most $TmK \cdot \widetilde{\mathcal{O}}\left(\frac{\varepsilon\delta'}{dq}\right) \leq \frac{\delta'}{10}$. Preparing a KP-tree that allows a similar precision for the preparation of $g_j^{(k)}/\|g_j^{(k)}\|$ likewise induces at most an additional $\frac{\delta'}{10}$ total variation distance, implying that our algorithm outputs a sufficiently precise answer with probability at least $1-\delta'$ when all approximations are considered. The quantum time complexity comes entirely from Corollary 4.7, which is time efficient, while the final computation requires computing the SVD of a $d \times m$ matrix, which can be performed in $\widetilde{\mathcal{O}}\left(dm^2\right) = \widetilde{\mathcal{O}}\left(dq^2\right)$ classical time. □

Note that Algorithm 6 uses QRAM for time-efficiently preparing the quantum state $|\psi\rangle$. One can use Theorem 2.12 to avoid the usage of QRAM in Theorem 4.11. Though this will cost extra $\tilde{\mathcal{O}}(d)$ factor quantum gate complexity overhead, the number of applications of controlled $U_\Pi^{\pm}$ will still remain the same.

Using the following lemma and basic quantum information theory, one can see that recovering the q-dimensional subspace with small constant error $\varepsilon = 1/6$ gains us $\Omega(dq)$ bits of information about the subspace, and hence requires $\widetilde{\Omega}(dq)$ quantum queries. This shows that the query complexity of our previous algorithm is essentially optimal in its dq-dependence.

Lemma 4.12 (ε-net of subspaces)**.** *Let $q \leq d/2$. There exists a set S of q-dimensional subspaces of $\mathbb{R}^d$ of size $\exp(\Omega(dq))$ such that for any distinct $s, r \in S$ we have $\|\Pi_s - \Pi_r\| > \frac{1}{3}$, where Π_t denotes the projector to the subspace t.*

Proof. We can assume without loss of generality that $d \geq 128$[3].

First let us assume that $q \leq d/64$; we show the existence of such a set S via the probabilistic method, by showing that for any set S of subspaces, if $|S| < \exp(qd/32 - 1)$, then with non-zero probability a Haar-random q-dimensional subspace r satisfies $\|\Pi_s - \Pi_r\| > \frac{1}{3}$ for every $s \in S$ (thereby we can take $S \leftarrow S \cup \{r\}$). We sample r as follows: generate a random matrix $R \in \mathbb{R}^{d\times q}$ with i.i.d. standard normal entries, and accept R only if $\varsigma_{\min}(R) \geq \frac{3}{4}\sqrt{d}$ (i.e., take a sample conditioned on this happening – we know by Theorem 2.29 that this happens with probability $\geq \frac{1}{e}$).

Upon acceptance we compute a singular value decomposition $R = U\Sigma V^\dagger$ and define $\Pi = UU^\dagger$ as the projector corresponding to the subspace. Since $\Pi = UU^\dagger$ is an orthogonal projection to the image of $R = U\Sigma V^\dagger$ we have $\Pi r_i = r_i$ for all columns r_i of

[16] With more careful tracking of error spreading in the estimated vectors it might be possible to show that it suffices to have access to a block-encoding satisfying the weaker condition $\|(\langle 0^a| \otimes I)U_\Pi(|0^a\rangle \otimes I) - \Pi\| \leq \frac{\varepsilon'}{\sqrt{dm}}$.

R and thus

$$\begin{aligned}\|\Pi_s r_i\| &\geq \|\Pi r_i\| - \|(\Pi - \Pi_s) r_i\| \\ &= \|r_i\| - \|(\Pi - \Pi_s) r_i\| \\ &\geq (1 - \|\Pi - \Pi_s\|)\|r_i\| \\ &\geq (1 - \|\Pi - \Pi_s\|)\varsigma_{\min}(R) \\ &\geq \frac{3}{4}\sqrt{d}(1 - \|\Pi - \Pi_s\|).\end{aligned}$$

Hence $\|\Pi - \Pi_s\| \leq \frac{1}{3}$ is only possible if $\|\Pi_s r_i\| \geq \sqrt{d}/2$ for all columns of R. Without the conditioning (on $\varsigma_{\min}(R) \geq \frac{3}{4}\sqrt{d}$, the condition we accept R), $\Pi_s r_i$ is effectively a random q-dimensional vector with i.i.d. standard normal entries. Since $\sqrt{d}/2 - \sqrt{q} \geq \sqrt{d}/4$, by Proposition 2.31 for any $s \in S$ we have $\Pr[\|\Pi_s r_i\| \geq \sqrt{d}/2] \leq \exp(-d/32)$, and due to the independece of the columns the probability that this happens for all $i \in [q]$ is less than $\exp(1 - qd/32)$ even after conditioning. Taking the union bound over all $s \in S$ we can conclude that with non-zero probability $\|\Pi - \Pi_s\| > \frac{1}{3}$ for all $s \in S$.

The statement for $q = \Omega(d)$ follows from [HHJ+17, Lemma 8]. Alternatively, if $q > d/64$, we can set $q' := \lceil q/128 \rceil$, $d' := d - (q - q')$ so that $q' \leq d'/64$. Then from a large set S' of q'-dimensional subspaces of $\mathbb{R}^{d'}$ satisfying $\|\Pi_{s'} - \Pi_{r'}\| > \frac{1}{3}$ for any distinct $s', r' \in S'$ we construct $S := \{s \colon \Pi_s = \Pi_{s'} \oplus I_{q-q'} \text{ for some } s' \in S'\}$ so that also $\|\Pi_s - \Pi_r\| > \frac{1}{3}$ for any distinct $s, r \in S$. □

4.4.4 Approximating the subspace spanned by top-q eigenvectors

In this subsection, we give a quantum algorithm to "approximate the top-q eigenvectors" in a strong sense using $qd^{1.5+o(1)}$ time (and $q\sqrt{s}d^{1+o(1)}$ time if the matrix is s-sparse). In particular, when $q = 1$, this algorithm outputs a vector that approximates the top eigenvector using $d^{1.5+o(1)}$ time. This is what we referred to as our "second algorithm" in Section 4.1.1.

Consider the following situation: for $q \in [d]$, suppose we only know there is a significant eigenvalue gap between the qth eigenvalue λ_q and the $(q+1)$th eigenvalue λ_{q+1}. Is there a way we can learn the subspace spanned by the top-q eigenvectors? Here we consider the subspace instead of the top-q eigenvectors directly, because there might be degeneracy among $\lambda_1, \ldots, \lambda_q$, in which case the set of the top-q eigenvectors is not uniquely defined.

We first estimate the magnitude of λ_q (with additive error $\gamma/100$) using the following theorem.

Theorem 4.13. *Let $\delta \in (0,1)$, $q < d$, $A \in \mathbb{C}^{d \times d}$ be a Hermitian matrix with operator norm at most 1, $v_1, \ldots, v_d$ be an orthonormal basis of eigenvectors of A, and corresponding eigenvalues $\lambda_1, \ldots, \lambda_d$ such that $|\lambda_1| \geq \cdots \geq |\lambda_d|$, where we know the gap $\gamma = |\lambda_q| - |\lambda_{q+1}|$. Suppose $U_A = \exp(\pi i A)$. There is a quantum algorithm that with probability at least*

$1-\delta$, *estimates* $|\lambda_q|$ *with additive error* $\gamma/100$, *using* $\mathcal{O}\Big(\frac{\sqrt{qd}}{\gamma}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{d}{\delta})\log(\frac{1}{\gamma})\Big)$ *controlled applications of* $U_A^{\pm}$ *and* $\tilde{\mathcal{O}}\Big(\frac{\sqrt{qd}}{\gamma}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{d}{\delta})\log(\frac{1}{\gamma})\Big)$ *time.*

Proof. Let $\delta' = \delta/(10d)$, $A = \sum_{i\in[d]} \lambda_i |v_i\rangle\langle v_i|$, and $T = 2^{\lceil \log(200\log(1/\delta')/\gamma)\rceil+2}$. Unitary $W = \sum_{t=0}^{T-1} |t\rangle\langle t| \otimes \exp(\pi i t A)$ does Hamiltonian simulation according to A on the second register, for an amount of time specified in the first register. Observe that $\frac{1}{\sqrt{d}}\sum_{i\in[d]} |i\rangle|i\rangle = \frac{1}{\sqrt{d}}\sum_{i\in[d]} |v_i\rangle|v_i^*\rangle$ because of the invariance of maximally entangled states under unitaries of the form $U\otimes U^\dagger$. Hence we can apply phase estimation with precision $\gamma/200$ and failure probability δ' to the quantum state $\frac{1}{\sqrt{d}}\sum_{i\in[d]} |v_i\rangle|v_i^*\rangle|0\rangle$ using W, to obtain the state

$$\frac{1}{\sqrt{d}}\sum_{i\in[d]} |v_i\rangle|v_i^*\rangle|L_i\rangle, \tag{4.14}$$

where the state $|L_i\rangle$ contains a superposition over different estimates $\tilde{\lambda}_i$ of λ_i. For each $i\in[d]$, if we were to measure $|L_i\rangle$ in the computational basis, then with probability at least $1-\delta'$ we get an outcome $\tilde{\lambda}_i$ such that $|\lambda_i - \tilde{\lambda}_i| \le \gamma/200$.[17] Let $\mu\in[0,1]$, and R_μ be a unitary that marks whether a number's absolute value is $<\mu$, i.e., for every $a\in[-1,1]$

$$R_\mu|a\rangle|0\rangle = \begin{cases} |a\rangle|0\rangle, \text{ if } |a|\ge\mu \\ |a\rangle|1\rangle, \text{ otherwise.}\end{cases}$$

This unitary can be implemented up to negligibly small error by $\tilde{\mathcal{O}}(1)$ elementary gates. Applying R_μ on the last register of the state of Eq. (4.14) and an additional $|0\rangle$, we obtain $\sqrt{p_\mu}|\phi_0\rangle|0\rangle + \sqrt{1-p_\mu}|\phi_1\rangle|1\rangle$ for some $|\phi_0\rangle$ and $|\phi_1\rangle$, where p_μ is the probability of outcome 0 if we were to measure the last qubit. Note that if $\mu \ge |\lambda_q| + \gamma/150 > |\lambda_q| + \gamma/200$, then $p_\mu \le (q-1)/d + (d-q+1)\delta'/d$, where the first term on the right-hand side is the maximal contribution (to the probability p_μ of getting outcome 0 for the last qubit) coming from $|L_i\rangle$ with $i\le q-1$ and the second term is the maximal contribution coming from $|L_i\rangle$ with $i>q-1$. On the other hand, if $\mu \le |\lambda_q| - \gamma/150 < |\lambda_q| - \gamma/200$, then $p_\mu \ge (q/d)\cdot(1-\delta')$, which is the minimal contribution coming from $|L_i\rangle$ with $i\le q$. The difference between the square-roots of these two values is therefore

$$\sqrt{\frac{q}{d}\cdot(1-\delta')} - \sqrt{\frac{q-1}{d} + \frac{d-q+1}{d}\delta'} \ge \sqrt{\frac{q}{d}-\delta'} - \sqrt{\frac{q-1}{d}+\delta'} = \frac{\frac{1}{d}-2\delta'}{\sqrt{\frac{q}{d}-\delta'}+\sqrt{\frac{q-1}{d}+\delta'}}, \tag{4.15}$$

[17] There's a small technical issue here: the unitary $e^{\pi i A}$ (to which we apply phase estimation) has phases ranging between $-\pi$ and π because the λ_j range between -1 and 1, and phase estimation treats $-\pi$ and π the same. However, we can easily fix that by applying phase estimation to the unitary $e^{\pi i A/2}$, whose phases range between $-\pi/2$ and $\pi/2$.

where the last equality is because $a-b=(a^2-b^2)/(a+b)$. Because $\delta'=\delta/(10d)\in(0,1/(10d))$, both terms in the denominator are $\leq\sqrt{q/d}$, and hence the right-hand side of Eq. (4.15) is at least $2/(5\sqrt{qd})$. To estimate $|\lambda_q|$ with additive error $\gamma/100$, it therefore suffices to do binary search over the values of μ (with precision $\gamma/300$, that is, binary search over $\mu\in\{0,\gamma/300,2\gamma/300,\ldots,1\}$), in each iteration estimating $\sqrt{p_\mu}$ to within $\pm 1/(5\sqrt{qd})$. We can implement the unitary that maps $|0\rangle|0\rangle\to\sqrt{p_\mu}|\phi_0\rangle|0\rangle+\sqrt{1-p_\mu}|\phi_1\rangle|1\rangle$ using one application of W and $\tilde{\mathcal{O}}(1)$ time. Let $\delta''=\delta/(10\log(300/\gamma))>0$. By Theorem 2.3 with additive error $\eta=1/(5\sqrt{qd})=\Theta(1/\sqrt{qd})$ and with failure probability δ'', one iteration of the binary search succeeds with probability at least $1-\delta''$ and uses $\mathcal{O}(\log(1/\delta'')/\eta)=\mathcal{O}\big(\sqrt{qd}\log(\frac{\log(1/\gamma)}{\delta})\big)$ applications of W and $W^\dagger$, and $\tilde{\mathcal{O}}\big(\sqrt{qd}\log(\frac{\log(1/\gamma)}{\delta})\big)$ time. Therefore by the union bound, with probability at least $1-(\lceil\log(300/\gamma)\rceil+1)\cdot\delta''\geq 1-\delta$, all iterations of the binary search give a sufficiently good estimate of the value $\sqrt{p_\mu}$ of that iteration, so the binary search gives us an estimate of $|\lambda_q|$ to within $\pm(\gamma/150+\gamma/300)=\pm\gamma/100$.

Since we use $O(\log(1/\gamma))$ iterations of binary search, we use $\mathcal{O}\Big(\sqrt{qd}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{1}{\gamma})\Big)$ applications of W and $W^\dagger$ and $\tilde{\mathcal{O}}\Big(\sqrt{qd}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{1}{\gamma})\Big)$ time for the whole binary search. We can implement W and $W^\dagger$ using $\mathcal{O}(T)=\mathcal{O}(\frac{\log(d/\delta)}{\gamma})$ controlled applications of $U_A^\pm$ and $\tilde{\mathcal{O}}(\frac{\log(d/\delta)}{\gamma})$ time. Thus we obtain a good estimate of $|\lambda_q|$ with probability $\geq 1-\delta$, using $\mathcal{O}\Big(\frac{\sqrt{qd}}{\gamma}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{d}{\delta})\log(\frac{1}{\gamma})\Big)$ controlled applications of $U_A^\pm$ and $\tilde{\mathcal{O}}\Big(\frac{\sqrt{qd}}{\gamma}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{d}{\delta})\log(\frac{1}{\gamma})\Big)$ time. □

The above theorem assumes perfect access to $\exp(\pi iA)=U_A$ for doing phase estimation. If we only assume we have sparse-query-access to A, then by Theorem 2.17 we can implement a unitary $\widetilde{U}_A$ such that $\|\widetilde{U}_A-\exp(\pi iA)\|\leq\varepsilon$ using $\tilde{\mathcal{O}}(s^{0.5+o(1)}/\varepsilon^{o(1)})$ time and queries.

Since the procedure in Theorem 4.13 makes use of $D=\mathcal{O}\Big(\frac{\sqrt{qd}}{\gamma}\log(\frac{\log(1/\gamma)}{\delta})\log(\frac{d}{\delta})\log(\frac{1}{\gamma})\Big)$ controlled applications of $U_A^\pm$, if we replace U_A with $\widetilde{U}_A$, the algorithm still outputs the desired answer with success probability at least $1-\delta-D\varepsilon$.[18] By plugging in the time complexity for constructing $\widetilde{U}_A$ with $\varepsilon=\Theta\Big(\frac{\delta\gamma}{\sqrt{qd}\log(\log(1/\gamma)/\delta)\log(d/\delta)\log(1/\gamma)}\Big)$, with the constant in the $\Theta(\cdot)$ chosen such that $T\varepsilon\leq\delta$ (and rescaling δ by factor of 2), we immediately have the following corollary.

Corollary 4.14. *Let $q<d$ and $A\in\mathbb{C}^{d\times d}$ be a Hermitian matrix with operator norm at most 1, $v_1,\ldots,v_d$ be an orthonormal basis of eigenvectors of A, and eigenvalues $\lambda_1,\ldots,\lambda_d$ such that $|\lambda_1|\geq\cdots\geq|\lambda_d|$, where we know the gap $\gamma=|\lambda_q|-|\lambda_{q+1}|$, and $\delta\in(0,1)$. Suppose A has sparsity s and we have sparse-query-access to A. There is a quantum algo-*

[18] Here we use the fact that if two unitaries are ε-close in operator norm, and they are applied to the same quantum state, then the resulting two states are ε-close in Euclidean norm, and the two probability distributions obtained by measuring the resulting two states in the computational basis are ε-close in total variation distance.

rithm that with success probability at least $1-\delta$*, estimates* $|\lambda_q|$ *with additive error* $\gamma/100$*, using* $\tilde{\mathcal{O}}\left(\frac{1}{\delta^{o(1)}}\left(\frac{\sqrt{dqs}}{\gamma}\right)^{1+o(1)}\right)$ *queries and time.*

The following proposition shows that every bounded-error quantum algorithm needs $\Omega(\sqrt{ds})$ sparse-access queries to estimate the top eigenvalue of an s-sparse matrix with constant additive error. This implies the above corollary is near-optimal when $q = 1$.

Proposition 4.15. *Let* $A \in \mathbb{C}^{d\times d}$ *be a Hermitian matrix with operator norm at most* 3*. Suppose* A *has sparsity* s *and we have sparse-query-access to* A*. Every bounded-error quantum algorithm that estimates the top eigenvalue of* A *with additive error* 0.1 *uses* $\Omega(\sqrt{ds})$ *queries.*

Proof. For simplicity and without loss of generality we assume A has sparsity $2s+1$ and $d \geq 2s+1$ is a multiple of s. The idea is to encode an $s(d-s)$-bit Boolean string into a $2s+1$-sparse $d \times d$ matrix. Given a Boolean string $X \in \{0,1\}^{s(d-s)} := X^{(1)}X^{(2)}\dots X^{(d/s)-1}$ with Hamming weight either 0 or 1, where $X^{(k)}$ is an s^2-bit Boolean string for each $k \in [d/s-1]$. For every $k \in [d/s-1]$, define $Y^{(k)} \in \{0,1\}^{s\times s}$ as $(Y^{(k)})_{ij} = X^{(k)}_{s\cdot i+j}$. Let A be defined by $d/s \cdot d/s = d^2/s^2$ many $s \times s$ square matrices such that for $i \geq j$

$$A_{ij} = \begin{cases} I_s & \text{if } i = j \\ Y^{(i)} + 2^{-d}\cdot J_s & \text{if } i = j+1 \\ 0_s & \text{otherwise,} \end{cases}$$

where I_s is the $s \times s$ identity matrix, J_s is the $s \times s$ all-1 matrix, and 0_s is the $s \times s$ all-0 matrix; and for $i < j$, $A_{ij} = A^T_{ji}$. One can easily see that A has sparsity $2s+1$, and because the Hamming weight $\mathrm{Ham}(X)$ of X is at most 1, the operator norm of A is at most $2 + 2s\cdot 2^{-d} \leq 2 + d\cdot 2^{-d} \leq 3$. Note that given access to the oracle O_X that maps $|i\rangle|0\rangle \to |i\rangle|X_i\rangle$ for every $i \in [s(d-s)]$, one can construct an oracle that allows us to make sparse-query-access to A using 2 applications of $O_X^{\pm}$.

Observe that if $\mathrm{Ham}(X)$=0, then the operator norm of A is at most $1 + 2s\cdot 2^{-d} \leq 1 + d\cdot 2^{-d} \leq 1 + 1/(e\cdot \ln 2) < 1.6$, while if $\mathrm{Ham}(X)$=1, then the operator norm of A is at least 2. Therefore, if there exists a T-query quantum algorithm $\mathcal{A}$ that estimates the top eigenvector of A with additive error 0.1, then $\mathcal{A}$ can also be used to distinguish if $\mathrm{Ham}(X)$ is 0 or 1 using $2T$ queries to O_X. On the other hand, by adversary method (see Theorem 6.4), every bounded-error quantum algorithm uses $\Omega(\sqrt{ds})$ queries to decide if the Hamming weight $\mathrm{Ham}(X)$ of an $s(d-s)$-bit string X is 0 or 1. Combining the above two arguments, we know $T = \Omega(\sqrt{ds})$. □

Once we know $|\lambda_q|$, we can apply [GSL+19, Theorem 31] to implement a block-encoding of U_Π using $\widetilde{\mathcal{O}}(1/\gamma)$ applications of a block-encoding of A, where $\Pi = \sum_{i\in[q]} v_i v_i^\dagger$. Combining the above argument with Theorem 2.19, we directly get the following corollary of Theorem 4.11:

Corollary 4.16. *Let* $q < d$ *and* $A \in \mathbb{C}^{d\times d}$ *be a Hermitian matrix with* $\|A\| \leq 1$*,* $v_1,\dots,v_d$ *be an orthonormal eigenbasis of* A *with respective eigenvalues* $\lambda_1,\dots,\lambda_d$ *such that* $|\lambda_1| \geq$

$\cdots \geq |\lambda_d|$, where we know the gap $\gamma = |\lambda_q| - |\lambda_{q+1}|$. Let $\varepsilon, \delta \in (0,1)$ and $\Pi = \sum_{i\in[q]} v_i v_i^\dagger$. Suppose A has sparsity s and we have sparse-query-access to A. There exists a quantum algorithm that outputs a $d \times q$ matrix W with orthonormal columns such that, with probability $\geq 1-\delta$, $\|WW^\dagger - \Pi\| \leq \varepsilon$, using $\widetilde{\mathcal{O}}\left(\left(\frac{d\sqrt{s}q}{\gamma\varepsilon}\right)^{1+o(1)} + \frac{1}{\delta^{o(1)}}\left(\frac{\sqrt{dqs}}{\gamma}\right)^{1+o(1)} + dq^2\right)$ time and $\tilde{\mathcal{O}}(d)$ QRAM bits.

For the case of dense matrix A, we can set $s = d$ to get time complexity roughly $qd^{1.5}$. The special case $q = 1$ gives our main result for approximating the top eigenvector (with additive ℓ_2-error ε)[19] in time $\widetilde{\mathcal{O}}\left(\left(d^{1.5}/(\gamma\varepsilon)\right)^{1+o(1)}\right)$. The ε-dependency is slightly worse than the algorithm in Section 4.4.2, while both the d-dependency and γ-dependency are significantly better (for d, the power is $1.5 + o(1)$ instead of 1.75; for γ, the power is $1 + o(1)$ instead of 2). In Chapter 7 we show that its d-dependence to be essentially optimal. However, the complexity with respect to q is sub-optimal for q close to d, because one can diagonalize the entire matrix A classical in matrix-multiplication time $\mathcal{O}(d^\omega)$.

Again we note that if we only care about the number of queries to entries of the input matrix (instead of time complexity), then we can get an $\mathcal{O}\left(d^{1.5+o(1)}\right)$-vs-$\Omega(d^2)$ quantum-classical query-complexity separation (see Section 7.3 for classical query lower bounds) for approximating the top eigenvector without using any QRAM, because we can prepare $|\frac{g_j^{(k)}}{\|g_j^{(k)}\|}\rangle$ (of Algorithm 6) from its classical description using a circuit of $\widetilde{\mathcal{O}}(d)$ gates that uses no QRAM and no queries to entries of the input matrix.

4.5 Open problems

Here we mention some questions for future work.

- Can we improve the d-dependence to $d^{1.5}$, without the $o(1)$ in the exponent? And maybe also without using QRAM? Our upper bound of roughly $qd^{1.5}$ for finding the subspace spanned by the top-q eigenvectors is essentially optimal for constant q, but it cannot be optimal for large q (i.e., $q = \Omega(d)$) because diagonalization finds all d eigenvectors exactly in time roughly $d^{2.37}$, which is less than $d^{2.5}$. We should try to improve our algorithm for large q.

- Matrix-vector multiplication is a very basic and common operation in many algorithms. So far there has not been much work on speeding this up quantumly, possibly because easy lower bounds preclude quantum speed-ups for *exact* matrix-vector multiplication.[20] Can we find other applications of our polynomially faster *approximate* matrix-vector multiplication? One such application is computing

[19] Note that for every unit $w, v \in \mathbb{C}^d$ it holds that $\|ww^\dagger - vv^\dagger\| = 2\sqrt{1-|\langle w,v\rangle|^2}$, thus $\varepsilon \geq \|ww^\dagger - vv^\dagger\|$ implies $|\langle w,v\rangle| \geq \sqrt{1-\varepsilon^2/4} \geq 1-\varepsilon^2/4 \geq 1-\varepsilon^2/2$.

[20] Exact matrix-vector multiplication takes $\Omega(d^2)$ quantum queries to entries of A (and hence has no

an approximate matrix-matrix product AB in time roughly $d^{2.5}$, by separately computing AB_i for each of the d columns B_i of B. This would not beat the current-best (but wholly impractical) matrix-multiplication techniques, which take time $d^{2.37\cdots}$, but it would be a very different approach for going beyond the basic $\mathcal{O}\left(d^3\right)$ matrix-multiplication algorithm. A related application is to matrix-product *verification*: we can decide whether AB is close to C in Frobenius norm for given $d \times d$ matrices A, B, C, in quantum time roughly $d^{1.5}$, by combining our approximate matrix-vector computation with Freivalds's algorithm [Fre77]. This should be compared with the quantum algorithm of Buhrman and Špalek [BŠ06] that tests if AB is *equal* to C (over an arbitrary field) using $\widetilde{\mathcal{O}}\left(d^{5/3}\right)$ time.

speed-up). This is easy to see by taking $A \in \{0, 1/d\}^{d \times d}$ and $w = (\frac{1}{\sqrt{d}}, \ldots, \frac{1}{\sqrt{d}})^T$, because then $d^{1.5}Aw$ gives the number of nonzero entries in A. It is well-known that $\Omega(d^2)$ quantum queries are needed to count this number exactly.

CHAPTER 5

Quantum algorithms for SVP

5.1 Introduction

The most important computational problem on lattices is the Shortest Vector Problem (SVP). Given a basis for a lattice $\mathcal{L} \subseteq \mathbb{R}^n$,[1] SVP asks us to compute a nonzero vector in $\mathcal{L}$ with the smallest Euclidean norm. Starting from the '80s, the use of approximate and exact SVP solvers (and other lattice problems) gained prominence for their applications in algorithmic number theory [LLL82], convex optimization [Len83; Kan87b; FT87], coding theory [Bud89], and cryptanalysis [Sha84; Bri84; LO85]. The security of many cryptographic primitives is based on the worst-case hardness of (a decision variant of) approximate SVP to within polynomial factors [Ajt96; MR07; Reg09; Reg06; MR08; Gen09; BV14] in the sense that any cryptanalytic attack on these cryptosystems that runs in time polynomial in the security parameter implies a polynomial-time algorithm to solve approximate SVP to within polynomial factors. Such cryptosystems have attracted a lot of research interest because those lattice problems are conjectured to be resistant to quantum attacks.

The SVP is a well-studied computational problem in both its exact and approximate (decision) versions. By a randomized reduction, it is known to be NP-hard to approximate within any constant factor, and hard to approximate even within a factor $n^{c/\log\log n}$ for some $c > 0$ under reasonable complexity-theoretic assumptions [Mic00; Kho05; HR12].[2] For an approximation factor $2^{\mathcal{O}(n)}$, one can solve SVP in time polynomial in n using the celebrated LLL lattice basis reduction algorithm [LLL82]. In general, the fastest known algorithm(s) for approximating SVP within factors polynomial in n rely on (a variant of) the BKZ lattice basis reduction algorithms [Sch87; SE94; AKS01; GN08; HPS11; ALN+20; ALS21], which can be seen as generalizations of the LLL algorithm and gives an $r^{n/r}$ approximation in $2^{\mathcal{O}(r)}\text{poly}(n)$ time. All these algorithms internally use an algorithm for solving (near) exact SVP in lower-dimensional lattices.

[1] By the discussion at the beginning of Section 2.9.2, we assume the lattice is full rank and $d = n$.

[2] When we say we "approximate" the shortest vector within a factor γ, it means we find a nonzero lattice vector whose length is at most $\gamma \cdot \lambda_1$, where λ_1 is the length of the shortest vector.

Therefore, finding faster algorithms to solve exact SVP is critical to choosing security parameters of cryptographic primitives.

As one would expect from the hardness results above, all known algorithms for solving exact SVP, including the ones we present here, require at least exponential time. There has been some recent evidence [AS18] showing that one cannot hope to get a $2^{o(n)}$ time algorithm for SVP if one believes in complexity-theoretic conjectures such as the (Gap) Exponential Time Hypothesis (see Chapter 8 for more details about the Exponential Time Hypothesis). Most of the known algorithms for SVP can be broadly classified into two classes: (i) the algorithms that require memory polynomial in n but run in time $n^{\mathcal{O}(n)}$ and (ii) the algorithms that require memory $2^{\mathcal{O}(n)}$ and run in time $2^{\mathcal{O}(n)}$. The first class, initiated by Kannan [Kan87b; Hel85; HS07; GNR10; MW15], combines basis reduction with exhaustive enumeration inside Euclidean balls. While enumerating vectors requires $2^{\mathcal{O}(n\log n)}$ time, it is much more space-efficient than other kinds of algorithms for exact SVP.

The second class of algorithms, and currently the fastest, is based on sieving. First developed by Ajtai, Kumar, and Sivakumar [AKS01], they generate many lattice vectors and then divide-and-sieve to create shorter and shorter vectors iteratively. A sequence of improvements [Reg04; NV08; MV10; PS09; ADR+15; AS18], has led to a $2^{n+o(n)}$ time and space algorithm by sieving the lattice vectors and carefully controlling the distribution of the output, thereby outputting a set of lattice vectors that contains the shortest vector with overwhelming probability.

There are variants [NV08; MV10; BDG+16; BCS+23] of the above-mentioned classical sieving algorithms that, under some heuristic assumptions, have an asymptotically smaller (but still $2^{\Theta(n)}$) time and space complexity than their provable counterparts. A number of works have also investigated the potential quantum speedups for lattice algorithms (under some heuristic assumptions), and SVP in particular [LMP15; CL21]. A similar landscape to the classical one exists, although the quantum memory model needs additional attention because it comes in a number of different kinds. While quantum enumeration algorithms only require qubits [ANS18], sieving algorithms usually require the usage of QRAM memory [LMP15; KMP+19].

5.1.1 Main results and high-level intuition

Our main result is to provide two *provable* quantum algorithms for SVP that both improve over the current *fastest quantum algorithm* for SVP [ADR+15] (the algorithm in [ADR+15] is still the fastest classical algorithm for SVP). The first one takes $2^{0.9497n+o(n)}$ time, which is slower than the second one but it does not require the usage of QRAM memory. The second one takes time $2^{0.8345n+o(n)}$ and uses $2^{0.293n+o(n)}$ QRAM bits, which is the best known provable quantum algorithm for solving SVP.

The time complexity of both our quantum algorithms is obtained using a known upper bound on a quantity $\beta(\mathcal{L})^n$ (also known as the lattice kissing number, see Section 2.10.2), which depends on the lattice and is always upper-bounded by $2^{0.402n}$. We analyzed the dependency of the running time of our algorithm on this quantity $\beta(\mathcal{L})$

and plotted (Figure 5.2) the graph of the complexity exponent as a function of $\beta(\mathcal{L})$. In practice, for most lattices, $\beta(\mathcal{L})^n$ is often $2^{o(n)}$ (see the discussion in Section 2.10.2). In this case, the running time of our algorithm is significantly better than when using the generic upper bound on $\beta(\mathcal{L})$. Precisely, our first quantum algorithm (the one without using QRAM) runs in time $2^{0.750n+o(n)}$ and our second quantum algorithm uses only $2^{0.667n+o(n)}$ time and $2^{0.167n+o(n)}$ QRAM bits.

Our high-level idea is to generalize the result in [CCL18]: if we have an α-BDD oracle, then by Theorem 2.45, $\lceil \alpha^{-1} \rceil^n := q^n$ calls to this oracle can enumerate all lattice points within $q \cdot \alpha\lambda_1 \geq \lambda_1$ distance to the origin. If we want to speed up this enumeration algorithm proposed by [CCL18], then we can either try to improve the running time of an α-BDD oracle or try to reduce $q = \lceil \alpha^{-1} \rceil$.

Quantum algorithm for BDDP in QRAM.

In [DRS14], Dadush, Regev, and Stephens-Davidowitz gave an algorithm for BDD with preprocessing (or BDDP), which requires advice containing discrete Gaussian samples over the dual lattice. The idea is to use the *periodic Gaussian function* f (defined in Section 5.2) to go near the closest lattice vector. The function f is periodic over the lattice, and its value depends only on the distance between the input vector and the lattice. Now we can do the gradient ascent by iteratively updating the target vector using values of ∇f and f such that the distance of the target vector from the closest lattice vector decreases. Here we show that we can reduce the time complexity of this algorithm by using quantum amplitude estimation assuming that the advice string is stored in a QRAM memory. More specifically, we show that just by using $\sim \mathcal{O}(\sqrt{N})$ arithmetic operations in QRAM, we can solve BDDP where N is the size of the advice string required in [DRS14].

Covering the surface of a ball by spherical caps.

This result improves the quantum algorithm from [CCL18]. As we mentioned above, one can enumerate all lattice points within a $q\alpha$ distance to a target $\boldsymbol{t}$ by querying an α-BDD oracle q^n times. However, to make use of Theorem 2.45, we should ensure the target is uniquely decodable, or equivalently, $\alpha < 1/2$. [3] Therefore, if we choose $\boldsymbol{t}$ to be $\mathbf{0}$, then q has to be at least 3 to ensure that the shortest vector is one of the vectors output by the enumeration algorithm mentioned above.

We observe here that if we choose a target $\boldsymbol{t}$ to be a random vector on a sphere of a well-chosen radius centered at the origin, then the shortest vector will be within a radius 2α from the target $\boldsymbol{t}$ with some probability P, and thus we can find the shortest vector by making $2^n/P$ calls to the BDD oracle (and hence $\sqrt{2^n/P}$ quantum queries to BDD oracle using quantum minimum-finding). An appropriate[4] choice of the target $\boldsymbol{t}$ and the factor α gives a quantum algorithm (with its corresponding optimized value

[3] Also note that the decoding distance of α-BDD oracle built by discrete Gaussian samples in [DRS14] only succeeds if the target vector is within a radius $\alpha\lambda_1(\mathcal{L})$ for $\alpha < 1/2$.

[4] The optimal choice of α is obtained by numerical optimization, see Section 5.5.

of α, which is different from the latter one) that runs in time $2^{n/2} \cdot 2^{0.4497n+o(n)}$ using poly(n) qubits. Finally, if we can store those DGS samples in a QRAM memory, we can obtain a further speedup and an algorithm that runs in time $2^{n/2} \cdot 2^{0.3345n+o(n)}$ and uses $2^{0.293n+o(n)}$ QRAM bits.

Dependency on a quantity related to the kissing number.

The running time of the above algorithms crucially depends on a quantity related to the kissing number of the input lattice. This quantity plays a role in the BDD-to-DGS reduction when relating the decoding radius α to the ε when sampling at the smoothing parameter η_ε. Our algorithms take significantly less time for the smaller values of this quantity. However, the only known upper bound on this quantity seems to be very pessimistic for most lattices. Since we have used this upper bound to derive the complexity of our algorithm (except in Section 5.5), this means that the actual running time of this algorithm might be much better for most lattices. For a more elaborate discussion on this, see Section 5.5.

Roadmap

In Section 5.2, we will explain how to speed up the gradient-ascent algorithm by [DRS14] assuming those DGS samples are stored in a QRAM memory. In Section 5.3 we discuss the tradeoff between the time complexity of α-BDD oracle (built by the gradient-ascent algorithm) and the decoding distance α. In Section 5.4 we combine the spherical cap covering idea with the enumeration algorithm of [CCL18], as well as the α-BDD oracle we built up in Section 5.3 and Section 5.2, to give two quantum algorithms for solving SVP. In Section 5.5, we show how the time complexity of our two quantum algorithms decreases as the lattice kissing number $\beta(\mathcal{L})$ becomes smaller.

Remark

For convenience, we sometimes will only describe a BDD oracle that succeeds with constant probability, while this success probability can be anyway increased to $1 - 2^{-\Omega(n)}$ by a polynomial number of repeated calls to a BDD oracle (and then by choosing the closest one). We sometimes describe both algorithms for BDD and BDDP in the same theorem, and in this case, we always explain the time complexity of BDD and then the time complexity of BDDP (the sentence will start with "Every extra call"). For preliminaries of lattice and related problems (w.r.t. ℓ_2-norm), see Section 2.9.2.

5.2 Quantum speedup for BDDP using QRAM

The goal of this section is to obtain a quantum speedup of Theorem 5.1.

Theorem 5.1 ([DRS14, Theorem 3.1])**.** *Let $\varepsilon \in (0, \frac{1}{200})$, lattice $\mathcal{L} \subset \mathbb{R}^n$, and $\alpha \leq \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$. There exists a classical algorithm that with probability $\geq 1 - 2^{-\Omega(n)}$, solves α-BDDP using*

$m \cdot poly(n)$ time where $m = \mathcal{O}(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}})$. The preprocessing advice consists of m vectors sampled from $D_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$.[5]

We improve the time complexity of the algorithm by almost a square root factor, but we also require the advice string to be stored in QRAM. We first give an overview of the known algorithms for BDDP.

Most of the known BDDP algorithms (including the one in [DRS14]) are based on the algorithm for decision-CVPP by Aharonov and Regev [AR05]. We first revisit the algorithm for decision-CVPP. In their work, the authors introduced the *periodic Gaussian function* $f : \mathbb{R}^n \to \mathbb{R}^+$,

$$f(\boldsymbol{t}) := \frac{\rho(\boldsymbol{t}+\mathcal{L})}{\rho(\mathcal{L})}$$

towards giving an algorithm for $\mathcal{O}(\sqrt{n/\log n})$-approximation of decision-CVPP. They observed that by the Poisson summation formula, we get the identity

$$f(\boldsymbol{t}) = \mathop{\mathbb{E}}_{\boldsymbol{w}\sim\mathcal{D}_{\mathcal{L}^*}}[\cos(2\pi\langle \boldsymbol{w}, \boldsymbol{t}\rangle)]. \tag{5.1}$$

They also showed that when the distance of the target vector $\boldsymbol{t}$ from lattice $\mathcal{L}$ is at least $\sqrt{n}$, then $f(\boldsymbol{t})$ is negligible, and when the distance between $\boldsymbol{t}$ and lattice $\mathcal{L}$ is at most $\sqrt{\log n}$, then $f(\boldsymbol{t})$ is non-negligible (and the ratio between those two values $= \sqrt{n/\log n}$ is the approximation factor for decision-CVP). This function f evaluated on any vector $\boldsymbol{t}$ is an infinite sum, and is not easy to evaluate efficiently. Their algorithm crucially relied on the observation in Eq (5.1) that shows that the function f can be estimated by using a polynomial-size advice string with at most $1/\text{poly}(n)$ error. They gave the estimator

$$f_W(\boldsymbol{t}) = \frac{1}{N}\sum_{i=1}^{N}\cos(2\pi\langle \boldsymbol{w}_i, \boldsymbol{t}\rangle) \tag{5.2}$$

where $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N) \in \mathcal{L}^*$ are i.i.d. samples from $\mathcal{D}_{\mathcal{L}^*}$; and showed that $f_W \approx f$ with at most $1/\text{poly}(n)$ error when N is a large enough number upper bounded by a polynomial in n.

Later, Liu, Lyubashevsky, and Micciancio [LLM06] gave an algorithm for the approximation of search BDDP. The idea is to iteratively update the target vector $\boldsymbol{t}$ such that its distance from the closest lattice vector decreases and eventually, it is easy to efficiently find the closest lattice vector. They are able to solve the α-BDDP for $\alpha \leq$

[5] We are going to use this reduction in the superpolynomial regime: typically m will be exponential in n because ε will be exponentially small in n. This leaves unclear the space complexity of the reduction. The reduction works by evaluating a polynomial number of times functions of the form $\sum_{i=1}^{m} f_i(\boldsymbol{x})$ where each f_i is a polynomial-time computable function that depends on the i^{th} DGS sample. Furthermore, all the complexities above are in terms of arithmetic operations, not bit complexity. If we assume that all the DGS samples have $\text{poly}(n)$ bit-size then the reduction has time complexity $m \cdot \text{poly}(n)$ and space complexity $O(\text{poly}(n) + \log m)$ *excluding the storage space of the m vectors provided by the DGS*. Finally, as noted in the proof of the theorem in [DRS14], only the preprocessing is probabilistic and with probability at least $1 - 2^{-\Omega(n)}$ over the choice of the samples, the algorithm will solve all α-BDD instances.

$\mathcal{O}\left(\sqrt{\frac{\log n}{n}}\right)$. Dadush, Regev, and Stephens-Davidowitz gave an improvement by a careful analysis of the function $f(\boldsymbol{t})$. They proposed that by iteratively updating $\boldsymbol{t}$ by an approximation of

$$\boldsymbol{t} + \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})},$$

we can go near the closest lattice vector. Their algorithm solves α-BDDP, for $\alpha = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$. The advice string consists of N vectors from $\mathcal{D}_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$ and the algorithm performs $\mathcal{O}(N+\mathrm{poly}(n))$ arithmetic operations where $N = \mathcal{O}\left(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}}\right)$. In this section, we will show that if the advice string is stored in QRAM, then we can achieve the same approximation of BDDP by using only $\mathcal{O}(\sqrt{N}+\mathrm{poly}(n))$ arithmetic operations.

We will start with listing some of the lemmas and theorems from [DRS14] that we will directly use in our proof. After that, we will show the quantum improvement in the estimation of function f_W and ∇f_W. We will present the main result in the last part of this section.

5.2.1 Results from [DRS14]

Let $\rho(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}\|^2)$ be a perfect (unnormalized) Gaussian, then for all vectors $\boldsymbol{t}$, $\boldsymbol{t} + \frac{\nabla\rho(\boldsymbol{t})}{2\pi\rho(\boldsymbol{t})} = 0$, which means one step of the gradient ascent will immediately decode a vector $\boldsymbol{t}$ back to $\boldsymbol{0}$. The proof idea of [DRS14] is to first show that "locally" the periodic Gaussian function f behaves really like a Gaussian (and hence the gradient ascent can help you move closer to the lattice). The following lemma tells us that the periodic Gaussian function is always "above" the perfect Gaussian.

Lemma 5.2. *[DRS14, Lemma 2.14] Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice. Then, for all $\boldsymbol{t} \in \mathbb{R}^n$, $f(\boldsymbol{t}) \geq \rho(\boldsymbol{t})$.*

The following theorem tells us that if we can compute $f(\boldsymbol{t})$ and its gradient, and suppose that $\boldsymbol{t}$ is sufficiently close to the lattice (here we denote this distance as $s_\varepsilon/2$), then we can simply apply the gradient ascent algorithm to find a vector $\boldsymbol{t}' = \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t}$ such that the new vector $\boldsymbol{t}'$ is even closer to the closest lattice vector of $\boldsymbol{t}$. Note that since the periodic Gaussian f is periodic over the lattice, without loss of generality we can always assume the closest vector is $\boldsymbol{0}$ and the vector $\boldsymbol{t}$ gets shorter after one step of the gradient ascent.

Theorem 5.3. *[DRS14, Corollary 4.3] Let $\varepsilon \in (0, 1/400)$ and $\mathcal{L} \subset \mathbb{R}^n$ a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$. Let $s_\varepsilon = \left(\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon}\right)^{1/2}$. Then for all $\boldsymbol{t} \in \mathbb{R}^n$ satisfying $\|\boldsymbol{t}\| \leq s_\varepsilon/2$,*

$$\left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t} \right\| \leq 12(\varepsilon/2)^{1-2\delta(\boldsymbol{t})}\|\boldsymbol{t}\|,$$

where $\delta(\boldsymbol{t}) = \max(1/8, \|\boldsymbol{t}\|/s_\varepsilon)$. In particular, for $\delta(\boldsymbol{t}) \leq 1/2 - 2/(\pi s_\varepsilon^2)$,

$$\left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t} \right\| \leq \|\boldsymbol{t}\|/4.$$

Once we know $\boldsymbol{t}' = \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t}$ gets closer to the lattice as we wanted, it then suffices to show $\frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})}$ is very close to $\frac{\nabla f_W(\boldsymbol{t})}{2\pi f_W(\boldsymbol{t})}$ with respect to Euclidean norm. To this, it suffices to show that both f_W and f are not too small (because both f, f_W are denominators) and that both $f(\boldsymbol{t}) \approx f_W(\boldsymbol{t})$ and $\nabla f(\boldsymbol{t}) \approx \nabla f_W(\boldsymbol{t})$ simultaneously for all relevant $\boldsymbol{t}$ (with overwhelming probability). Here we have two regions for the length of $\boldsymbol{t}$: when the length of $\boldsymbol{t}$ is relatively small, we can already show that doing the gradient ascent on f_W helps us to find a closer point.

Lemma 5.4. *[DRS14, Lemma 4.7] Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ for $\varepsilon \in (0, 1/400)$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$ with $N \geq \Omega(n/\sqrt{\varepsilon})$. Then,*

$$\Pr[\exists \boldsymbol{t}, \|\boldsymbol{t}\| \leq \varepsilon^{1/8}/(1000n) : \|\nabla f_W(\boldsymbol{t})/(2\pi f_W(\boldsymbol{t})) + \boldsymbol{t}\| > \varepsilon^{0.25}\|\boldsymbol{t}\|] \leq 2^{-\Omega(n)}.$$

When $\boldsymbol{t}$ is relatively far away from (but still close enough to) $\mathbf{0}$, we do not directly get a result similar to the lemma above, but still we can show the following four things hold simultaneously for all relevant $\boldsymbol{t}$ (with overwhelming probability): $\nabla f \approx \nabla f_W$, $f \approx f_W$, $f_W \sim 1$, and $\|\nabla f_W\|$ is not too big.

Lemma 5.5. *[DRS14, Lemma 4.10] Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ with $\varepsilon \in (0, 1/400)$. Let $s_\varepsilon = \left(\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon}\right)^{0.5}$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then, for $\varepsilon^2 \leq s \leq 10$, if $N \geq \Omega(n\ln(1/\varepsilon)/s^2)$,*

$$\Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \varepsilon^{1/8}/(1000n) \leq \|\boldsymbol{t}\| \leq s_\varepsilon : \|\nabla f_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| > s\|\boldsymbol{t}\|\right] \leq 2^{-\Omega(N\cdot s^2)}.$$

Lemma 5.6. *[DRS14, Lemma 4.12] Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ with $\varepsilon \in (0, 1/400)$. Let $s_\varepsilon = (\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon})^{0.5}$. Let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then, for $\varepsilon^2 \leq s \leq 10$, if $N \geq \Omega(n\ln(1/\varepsilon)/s^2)$, then*

$$\Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \leq s_\varepsilon : |f_W(\boldsymbol{t}) - f(\boldsymbol{t})| > s\right] \leq 2^{-\Omega(N\cdot s^2)}.$$

Lemma 5.7. *[DRS14, Consequence of the proof of Lemma 4.7] Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ for some $\varepsilon > 0$, and $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$ with $N = \Omega(n/\sqrt{\varepsilon})$. Then we have*

$$\Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \leq \frac{\varepsilon^{1/8}}{1000n} : \|\nabla f_W(\boldsymbol{t})\| > (2\pi + 4\varepsilon^{0.25})\|\boldsymbol{t}\|\right] \leq 2^{-\Omega(n)},$$

and

$$\Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \leq \frac{\varepsilon^{1/8}}{1000n} : |f_W(\boldsymbol{t})| < 1 - \frac{\varepsilon^{0.25}}{100}\right] \leq 2^{-\Omega(n)}.$$

We also include the following lemma.

Lemma 5.8. *[DRS14, First item of Lemma 4.5] Let $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$ for some $\varepsilon > 0$, and let $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$. Then, for $s \geq 0$, $N\min(s, s^2) \geq \Omega(n)$, and $\Delta_\varepsilon = \frac{4\pi\varepsilon}{1+\varepsilon}(\ln\frac{2+2\varepsilon}{\varepsilon} + 1)$, we have*

$$\Pr\left[\|Hf_W(\mathbf{0}) + 2\pi\mathbf{I}_n\| > \Delta_\varepsilon + s\right] \leq 2^{-\Omega(N\min\{s, s^2\})},$$

where Hf_W denotes the Hessian matrix of f_W.

5.2.2 Estimation of f_W and ∇f_W using QRAM

In this subsection, we will show how to estimate f_W and ∇f_W faster using a quantum computer assuming W is stored in QRAM.

Theorem 5.9. *Let $\varepsilon', \delta \in (0,1)$, N be a positive integer, lattice $\mathcal{L} \subset \mathbb{R}^n$ and $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ be a set of vectors from $\mathcal{L}^*$. Suppose we can make quantum queries to $O_W : |j\rangle|0\rangle \to |j\rangle|\boldsymbol{w}_j\rangle$. There exists a quantum algorithm that given target $\boldsymbol{t} \in \mathbb{R}^n$ outputs $\widetilde{f_W}(\boldsymbol{t})$ such that, with probability $\geq 1-\delta$, $|\widetilde{f_W}(\boldsymbol{t}) - f_W(\boldsymbol{t})| \leq \varepsilon'$, using $\mathcal{O}(\frac{1}{\varepsilon'}\log\frac{1}{\delta})$ applications of $O_W^{\pm}$ and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon'}\log\frac{1}{\delta})$ time.*

Proof. We define the positive controlled rotation unitary as, for any $a \in \mathbb{R}$

$$U_{CR^+} : |a\rangle|0\rangle \to \begin{cases} |a\rangle(\sqrt{a}|1\rangle + \sqrt{1-a}|0\rangle), & \text{if } a > 0 \\ |a\rangle|0\rangle, & \text{otherwise,} \end{cases}$$

which can be implemented up to negligible error by $\widetilde{\mathcal{O}}(1)$ quantum elementary gates. Also, we define the cosine inner product unitary as for any $\boldsymbol{t}, \boldsymbol{w} \in \mathbb{R}^n$

$$U_{\cos} : |\boldsymbol{w}\rangle|\boldsymbol{t}\rangle|0\rangle \to |\boldsymbol{w}\rangle|\boldsymbol{t}\rangle|\cos(2\pi\langle \boldsymbol{w}, \boldsymbol{t}\rangle)\rangle,$$

which can also be implemented up to negligible error by $\widetilde{\mathcal{O}}(1)$ quantum elementary gates. Preparing the state $\frac{1}{\sqrt{N}}\sum_{j=1}^{N}|j\rangle|0\rangle|\boldsymbol{t}\rangle|0\rangle|0\rangle$, applying O_W on the first and second registers, applying $U_{\cos}$ on the second, third, fourth registers, and applying U_{CR^+} on the fourth and fifth registers, we obtain

$$\frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)>0}} |j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle\left(\sqrt{\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle}|1\rangle + \sqrt{1-\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle}|0\rangle\right)$$
$$+ \frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\leq 0}} |j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle\rangle|0\rangle.$$

By rearranging the equation, the above is equal to

$$\frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)>0}} \sqrt{\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle}|j, \boldsymbol{w}_j, \boldsymbol{t}\rangle|\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle|1\rangle$$
$$+\frac{1}{\sqrt{N}}\Big(\sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)>0}} \sqrt{1-\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)}|j, \boldsymbol{w}_j, \boldsymbol{t}\rangle|\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)\rangle + \sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle\leq 0}} |j, \boldsymbol{w}_j, \boldsymbol{t}\rangle|\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle\rangle\Big)|0\rangle$$
$$= \sqrt{a^+}|\phi_1\rangle|1\rangle + \sqrt{1-a^+}|\phi_0\rangle|0\rangle,$$

where $a^+ = \sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)>0}} \frac{\cos(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle)}{N}$. By applying Theorem 2.3, we can estimate a^+ with additive error $\varepsilon'/2$ by using $\mathcal{O}(\varepsilon'^{-1})$ applications of O_W, $O_W^{\dagger}$, and $\tilde{\mathcal{O}}(\varepsilon'^{-1})$ time.

Following the same strategy, we can also estimate $a^- = \sum_{\substack{j\in[N] \text{ and} \\ \cos(2\pi\langle w_j,t\rangle)<0}} \frac{\cos(2\pi\langle w_j,t\rangle)}{N}$ with the same additive error and by using the same amount of queries and time. Therefore, we can estimate $a^+ + a^- = \sum_{j\in[N]} \frac{\cos(2\pi\langle w_j,t\rangle)}{N}$ with additive error ε'. By repeating the procedure $\Theta(\log\frac{1}{\delta})$ times and taking the median among them, we finish the proof. □

Theorem 5.10. *Let $\varepsilon', \delta \in (0,1)$, N be a positive integer, lattice $\mathcal{L} \subset \mathbb{R}^n$, and $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ be the set of vectors in $\mathcal{L}^*$. Suppose we can make quantum queries to $O_W : |j\rangle|0\rangle \to |j\rangle|\boldsymbol{w}_j\rangle$ and suppose we know a number $w_{max} \geq \max_{j\in[N]} \|\boldsymbol{w}_j\|$. Then for all $i \in [n]-1$, there exists a quantum algorithm that given target $\boldsymbol{t} \in \mathbb{R}^n$ outputs $\widetilde{\nabla_i f_W}(\boldsymbol{t})$, such that, with probability $\geq 1-\delta$, $|\widetilde{\nabla_i f_W}(\boldsymbol{t}) - \nabla_i f_W(\boldsymbol{t})| \leq 2\pi\varepsilon'\|\boldsymbol{t}\| \cdot w_{max}^2$, using $\mathcal{O}(\frac{1}{\varepsilon'}\log\frac{1}{\delta})$ applications of $O_W^{\pm}$ and $\tilde{\mathcal{O}}(\frac{1}{\varepsilon'}\log\frac{1}{\delta})$ time.*

Proof. From Eq (5.2), we get $\nabla_i f_W(\boldsymbol{t}) = \frac{-1}{N}\sum_{j=1}^{N} \sin(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) \cdot (w_j)_i$ for $i \in [n]-1$. Also observe that for all $i \in [n]-1$ and $j \in [N]$,

$$|\sin(2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle) \cdot (w_j)_i| \leq |2\pi\langle \boldsymbol{w}_j, \boldsymbol{t}\rangle \cdot w_{max}| \leq 2\pi\|\boldsymbol{t}\| \cdot w_{max}^2,$$

and hence $\left|\frac{\sin(2\pi\langle \boldsymbol{w}_j,\boldsymbol{t}\rangle)\cdot(w_j)_i}{2\pi\|\boldsymbol{t}\|\cdot w_{max}^2}\right| \leq 1$ for all $i \in [n]-1$ and $j \in [N]$. For simplicity, we define $g_i(\boldsymbol{w},\boldsymbol{t}) = \frac{\sin(2\pi\langle \boldsymbol{w},\boldsymbol{t}\rangle)\cdot(w)_i}{2\pi\|\boldsymbol{t}\|\cdot w_{max}^2}$. Then we define the sine inner product unitary as for any $\boldsymbol{t}, \boldsymbol{w} \in \mathbb{R}^n$ and $i \in [n]-1$

$$U_{\sin} : |\boldsymbol{w}\rangle|\boldsymbol{t}\rangle|i\rangle|0\rangle \to |\boldsymbol{w}\rangle|\boldsymbol{t}\rangle|i\rangle|g_i(\boldsymbol{w},\boldsymbol{t})\rangle,$$

which can be implemented by $\widetilde{\mathcal{O}}(1)$ quantum elementary gates. Also, we would like to use the positive controlled rotation unitary U_{CR^+} defined in the proof of Theorem 5.9, which again can be implemented up to negligible error by $\widetilde{\mathcal{O}}(1)$ quantum elementary gates.

Preparing the state $\frac{1}{\sqrt{N}}\sum_{j=1}^{N} |j\rangle|\boldsymbol{0}\rangle|\boldsymbol{t}\rangle|i\rangle|0\rangle|0\rangle$, applying O_W on the first and second registers, applying $U_{\sin}$ on the second, third, fourth, fifth registers, and applying U_{CR^+}

on the fifth and sixth registers, we obtain

$$\begin{aligned}
&\frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} |j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|i\rangle|g_i(\boldsymbol{w}_j,\boldsymbol{t})\rangle\left(\sqrt{g_i(\boldsymbol{w}_j,\boldsymbol{t})}|1\rangle+\sqrt{1-g_i(\boldsymbol{w}_j,\boldsymbol{t})}|0\rangle\right) \\
&+\frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})\le 0}} |j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|i\rangle|g_i(\boldsymbol{w}_j,\boldsymbol{t})\rangle|0\rangle \\
=&\frac{1}{\sqrt{N}} \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} \sqrt{g_i(\boldsymbol{w}_j,\boldsymbol{t})}|j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|i\rangle|g_i(\boldsymbol{w}_j,\boldsymbol{t})\rangle|1\rangle \\
&+\frac{1}{\sqrt{N}}\Big(\sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} \sqrt{1-g_i(\boldsymbol{w}_j,\boldsymbol{t})}|j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|i\rangle|g_i(\boldsymbol{w}_j,\boldsymbol{t})\rangle+\sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})\le 0}} |j\rangle|\boldsymbol{w}_j\rangle|\boldsymbol{t}\rangle|i\rangle|g_i(\boldsymbol{w}_j,\boldsymbol{t})\rangle\Big)|0\rangle \\
=&\sqrt{a^+}|\phi_1\rangle|1\rangle+\sqrt{1-a^+}|\phi_0\rangle|0\rangle,
\end{aligned}$$

where $a^+ = \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})>0}} \frac{g_i(\boldsymbol{w}_j,\boldsymbol{t})}{N}$. By applying Theorem 2.3, we can estimate a^+ with additive error $\varepsilon'/2$ using $\mathcal{O}(\varepsilon'^{-1})$ applications of O_W, $O_W^\dagger$, and $\tilde{\mathcal{O}}(\varepsilon'^{-1})$ time. Following the same strategy, we can also estimate $a^- = \sum_{\substack{j\in[N] \text{ and} \\ g_i(\boldsymbol{w}_j,\boldsymbol{t})<0}} \frac{g_i(\boldsymbol{w}_j,\boldsymbol{t})}{N}$ with the same additive error and by using the same amount of queries and time. Therefore, we can estimate $a^+ + a^- = \sum_{j\in[N]} \frac{g_i(\boldsymbol{w}_j,\boldsymbol{t})}{N} = \frac{-\nabla_i f_W(\boldsymbol{t})}{2\pi\|\boldsymbol{t}\|\cdot w_{max}^2}$ with additive error ε'. By repeating the procedure $\Theta(\log\frac{1}{\delta})$ times and taking the median among them, we finish the proof. □

5.2.3 Building BDDP using QRAM

From the previous subsection, given $\boldsymbol{t}$, we can estimate f_W and its gradient with small additive error. In this subsection, we will replace f_W and its gradient with the approximate ones, and show that doing gradient ascent on the approximation function $\widetilde{f_W}(\boldsymbol{t})$ still helps us to find the closest vector, and hence we can use $\widetilde{f_W}$ to solve BDDP.

Theorem 5.11. *Let $\varepsilon \in (0, 1/400)$, $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$, N be a positive integer, $s_\varepsilon = \left(\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon}\right)^{0.5}$, $\delta_{\max} = 0.5 - \frac{2}{\pi s_\varepsilon^2}$, and $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be a set of vectors from $\mathcal{L}^*$. Suppose that for some $\gamma > 0$, $\boldsymbol{t} \in \mathbb{R}^n$, one can compute $\widetilde{f_W}(\boldsymbol{t})$ and $\widetilde{\nabla f_W}(\boldsymbol{t})$, and it holds that*

1. $\|\boldsymbol{t}\| \le \min\{\delta_{\max} s_\varepsilon, \sqrt{\ln(1/(4\gamma))/\pi}\}$,
2. $\|\nabla f_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| \le \frac{\pi}{2}\gamma\|\boldsymbol{t}\|$,
3. $\|\nabla f_W(\boldsymbol{t}) - \widetilde{\nabla f_W}(\boldsymbol{t})\| \le \frac{\pi}{2}\gamma\|\boldsymbol{t}\|$,
4. $|f_W(\boldsymbol{t}) - f(\boldsymbol{t})| \le \gamma$,

5. $|f_W(\boldsymbol{t}) - \widetilde{f_W}(\boldsymbol{t})| \le \gamma$.

Then,

$$\left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} - \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \le \frac{6\gamma}{\rho(\boldsymbol{t})} \|\boldsymbol{t}\|.$$

Proof. By triangle inequality, the second and third conditions imply

$$\|\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| \le \pi\gamma\|\boldsymbol{t}\|, \tag{5.3}$$

and the fourth and fifth imply

$$|\widetilde{f_W}(\boldsymbol{t}) - f(\boldsymbol{t})| \le 2\gamma. \tag{5.4}$$

From Lemma 5.2 and the condition on the length of $\boldsymbol{t}$, we get $f(\boldsymbol{t}) \ge \rho(\boldsymbol{t}) \ge 4\gamma$. By triangle inequality, we get

$$\begin{aligned} \left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} - \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| &= \left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \frac{f(\boldsymbol{t})}{\widetilde{f_W}(\boldsymbol{t})} + \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \left(\frac{f(\boldsymbol{t})}{\widetilde{f_W}(\boldsymbol{t})} - 1 \right) \right\| \\ &\le \left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \frac{f(\boldsymbol{t})}{\widetilde{f_W}(\boldsymbol{t})} + \left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \left| \frac{f(\boldsymbol{t})}{\widetilde{f_W}(\boldsymbol{t})} - 1 \right| \end{aligned} \tag{5.5}$$

To bound the first term in Eq (5.5), by using Eq (5.3) and Eq (5.4), we get

$$\left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \frac{f(\boldsymbol{t})}{\widetilde{f_W}(\boldsymbol{t})} \le \frac{\pi\gamma\|\boldsymbol{t}\|}{2\pi f(\boldsymbol{t})} \frac{f(\boldsymbol{t})}{f(\boldsymbol{t}) - 2\gamma} = \frac{\gamma\|\boldsymbol{t}\|}{2(f(\boldsymbol{t}) - 2\gamma)}. \tag{5.6}$$

For the second term in Eq (5.5), we use Theorem 5.3 ($\|\boldsymbol{t}\| \le \delta_{\max} s_\varepsilon < s_\varepsilon/2$) and Eq (5.4),

$$\left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \left| \frac{f(\boldsymbol{t})}{\widetilde{f_W}(\boldsymbol{t})} - 1 \right| \le \frac{5}{4}\|\boldsymbol{t}\| \left(\frac{f(\boldsymbol{t})}{f(\boldsymbol{t}) - 2\gamma} - 1 \right) = \frac{10\gamma}{4(f(\boldsymbol{t}) - 2\gamma)} \|\boldsymbol{t}\|. \tag{5.7}$$

From Eq (5.5), Eq (5.6), Eq (5.7), and the fact $f(\boldsymbol{t}) \ge \rho(\boldsymbol{t}) \ge 4\gamma$, we get

$$\left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} - \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} \right\| \le \frac{3\gamma \cdot \|\boldsymbol{t}\|}{f(\boldsymbol{t}) - 2\gamma} \le \frac{6\gamma}{\rho(\boldsymbol{t})} \|\boldsymbol{t}\|.$$

□

The following lemma shows that if the target vector $\boldsymbol{t}$ is very close to the lattice, then even if we make some additive errors for estimating f_W and ∇f_W, one step of the gradient ascent still shrinks the distance by a desired factor.

Lemma 5.12. *Let $\varepsilon \in (0,1/400)$, $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$, $\boldsymbol{t} \in \mathbb{R}^n$ such that $\|\boldsymbol{t}\| \le \varepsilon^{1/8}/(1000n)$, $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ with $\boldsymbol{w}_i$ sampled i.i.d. from $D_{\mathcal{L}^*}$ with $N = \Omega(n/\sqrt{\varepsilon})$. Suppose $\widetilde{f_W}(\boldsymbol{t})$ and $\widetilde{\nabla f_W}(\boldsymbol{t})$ satisfy both*

$$|\widetilde{f_W}(\boldsymbol{t}) - f_W(\boldsymbol{t})| \le \frac{\varepsilon^{1/4}}{100} \text{ and } \|\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})\| \le \frac{\varepsilon^{1/4}}{100}\|\boldsymbol{t}\|.$$

Then[6]

$$\Pr\left[\exists \boldsymbol{t} \in \mathbb{R}^n, \|\boldsymbol{t}\| \le \frac{\varepsilon^{1/8}}{1000n} : \left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} + \boldsymbol{t} \right\| \ge 3\varepsilon^{1/4}\|\boldsymbol{t}\| \right] \le 2^{-\Omega(n)}.$$

Proof. By Lemma 5.7, with probability at least $1 - 2\cdot 2^{-\Omega(n)}$, $\|\nabla f_W(\boldsymbol{t})\| \le (2\pi + 4\varepsilon^{1/4})\|\boldsymbol{t}\|$ and $|f_W(\boldsymbol{t})| > 1 - \frac{\varepsilon^{1/4}}{100}$ hold simultaneously for all $\boldsymbol{t} \le \varepsilon^{1/8}/(1000n)$. By triangle inequality and both the assumptions, with probability at least $1 - 2\cdot 2^{-\Omega(n)}$,

$$\|\widetilde{\nabla f_W}(\boldsymbol{t})\| \le (2\pi + 5\varepsilon^{1/4})\|\boldsymbol{t}\| \text{ and } |\widetilde{f_W}(\boldsymbol{t})| > 1 - \frac{\varepsilon^{1/4}}{50}, \tag{5.8}$$

holds simultaneously, which implies

$$\begin{aligned}\left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} - \frac{\nabla f_W(\boldsymbol{t})}{2\pi f_W(\boldsymbol{t})} \right\| &\le \left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} \right\| + \frac{\|\nabla f_W(\boldsymbol{t})\|}{2\pi} \left| \frac{1}{\widetilde{f_W}(\boldsymbol{t})} - \frac{1}{f_W(\boldsymbol{t})} \right| \\ &\le \frac{\varepsilon^{1/4}\|\boldsymbol{t}\|/100}{2\pi(1-\varepsilon^{1/4}/50)} + \frac{(2\pi + 5\varepsilon^{1/4})\|\boldsymbol{t}\|}{2\pi} \cdot \frac{\varepsilon^{1/4}/100}{(1-\varepsilon^{1/4}/100)(1-\varepsilon^{1/4}/50)} \le 2\varepsilon^{1/4}\|\boldsymbol{t}\|.\end{aligned}$$

Also, by Lemma 5.4 we know that, with at least $1 - 2^{-\Omega(n)}$ probability, $\left\| \frac{\nabla f_W(\boldsymbol{t})}{2\pi f_W(\boldsymbol{t})} + \boldsymbol{t} \right\| \le \varepsilon^{1/4}\|\boldsymbol{t}\|$. Hence, by triangle inequality and union bound, we finish the proof. □

We further extend the above lemma to ensure that even if the target is relatively far from (but reasonably close to) the lattice, the gradient ascent still works.

Theorem 5.13. *Let $\varepsilon \in (0,1/400)$, $\mathcal{L} \subset \mathbb{R}^n$ be a lattice with $\rho(\mathcal{L}) = 1+\varepsilon$, $s_\varepsilon = (\frac{1}{\pi}\ln\frac{2(1+\varepsilon)}{\varepsilon})^{0.5}$, $\delta_{\max} = 0.5 - \frac{2}{\pi s_\varepsilon^2}$, $\boldsymbol{t} \in \mathbb{R}^n$ such that $\|\boldsymbol{t}\| \le \delta_{\max} s_\varepsilon$, $\delta(\boldsymbol{t}) = \max\{1/8, \|\boldsymbol{t}\|/s_\varepsilon\} < 1/2$, and $W = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N)$ be sampled independently from $\mathcal{D}_{\mathcal{L}^*}$ with $N \ge \Omega(n\ln(1/\varepsilon)/\sqrt{\varepsilon})$. Suppose $\widetilde{f_W}(\boldsymbol{t})$ and $\widetilde{\nabla f_W}(\boldsymbol{t})$ satisfy both*

$$|\widetilde{f_W}(\boldsymbol{t}) - f_W(\boldsymbol{t})| \le \frac{\varepsilon^{1/4}}{100} \text{ and } \|\widetilde{\nabla f_W}(\boldsymbol{t}) - \nabla f_W(\boldsymbol{t})\| \le \frac{\|\boldsymbol{t}\| \cdot \varepsilon^{1/4}}{100}.$$

Then with probability at least $1 - 2^{-\Omega(n)}$, $\left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} + \boldsymbol{t} \right\| \le \varepsilon^{(1-2\delta(\boldsymbol{t}))/4}\|\boldsymbol{t}\|$.

Proof. Lemma 5.12 already shows that the theorem is satisfied for all $\boldsymbol{t}$ with $\|\boldsymbol{t}\| \le \varepsilon^{1/8}/(1000n)$. So it suffices to show the case when $\varepsilon^{1/8}/(1000n) < \|\boldsymbol{t}\| \le \delta_{\max} s_\varepsilon$. By Lemma 5.2, for such $\boldsymbol{t}$,

$$f(\boldsymbol{t}) \ge \rho(\boldsymbol{t}) \ge e^{-\pi\delta_{\max}^2 s_\varepsilon^2} > \varepsilon^{\delta_{\max}^2/2} \ge \varepsilon^{1/4}/2.$$

Also by Lemma 5.5 and Lemma 5.6 (choosing $s = \varepsilon^{1/4}/100$) we know that

[6] The probability is over the choice of the set W.

- $\|\nabla f_W(\boldsymbol{t}) - \nabla f(\boldsymbol{t})\| \le \varepsilon^{1/4}\|\boldsymbol{t}\|/100$ holds with probability $\ge 1 - 2^{-\Omega(\varepsilon^{1/2}N/100^2)} = 1 - 2^{-\Omega(n)}$.
- $\|f(\boldsymbol{t}) - f(\boldsymbol{t})\| \le \varepsilon^{1/4}/100$ holds with probability $\ge 1 - 2^{-\Omega(\varepsilon^{1/2}N/100^2)} = 1 - 2^{-\Omega(n)}$.

Therefore, by using Theorem 5.11 with $\gamma = \frac{\varepsilon^{1/4}}{50\pi}$, we have that with probability $\ge 1 - 2^{-\Omega(n)}$,

$$\begin{aligned}
\left\| \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi \widetilde{f_W}(\boldsymbol{t})} + \boldsymbol{t} \right\| &\le \frac{6\gamma}{\rho(\boldsymbol{t})}\|\boldsymbol{t}\| + \left\| \frac{\nabla f(\boldsymbol{t})}{2\pi f(\boldsymbol{t})} + \boldsymbol{t} \right\| \\
&\le \frac{3\varepsilon^{1/4}}{25} \cdot e^{\pi\|\boldsymbol{t}\|^2}\|\boldsymbol{t}\| + 12(\varepsilon/2)^{1-2\delta(\boldsymbol{t})}\|\boldsymbol{t}\| && \text{(by Theorem 5.3)} \\
&\le \frac{3\varepsilon^{1/4}}{25}\left(\frac{2(1+\varepsilon)}{\varepsilon}\right)^{\delta(\boldsymbol{t})^2}\|\boldsymbol{t}\| + 12(\varepsilon/2)^{1-2\delta(\boldsymbol{t})}\|\boldsymbol{t}\| && \text{(since } \delta(\boldsymbol{t}) \ge \|\boldsymbol{t}\|/s_\varepsilon\text{)} \\
&\le \frac{\varepsilon^{0.25-\delta(\boldsymbol{t})^2}}{6} \cdot \|\boldsymbol{t}\| + \frac{3}{10}\varepsilon^{(1-2\delta(\boldsymbol{t}))/4}\|\boldsymbol{t}\| \\
&\le \varepsilon^{(1-2\delta(\boldsymbol{t}))/4}\|\boldsymbol{t}\| && \text{(since } \delta(\boldsymbol{t})/2 \ge \delta(\boldsymbol{t})^2\text{)},
\end{aligned}$$

where the fourth equation holds because $(\frac{\varepsilon}{2})^{\frac{3(1-2\delta(\boldsymbol{t}))}{4}} \le (\frac{1}{800})^{\frac{9}{16}} < \frac{1}{40}$ and $(\frac{\varepsilon}{2})^{\frac{(1-2\delta(\boldsymbol{t}))}{4}} < \varepsilon^{\frac{(1-2\delta(\boldsymbol{t}))}{4}}$. □

Now we show how to build up a BDD oracle, given i.i.d. samples drawn from $\mathcal{D}_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$ assuming they are all stored in a QRAM memory and we are allowed quantum queries.

Theorem 5.14. *Let $\varepsilon \in (e^{-n^2}, 1/400)$, lattice $\mathcal{L} \subset \mathbb{R}^n$, $\phi(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)}$, $N = \frac{100n^2\ln(1/\varepsilon)}{\sqrt{\varepsilon}}$, and $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\} \subset \mathcal{L}^*$ with $\boldsymbol{w}_i$ sampled i.i.d. from $\mathcal{D}_{\mathcal{L}^*,\eta_\varepsilon(\mathcal{L}^*)}$. Suppose we can make quantum queries to $O_W : |j\rangle|0\rangle \to |j\rangle|\boldsymbol{w}_j\rangle$. There exists a quantum algorithm that with probability $\ge 1 - 2^{-\Omega(n)}$, solves $\phi(\mathcal{L})/\lambda_1(\mathcal{L})$-BDD,[7] using $\mathcal{O}(\frac{n^{2.5}\log n}{\varepsilon^{0.25}}\log\log(\frac{1}{\varepsilon}))$ applications of $O_W^{\pm}$ and $\tilde{\mathcal{O}}(\frac{n^{2.5}\log n}{\varepsilon^{0.25}}\log\log(\frac{1}{\varepsilon}) + n^3)$ time.*

Proof. Let $s_\varepsilon = (\frac{1}{\pi}\log(\frac{2(1+\varepsilon)}{\varepsilon}))^{1/2}$ and $\delta_{max} = \frac{1}{2} - \frac{2}{\pi s_\varepsilon^2}$. The algorithm takes target $\boldsymbol{t} \in \mathbb{R}^n$. It then iteratively updates $\boldsymbol{t} \leftarrow \boldsymbol{t} + \frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi\widetilde{f_W}(\boldsymbol{t})}$ for $1 + \lceil 8\log(\sqrt{n}s_\varepsilon)/\log(1/\varepsilon)\rceil$ times, where $\widetilde{\nabla f_W}(\boldsymbol{t})$ and $\widetilde{f_W}(\boldsymbol{t})$ are computed by Theorem 5.9 and Theorem 5.10 with additive error $\varepsilon' = \varepsilon^{1/4}/(800\pi n^{2.5})$ and failure probability $\delta = 2^{-100n}$. It then queries the first $100n^2$ vectors in W and takes the first n linearly independent vectors of length bounded by $\sqrt{n}\cdot\eta_\varepsilon(\mathcal{L}^*)$ as set $V^* = (\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_n^*) \subset W$. If no such set exists then abort. Compute $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ such that $\langle \boldsymbol{v}_j, \boldsymbol{v}_k^* \rangle = \delta_{j,k}$ and return $\sum_{j\in[n]} c_j\boldsymbol{v}_j$ where $c_j = \lfloor \boldsymbol{v}_j^* \cdot \boldsymbol{t} \rceil$ is the nearest integer of $\boldsymbol{v}_j^* \cdot \boldsymbol{t}$. For each iteration, the algorithm uses $\mathcal{O}(1/\varepsilon') = \mathcal{O}(n^{2.5}/\varepsilon^{0.25})$ applications of $O_W^{\pm}$ and $\tilde{\mathcal{O}}(n^{2.5}/\varepsilon^{0.25})$ time to estimate $\widetilde{\nabla f_W}(\boldsymbol{t})$ and $\widetilde{f_W}(\boldsymbol{t})$. Since it repeats the iterative update $1 + \lceil 8\log(\sqrt{n}s_\varepsilon)/\log(1/\varepsilon)\rceil = \mathcal{O}((\log n)\log\log(1/\varepsilon))$ times, and

[7] Note that by the left equality of Eq (2.8), $\phi(\mathcal{L})/\lambda_1(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})} < 1/2$.

since it makes $100n^2$ queries to O_W and uses $\mathcal{O}(n^3)$ time to compute V^* and V by Gaussian elimination, the algorithm therefore uses $\mathcal{O}(\frac{n^{2.5}\log n}{\varepsilon^{0.25}}\log\log(\frac{1}{\varepsilon}))$ applications of $O_W^{\pm}$ as well as $\tilde{\mathcal{O}}(\frac{n^{2.5}\log n}{\varepsilon^{0.25}}\log\log(\frac{1}{\varepsilon})+n^3)$ time.

Now we show the correctness. By scaling the lattice appropriately, we can assume without loss of generality that $\rho(\mathcal{L})=1+\varepsilon$, so that $\eta_\varepsilon(\mathcal{L}^*)=1$. Let $\boldsymbol{t}'=\boldsymbol{t}-\boldsymbol{w}$ such that $\|\boldsymbol{t}'\|\le\delta_{max}s_\varepsilon$ for some $\boldsymbol{w}\in\mathcal{L}$. There exists such a vector $\boldsymbol{t}'$ because of the promise of the ϕ/λ_1-BDD and the fact that $\phi(\mathcal{L})\le\delta_{max}s_\varepsilon$. By Lemma 2.47 with $t=\sqrt{2\pi n}$ and the union bound, we get that with probability $1-N\cdot e^{-2n^2}\ge 1-2^{\Omega(n)}$, $\forall i\in[N], \|\boldsymbol{w}_i\|\le 2n+1$. Then by Theorem 5.9 and Theorem 5.10, we can estimate $\widetilde{f_W}(\boldsymbol{t})$ and $\widetilde{\nabla f_W}(\boldsymbol{t})$ such that, with probability $1-2^{-\Omega(n)}$,

$$|\widetilde{f_W}(\boldsymbol{t})-f_W(\boldsymbol{t})|\le\frac{\varepsilon^{1/4}}{100}\text{ and }\|\widetilde{\nabla f_W}(\boldsymbol{t})-\nabla f_W(\boldsymbol{t})\|\le\frac{\varepsilon^{1/4}}{100}\|\boldsymbol{t}\|.$$

Note that $f_W(\boldsymbol{t})=f_W(\boldsymbol{t}')$ and $\nabla f_W(\boldsymbol{t})=\nabla f_W(\boldsymbol{t}')$ because both f_W and ∇f_W are periodic over the lattice by their definition (see Eq (5.2)), and hence $\widetilde{f_W}(\boldsymbol{t})$ and $\widetilde{\nabla f_W}(\boldsymbol{t})$ are also good estimators for $f_W(\boldsymbol{t}')$ and $\nabla f_W(\boldsymbol{t}')$. Therefore, by Theorem 5.13 we can say that by every update of $\boldsymbol{t}$ by $\boldsymbol{t}\leftarrow\boldsymbol{t}+\frac{\widetilde{\nabla f_W}(\boldsymbol{t})}{2\pi\widetilde{f_W}(\boldsymbol{t})}=\boldsymbol{w}+\boldsymbol{t}'+\frac{\widetilde{\nabla f_W}(\boldsymbol{t}')}{2\pi\widetilde{f_W}(\boldsymbol{t}')}$, the output vector is of the form $\boldsymbol{w}+\boldsymbol{t}^*$ where $\|\boldsymbol{t}^*\|$ shrinks by a factor of at least $\varepsilon^{(1-2(1/4))/4}=\varepsilon^{1/8}$ with probability $1-2^{-\Omega(n)}$ for every update. Hence by $1+\lceil 8\log(\sqrt{n}s_\varepsilon)/\log(1/\varepsilon)\rceil$ updates, we get $\boldsymbol{t}=\boldsymbol{w}+\boldsymbol{t}^*$ such that $\|\boldsymbol{t}^*\|<1/(2\sqrt{n})$. Since $\boldsymbol{v}_j^*\in\mathcal{L}^*$ for all $j\in[n]$, we obtain $\lfloor\langle\boldsymbol{v}_j^*,\boldsymbol{t}\rangle\rceil=\lfloor\langle\boldsymbol{v}_j^*,\boldsymbol{t}^*\rangle\rceil+\langle\boldsymbol{v}_j^*,\boldsymbol{w}\rangle$ for every $j\in[n]$. Also, because $\|\boldsymbol{t}^*\|<1/(2\sqrt{n})$ and $\|\boldsymbol{v}_j^*\|\le\sqrt{n}$ for all $j\in[n]$, by Cauchy-Schwarz we get $\lfloor\langle\boldsymbol{v}_j^*,\boldsymbol{t}^*\rangle\rceil=0$ for every $j\in[n]$, implying $\sum_{j\in[n]}\langle\boldsymbol{v}_j^*,\boldsymbol{w}\rangle\boldsymbol{v}_j=w$. Hence, we get the vector $\boldsymbol{w}$ as the output with probability greater than $1-2^{-\Omega(n)}$.

It remains to show that $\{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_{100n^2}\}$ contains n linearly independent vectors of length at most $\sqrt{n}$ with probability at least $1-2^{-\Omega(n^2)}$. Let $W'=(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_{100n^2})$. By Lemma 2.47 and union bound, with at least $1-100n^2\cdot e^{-n}$ probability, all vectors in W' have length at most $\sqrt{n}$. From Lemma 5.8, we know that

$$\|Hf_{W'}(\boldsymbol{0})+2\pi\mathbf{I}_n\|\le\frac{4\pi\varepsilon}{1+\varepsilon}\left(\log\left(\frac{2(1+\varepsilon)}{\varepsilon}+1\right)+1\right)<2\pi$$

with probability at least $1-2^{-\Omega(n^2)}$. Hence, we have that $Hf_{W'}(\boldsymbol{0})$ is invertible and $Hf_{W'}(\boldsymbol{0})=\frac{-4\pi^2}{m}\sum_{i=1}^m\boldsymbol{w}_i\boldsymbol{w}_i^T$ (from Eq (5.2)). It implies that W' spans $\mathbb{R}^n$, which completes the proof. □

5.3 Improved quantum algorithms for BDD

In the previous section, we showed how to construct a faster BDD solver using a quantum computer assuming we already have a bunch of samples stored in a QRAM memory, while the decoding distance is explicitly stated (with respect to λ_1) in Theorem 5.14. In this section, we will explicitly give a lower bound for the decoding distance stated in Theorem 5.14.

The decoding distance of our BDD solver heavily depends on the quantity $\beta(\mathcal{L})$ which is related to the kissing number of the lattice (see Section 2.10.2). For this reason, we will first provide complexity bounds that depend on $\beta(\mathcal{L})$ and then obtain complexity bounds in the worst case ($\beta(\mathcal{L}) \leqslant 2^{0.401+o(1)}$) as corollaries.

5.3.1 Reduction from BDDP to honest DGS

In Theorem 5.14, we gave an algorithm for BDDP which requires a sampler from the discrete Gaussian distribution *exactly at the smoothing parameter* $\eta_\varepsilon(\mathcal{L}^*)$, which is generally not known. In this subsection, we present a modification of Theorem 5.1 and Theorem 5.14 that gives a reduction from BDDP (with preprocessing) to hDGS, which does not require knowing the exact value of $\eta_\varepsilon(\mathcal{L}^*)$. The idea is simple: We just try all possible values of the smoothing parameter.

Theorem 5.15. *For all $\alpha \in (0,2)$, ε satisfying $e^{-n^2} \leqslant \varepsilon \leqslant \min(e^{-n^\alpha}, 1/200)$, there exist two algorithms (one is classical and the other is quantum) that, on input a basis $\mathbf{B}$ of $\mathcal{L}(\mathbf{B}) \subset \mathbb{R}^n$, with constant probability, construct a $\frac{\phi(\mathcal{L})}{\lambda_1(\mathcal{L})}$-BDDP oracle, where $\phi(\mathcal{L}) \equiv \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)}$. The classical algorithm in the preprocessing stage makes poly(n) calls to a 0.5-hDGS$^m_{\eta_\varepsilon}$ sampler on the lattice $\mathcal{L}^*$, takes poly(n) time, and stores all of those samples using $m \cdot$ poly(n) space. Each extra call to the BDD oracle constructed by the classical algorithm after the preprocessing stage uses $m \cdot$ poly(n) time and $\mathcal{O}($poly$(n) + \ln m)$ space. The quantum algorithm in the preprocessing stage makes poly(n) calls to a 0.5-hDGS$^m_{\eta_\varepsilon}$ sampler on the lattice $\mathcal{L}^*$, takes poly(n) time, and stores all of those samples in a QRAM memory using $m \cdot$ poly(n) QRAM bits. Each extra call to the BDD oracle constructed by the quantum algorithm after the preprocessing stage uses $\sqrt{m} \cdot$ poly(n) time, $\mathcal{O}($poly$(n) + \ln m)$ classical space, poly(n) qubits and $\sqrt{m} \cdot$ poly(n) queries to the samples stored in a QRAM memory.*

Proof. First we note that we can easily identify an interval $I = [a,b]$ such that $\eta_\varepsilon(\mathcal{L}^*) \in [a,b]$ and $\frac{b}{a} \leqslant 2^{n+o(n)}$. By Eq (2.8) in Lemma 2.58, one has

$$\sqrt{\ln(1/\varepsilon)/\pi} \leqslant \lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) \leqslant \sqrt{n} \cdot 2^{0.402} \varepsilon^{-1/n} / \sqrt{2\pi e}$$

so $\eta_\varepsilon(\mathcal{L}^*) \in \frac{1}{\lambda_1(\mathcal{L})}[a',b']$ where $\frac{b'}{a'} = \tilde{\mathcal{O}}(n^{0.5+o(1)})$. Furthermore, by Theorem 2.40, we can obtain a length ℓ such that $2^{-n}\ell \leqslant \lambda_1(\mathcal{L}) \leqslant \ell$ in poly(n) time. It follows that $\eta_\varepsilon(\mathcal{L}^*) \in [a,b] := [\frac{a'}{\ell}, \frac{2^n \cdot b'}{\ell}]$ and $\frac{b}{a} = 2^n \frac{b'}{a'} = 2^{n+o(n)}$.

Now let $c = 2+\alpha$, $\Delta = 1 + \frac{1}{n^c}$ and $N = \left\lceil \frac{\ln(b/a)}{\ln \Delta} \right\rceil$. Note that $N =$ poly(n) since $\frac{b}{a} = 2^{n+o(n)}$. Furthermore, if we let $s_i = a\Delta^i$ for $i = 1, \ldots, N$ then there must exist some i_0 such that

$$\tfrac{1}{\Delta} s_{i_0} \leqslant \eta_\varepsilon(\mathcal{L}^*) \leqslant s_{i_0}. \tag{5.9}$$

The preprocessing stage of both the classical algorithm $\mathcal{A}_c$ and the quantum algorithm $\mathcal{A}_q$ consists in calling the hDGS$^m_{\eta_\varepsilon}$ sampler with $\sigma = s_i$ for each $i = 1, \ldots, N$,

to obtain lists L_i of m vectors,[8] and storing all the lists in either classical memory or in a QRAM memory. This requires $N = \text{poly}(n)$ calls and we need to store $m \cdot \text{poly}(n)$ vectors.

We now describe the algorithms for ϕ/λ_1-BDDP after the preprocessing stage. We start with the classical one $\mathcal{A}_c$: On input $\boldsymbol{t} \in \mathbb{R}^n$, for each $i = 1, \dots, N$, $\mathcal{A}_c$ calls the algorithm of Theorem 5.1 on input $\boldsymbol{t}$ and provides the list L_i to the algorithm of Theorem 5.1 in place of the DGS samples. Hence, for each i, we either obtain a lattice vector $\boldsymbol{y}_i$ or the algorithm of Theorem 5.1 fails and we let $\boldsymbol{y}_i = \boldsymbol{0}$ in that case. Finally, $\mathcal{A}_c$ returns the point closest to $\boldsymbol{t}$ in the list $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N$.[9]

As for the quantum one $\mathcal{A}_q$, we use Theorem 5.14 instead of Theorem 5.1 which gives exactly the same result except for the exponent in the complexity of the oracle and the quantum queries to the samples stored in a QRAM memory. The running time of both $\mathcal{A}_c$ and $\mathcal{A}_q$ described above is therefore clear, so it remains to prove that the algorithms actually solve $\phi(\mathcal{L})/\lambda_1(\mathcal{L})$ BDDP on $\mathcal{L}$.

We first note that when called on s_{i_0}, the hDGS_σ^m sampler will return m vectors whose joint distribution is 0.5-close $\mathcal{D}^m_{\mathcal{L},s_{i_0}}$ since $s_{i_0} \geqslant \sigma(\mathcal{L}) = \eta_\varepsilon(\mathcal{L}^*)$ by Eq (5.9). Furthermore, by Lemma 2.60 we have

$$t\eta_\varepsilon(\mathcal{L}^*) \leqslant \eta_{\varepsilon^{\Delta^2} f}(\mathcal{L}^*),$$

where f is defined in Lemma 2.60. It follows by Eq (5.9) that

$$\eta_\varepsilon(\mathcal{L}^*) \leqslant s_{i_0} \leqslant \Delta\eta_\varepsilon(\mathcal{L}^*) \leqslant \eta_{\varepsilon^{\Delta^2} \cdot f}(\mathcal{L}^*).$$

Also, the map $\varepsilon \mapsto \eta_\varepsilon(\mathcal{L})$ is continuous and decreasing, so it follows that there exists ε' s.t.

$$s_{i_0} = \eta_{\varepsilon'}(\mathcal{L}^*) \quad \text{and } \varepsilon^{\Delta^2} f \leqslant \varepsilon' \leqslant \varepsilon.$$

Therefore by Theorem 5.1(Theorem 5.14 for the quantum algorithm, but for the analysis of the decoding distance it is the same as the classical one), Footnote 5 and those m samples whose joint distribution is 0.5-close $\mathcal{D}^m_{\mathcal{L},s_{i_0}}$, with constant probability over the choice of L_{i_0}, the algorithm of Theorem 5.1(Theorem 5.14) solves $\psi(\mathcal{L})/\lambda_1(\mathcal{L})$-BDD when given L_{i_0}, where

$$\psi(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon')/\pi - o(1)}}{2\eta_{\varepsilon'}(\mathcal{L}^*)}.$$

[8] Note that the hDGS sampler is allowed to return fewer than m samples if $s_i < \eta_\varepsilon$: in this case, we do not care about the distribution of the vectors anyway so we can add random vectors until we get m samples when that happens.

[9] As noted in Footnote 5, if we assume that all DGS samples have $\text{poly}(n)$ bit-size, then the reduction from Theorem 5.1 has time complexity $m \cdot \text{poly}(n)$ and space complexity $O(\text{poly}(n) + \ln m)$ *excluding the storage space of the m vectors provided by the DGS*. Furthermore, we can ensure that all DGS samples have $\text{poly}(n)$ bit-size by first generating more samples (say twice the amount) and throwing away all samples of norm larger than $\exp(\Omega(n^2))$. Since $\mathcal{D}_{\mathcal{L},s}$ is sub-Gaussian with parameter s by [MP12, Lemma 2.8] and the vectors are sampled from $\mathcal{D}_{\mathcal{L},s}$ with $s = \exp(O(n))$, the error induced by throwing away the tail of the distribution is smaller than $2^{-\Omega(n^2)}$ in total variation distance.

Here we use the fact that $m = |L_{i_0}| \geqslant m' := O(\frac{n\ln(1/\varepsilon')}{\sqrt{\varepsilon'}})$ which holds because

$$\frac{n\ln(1/\varepsilon')}{\sqrt{\varepsilon'}} \leqslant \frac{n\Delta^2\ln(1/\varepsilon)+o(n)}{\sqrt{\varepsilon^{\delta^2}f}} \leqslant \frac{n\ln(1/\varepsilon)+n^{1-c}\ln(1/\varepsilon)+o(n)}{\sqrt{\varepsilon}\varepsilon^{n^{-c}/2}e^{o(1)}} = \mathcal{O}\left(\frac{n\ln(1/\varepsilon)}{\sqrt{\varepsilon}}\right)$$

by Footnote 9, $n^{1-c}\ln\varepsilon = n^{1-c+\alpha} = o(1)$ and $\varepsilon^{n^{-c}/2} = e^{n^{\alpha-c}/2} = e^{o(1)}$. It follows that, with constant probability over the preprocessing stage, $\mathcal{A}_c(\mathcal{A}_q)$ solves $\psi(\mathcal{L})/\lambda_1(\mathcal{L})$-BDDP. We can verify that $\varepsilon^{\delta^2}f \geqslant \varepsilon^{\delta^2+\ln f}$ since $\varepsilon < \frac{1}{e}$ and thus by Lemma 2.51 and Footnote 9,

$$\eta_{\varepsilon^{\delta^2}f}(\mathcal{L}^*) \leqslant \eta_{\varepsilon^{\delta^2+\ln f}}(\mathcal{L}^*) \leqslant (\delta^2+\ln f)\eta_\varepsilon(\mathcal{L}^*) \leqslant (1+o(1))\eta_\varepsilon(\mathcal{L}^*)$$

since $\delta = 1+o(1)$. Hence we have

$$\psi(\mathcal{L}) \geqslant \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2(1+o(1))\eta_\varepsilon(\mathcal{L}^*)} \equiv \phi(\mathcal{L}).$$

□

5.3.2 Tradeoff between the time complexity of BDDP and its decoding distance

In the previous subsection, we showed how to create a BDDP oracle by DGS samples without knowing the exact smoothing parameter, and in this subsection, we will try to give explicit lower bounds for the decoding distance of the BDDP oracle constructed in the previous section and subsection, and then give a tradeoff between time and decoding distance. We will discuss this tradeoff by splitting the interval of possible values of ε (of the smoothing parameter η_ε) into two parts. We will heavily use Theorem 5.15 and Lemma 2.58 to relate the smoothing parameter to other parameters (ε and β) of the lattice.

When ε is small ($\varepsilon \leq (e/\beta)^{-n}$)

A small difficulty when applying Lemma 2.58 is the case distinction on ε. We will start by using Eq (2.9), which will require taking very small values of ε when sampling discrete Gaussian samples.

Lemma 5.16. *Let $n \geq 17$. There exist two algorithms (one classical and one quantum) that, on input a basis $\mathbf{B}$ of lattice $\mathcal{L} \subset \mathbb{R}^n$, with constant probability, create a classical and a quantum α-BDD oracle respectively (α will be specified later). Both algorithms take time $2^{(A+1)n/2+o(n)}$, space $2^{0.5n+o(n)}$, and the quantum one uses additional $2^{An/2+o(n)}$ QRAM bits, where $b = \log_2\beta(\mathcal{L})$, A is an arbitrary positive value satisfying $\frac{1}{2\ln 2} - b + o(1) \leqslant A \leqslant 1$, and $\alpha = \frac{1}{2}\sqrt{\frac{A}{A+b}}$. Every extra call to the classical oracle takes time $2^{An/2+o(n)}$. Every extra call to the quantum oracle takes time $2^{An/4+o(n)}$ and $2^{An/4+o(n)}$ queries to the preprocessed data stored in a QRAM memory.*

Proof. Let $\varepsilon = 2^{-An}$, $A \leqslant 1$ to be fixed later. We know that $\eta_\varepsilon(\mathcal{L}^*) > \eta_{1/3}(\mathcal{L}^*)$ for any sufficiently large n ($n > \frac{1}{A}\log_2 3$) by the monotonicity of the smoothing parameter function. Hence for any $m \in \mathbb{N}$, the $\mathrm{DGS}^m_{\eta_{1/3}}$ sampler from Lemma 2.55 can be used as a $\mathrm{DGS}^m_{\eta_\varepsilon}$ sampler.

By Theorem 5.15 we can construct an α-BDD such that each call takes time $m \cdot \mathrm{poly}(n) = 2^{An/2+o(n)}$ and space $\mathrm{poly}(n)$, where $\alpha = \phi(\mathcal{L})/\lambda_1(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$ and $m = O(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}}) = 2^{An/2+o(n)}$. The preprocessing stage consists of $\mathrm{poly}(n)$ calls to the $\mathrm{hDGS}^m_{\eta_\varepsilon}$ sampler described above and uses $m \cdot \mathrm{poly}(n)$ space. Hence the total complexity is $\mathrm{poly}(n) \cdot m \cdot 2^{n/2+o(n)} = 2^{(A+1)n/2+o(n)}$ in time and $2^{An/2+o(n)} \leqslant 2^{n/2+o(n)}$ in space. By using Eq (2.9) in Lemma 2.58, which is only valid when $\varepsilon \leq (e/\beta(\mathcal{L})^2 + o(1))^{-\frac{n}{2}}$ (and hence gives a lower bound for A), we have that

$$\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L}) < \sqrt{\frac{\ln(1/\varepsilon) + n\ln\beta(\mathcal{L}) + o(n)}{\pi}}.$$

Hence we can guarantee that

$$\alpha = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})} > \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\sqrt{\frac{\ln(1/\varepsilon)+n\ln\beta(\mathcal{L})+o(n)}{\pi}}} = \frac{1}{2}\sqrt{\frac{\ln(1/\varepsilon)+o(1)}{\ln(1/\varepsilon)+n\ln\beta(\mathcal{L})}} = \frac{1}{2}\sqrt{\frac{A}{A+b}} + o(1)$$

where $b = \log_2 \beta(\mathcal{L})$. Furthermore, as noted above, this inequality holds when $\varepsilon \leq (e/\beta^2 + o(1))^{-\frac{n}{2}}$, that is $A \geqslant \frac{1}{2\ln 2} - b + o(1)$. Since $b \leqslant 0.401$, we must have $A \geqslant 0.32$ and the inequality holds as soon as $n \geqslant 5 \geqslant \frac{1}{A}\log_2 3$. Finally note that Theorem 5.15 requires $\varepsilon \leqslant 1/200$ which holds as soon as $n \geqslant 17 \geqslant \frac{1}{A}\ln 200$.

The quantum algorithm is exactly the same but using the quantum oracle of Theorem 5.15, which uses $m \cdot \mathrm{poly}(n) = 2^{An/2+o(n)}$ QRAM bits in the preprocessing stage and each call after the preprocessing stage takes time $\sqrt{m} \cdot \mathrm{poly}(n) = 2^{An/4+o(n)}$ (and the same number of queries to the preprocessing data stored in a QRAM memory). □

We can reformulate the previous lemma by expressing the complexity in terms of α instead of some arbitrary constant A.

Corollary 5.17. *Let $n \geqslant 17$. There exist two algorithms (one classical and one quantum) that, on input a basis $\mathbf{B}$ of lattice $\mathcal{L} \subset \mathbb{R}^n$, with constant probability, create a classical and a quantum α-BDD oracle respectively (α will be specified later). Both algorithms take time $2^{(A+1)n/2+o(n)}$ and space $2^{0.5n+o(n)}$, and the quantum one uses additional $2^{An/2+o(n)}$ QRAM bits, where $b = \log_2 \beta(\mathcal{L})$, α is an arbitrary value satisfying $\frac{1}{2}\sqrt{1-2b\ln 2} + o(1) \leqslant \alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$, and $A = \frac{4b\alpha^2}{1-4\alpha^2}$. Every extra call to the classical oracle takes time $2^{An/2+o(n)}$; and every extra call to the quantum oracle takes time $2^{An/4+o(n)}$ and $2^{An/4+o(n)}$ queries to the preprocessed data stored in a QRAM memory.*

Proof. Apply Lemma 5.16 for some A to be fixed later. Observe that $\alpha = \frac{1}{2}\sqrt{\frac{A}{A+b}}$ so

$A = \frac{4b\alpha^2}{1-4\alpha^2}$. Now the constraints $\frac{1}{2\ln 2} - b + o(1) \leqslant A \leqslant 1$ become

$$\begin{aligned}\tfrac{1}{2\ln 2} - b + o(1) \leqslant \tfrac{4b\alpha^2}{1-4\alpha^2} &\Leftrightarrow (\tfrac{1}{2\ln 2} - b + o(1))(1-4\alpha^2) \leqslant 4b\alpha^2 \\ &\Leftrightarrow \tfrac{1}{4\ln 2} - \tfrac{b}{2} + o(1) \leqslant \tfrac{\alpha^2}{\ln 2} \\ &\Leftrightarrow \tfrac{1}{2}\sqrt{1-2b\ln 2} + o(1) \leqslant \alpha\end{aligned}$$

and

$$\tfrac{4b\alpha^2}{1-4\alpha^2} \leqslant 1 \Leftrightarrow 4(1+b)\alpha^2 \leqslant 1 \Leftrightarrow \alpha \leqslant \tfrac{1}{2}\sqrt{\tfrac{1}{1+b}}.$$

□

When ε is large ($\varepsilon \geq (e/\beta)^{-n}$)

The Eq (2.9) in Lemma 2.58 tells us that if we take an extremely small ε to compute the BDD oracle, we can find a BDD oracle with $\alpha(\mathcal{L})$ almost 1/2. However, the time complexity for each call of the oracle will be very costly. On the other hand, if we use the Eq (2.8) in Lemma 2.58 with a larger ε, then each call of the oracle will take much less time, but the constraint on the decoding coefficient α will be different. It is, therefore, important to study this second regime as well. Note that Eq (2.8) actually applies to all $\varepsilon \in (0,1)$ but is mostly useful when ε is large.

Lemma 5.18. *Let $n \geq 17$. There exist two algorithms (one is classical and one is quantum) that, on input a basis $\mathbf{B}$ of lattice $\mathcal{L} \subset \mathbb{R}^n$, with constant probability, create a classical and a quantum α-BDD oracle respectively (α will be specified later). Both algorithms take time $2^{(A+1)n/2+o(n)}$, space $2^{0.5n+o(n)}$, and the quantum one uses additional $2^{An/2+o(n)}$ QRAM bits, where A is an arbitrary value satisfying $\frac{1}{n}\log_2 3 \leqslant A \leqslant 1$ and $\alpha = \frac{2^{-A}\sqrt{A}\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} - o(1)$. Every extra call to the classical oracle takes time $2^{An/2+o(n)}$. Every extra call to the quantum oracle takes time $2^{An/4+o(n)}$ and $2^{An/4+o(n)}$ queries to the preprocessed data stored in a QRAM memory.*

Proof. Let $\varepsilon = 2^{-An}$, $A \leqslant 1$ to be fixed later. We know that $\eta_\varepsilon(\mathcal{L}^*) > \eta_{1/3}(\mathcal{L}^*)$ for any sufficiently large n ($n > \frac{1}{A}\log_2 3$) by the monotonicity of the smoothing parameter function. Hence for all $m \in \mathbb{N}$, the $\mathrm{DGS}^m_{\eta_{1/3}}$ sampler from Lemma 2.55 can be used as a $\mathrm{DGS}^m_{\eta_\varepsilon}$ sampler.

By Theorem 5.15 we can construct a α−BDD such that each call takes time $m \cdot \mathrm{poly}(n) = 2^{An/2+o(n)}$ and space $\mathrm{poly}(n)$, where $\alpha = \phi(\mathcal{L})/\lambda_1(\mathcal{L}) = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})}$ and $m = O(\frac{n\log(1/\varepsilon)}{\sqrt{\varepsilon}}) = 2^{An/2+o(n)}$. The preprocessing stage consists of $\mathrm{poly}(n)$ calls to the $\mathrm{DGS}^m_{\eta_\varepsilon}$ sampler described above and uses $m \cdot \mathrm{poly}(n)$ space. Hence the total complexity is $\mathrm{poly}(n) \cdot m \cdot 2^{n/2+o(n)} = 2^{(A+1)n/2+o(n)}$ in time and $2^{An/2+o(n)} \leqslant 2^{n/2+o(n)}$ in space. By using Eq (2.8) in Lemma 2.58, we have that

$$\lambda_1(\mathcal{L})\eta_\varepsilon(\mathcal{L}^*) < \sqrt{\frac{\beta(\mathcal{L})^2 n}{2\pi e}} \cdot \varepsilon^{-1/n}(1+o(1)).$$

Hence we can guarantee that

$$\alpha = \frac{\sqrt{\ln(1/\varepsilon)/\pi - o(1)}}{2\eta_\varepsilon(\mathcal{L}^*)\lambda_1(\mathcal{L})} > \frac{1}{2}\sqrt{\frac{2e\ln\frac{1}{\varepsilon} - o(1)}{n}} \cdot \beta(\mathcal{L})^{-1}\varepsilon^{\frac{1}{n}} \cdot (1-o(1)) = \frac{2^{-A}\sqrt{A}\cdot\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} - o(1).$$

The quantum algorithm is exactly the same but using the quantum oracle of Theorem 5.15, which uses $m \cdot \mathrm{poly}(n) = 2^{An/2+o(n)}$ QRAM bits in the preprocessing stage, and each call after the preprocessing stage takes time $\sqrt{m}\cdot\mathrm{poly}(n) = 2^{An/4+o(n)}$ (and the same amount of the queries to the preprocessing data stored in a QRAM memory). □

We again reformulate the previous lemma by expressing the complexity in terms of α instead of some arbitrary constant A. To express A in terms of α, we have to introduce the principal branch of the Lambert W function, which is defined to be the function satisfying $W(z)\cdot\exp(W(z)) = z$ for arbitrary $z \in \mathbb{R}$.

Corollary 5.19. *Let $n \geqslant 17$. There exist two algorithms (one is classical and one is quantum) that, on input a basis $\mathbf{B}$ of lattice $\mathcal{L} \subset \mathbb{R}^n$, with constant probability, create a classical and a quantum $(\alpha+o(1))$-BDD oracle (α will be specified later). Both algorithms take time $2^{(A+1)n/2+o(n)}$ and space $2^{n/2+o(n)}$, and the quantum one uses additional $2^{An/2+o(n)}$ QRAM bits, where α is an arbitrary value satisfying $\frac{\sqrt{e\ln 3}}{\sqrt{2n}\beta(\mathcal{L})} \leqslant \alpha \leqslant \frac{1}{2\beta(\mathcal{L})}$. Every extra call to the classical oracle takes time $2^{An/2+o(n)}$; and every extra call to the quantum oracle takes time $2^{An/4+o(n)}$ and $2^{An/4+o(n)}$ queries to the preprocessed data stored in a QRAM memory, where*

$$A = -\frac{1}{2\ln 2}W\left(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e}\right).$$

Furthermore, the above expression of A is a continuous and increasing function of $\beta(\mathcal{L})$.

Proof. By Lemma 5.18, we can build an oracle for any $\frac{1}{n}\log_2 3 \leqslant A \leqslant 1$ such that the decoding radius is $\alpha = \frac{2^{-A}\sqrt{A}\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} - o(1)$. Hence, we want to find A such that

$$\frac{2^{-A}\sqrt{A}\sqrt{2e\ln 2}}{2\beta(\mathcal{L})} = \alpha \qquad \text{and} \qquad \tfrac{1}{n}\log_2 3 \leqslant A \leqslant 1.$$

Let $f : A \mapsto 2^{-A}\sqrt{A}$ so that the first condition is equivalent to

$$f(A) = \frac{2\alpha\beta(\mathcal{L})}{\sqrt{2e\ln 2}}. \tag{5.10}$$

Now assume that Eq (5.10) holds and let $y = -2A\ln(2)$, then it is equivalent to

$$e^y y = -2\ln(2)\frac{2\alpha^2\beta(\mathcal{L})^2}{e\ln 2},$$

that is

$$e^y y = -\frac{4\alpha^2\beta(\mathcal{L})^2}{e}. \tag{5.11}$$

This equation admits a solution if and only if

$$-\frac{4\alpha^2\beta(\mathcal{L})^2}{e} \geqslant -\frac{1}{e}, \quad \text{which can be reformulated to} \quad \alpha \leqslant \frac{1}{2\beta(\mathcal{L})}. \tag{5.12}$$

Assuming this is the case, Eq (5.11) can admit up to two solutions. However, since the complexity increases with A, we want the solution that minimizes A, i.e. that maximizes y. The largest of the (up to) two solutions of Eq (5.11) is always given by the principal branch W of the Lambert W function:

$$y = W(-\tfrac{4\alpha^2\beta(\mathcal{L})^2}{e}) \quad \text{that is} \quad A = -\tfrac{1}{2\ln 2} W(-\tfrac{4\alpha^2\beta(\mathcal{L})^2}{e}), \tag{5.13}$$

and always satisfies $y \geqslant -1$. In particular, we always have $A \leqslant \frac{1}{2\ln 2}$. Note that f is strictly increasing over $[0, \frac{1}{2\ln 2}]$. Hence, the condition $\frac{1}{n}\log_2 3 \leqslant A$ is equivalent to

$$\begin{aligned}
& f(\tfrac{1}{n}\log_2 3) \leqslant f(A) \\
& \Leftrightarrow f(\tfrac{1}{n}\log_2 3)^2 \leqslant \left(\tfrac{2\alpha\beta(\mathcal{L})}{\sqrt{2e\ln 2}}\right)^2 && \text{by Eq (5.10)} \\
& \Leftrightarrow 2^{-\frac{2}{n}\log_2 3}\tfrac{1}{n}\log_2 3 \leqslant \tfrac{2\alpha^2\beta(\mathcal{L})^2}{e\ln 2} \\
& \Leftrightarrow \tfrac{e\ln 3}{2n\beta(\mathcal{L})^2} 9^{-\frac{1}{n}} \leqslant \alpha^2 \\
& \Leftarrow \tfrac{e\ln 3}{2n\beta(\mathcal{L})^2} \leqslant \alpha^2. && (5.14)
\end{aligned}$$

In summary, we can always take A as in Eq (5.13) assuming Eq (5.12) and Eq (5.14) hold. □

5.3.3 Putting everything together

We have analyzed the construction of α-BDD oracles in two regimes, based on Lemma 2.55. It is not a priori clear which construction is better, and in fact, we will see that it depends in a nontrivial way on the relation between α and $\beta(\mathcal{L})$.

Theorem 5.20. *Let $n \geq 17$. There exist two algorithms (one is classical and one is quantum) that, on input a basis $\mathbf{B}$ of lattice $\mathcal{L} \subset \mathbb{R}^n$, with constant probability, creates a classical and a quantum α-BDD oracle (α will be specified later). Both algorithms take time $2^{(A+1)n/2+o(n)}$ and space $2^{n/2}$, and the quantum one uses additional $2^{An/2+o(n)}$ QRAM bits, where $b = \log_2 \beta(\mathcal{L})$ and $\frac{\sqrt{e\ln 3}}{\sqrt{2n}\beta(\mathcal{L})} \leqslant \alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$. Every extra call to the classical oracle takes time $2^{An/2+o(n)}$. Every extra call to the quantum oracle takes time $2^{An/4+o(n)}$ and $2^{An/4+o(n)}$ queries to the preprocessed data stored in a QRAM memory, where*

$$A = \begin{cases} -\frac{1}{2\ln 2} W(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e}) & \text{when } b < \frac{1-4\alpha^2}{2\ln 2} \\ \frac{4\alpha^2}{1-4\alpha^2} b & \text{when } b \geqslant \frac{1-4\alpha^2}{2\ln 2}. \end{cases}$$

Furthermore, the above expression of A is a continuous and increasing function of b.

Proof. Let $\frac{\sqrt{e\ln 3}}{\sqrt{2n}\beta(\mathcal{L})} \leqslant \alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$ and $b = \log_2 \beta(\mathcal{L})$. By Corollary 5.17, we can build an α-BDD if $\frac{1}{2}\sqrt{1-2b\ln 2} \leqslant \alpha$, in which case the complexity will depend on $A = A_1(\alpha, b) := \frac{4b\alpha^2}{1-4\alpha^2}$. By Corollary 5.19, we can build an α-BDD if $\alpha < \frac{1}{2}\sqrt{1-2b\ln 2}$, in which case the

complexity will depend on $A = A_2(\alpha, b) := -\frac{1}{2\ln 2} W(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e})$. In both cases, the BDD oracle can be created in time $2^{(A+1)n/2+o(n)}$, space $2^{0.5n+o(n)}$ and each call takes time $2^{An/2+o(n)}$. Now observe that

$$\alpha < \tfrac{1}{2}\sqrt{1-2b\ln 2} \Leftrightarrow 4\alpha^2 < 1-2b\ln 2 \Leftrightarrow b < \tfrac{1-4\alpha^2}{2\ln 2}.$$

Let $b^* := \frac{1-4\alpha^2}{2\ln 2}$. Then there are two cases:

- If $b \geqslant b^*$ then $\frac{1}{2}\sqrt{1-2b\ln 2} \leqslant \alpha$ so only Corollary 5.17 applies and we can build a α-BDD. In this case the complexity exponent is $A_1(\alpha, b) = \frac{4\alpha^2}{1-4\alpha^2} b$.
- If $b < b^*$ then $\alpha < \frac{1}{2}\sqrt{1-2b\ln 2}$ so Corollary 5.19 applies but Corollary 5.17 does not for this particular value of α. However, we can apply Corollary 5.17 to build an α'-BDD oracle with $\alpha' \geqslant \alpha_1^{min}(b) := \frac{1}{2}\sqrt{1-2b\ln 2} > \alpha$. We will show that the α-BDD of Corollary 5.19 is always more efficient than the α'-BDD of Corollary 5.17 in this case and the complexity exponent will thus be $A_2(\alpha, b) = -\frac{1}{2\ln 2} W(-\frac{4\alpha^2\beta(\mathcal{L})^2}{e})$.

Assume that $b < b^*$, we claim that $A_1(\alpha, b) \geqslant A_2(\alpha', b)$ for any $\alpha' \geqslant \alpha_1^{min}(b)$. Indeed, on the one hand A_2 is an increasing function of b so

$$A_2(\alpha, b) < A_2(\alpha, \tfrac{5}{18\ln 2}) = -\tfrac{1}{2\ln 2} W(-4\alpha^2 e^{-4\alpha^2}) = \tfrac{4\alpha^2}{2\ln 2} = \tfrac{2\alpha^2}{\ln 2} \qquad \text{since } W(xe^x) = x.$$

On the other hand, A_1 is an increasing function of α so

$$A_1(\alpha, b) \geqslant A_1(\alpha_1^{min}(b), b) = \tfrac{1-2b\ln 2}{2\ln 2}$$

which is a decreasing function of b, therefore

$$A_1(\alpha, b) \geqslant A_1(\alpha_1^{min}(b^*), b^*) = \tfrac{1-2b^*\ln 2}{2\ln 2} = \tfrac{2\alpha^2}{\ln 2} > A_2(\alpha, b).$$

□

5.4 Solving SVP by spherical caps on the sphere

In this section, we will show how to use the BDD oracles we constructed in the previous section to solve SVP. One can use Theorem 2.45 to directly solve SVP using BDD by choosing $q = 3$ and $\alpha = 1/3$. In this case, we make 3^n ($3^{n/2}$ quantum) calls to the BDD oracle and the time complexity is simply $|1/3\text{-BDD}| + 3^{0.5n} \cdot |1/3\text{-BDDP}|$ using quantum minimum-finding, where $|1/3\text{-BDD}|$ is the time complexity of building up 1/3-BDD and $|1/3\text{-BDDP}|$ is the cost of extra queries to this 1/3-BDD oracle. One can already achieve faster quantum algorithms (than the $2^{n+o(n)}$-time classical algorithm of [ADR+15]) using this simple strategy. Since the decoding distance α needs to be $< 1/2$ (if we want to use Theorem 2.45), seemingly we cannot avoid 3^n ($3^{n/2}$ quantum) calls to the BDD oracle. Or can we?

We now explain how to further reduce the number of queries to the α-BDD oracle as follows. Consider a target vector $\boldsymbol{t}$ uniformly at random on the sphere of the ball with radius $r\lambda_1$. We enumerate all lattice vectors within distance $2\alpha\lambda_1(\mathcal{L})$ of $\boldsymbol{t}$ and keep only the shortest nonzero one. We show that for any $\alpha \geqslant \frac{1}{3}$, we will get the shortest nonzero vector of the lattice with probability at least $2^{-cn+o(n)}$ for some c that depends on α. This probability is typically the ratio of the area of the spherical cap to the whole sphere area (see Figure 5.1).

One can simply generate $2^{cn+o(n)}$ samples uniformly at random on $r\lambda_1 \cdot S^{n-1}$ to ensure that we can find the shortest vector. To achieve a further quadratic quantum speedup over the sphere, we would like to have a unitary that prepares a quantum state with the property that if we measure its last register in the computational basis, the measurement outcome will distribute close to a sample that is uniform over $r\lambda_1 \cdot S^{n-1}$. To achieve that, it suffices to first sample a Gaussian vector $\boldsymbol{g}$, prepare the quantum state $\sum_{\boldsymbol{x}\in\mathbb{Z}^n} \sqrt{\frac{\rho_s(\boldsymbol{x})}{\rho_s(\mathbb{Z}^n)}}\,|\boldsymbol{x}\rangle$, and then output $\sum_{\boldsymbol{x}\in\mathbb{Z}^n} \sqrt{\frac{\rho_s(\boldsymbol{x})}{\rho_s(\mathbb{Z}^n)}}\,|\boldsymbol{x}\rangle\,|\boldsymbol{g}\rangle\,|\boldsymbol{x}+\boldsymbol{g}\rangle\,|r\lambda_1\frac{\boldsymbol{x}+\boldsymbol{g}}{\|\boldsymbol{x}+\boldsymbol{g}\|}\rangle$. By Lemma 2.50, one can see that if both widths of continuous Gaussian and discrete Gaussian are big enough, the measurement outcome of the third register will be close to a sample drawn from a continuous Gaussian, and hence its normalized outcome will be close to a sample drawn uniformly at random from the sphere. Here, we include the following theorem to show how to efficiently prepare the discrete Gaussian state over $\mathbb{Z}^n$.

Theorem 5.21. *Let n be an integer, $\delta \in (0,1)$, $s > 0$, and $|\rho_s(\mathbb{Z}^n)\rangle = \sum_{\boldsymbol{x}\in\mathbb{Z}^n} \sqrt{\frac{\rho_s(\boldsymbol{x})}{\rho_s(\mathbb{Z}^n)}}\,|\boldsymbol{x}\rangle$. There exists a quantum algorithm that prepares $|\psi\rangle$ such that $\| |\psi\rangle - |\rho_s(\mathbb{Z}^n)\rangle \| \leq n\delta$, using $\tilde{\mathcal{O}}(ns \cdot \mathrm{poly}\log(ns/\delta))$ time.*

Proof. One can see that $\sum_{\boldsymbol{x}\in\mathbb{Z}^n} \sqrt{\frac{\rho_s(\boldsymbol{x})}{\rho_s(\mathbb{Z}^n)}}\,|\boldsymbol{x}\rangle = \bigotimes_{i\in[n]} \sum_{x_i\in\mathbb{Z}} \sqrt{\frac{\rho_s(x_i)}{\rho_s(\mathbb{Z})}}\,|x_i\rangle$. So it suffices to show how to prepare a quantum state that is δ-close to $\sum_{x\in\mathbb{Z}} \sqrt{\frac{\rho_s(x)}{\rho_s(\mathbb{Z})}}\,|x\rangle$ in $\tilde{\mathcal{O}}(s\cdot\mathrm{poly}\log(s/\delta))$ time.

Let $N = \lceil 2s\cdot\sqrt{\ln(2/\delta)}\rceil$ and set $S = \{-N, -N+1, \ldots, N\}$. By Theorem 2.24 and the discussion below Theorem 2.24, one can see that the state $\sum_{x\in S} \sqrt{\frac{\rho_s(x)}{\rho_s(S)}}\,|x\rangle$ is δ-close to $\sum_{x\in\mathbb{Z}} \sqrt{\frac{\rho_s(x)}{\rho_s(\mathbb{Z})}}\,|x\rangle$. To prepare the state $\sum_{x\in S} \sqrt{\frac{\rho_s(x)}{\rho_s(S)}}\,|x\rangle$, one can compute $\rho_s(x)$ for all $x \in S$ and then store those values in the leaves of a KP-tree with respect to ℓ_2-norm (see Definition 2.13).[10] Since the sparsity of this KP-tree is $2N+1$, using Theorem 2.14 and Theorem 2.12, one can prepare the state $\sum_{x\in S} \sqrt{\frac{\rho_s(x)}{\rho_s(S)}}\,|x\rangle$ using $\mathcal{O}(N\log N) = \tilde{\mathcal{O}}(s\cdot\mathrm{poly}\log(s/\delta))$ time. □

Now we are ready to reduce SVP to α-BDD. The optimal choice of α (for solving SVP most efficiently) is not obvious and is deferred to the end of this section and Section 5.5.

[10] Note that the KP-tree here does not use any QRAM bits. It is just a pure classical data structure if we

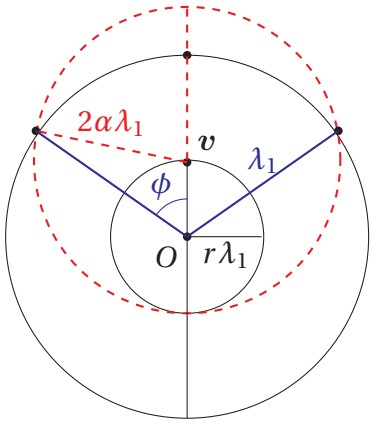

Figure 5.1: One can cover the sphere of radius λ_1 by balls of radius $2\alpha\lambda_1$, where $\frac{1}{3} \leqslant \alpha < \frac{1}{2}$, whose centers (here $\boldsymbol{v}$) are at distance $r\lambda_1$ from the origin O. Each such ball covers a spherical cap of half-angle ϕ.

Theorem 5.22. *Suppose we can create a (quantum) α-BDD oracle, with $\alpha \geqslant \frac{1}{3}$, in time T, space S (and using poly(n) qubits and S_q QRAM bits) such that each extra call takes time T_o. Then there exists a quantum algorithm that with constant probability, solves SVP in classical space S (and using poly(n) qubits and S_q QRAM bits) and in time $2^{o(n)} \cdot (T + T_o\alpha^{-n/2})$.*

Proof. On input lattice $\mathcal{L}(\mathbf{B})$, use Theorem 2.40 to get a number d that satisfies $\lambda_1(\mathcal{L}) \leq d \leq 2^n\lambda_1(\mathcal{L})$. For $i = 1,\ldots,2n^2$, let $d_i = d/(1+\frac{1}{2n})^i$. Also let BDD_α denote the α-BDD oracle and $f : \mathbb{Z}_2^n \times \mathbb{R}^n \to \mathcal{L}$ defined by [11]

$$f(\boldsymbol{x},\boldsymbol{v}) = \mathbf{B}\boldsymbol{x} - 2\cdot\mathrm{BDD}_\alpha(\mathcal{L},(\mathbf{B}\boldsymbol{x}-\boldsymbol{v})/2). \tag{5.15}$$

Let $s = 100n^{100}$, $\nu_s := \rho_s/s^n$ be the n-dimensional continuous Gaussian probability density function with width s and $U_G : |r,\boldsymbol{g},0,0\rangle \to \sum_{\boldsymbol{x}\in\mathbb{Z}^n} \sqrt{\frac{\rho_s(\boldsymbol{x})}{\rho_s(\mathbb{Z}^n)}}\,|r,\boldsymbol{g}\rangle\,|\boldsymbol{x}\rangle\,|r\cdot\frac{\boldsymbol{x}+\boldsymbol{g}}{\|\boldsymbol{x}+\boldsymbol{g}\|}\rangle$. Note that by Theorem 5.21 with $\delta = 2^{-100n^2}$, one can implement U_G up to $2^{-\Theta(n^2)}$ error by poly(n) quantum elementary gates; then we define

$$U_0 : |\boldsymbol{x},0\rangle \to \begin{cases} |\boldsymbol{x},\|\boldsymbol{x}\|\rangle & \text{if } \|\boldsymbol{x}\| \neq 0 \\ |\boldsymbol{x},2d\rangle & \text{otherwise}, \end{cases}$$

which can be implemented up to negligible error by poly(n) elementary quantum gates. Also we define $U_f : |\boldsymbol{x},\boldsymbol{v},0\rangle \to |\boldsymbol{x},\boldsymbol{v},f(\boldsymbol{x},\boldsymbol{v})\rangle$, which can be implemented by first using

do not implement an oracle that allows quantum queries. See the discussion above Theorem 2.12 for more details.

[11] Even though a BDD oracle can output anything if the target is not close enough to the lattice, one can easily transfer a non-lattice vector to a lattice vector by simply using the basis vectors $\boldsymbol{b_1},\ldots,\boldsymbol{b_n} \in \mathbf{B}$ to round the output, and then convert it to a lattice vector. So here, without loss of generality, we assume that the output of f is always a lattice vector.

$\tilde{\mathcal{O}}(T)$ time in the preprocessing stage and then $\tilde{\mathcal{O}}(T_o)$ time for each application after the preprocessing stage. We now describe the algorithm. For $i = 1, \ldots, n^2$ we do the following. Let r_i be a value to be fixed later. Sample a Gaussian sample $\boldsymbol{g}$ from ν_s. Starting with $|0, r_i d_i, \boldsymbol{g}, 0, 0, 0, 0\rangle$, we apply $H^{\otimes n}$ on the first register, apply U_G on the second, third, fourth, and fifth registers, apply U_f on the first, fifth, and sixth registers, and U_0 on the last two registers. After that, we apply Theorem 2.4 to find the minimum of the last register and collect the corresponding vector $\boldsymbol{w}_i$ (which is on the sixth register). We output the shortest nonzero $\boldsymbol{w}_i$ at the end of the algorithm.

Since $d_i = d/(1+\frac{1}{2n})^i$ for all $i \in [2n^2]$ and $\lambda_1(\mathcal{L}) \le d \le 2^n \lambda_1(\mathcal{L})$, there exists $k \in [2n^2]$ such that $\lambda_1(\mathcal{L}) \le d_k \le (1+\frac{1}{n})\lambda_1(\mathcal{L})$. We assume we are working on this d_k for the rest of the proof. Let P_ϕ be the probability that the cosine of the angle between $\boldsymbol{x}+\boldsymbol{g}$ and the shortest vector is at most ϕ, where $\boldsymbol{x} \sim \mathcal{D}_{\mathbb{Z},s}$ and $\boldsymbol{g} \sim \nu_s$. To show the correctness and the time complexity of the algorithm, it suffices to lower bound the probability P_ϕ. This is because if the angle between the shortest vector and $\boldsymbol{v} := \boldsymbol{x}+\boldsymbol{g}$ is at most ϕ (of Figure 5.1), then by the correctness of Theorem 2.45, $\{f(\boldsymbol{x}, \boldsymbol{v})\}_{\boldsymbol{x} \in \mathbb{Z}_2^n}$ must include the shortest vector. Also, when measuring the last register, the smallest possible value is λ_1 because the last register stores the length of a nonzero lattice vector, so the generalized quantum minimum-finding can find the shortest vector using $\mathcal{O}(\sqrt{2^n/P_\phi})$ applications of $U_G^{\pm}$, $U_f^{\pm}$, $U_0^{\pm}$ and $\tilde{\mathcal{O}}(\sqrt{2^n/P_\phi})$ time. Since both U_G and U_0 can be implemented in polynomial time, the running time of the algorithm is therefore dominated by the cost of implementing U_f, and hence is $\mathcal{O}(T + T_o\sqrt{2^n/P_\phi}) \cdot \mathrm{poly}(n)$.

To lower bound the probability P_ϕ, we have to first choose the optimal r_k in our algorithm described above. Observe that a ball of radius $2\alpha\lambda_1$ at center $\boldsymbol{v} \in r_k d_k \cdot S^{n-1}$ will cover the spherical cap with angle $\phi = \arccos(\frac{\lambda_1^2 + r_k^2 d_k^2 - 4\alpha^2\lambda_1^2}{\lambda_1 r_k d_k})$ of the ball of radius λ_1 by the law of cosines. For convenience, we write $r_k d_k(1-\frac{1}{2n}) = r\lambda_1$ for some $r_k \le r \le (1+1/n)r_k$. We can now calculate the optimal choice of $r\lambda_1$ by noting that if we take the center of the caps to be at distance $r\lambda_1$ then the angle ϕ satisfies $\cos\phi = \frac{1+r^2-4\alpha^2}{2r} := f(r)$. We want to maximize the angle ϕ, since the area we can cover increases with ϕ. We can verify that that $f(r)$ is decreasing until $r = \sqrt{1-4\alpha^2}$ and then increasing. We conclude that the optimal radius r' is when $r' = \sqrt{1-4\alpha^2}$ and this gives an optimal angle ϕ' such that $\cos\phi' = \sqrt{1-4\alpha^2}$ and therefore $\sin\phi' = 2\alpha$. Note that $d_k \in (1 \pm 1/n)\lambda_1$. Hence when we choose our $r_k = \sqrt{1-4\alpha^2}$, $r_k d_k/\lambda_1 \in (1 \pm 1/n)r'$, and hence the corresponding $\sin\phi \in 2\alpha \pm o(1)$.

Let $\varepsilon = 2^{-100n^2}$, $\boldsymbol{u}_1$ denote the shortest vector. By Lemma 2.50, Lemma 2.49, and the fact $\lambda_n(\mathbb{Z}^n) = 1$, we have that $s/\sqrt{2} \ge \eta_\varepsilon(\mathbb{Z}^n)$ and hence $\boldsymbol{v}/\|\boldsymbol{v}\| = (\boldsymbol{x}+\boldsymbol{g})/\|\boldsymbol{x}+\boldsymbol{g}\|$ distributes 4ε-close to a vector drawn from S^{n-1} uniformly at random. Therefore we

obtain

$$\begin{aligned}
P_\phi &= \Pr_{x\sim\mathcal{D}_{\mathbb{Z}^n,s},\boldsymbol{g}\sim\nu_s}\left[\langle \boldsymbol{x}+\boldsymbol{g},\boldsymbol{u}_1\rangle \geq \cos\phi\cdot\|\boldsymbol{x}+\boldsymbol{g}\|\cdot\|\boldsymbol{u}_1\|\}|\right] \\
&\geq \Pr_{\boldsymbol{v}'\sim\nu_{\sqrt{2}s}}\left[\langle \boldsymbol{v}',\boldsymbol{u}_1\rangle \geq \cos\phi\cdot\|\boldsymbol{v}'\|\cdot\|\boldsymbol{u}_1\|\}|\right] - 4\varepsilon \\
&= \frac{A_n(\phi)}{2\cdot A_n(\pi/2)} - 4\varepsilon \\
&\geq 2^{-o(n)}(\sin\phi)^{-n} - 4\varepsilon \geq 2^{-o(n)}(2\alpha - o(1))^{-n} - 2^{-\Omega(n^2)} \\
&= 2^{-o(n)}\cdot(2\alpha)^{-n},
\end{aligned}$$

where the second equation holds because $\boldsymbol{x}+\boldsymbol{g}$ is 4ε-close to $\boldsymbol{v}'$ and fourth equation holds by the discussion in Section 2.10.1. Therefore the running time of the algorithm is $\mathcal{O}(T + T_o\sqrt{2^n/P_\phi})\cdot\mathrm{poly}(n) = 2^{o(n)}\cdot(T + T_o\alpha^{-n/2})$. □

By plugging in the best choice of the parameters, we obtain the following corollary.

Corollary 5.23. *There is a quantum algorithm without using QRAM that, with constant probability, solves SVP in time* $2^{0.9497n+o(n)}$*, classical space* $2^{0.5n+o(n)}$*, and poly*(n) *qubits. There is a quantum algorithm using QRAM that, with constant probability, solves the SVP in time* $2^{0.8345n+o(n)}$*, using* $2^{0.293n+o(n)}$ *QRAM bits, poly*(n) *qubits and classical space* $2^{0.5n}$.

Proof. Consider the classical BDD oracle Theorem 5.20 with $\alpha = 0.3473$:[12] since $0 \leqslant b = \log_2\beta(\mathcal{L}) \leqslant 0.401 + o(1)$, we indeed have that $\alpha < \frac{1}{2}\sqrt{\frac{1}{1+b}}$ so we can create a $(\alpha + o(1))$-BDD oracle in time $T_c = 2^{(A+1)n/2+o(n)}$, classical space $S_c = 2^{0.5n+o(n)}$ such that each extra call takes time $T_o = 2^{An/2+o(n)}$ where $A = A(b)$ is given by Theorem 5.20. The theorem also guarantees that $A(b)$ increases with b so $A \leqslant A(0.401)$. But $0.401 \geqslant \frac{1-4\alpha^2}{2\ln 2} \approx 0.3733$ so $A(0.401) = \frac{4\alpha^2}{1-4\alpha^2}\cdot 0.401$ by Theorem 5.20. Then we can apply Theorem 5.22 to get a quantum algorithm (without using QRAM) that solves SVP in classical space S_c, polynomial qubits, and in time

$$T := T_c + \frac{2^{o(n)}T_o}{\alpha^n} = 2^{(A+1)n/2+o(n)} + 2^{An/2-n/2\log_2\alpha+o(n)} = 2^{0.9497n+o(n)}.$$

For the quantum algorithm using QRAM, we consider the quantum BDD oracle in Theorem 5.20 with $\alpha' = 0.3853$: each extra call to the quantum oracle now takes $T_o' = 2^{A'n/4+o(n)}$ and it uses $T_q = 2^{A'n/2+o(n)}$ QRAM bits in the preprocessing stage, where $A' \approx 0.5862$ now. Then we apply Theorem 5.22 to get a quantum algorithm (without using QRAM) in time

$$T' := T_c + \frac{2^{o(n)}T_o'}{\alpha^n} = 2^{(A'+1)n/2+o(n)} + 2^{A'n/4-n/2\log_2\alpha+o(n)} = 2^{0.8345n+o(n)}$$

and using classical space $S_c = 2^{0.5n+o(n)}$, $T_q = 2^{0.293n+o(n)}$ QRAM bits and polynomially many qubits. □

[12] The optimal value of α was found numerically, see Section 5.5.

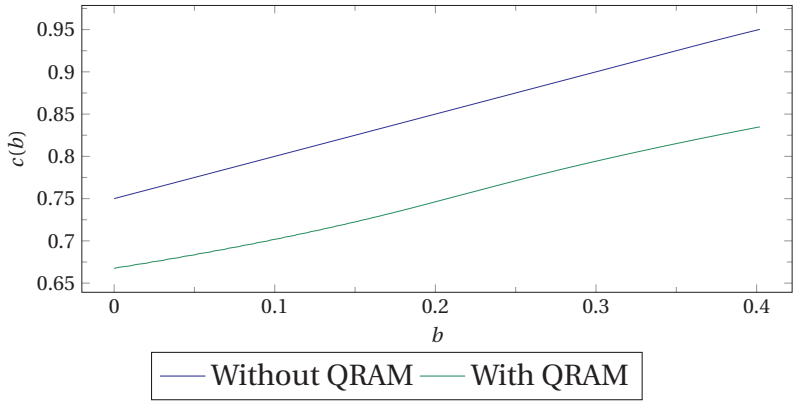

Figure 5.2: (Exponent $c(b)$ of the) time complexity of the spherical capping algorithm, plotted against $b = \log_2 \beta(\mathcal{L})$. The complexity of the algorithms is $2^{c(b)n+o(n)}$.

5.5 Dependence of our SVP solver on a quantity related to the kissing number

In the previous sections, we obtained several algorithms for SVP and bounded their complexity using the only known bound on the quantity $\beta(\mathcal{L})$, which is related to the lattice kissing number (see Equation 2.7): $\beta(\mathcal{L}) \leqslant 2^{0.402}$. The complexity of those algorithms is highly affected by this quantity and since $\beta(\mathcal{L})$ can be anywhere between 1 and $2^{0.402}$ (see Section 2.10.2), we will study the dependence of the time complexity on $\beta(\mathcal{L})$. Recall that $b = \log_2 \beta(\mathcal{L})$.

In order to avoid doing the analysis two times, we introduce a factor ξ that is 1 for the quantum algorithm without using QRAM, and $\frac{1}{2}$ for quantum algorithms using QRAM. We now can reformulate the time complexity in Theorem 5.22 as

$$T_c + \frac{2^{o(n)} T_o}{\alpha^{n/2}} \tag{5.16}$$

We instantiate the algorithm in Theorem 5.22 with the α-BDD oracle provided by Theorem 5.20 which satisfies

$$T_c = 2^{(A+1)n/2+o(n)}, \qquad T_o = 2^{A\xi n/2+o(n)} \quad \text{and} \quad S_c = 2^{0.5n+o(n)}$$

where

$$A = \begin{cases} -\frac{1}{2\ln 2} W(-\frac{4\alpha^2 \beta(\mathcal{L})^2}{e}) & \text{when } b < \frac{1-4\alpha^2}{2\ln 2} \\ \frac{4\alpha^2}{1-4\alpha^2} b & \text{when } b \geqslant \frac{1-4\alpha^2}{2\ln 2} \end{cases}.$$

The new expression of the time complexity is

$$2^{(A+1)n/2+o(n)} + 2^{(A\xi/2-0.5\log_2 \alpha)n+o(n)}. \tag{5.17}$$

Note that for the quantum algorithm without using QRAM, the cost of preprocessing stage is always negligible compared to the cost of the queries but this is not necessarily

the case when using QRAM memory. The optimal choice of α is not obvious: by increasing the decoding radius, we reduce the number of queries but increase the cost of each query. While we could, in principle, obtain closed-form expressions for the optimal value of α, those are too complicated to be really helpful. Instead, we express it as an optimization program. Formally, we have $T = 2^{c(b,\xi)n+o(n)}$ where

$$c(b,\xi) = \min_{\alpha\in[\frac{1}{3},\frac{1}{2})} \max\left(\tfrac{A+1}{2}, \tfrac{A\xi}{2} - 0.5\log_2\alpha\right)$$

where A and $\cos\phi$ are given by the expressions above that depend on α. We numerically computed the graph of this function and plotted the result in Figure 5.2 for quantum algorithms with and without QRAM, respectively. In particular, we obtain the following result when $\gamma(\mathcal{L}) = \beta(\mathcal{L})^n$ is subexponential in n:

Theorem 5.24. *For any family $(\mathcal{L}_n)_n$ of full-rank lattices such that $\mathcal{L}_n \subseteq \mathbb{R}^n$ and $\beta(\mathcal{L}_n)^n = 2^{o(n)}$, there are quantum algorithms to solve the SVP on $\mathcal{L}_n$:*

- *in time $2^{0.750n+o(n)}$, classical space $2^{0.5n+o(n)}$ and poly(n) qubits,*
- *in time $2^{0.667n+o(n)}$, classical space $2^{0.5n+o(n)}$, poly(n) qubits, and $2^{0.167n+o(n)}$ QRAM bits.*

5.6 Open problems

We mention a few directions for future work:

- Can our approach also give a faster provable quantum algorithm for CVP? The current fastest classical algorithm (and also the fastest quantum algorithm at the same time) still runs in $2^{n+o(n)}$ time.
- Can we further improve our quantum algorithms for SVP? A natural limit of our approach is $2^{0.5n}$ since to use Theorem 2.45, we have to at least choose $p = 2$ and in this case, the number of quantum queries to the BDD oracle is at least $2^{0.5n}$. There is still a huge gap with the best heuristic algorithm for SVP, which takes time $2^{0.292n+o(n)}$ classically by [BDG+16] ($2^{2563n+o(n)}$ quantumly by [BCS+23]).
- Is it possible to give a better upper bound for the lattice kissing number?

Part II

Quantum query lower bounds

CHAPTER 6

Query lower bounds for linear regression with norm constraints

6.1 Introduction

In Chapter 3, we discussed quantum algorithms for linear regression with norm constraints, and in this chapter, we prove quantum query lower bounds for those problems (Lasso and Ridge). In the quantum query access model, we assume we can have quantum access to entries of the samples, and we want to know how many quantum queries are necessary for outputting an ε-minimizer (a classical d-dimensional vector) for Lasso and Ridge. In the classical case, people usually care about how many samples we need for outputting an ε-minimizer, while in the quantum case, we assume those samples are already stored in a QRAM, and we care about how many queries we need for finding an ε-minimizer.

Cesa-Bianchi, Shalev-Shwartz, and Shamir [CSS11] provided a $\Omega(d/\varepsilon)$ classical query lower bound for both Lasso and Ridge (assuming query access to entries of X and y), and soon after Hazan and Koren [HK12] obtained an optimal (up to polylog) $\tilde{\Theta}(d/\varepsilon^2)$ classical query lower bound for Ridge.[1]

6.1.1 Main results and high-level intuition

We prove a lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ quantum queries for Lasso, showing that the d-dependence of our quantum algorithm is essentially optimal by combining the results in Chapter 3, while our ε-dependence might still be slightly improvable. Our lower bound strategy "hides" a subset of the columns of the data matrix X by letting those

[1] In [CSS11] and [HK12], the authors are actually instered in the limited attribute observation model (LAO). In the LAO model, for each given pair (X_i, y_i), the learner can access y_i, but only limited number of entries of X_i. The goal is to output an ε-minimizer using fewest samples. While the authors in [CSS11] and [HK12] concerned the lower bound for Lasso/Ridge in the LAO model, they actually provided lower bounds for Lasso/Ridge in query model, because a lower bound in the query model implies a lower bound in the LAO model.

columns have slightly more +1s than −1, and observes that an approximate minimizer for Lasso allows us to recover this hidden set. We then use the composition property of the adversary lower bound [BL20] together with a worst-case to average-case reduction to obtain a quantum query lower bound for this hidden-set-finding problem, and hence for Lasso.

Somewhat surprisingly, no tight *classical* lower bound was known for Lasso prior to this work. To the best of our knowledge, the previous-best classical lower bound was $\Omega(d/\varepsilon)$, due to Cesa-Bianchi, Shalev-Shwartz, and Shamir [CSS11]. As a byproduct of our quantum lower bound, we use the same set-hiding approach to prove for the first time the optimal (up to logarithmic factors) lower bound of $\tilde{\Omega}(d/\varepsilon^2)$ queries for classical algorithms for Lasso (Section 6.4).

What about Ridge? Because ℓ_2 is a more natural norm for quantum states than ℓ_1, one might hope that Ridge is more amenable to quantum speedup than Lasso. Unfortunately this turns out to be wrong: we prove a quantum lower bound of $\Omega(d/\varepsilon)$ queries for Ridge, using a similar strategy as for Lasso. This shows that the classical linear dependence of the runtime on d cannot be improved on a quantum computer (recall Hazan and Koren showed a $\Omega(d/\varepsilon^2)$ query lower bound). As we mentioned in Section 3.1, Ridge regression tends to keep all features in the model with reduced weights, allowing us to hide a subset of the columns with size $\Theta(d)$, while in the Lasso case, we can only hide a subset with size $\Theta(1/\varepsilon)$ (because ℓ_1-regularizer ensures that there is always an ε-minimizer with sparsity $\mathcal{O}(1/\varepsilon)$). Whether the ε-dependence can be improved remains an open question.

Roadmap

We show a Lasso solver can help us to solve the distributional set-finding problem (defined in the beginning of Section 6.2.1) approximately in Section 6.2.1, and a quantum query lower bound for the hidden-set finding problem in Section 6.2.2. After that, we provide a worst-case to average-case reduction for the set-finding problem in Section 6.2.3 and conclude a quantum query lower bound for Lasso. We use almost the same strategy to show a quantum query lower bound for Ridge in Section 6.3. We further dequantize our strategy to get a tight classical query lower bound (up to polylog factor), stated in Section 6.4

Remark

For the whole Chapter 6, we inherit the regression model in Section 3.2.2. In this chapter, when we say a (quantum) algorithm is bounded-error, it always means that it returns a correct output with probability 9/10; choosing 9/10 instead of 2/3 is to simplify parts of our statements and proofs.

6.2 Quantum query lower bounds for Lasso

In this section we prove a quantum lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ queries for Lasso. To show such a lower bound, we define a certain set-finding problem, and show how it can be solved by an algorithm for Lasso. After that, we show that the worst-case set-finding problem can be seen as the composition of two problems, which have query complexities $\Omega(\sqrt{d/\varepsilon})$ and $\Omega(1/\varepsilon)$, respectively. Then the composition property of the quantum adversary bound implies a $\Omega(\sqrt{d/\varepsilon}\cdot 1/\varepsilon) = \Omega(\sqrt{d}/\varepsilon^{1.5})$ query lower bound for Lasso.

6.2.1 Finding a hidden set W using a Lasso solver

In this subsection we define the *distributional set-finding problem*, and show how to reduce this to Lasso. Let $p \in (0,1/2)$, $W \subset [d]-1$, and $\overline{W} = [d]-1 \setminus W$. Define the distribution $\mathcal{D}_{p,W}$ over $(x,y) \in \{-1,1\}^d \times \{-1,1\}$ as follows. For each $j' \in \overline{W}$, $x_{j'}$ is generated according to $\Pr[x_{j'}=1] = \Pr[x_{j'}=-1] = 1/2$, and for each $j \in W$, x_j is generated according to $\Pr[x_j = 1] = 1/2 + p$. And y is generated according to $\Pr[y=1]=1$. The goal of the distributional set-finding problem $\mathrm{DSF}_{\mathcal{D}_{p,W}}$ with respect to $\mathcal{D}_{p,W}$ is to output a set $\tilde{W}$ such that $|\tilde{W}\Delta W| \le w/200$, given M samples from $\mathcal{D}_{p,W}$. One can think of the $M \times d$ matrix of samples as "hiding" the set W: the columns corresponding to $j \in W$ are likely to have more 1s than -1s, while the columns corresponding to $j \in \overline{W}$ have roughly as many 1s as -1s.

We first show some basic properties of $L_{\mathcal{D}_{p,W}}$. In this subsection, let

$$\theta^* = v \cdot e_W, \text{ where } v = 2p/(1+4p^2(w-1)) \text{ and } e_W = \sum_{j\in W} e_j.$$

Theorem 6.1. *Let θ be a vector in $\mathbb{R}^d$. We have*

- $L_{\mathcal{D}_{p,W}}(\theta) = \sum_{j'\in\overline{W}} \theta_{j'}^2 + \sum_{j\in W} (\theta_j - 2p)^2 + 4p^2 \sum_{j_1\in W} \sum_{j_2 \in W\setminus\{j_1\}} \theta_{j_1}\theta_{j_2} - 4p^2 w + 1$, *where* $w = |W|$.
- $\nabla_j L_{\mathcal{D}_{p,W}}(\theta) = \begin{cases} 2\theta_j - 4p + 8p^2 \sum_{\ell\in W\setminus\{j\}} \theta_\ell, & \text{if } j \in W, \\ 2\theta_j, & \text{otherwise.} \end{cases}$
- $[\nabla^2 L_{\mathcal{D}_{p,W}}(\theta)]_{jk} = \begin{cases} 2\delta_{jk}, & \text{if } j,k \in \overline{W}, \\ (2-8p^2)\delta_{jk} + 8p^2, & \text{if } j,k \in W, \\ 0, & \text{otherwise,} \end{cases}$

If we rearrange the order of indices such that $\{0,1,\dots,w-1\} \in W$ and $\{w, w+1, \dots, d-1\} \in \overline{W}$, then the Hessian of $L_{\mathcal{D}_{p,W}}$ is

$$\nabla^2 L_{\mathcal{D}_{p,W}} = \begin{pmatrix} 8p^2 J_w + (2-8p^2) I_w & 0_{w\times\bar{w}} \\ 0_{\bar{w}\times w} & 2I_{\bar{w}}, \end{pmatrix}$$

where $\bar{w} = d - w$. *Note that this is independent of* θ, *and* $\nabla^2 L_{\mathcal{D}_{p,W}} \succeq (2-8p^2) I_d$. *The unique global minimizer of* $L_{\mathcal{D}_{p,W}}$ *is* θ^*.

Proof. Note that for all distinct $j_1, j_2 \in W$,

$$\begin{aligned}\mathbb{E}_{(x,y)\sim\mathcal{D}_{p,W}}[x_{j_1}x_{j_2}] &= \Pr_{(x,y)\sim\mathcal{D}_{p,W}}[x_{j_1} = x_{j_2}]\cdot 1 + \Pr_{(x,y)\sim\mathcal{D}_{p,W}}[x_{j_1} = -x_{j_2}]\cdot(-1)\\ &= ((1/2+p)^2 + (1/2-p)^2) - 2(1/2+p)(1/2-p) = 4p^2.\end{aligned}$$

For all distinct j, j' such that $j \in [d]-1$ but $j' \in \overline{W}$, we have $\mathbb{E}_{(x,y)\sim\mathcal{D}_{p,W}}[x_j x_{j'}] = 0$.

We first prove the first bullet. By definition and the facts we mentioned above,

$$\begin{aligned}L_{\mathcal{D}_{p,W}}(\theta) &= \mathbb{E}_{(x,y)\sim\mathcal{D}_{p,W}}[(\langle x,\theta\rangle - y)^2] = \mathbb{E}_{(x,y)\sim\mathcal{D}_{p,W}}[(\sum_{j\in[d]-1} x_j\theta_j - y)^2]\\ &= \mathbb{E}_{(x,y)\sim\mathcal{D}_{p,W}}\left[\sum_{j\in[d]-1} x_j^2\theta_j^2 + \sum_{j_1\in[d]-1}\sum_{j_2\in[d]-1\setminus\{j_1\}} \theta_{j_1}\theta_{j_2}x_{j_1}x_{j_2} - 2\sum_{j\in[d]-1} x_j\theta_j y + y^2\right]\\ &= \sum_{j\in[d]-1}\theta_j^2 + 4p^2\sum_{j_1\in W}\sum_{j_2\in W\setminus\{j_1\}}\theta_{j_1}\theta_{j_2} - 4p\sum_{j\in W}\theta_j + 1\\ &= \sum_{j'\in\overline{W}}\theta_{j'}^2 + \sum_{j\in W}(\theta_j - 2p)^2 + 4p^2\sum_{j_1\in W}\sum_{j_2\in W\setminus\{j_1\}}\theta_{j_1}\theta_{j_2} - 4p^2\cdot|W| + 1.\end{aligned}$$

The last equation holds because $\sum_{j\in W}\theta_j^2 - 4p\sum_{j\in W}\theta_j = \sum_{j\in W}(\theta_j - 2p)^2 - 4p^2\cdot|W|$ by completing the square. The second and third bullets are easy to see by taking first and second partial derivatives of the expression of the first bullet. To see $\nabla^2 L_{\mathcal{D}_{p,W}} \succeq (2-8p^2) I_d$, note that $\nabla^2 L_{\mathcal{D}_{p,W}} - (2-8p^2) I_d$ is block-diagonal, where the $W \times W$ block is $4p^2$ times the all-1 matrix (which is positive semidefinite) and the $\overline{W} \times \overline{W}$ block is diagonal with diagonal entries $8p^2$ (which is positive definite).

The minimizer θ^* is a solution of the linear system one gets by setting all derivatives of the second bullet to 0. Because the Hessian is positive definite (we assumed $p < 1/2$, hence $2 - 8p^2 > 0$), this solution is the unique minimizer. □

The following theorem relates the entries of an approximate minimizer θ for Lasso with respect to distribution $\mathcal{D}_{p,W}$ to the elements of the hidden set W.

Theorem 6.2. *Let* $\varepsilon \in (2/d, 1/100)$, w *be either* $\lfloor 1/\varepsilon\rfloor$ *or* $\lfloor 1/\varepsilon\rfloor - 1$, $p = 1/(2\lfloor 1/\varepsilon\rfloor)$, *and* $W \subset [d]-1$ *be a set of size* w. *For every* $\theta \in B_1^d$ *satisfying* $L_{\mathcal{D}_{p,W}}(\theta) - L_{\mathcal{D}_{p,W}}(\theta^*) \le \varepsilon/8000$, *we have*

$$\sum_{j\in W}(\theta_j - v)^2 \le \varepsilon/6400 \text{ and } \sum_{j'\in\overline{W}}\theta_{j'}^2 \le \varepsilon/6400.$$

Proof. Let $\theta^* = v\cdot e_W$ be the minimizer of $L_{\mathcal{D}_{p,W}}$ from Theorem 6.1. Because $\nabla_j L_{\mathcal{D}_{p,W}}(\theta^*) = 0$ for every $j \in [d]-1$ and $\nabla^2 L_{\mathcal{D}_{p,W}}$ is a constant matrix, independent of θ, we have that

$$\begin{aligned}\varepsilon/8000 &\ge L_{\mathcal{D}_{p,W}}(\theta) - L_{\mathcal{D}_{p,W}}(\theta^*)\\ &= \langle\nabla L_{\mathcal{D}_{p,W}}(\theta^*), \theta-\theta^*\rangle + \langle\nabla^2 L_{\mathcal{D}_{p,W}}\cdot(\theta-\theta^*), \theta-\theta^*\rangle/2\\ &\ge 0 + (2-8p^2)\|\theta-\theta^*\|_2^2/2,\end{aligned}$$

which implies $\|\theta-\theta^*\|_2^2 \le \varepsilon/(8000\cdot(1-4p^2)) \le \varepsilon/6400$ from the fact that $p \le 1/200$. Because $\theta^* = v\cdot e_W$, we have $\|\theta-\theta^*\|_2^2 = \sum_{j\in W}(\theta_j - v)^2 + \sum_{j'\in\overline{W}}\theta_{j'}^2 \le \varepsilon/6400$, and therefore

$$\sum_{j\in W}(\theta_j - v)^2 \le \varepsilon/6400 \text{ and } \sum_{j'\in\overline{W}}\theta_{j'}^2 \le \varepsilon/6400.$$

□

Note that if $\varepsilon \in (2/d, 1/100)$, $1 \le w \le \lfloor 1/\varepsilon\rfloor$, $p = 1/(2\lfloor 1/\varepsilon\rfloor)$, then

$$\|\theta^*\|_1 = vw \le \frac{\frac{1}{\lfloor 1/\varepsilon\rfloor}\cdot\lfloor\frac{1}{\varepsilon}\rfloor}{1+\frac{1}{\lfloor 1/\varepsilon\rfloor^2}(1-1)} \le 1,$$

implying that $\theta^* \in B_1^d$, so the global minimizer actually satisfies Lasso's norm constraint. Now we are ready to show that algorithms for Lasso also find a good approximation to the hidden set W.

Theorem 6.3. *Let $\varepsilon \in (2/d, 1/100)$, w be either $\lfloor 1/\varepsilon\rfloor$ or $\lfloor 1/\varepsilon\rfloor - 1$, $p = 1/(2\lfloor 1/\varepsilon\rfloor)$, and $W \subset [d]-1$ be a set of size w. Let θ be an $\varepsilon/8000$-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$. Then the set $\tilde{W}$ that contains the indices of the entries of θ whose absolute value is $\ge \varepsilon/3$ satisfies $|W\Delta\tilde{W}| \le w/200$.*

Proof. Because θ is an $\varepsilon/8000$-minimizer for Lasso, Theorem 6.2 implies $\sum_{j\in W}(\theta_j - v)^2 \le \varepsilon/6400$ and $\sum_{j'\in\overline{W}}\theta_{j'}^2 \le \varepsilon/6400$. Hence

- at most $w/400$ many $j' \in \overline{W}$ have $|\theta_{j'}| \ge \sqrt{400\varepsilon/(6400w)} \ge \sqrt{\varepsilon/\lfloor 1/\varepsilon\rfloor}/4$,
- at least $w - w/400$ many $j \in W$ have $\theta_j \ge v - \sqrt{400\varepsilon/(6400w)} \ge v - \sqrt{\varepsilon/(\lfloor 1/\varepsilon\rfloor - 1)}/4$.

Note that $v - \sqrt{\varepsilon/(\lfloor 1/\varepsilon\rfloor - 1)}/4 \ge \varepsilon/3 \ge \sqrt{\varepsilon/\lfloor 1/\varepsilon\rfloor}/4$ for both the cases that $w = \lfloor 1/\varepsilon\rfloor$ and $w = \lfloor 1/\varepsilon\rfloor - 1$. Hence the set $\tilde{W}$ that contains the indices of the entries whose absolute value is $\ge \varepsilon/3$, omits at most $w/400$ of the $j \in W$ and includes at most $w/400$ of the $j' \in \overline{W}$. Therefore $|W\Delta\tilde{W}| \le w/400 + w/400 = w/200$. □

This implies that algorithms that find an $\varepsilon/8000$-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ can also find a set $\tilde{W} \subset [d]-1$ such that $|W\Delta\tilde{W}| \le w/200$.

6.2.2 Worst-case quantum query lower bound for the set-finding problem

Here we will define the worst-case set-finding problem and then provide a quantum query lower bound for it. Before we step into the query lower bound for the worst-case set-finding problem, we have to introduce the adversary method and the lower bounds for two problems first.

Theorem 6.4 ([Amb02], modified Theorem 6)**.** *Let $f(x_0,\dots,x_{d-1})$ be a function of d inputs with values from some finite set, $\varepsilon \in (0,1/2)$, and $\mathcal{X}, \mathcal{Y}$ be two sets of valid inputs for f. Let $\mathcal{R} \subseteq \mathcal{X}\times\mathcal{Y}$ be a relation such that*

- *For every $(x,y)\in\mathcal{R}$, $f(x)\neq f(y)$.*
- *For every $x\in\mathcal{X}$, there exist at least m different $y\in\mathcal{Y}$ such that $(x,y)\in\mathcal{R}$.*
- *For every $y\in\mathcal{Y}$, there exist at least m' different $x\in\mathcal{X}$ such that $(x,y)\in\mathcal{R}$.*

Let $\ell_{x,j}$ be the number of $y\in\mathcal{Y}$ such that $(x,y)\in\mathcal{R}$, and $x_j\neq y_j$ and $\ell_{y,j}$ be the number of $x\in\mathcal{X}$ such that $(x,y)\in\mathcal{R}$ and $x_j\neq y_j$. Let ℓ_{max} be the maximum of $\ell_{x,j}\cdot\ell_{y,j}$ over all $(x,y)\in\mathcal{R}$ and $j\in[d]-1$ such that $x_j\neq y_j$. Then, every quantum algorithm that computes f with success probability $1-\varepsilon$ uses at least $(1/2-\sqrt{\varepsilon(1-\varepsilon)})\cdot\sqrt{mm'/\ell_{max}}$ queries.

Using this adversary bound, we can give a query lower bound for the *exact set-finding problem*: given input $x = x_0\dots x_{d-1}\in\{0,1\}^d$ with at most w 1s, find the set W of all indices j with $x_j = 1$ (equivalently, learn x). To see the query lower bound for this problem, we consider the identity function where both domain and codomain are $\mathcal{Z} = \{z\in\{0,1\}^d : |z| = w\}$, and give a lower bound for computing this. If we can compute the identity function, then we can simply check the output string $x_0, x_1,\dots,x_{d-1}$ and collect all indices j with $x_j = 1$.

Theorem 6.5. *Let w be an integer satisfying $0 < w \le d/2$, $W\subset[d]-1$ with size w, and $x\in\{0,1\}^d$ such that $x_j = 1$ if $j\in W$ and $x_{j'} = 0$ if $j'\in\overline{W}$. Suppose we have query access to x. Then every quantum bounded-error algorithm to find W makes at least $\frac{1}{8}\sqrt{dw}$ queries.*

Proof. Note that if we can compute W, then we can compute the identity function $f:\mathcal{Z}\to\mathcal{Z}$. Let $\mathcal{X} = \mathcal{Y} = \mathcal{Z}$, and consider the relation $\mathcal{R}\subset\mathcal{X}\times\mathcal{Y}$ such that for every $(x,y)\in R$, $d_H(x,y) = 2$ (which implies $f(x)\neq f(y)$). Then we know

- For every $x\in\mathcal{X}$, there exist at least $m = w\cdot(d-w)$ different $y\in\mathcal{Y}$ such that $(x,y)\in R$.
- For every $y\in\mathcal{Y}$, there exist at least $m' = w\cdot(d-w)$ different $x\in\mathcal{X}$ such that $(x,y)\in R$.
- $\ell_{max} = \max\limits_{\substack{(x,y)\in\mathcal{R}, j\in[d]-1\\ s.t. x_j\neq y_j}} \ell_{x,j}\cdot\ell_{y,j} = w\cdot(d-w)$.

By Theorem 6.4, every quantum algorithm that computes f with probability $\ge 9/10$ uses at least

$$(1/2-\sqrt{1/10\cdot 9/10})\cdot\sqrt{mm'/\ell_{max}} \ge \frac{1}{5}\sqrt{w\cdot(d-w)} \ge \frac{1}{8}\sqrt{dw}$$

queries. □

We now prove a lower bound for the *approximate* set-finding problem $\mathrm{ASF}_{d,w}$, which is to find a set $\tilde{W} \subset [d]-1$ such that $|W \Delta \tilde{W}| \leq w/200$. The intuition of the proof is that if we could find such a $\tilde{W}$ then we can "correct" it to W itself using a small number of Grover searches, so finding a good approximation $\tilde{W}$ is not much easier than finding W itself.

Theorem 6.6. *Let w be an integer satisfying $0 < w \leq d/2$, $W \subset [d]-1$ with size w, and $x \in \{0,1\}^d$ such that $x_j = 1$ if $j \in W$ and $x_{j'} = 0$ if $j' \in \overline{W}$. Suppose we have query access to x. Then every bounded-error quantum algorithm that outputs $\tilde{W} \subset [d]-1$ satisfying $|W \Delta \tilde{W}| \leq w/200$ makes $\Omega(\sqrt{dw})$ queries.*

Proof. Suppose there exists a T-query bounded-error quantum algorithm to find a set $\tilde{W}$ satisfying $|W \Delta \tilde{W}| \leq w/200$. Define a function f which marks the elements of $F = W \Delta \tilde{W}$. Since we have a classical description of $\tilde{W}$, we can implement a query to f using one query to x. Now use Corollary 2.2 (with $u = w/200$) to find all elements of F with probability 1, using $\frac{\pi}{2}\sqrt{dw/200} + w/200$ queries. This gives a bounded-error quantum algorithm that finds W itself using $T' = T + \frac{\pi}{2}\sqrt{dw/200} + w/200$ queries. By Theorem 6.5 we have $T' \geq \frac{1}{8}\sqrt{dw}$, implying $T = \Omega(\sqrt{dw})$. □

Next we consider the *Hamming-weight distinguisher problem* $\mathrm{HD}_{\ell,\ell'}$: given a $z \in \{0,1\}^N$ of Hamming weight ℓ or ℓ', distinguish these two cases. The adversary bound gives the following bound (a special case of a result of Nayak and Wu [NW99] based on the polynomial method [BBC+01]).

Theorem 6.7. *Let $N \in 2\mathbb{Z}_+$, $z \in \{0,1\}^N$, and $p \in (0, 0.5)$ be multiple of $1/N$. Suppose we have query access to z. Then every bounded-error quantum algorithm that computes $HD_{\frac{N}{2},N(\frac{1}{2}+p)}$ makes $\Omega(1/p)$ queries.*

Proof. Let $\mathcal{X} = \{x \in \{0,1\}^N : |x| = N/2\}$, $\mathcal{Y} = \{y \in \{0,1\}^N : |y| = N/2 + pN\}$, and consider the relation $\mathcal{R} = \{(x,y) : x \in \mathcal{X}, y \in \mathcal{Y}, x \leq y\}$, where $x \leq y$ if and only if $\forall i \in [N]-1$, $x_i \leq y_i$. We know

- For every $x \in \mathcal{X}$, there exist at least $m = \binom{N/2}{pN}$ different $y \in \mathcal{Y}$ such that $(x,y) \in R$.
- For every $y \in \mathcal{Y}$, there exist at least $m' = \binom{N/2+pN}{pN}$ different $x \in \mathcal{X}$ such that $(x,y) \in R$.
- $\ell_{max} = \max\limits_{\substack{(x,y)\in\mathcal{R}, j\in[d]-1 \\ s.t. x_j \neq y_j}} \ell_{x,j} \cdot \ell_{y,j} = \binom{N/2-1}{pN-1} \cdot \binom{N/2+pN-1}{pN-1}$.

Hence by Theorem 6.4, every bounded-error quantum algorithm that computes g uses at least

$$\left(\frac{1}{2} - \sqrt{\frac{9}{10}\cdot\frac{1}{10}}\right)\cdot\sqrt{\frac{mm'}{\ell_{max}}} = \Omega\left(\sqrt{\frac{\binom{N/2}{pN}\cdot\binom{N/2+pN}{pN}}{\binom{N/2-1}{pN-1}\cdot\binom{N/2+pN-1}{pN-1}}}\right) = \Omega\left(\sqrt{\frac{N/2\cdot(N/2+pN)}{pN\cdot pN}}\right) = \Omega\left(\frac{1}{p}\right)$$

queries. □

The above theorem implies a lower bound of $\Omega(1/p)$ queries for $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$. One can also think of the input bits as ± 1 and in this case, the goal is to distinguish whether the entries add up to 0 or to $2pN$. For convenience, we abuse the notation $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$ also for the problem with ± 1 inputs. Now we are ready to prove a lower bound for the *worst-case set-finding problem* $\mathrm{WSF}_{d,w,p,N}$: given a matrix $X \in \{-1,1\}^{N\times d}$ where each column-sum is either $2pN$ or 0, the goal is to find a set $\tilde{W} \subset [d]-1$ such that $|\tilde{W}\Delta W| \le w/200$, where W is the set of indices for those columns whose entries add up to $2pN$ and $w = |W|$. One can see that this problem is actually a composition of the approximate set-finding problem and the Hamming-weight distinguisher problem. Composing the relational problem $\mathrm{ASF}_{d,w}$ with d valid inputs of $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$, exactly w of which evaluate to 1, we can see that the d-bit string given by the values of $\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}$ on these d inputs, is a valid input for $\mathrm{ASF}_{d,w}$. In other words, the set of valid inputs for $\mathrm{WSF}_{d,w,p,N}$, or equivalently, the set of valid inputs for the composed problem $\mathrm{ASF}_{d,w} \circ (\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)})^d$ is

$$\{(x^{(1)},\dots,x^{(d)}) \in \mathcal{P}^d : |\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}(x^{(1)})\dots\mathrm{HD}_{\frac{N}{2},N(\frac{1}{2}+p)}(x^{(d)})| = w\},$$

where $\mathcal{P} = \{x \in \{0,1\}^N : |x| \in \{N/2, N/2+pN\}\}$. The next theorem by Belovs and Lee shows that the quantum query complexity of the composed problem $\mathrm{ASF}_{d,w} \circ (\mathrm{HD}_{\frac{N}{2},\frac{N+2pN}{2}})^d$ is at least the product of the complexities of the two composing problems:

Theorem 6.8 ([BL20], Corollary 27)**.** *Let $f \subseteq S \times T$, with $S \subseteq \{0,1\}^d$, be a relational problem with bounded-error quantum query complexity L. Assume that f is efficiently verifiable, that is given some $t \in T$ and oracle access to $x \in S$, there exists a bounded-error quantum algorithm that verifies whether $(x,t) \in f$ using $o(L)$ queries to x. Let $D \subseteq \{0,1\}^N$ and $g : D \to \{0,1\}$ be a Boolean function whose bounded-error quantum query complexity is Q. Then the bounded-error quantum query complexity of the relational problem $f \circ g^d$, restricted to inputs $x \in \{0,1\}^{dN}$ such that $g^d(x) \in S$, is $\Omega(LQ)$.*

Applying Theorem 6.8 with the lower bounds of Theorem 6.7 and Theorem 6.6, we obtain:

Corollary 6.9. *Let $N \in 2\mathbb{Z}_+$ and $p \in (0, 0.5)$ be an integer multiple of $1/N$. Given a matrix $X \in \{-1,+1\}^{N\times d}$ such that there exists a set $W \subseteq [d]-1$ with size w and*

- *For every $j \in W$, $\sum_{i\in[N]-1} X_{ij} = 2pN$.*
- *For every $j' \in \overline{W}$, $\sum_{i\in[N]-1} X_{ij'} = 0$.*

Suppose we have query access to X. Then every bounded-error quantum algorithm that computes $\tilde{W}$ such that $|W\Delta\tilde{W}| \le w/200$, uses $\Omega(\sqrt{dw}/p)$ queries to O_X.

6.2.3 Worst-case to average-case reduction for the set-finding problem

Our goal is to prove a lower bound for Lasso algorithms that have high success probability w.r.t. the distribution $\mathcal{D}_{p,W}$, yet the lower bound of the previous subsection is for *worst-case* instances. In this subsection, we will connect these by providing a worst-case to average-case reduction for the set-finding problem. After that, by simply combining with the query lower bound for the worst-case set-finding problem and the reduction from the distributional set-finding problem to Lasso, we obtain an $\Omega(\sqrt{d}/\varepsilon^{1.5})$ query lower bound for Lasso.

Theorem 6.10. *Let $N \in 2\mathbb{Z}_+$, $p \in (0, 0.5)$ be an integer multiple of $1/N$, w be a natural number between 2 to $d/2$, and M be a natural number. Suppose $X \in \{-1,+1\}^{N\times d}$ is a valid input for $\mathrm{WSF}_{d,w,p,N}$, and let $W \subset [d]-1$ be the set of the w indices of the columns of X whose entries add up to $2pN$. Let R be an $M \times d$ matrix whose entries are i.i.d. samples from $\mathcal{U}_N$, and define $X' \in \{-1,1\}^{M\times d}$ as $X'_{ij} = X_{R_{ij}j}$. Then the M vectors $(X'_i, 1)$, where X'_i is the ith row of X' and $i \in [M]-1$, are i.i.d. samples from $\mathcal{D}_{p,W}$.*

Proof. Every entry of R is a sample from $\mathcal{U}_N$, so $X_{R_{ij}j}$ is uniformly chosen from the entries of the jth column of X. Moreover, because every valid input W for $\mathrm{WSF}_{d,w,p,N}$ satisfies that for every $j \in W$, $\Pr_{i\sim\mathcal{U}_N}[X_{ij} = 1] = 1/2 + p$ and for every $j' \in \overline{W}$, $\Pr_{i\sim\mathcal{U}_N}[X_{ij'} = 1] = 1/2$, we know $(X'_i, 1)$ is distributed as $D_{p,W}$. □

The above theorem tells us that we can convert an instance of $\mathrm{WSF}_{d,w,p,N}$ to an instance of $\mathrm{DSF}_{\mathcal{D}_{p,W}}$. Note that we can produce matrix R offline and therefore we can construct the oracle $O_{X'} : |i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|X_{R_{ij}j}\rangle$ using 1 query to $O_X : |i\rangle|j\rangle|0\rangle \to |i\rangle|j\rangle|X_{ij}\rangle$ (and some other elementary gates, which is irrelevant to the number of queries). Also observe that if $M = 10^{12} \cdot \lceil \log d \rceil \cdot \lfloor 1/\varepsilon \rfloor^2 = \mathcal{O}((\log d)/\varepsilon^2)$ and hence $S' = \{(X'_i, 1)\}_{i=0}^{M-1}$ is a sample set with M i.i.d. samples from $\mathcal{D}_{p,W}$, then by Theorem 3.3, with probability $\geq 9/10$, an $\varepsilon/16000$-minimizer for Lasso with respect to S' is also an $\varepsilon/8000$-minimizer for Lasso with respect to distribution $\mathcal{D}_{p,W}$. By Theorem 6.3, an $\varepsilon/8000$-minimizer for Lasso with respect to distribution $\mathcal{D}_{p,W}$ can be used to output a set $\tilde{W} \subset [d]-1$ such that $|\tilde{W} \Delta W| \leq w/200$, where W is the set of indices for those columns of X whose entries add up to $2pN$. Hence we have a reduction from the worst-case set-finding problem to Lasso. By the reduction above and by plugging $w = \lfloor 1/\varepsilon \rfloor$ and $p = 1/(2\lfloor 1/\varepsilon \rfloor)$ in Corollary 6.9 (and N an arbitrary natural number such that $pN \in \mathbb{N}$), we obtain a lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ queries for $\mathrm{WSF}_{d,w,p,N}$, and hence the main result of this section: a lower bound of $\Omega(\sqrt{d}/\varepsilon^{1.5})$ for Lasso.

Corollary 6.11. *Let $\varepsilon \in (2/d, 1/100)$, $w = \lfloor 1/\varepsilon \rfloor$, $p = 1/(2\lfloor 1/\varepsilon \rfloor)$, and $W \subset [d]-1$ with size w. Every bounded-error quantum algorithm that computes an ε-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ uses $\Omega(\sqrt{d}/\varepsilon^{1.5})$ queries.*

Classical lower bound. In Section 6.4 we show how this quantum lower bound approach can be modified to prove, for the first time, a lower bound of $\tilde{\Omega}(d/\varepsilon^2)$ on the classical query complexity of Lasso. This lower bound is optimal up to logarithmic factors.

6.3 Quantum query lower bound for Ridge

Now we switch our attention from Lasso to Ridge. We will prove a lower bound of $\Omega(d/\varepsilon)$ queries for Ridge in a very similar way as our lower bound for Lasso. Recall that Ridge's setup assumes the vectors in the sample set are normalized in ℓ_2 rather than ℓ_∞ as in Lasso. We modify the distribution to $\mathcal{D}'_{p,W}$ over $(x,y) \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^d \times \{-1,1\}$ as follows. Let $p \in (0,1/4)$, $W \subset [d]-1$, and $\overline{W} = [d]-1 \setminus W$. For each $j' \in \overline{W}$, $x_{j'}$ is generated according to $\Pr[x_{j'} = -1/\sqrt{d}] = 1/2+p$; for each $j \in W$, x_j is generated according to $\Pr[x_j = 1/\sqrt{d}] = 1/2+p$; y is generated according to $\Pr[y=1]=1$. Now again we want to solve a distributional set-finding problem with respect to $\mathcal{D}'_{p,W}$, given M samples from $\mathcal{D}'_{p,W}$. Similar to the Lasso case, one can think of the $M \times d$ matrix of samples as "hiding" the set W: the columns corresponding to $j \in W$ are likely to have more $1/\sqrt{d}$'s than $-1/\sqrt{d}$'s, while the columns corresponding to $j \in \overline{W}$ are likely to have more $-1/\sqrt{d}$'s than $1/\sqrt{d}$'s.

In this section let

$$\theta^* = \sum_{j \in [d]-1} \frac{e_j}{\sqrt{d}} (-1)^{[j \in \overline{W}]}$$

and note that for every $\theta \in \mathbb{R}^d$,

$$\begin{aligned}
L_{\mathcal{D}'_{p,W}}(\theta) &= \mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle^2] - 2\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle] + 1 \\
&= (\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle^2] - \mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle]^2) + \mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle]^2 - 2\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle] + 1 \\
&= \|\theta\|_2^2 \cdot (1-4p^2)/d + (\mathbb{E}_{(x,y)\sim\mathcal{D}'_{p,W}}[\langle\theta,x\rangle] - 1)^2 \\
&= \|\theta\|_2^2 \cdot (1-4p^2)/d + (2p\langle\theta,\theta^*\rangle - 1)^2,
\end{aligned}$$

where the third equality holds because $\langle\theta,x\rangle$ is a sum of independent random variables and hence its variance is the sum of the variances of the terms $\theta_i x_i$ (which are $\theta_i^2(1-4p^2)/d$).

Next we show that θ^* is the minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$.

Theorem 6.12. *Let $w = \lfloor d/2 \rfloor$ and $W \subset [d]-1$ be a set of size w, and let $\varepsilon \in (1000/d, 1/10000)$ and $p = 1/\lfloor 1/\varepsilon \rfloor$. Then $\theta^* = \sum_{j\in[d]-1} \frac{e_j}{\sqrt{d}}(-1)^{[j\in\overline{W}]}$ is the minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$.*

Proof. Let $\theta = \sum_{j\in[d]-1} \theta_j e_j \in B_2^d$ be a minimizer. We want to show $\theta_j = \theta_j^*$ for every $j \in [d]-1$. Note that if $\theta_j \cdot (-1)^{[j\in\overline{W}]} < 0$, then we can flip the sign of θ_j to get a smaller objective value, that is,

$$\begin{aligned}
L_{\mathcal{D}'_{p,W}}(\theta') - L_{\mathcal{D}'_{p,W}}(\theta) &= (\|\theta'\|_2^2 - \|\theta\|_2^2)\cdot(1-4p^2)/d + (2p\langle\theta',\theta^*\rangle - 1)^2 - (2p\langle\theta,\theta^*\rangle - 1)^2 \\
&= (2p\langle\theta'-\theta,\theta^*\rangle)(2p\langle\theta'+\theta,\theta^*\rangle - 2) \\
&= (-4p\theta_j \cdot (-1)^{[j\in\overline{W}]})(2p\langle\theta'+\theta,\theta^*\rangle - 2) < 0,
\end{aligned}$$

where $\theta' = \sum_{k\in[d]-1\setminus\{j\}} \theta_k e_k - \theta_j e_j$, and the last inequality is because $-4p\theta_j \cdot (-1)^{[j\in\overline{W}]} > 0$ and $2p\langle\theta'+\theta,\theta^*\rangle \le 2p\|\theta'+\theta\|_2 \cdot \|\theta^*\|_2 \le 4p \le 1$. Since θ was assumed a minimizer, for all $j \in [d]-1$ the sign of θ_j must be $(-1)^{[j\in\overline{W}]}$.

Second, we show that we must have $|\theta_0| = |\theta_1| = \cdots = |\theta_{d-1}|$. Suppose, towards a contradiction, that this is not the case. Consider $\theta' = \sum_{j\in[d]-1} ue_j \cdot (-1)^{[j\in\overline{W}]}$, where $u = \sqrt{\sum_{j\in[d]-1} |\theta_j|^2/d}$. We have

$$\begin{aligned} L_{\mathcal{D}'_{p,W}}(\theta') - L_{\mathcal{D}'_{p,W}}(\theta) &= (2p\langle\theta'-\theta,\theta^*\rangle)(2p\langle\theta'+\theta,\theta^*\rangle - 2) \\ &= (2p/\sqrt{d}) \cdot (du - \sum_{j\in[d]-1} |\theta_j|) \cdot (2p\langle\theta'+\theta,\theta^*\rangle - 2) < 0. \end{aligned}$$

The last inequality holds because again $2p\langle\theta'+\theta,\theta^*\rangle \le 4p \le 1$ and in addition,

$$d \cdot \sum_{j\in[d]-1} |\theta_j|^2 > (\sum_{j\in[d]-1} |\theta_j|)^2$$

by the Cauchy–Schwarz inequality (which is strict if the $|\theta_j|$ are not all equal). Hence if θ is indeed a minimizer, then its entries must all have the same magnitude.

Now we know a minimizer θ must be in the same direction as θ^*, we just don't know yet that the magnitudes of its entries are $1/\sqrt{d}$. Suppose $\|\theta\|_2 = u \le 1$ and $\theta = u \cdot \theta^*$, then we have

$$\begin{aligned} L_{\mathcal{D}'_{p,W}}(\theta) &= \|\theta\|_2^2 \cdot (1-4p^2)/d + (2p\langle\theta,\theta^*\rangle - 1)^2 \\ &= (u^2(1-4p^2)/d + (2pu-1)^2). \end{aligned}$$

The discriminant of $f(u) = u^2(1-4p^2)/d + (2pu-1)^2$ is less than 0, and $u = \frac{2p}{4p^2+(1-4p^2)/d}$ is the global minimizer of $f(u)$. Note that $u = \frac{2p}{4p^2+(1-4p^2)/d} > 1$, and hence $f(1) \le f(u)$ for every $u \le 1$. Therefore we know θ^* is the minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$. □

Next we show that the inner product between the minimizer and an approximate minimizer for Ridge will be close to 1.

Theorem 6.13. *Let $w = \lfloor d/2 \rfloor$, $W \subset [d]-1$ be a set of size w, $\varepsilon \in (1000/d, 1/10000)$, and $p = 1/\lfloor 1/\varepsilon \rfloor$. Suppose $\theta \in B_2^d$ is an $\varepsilon/1000$-minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$. Then $\langle\theta,\theta^*\rangle \ge 0.999$.*

Proof. Because θ is an $\varepsilon/1000$-minimizer, we have

$$\begin{aligned} &0.001\varepsilon \ge L_{\mathcal{D}'_{p,W}}(\theta) - L_{\mathcal{D}'_{p,W}}(\theta^*) = (1-4p^2)\cdot(\|\theta\|_2^2 - 1)/d + (2p\langle\theta,\theta^*\rangle - 1)^2 - (2p-1)^2 \\ &\implies 2p\langle\theta,\theta^*\rangle \ge 1 - \sqrt{1-4p+4p^2+0.001\varepsilon - (1-4p^2)\cdot(\|\theta\|_2^2-1)/d}. \end{aligned}$$

Letting $z = 4p - 4p^2 - 0.001\varepsilon + (1-4p^2)\cdot(\|\theta\|_2^2 - 1)/d$, we have

$$\begin{aligned} 2p\langle\theta,\theta^*\rangle \geq& 1 - \sqrt{1-z} \geq 1 - (1 - z/2) = z/2 \\ =& 2p - 2p^2 + (1-4p^2)\cdot(\|\theta\|_2^2 - 1)/d - 0.001\varepsilon, \end{aligned}$$

where the second inequality holds because $z \in (0,1)$. Dividing both sides by $2p$, we have

$$\langle\theta,\theta^*\rangle \geq 1 - p + (1-4p^2)\cdot(\|\theta\|_2^2 - 1)/(2pd) - 0.0005\varepsilon/p.$$

Because $\theta \in B_2^d$, $p = 1/\lfloor 1/\varepsilon \rfloor$, and $\varepsilon \in (1000/d, 1/10000)$, the above implies $\langle\theta,\theta^*\rangle \geq 0.999$. □

Combining the above theorem with the following theorem, we can see how to relate the entries of an approximate minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$ to the elements of the hidden set W.

Theorem 6.14. *Suppose $\theta \in B_2^d$ satisfies $\langle\theta,\theta^*\rangle \geq 1 - 0.001$. Then $\#\{j \in [d]-1 \mid \theta_j \cdot \theta_j^* \leq 0\} \leq d/500$.*

Proof. If $\theta_j \cdot \theta_j^* \leq 0$ then $|\theta_j - \theta_j^*| \geq |\theta_j^*| = \frac{1}{\sqrt{d}}$, hence using Theorem 6.13 we have

$$\frac{1}{d}\#\{j \in [d]-1 \mid \theta_j \cdot \theta_j^* \leq 0\} \leq \|\theta - \theta^*\|_2^2 = \|\theta\|_2^2 + \|\theta^*\|_2^2 - 2\langle\theta,\theta^*\rangle \leq 2 - 2(1-0.001) = 1/500.$$

□

We know $\theta^* = \sum_{j\in[d]-1} \frac{e_j}{\sqrt{d}}(-1)^{\lfloor j \in \overline{W} \rfloor}$, so by looking at the signs of entries of θ, we can find an index set $\tilde{W} = \{j \in [d]-1 : \theta_j > 0\}$ satisfying that $|W \Delta \tilde{W}| \leq d/500 \leq w/200$ because $w = \lfloor d/2 \rfloor$. Therefore, once we have an $\varepsilon/1000$-minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$, we can solve $\mathrm{DSF}_{\mathcal{D}'_{p,W}}$.

With the reduction from $\mathrm{DSF}_{\mathcal{D}'_{p,W}}$ to Ridge, we here show (similar to Lasso) a lower bound for the *worst-case symmetric set-finding problem* $\mathrm{WSSF}_{d,w,p,N}$: given a matrix $X \in \{\frac{-1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\}^{N\times d}$ where each column-sum is either $2pN/\sqrt{d}$ or $-2pN/\sqrt{d}$, the goal is to find a set $\tilde{W} \subset [d]-1$ such that $|\tilde{W}\Delta W| \leq w/200$, where W is the set of indices for those columns whose entries add up to $2pN/\sqrt{d}$ and $w = |W|$. This problem is again a composition of the approximate set finding problem in Section 6.2.2 and the Hamming-weight distinguisher problem $\mathrm{HD}_{\ell,\ell'}$ with $\ell = \frac{N}{2} - pN$ and $\ell' = \frac{N}{2} + pN$ up to a scalar $1/\sqrt{d}$. Following the proof of Theorem 6.7, we prove a lower bound of $\Omega(1/p)$ queries for this problem.

Theorem 6.15. *Let $N \in 2\mathbb{Z}_+$, $z \in \{0,1\}^N$, and $p \in (0, 0.5)$ be an integer multiple of $1/N$. Suppose we have query access to z. Then every bounded-error quantum algorithm that computes $\mathrm{HD}_{\frac{N}{2}-pN,\frac{N}{2}+pN}$ makes $\Omega(1/p)$ queries.*

Again we think of the input bits as ± 1 and abuse the notation $\mathrm{HD}_{\frac{N}{2}-pN,\frac{N}{2}+pN}$ for the problem with ± 1 input. Also, by the composition property of the adversary bound from Belovs and Lee [BL20] (Theorem 6.8), we have a lower bound of $\Omega(\sqrt{dw}/p)$ for $\mathrm{WSSF}_{d,w,p,N}$ from the $\Omega(\sqrt{dw})$ lower bound for $\mathrm{ASF}_{d,w}$ and the $\Omega(1/p)$ lower bound for $\mathrm{HD}_{\frac{N}{2}-pN,\frac{N}{2}+pN}$.

Corollary 6.16. *Let $N \in 2\mathbb{Z}_+$ and $p \in (0, 0.5)$ be an integer multiple of $1/N$. Given a matrix $X \in \{-1/\sqrt{d}, +1/\sqrt{d}\}^{N\times d}$ such that there exists a set $W \subseteq [d]-1$ with size w and*

- *For every $j \in W$, $\sum_{i\in[N]-1} X_{ij} = 2pN/\sqrt{d}$.*
- *For every $j' \in \overline{W}$, $\sum_{i\in[N]-1} X_{ij'} = -2pN/\sqrt{d}$.*

Then every bounded-error quantum algorithm that computes $\tilde{W}$ such that $|W \Delta \tilde{W}| \leq w/200$, takes $\Omega(\sqrt{dw}/p)$ queries.

The final step for proving a lower bound for Ridge, using the same arguments as in Section 6.2.3, is to provide a worst-case to average-case reduction for the symmetric set-finding problem. We follow the same proof in Theorem 6.10 and immediately get the following theorem:

Theorem 6.17. *Let $N \in 2\mathbb{Z}_+$, $p \in (0, 0.5)$ be an integer multiple of $1/N$, w be a natural number between 2 to $d/2$, and M be a natural number. Suppose $X \in \{-1/\sqrt{d}, +1/\sqrt{d}\}^{N\times d}$ is a valid input for $WSSF_{d,w,p,N}$, and let $W \subset [d]-1$ be the set of the w indices of the columns of X whose entries add up to $2pN/\sqrt{d}$. Let R be an $M \times d$ matrix whose entries are i.i.d. samples from $\mathcal{U}_N$, and define $X' \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^{M\times d}$ as $X'_{ij} = X_{R_{ij}j}$. Then the vectors $(X'_i, 1)$, where X'_i is the ith row of X' and $i \in [M]-1$, are i.i.d. samples from $\mathcal{D}'_{p,W}$.*

By setting $M = 10^{10} \cdot \lceil \log d \rceil \cdot \lfloor 1/\varepsilon \rfloor^2 = \mathcal{O}((\log d)/\varepsilon^2)$ and letting $S' = \{(X'_i, 1)\}_{i=0}^{M-1}$ be a sample set with M i.i.d. samples from $\mathcal{D}'_{p,W}$, with probability $\geq 9/10$, an $\varepsilon/2000$-minimizer for Ridge with respect to S' is also an $\varepsilon/1000$-minimizer for Ridge with respect to distribution $\mathcal{D}'_{p,W}$ from Theorem 3.4. By Theorem 6.14 and Theorem 6.13, an $\varepsilon/1000$-minimizer for Ridge with respect to distribution $\mathcal{D}'_{p,W}$ gives us a set $\tilde{W} \subset [d]-1$ such that $|\tilde{W}\Delta W| \leq w/200$, where W is the set of indices for those columns of X whose entries add up to $2pN/\sqrt{d}$. Hence we have a reduction from the worst-case symmetric set-finding problem to Ridge. By this reduction and by plugging $w = \lfloor d/2 \rfloor$ and $p = 1/\lfloor 1/\varepsilon \rfloor$ in Corollary 6.16 (and N an arbitrary natural number such that $pN \in \mathbb{N}$), we obtain a lower bound of $\Omega(d/\varepsilon)$ queries for $\mathrm{WSSF}_{d,w,p,N}$, and hence for Ridge as well, which is the main result of this section.

Corollary 6.18. *Let $\varepsilon \in (2/d, 1/1000)$, $w = \lfloor d/2 \rfloor$, $p = 1/\lfloor 1/\varepsilon \rfloor$, and $W \subset [d]-1$ with size w. Every bounded-error quantum algorithm that computes an ε-minimizer for Ridge with respect to $\mathcal{D}'_{p,W}$ uses $\Omega(d/\varepsilon)$ queries.*

6.4 Classical lower bound for Lasso

In this section, we will give a classical lower bound for Lasso. Here we first introduce some tools we will use. The first one is the *hypergeometric distribution* $\mathrm{Hyp}(N,L,m)$ with parameters N, L, and m: the distribution of the number of marked balls drawn when m balls are drawn without replacement from a set of n balls, L of which are marked and $N-L$ are unmarked. On the other hand, the *binomial distribution* $\mathrm{Bin}(m,L/N)$ is the distribution on the number of marked balls drawn with replacement. Holmes showed that when m is small enough, the total variation distance between those two distribution will be very small.

Theorem 6.19 ([Hol03], Theorem 3.1)**.** *Let $N,L,m\in\mathbb{N}$. If $N\geq L\geq m$, then*

$$d_{TV}(Hyp(N,L,m),Bin(m,L/N))\leq (m-1)/(N-1)$$

We also use another distance between probability distributions.

Definition 6.20. Given two discrete probability distributions $\mathcal{P}$, $\mathcal{Q}$ over $[N]-1$, the Hellinger distance d_H between $\mathcal{P}$ and $\mathcal{Q}$ is defined as

$$d_H(\mathcal{P},\mathcal{Q}):=\sqrt{\frac{1}{2}\sum_{i\in[N]-1}(\sqrt{\mathcal{P}_i}-\sqrt{\mathcal{Q}_i})^2}=\sqrt{1-\sum_{i\in[N]-1}\sqrt{\mathcal{P}_i\mathcal{Q}_i}}.$$

From the definition above we also have the following property for product distributions:

$$d_H^2(\mathcal{P}^{\otimes m},\mathcal{Q}^{\otimes m})=1-(\sum_{i\in[N]-1}\sqrt{\mathcal{P}_i\mathcal{Q}_i})^m=1-(1-d_H^2(\mathcal{P},\mathcal{Q}))^m\leq m\cdot d_H^2(\mathcal{P},\mathcal{Q}). \quad (6.1)$$

The following lemma bridges the Hellinger distance and total variation distance.

Lemma 6.21. *For arbitrary discrete probability distributions $\mathcal{P}$, $\mathcal{Q}$ over $[N]-1$, we have*

$$d_H^2(\mathcal{P},\mathcal{Q})\leq d_{TV}(\mathcal{P},\mathcal{Q})\leq\sqrt{2}\,d_H(\mathcal{P},\mathcal{Q}).$$

Proof. First we prove the left inequality:

$$\begin{aligned}d_H^2(\mathcal{P},\mathcal{Q})&=\frac{1}{2}\sum_{i\in[N]-1}(\sqrt{\mathcal{P}_i}-\sqrt{\mathcal{Q}_i})^2\leq\frac{1}{2}\sum_{i\in[N]-1}(|\sqrt{\mathcal{P}_i}-\sqrt{\mathcal{Q}_i}|)(\sqrt{\mathcal{P}_i}+\sqrt{\mathcal{Q}_i})\\&=\frac{1}{2}\sum_{i\in[N]-1}|\mathcal{P}_i-\mathcal{Q}_i|=d_{TV}(\mathcal{P},\mathcal{Q}).\end{aligned}$$

The right inequality follows using Cauchy-Schwarz:

$$\begin{aligned}d_{TV}(\mathcal{P},\mathcal{Q})&=\frac{1}{2}\sum_{i\in[N]-1}|\sqrt{\mathcal{P}_i}-\sqrt{\mathcal{Q}_i}|(\sqrt{\mathcal{P}_i}+\sqrt{\mathcal{Q}_i})\\&\leq\frac{1}{2}\sqrt{\sum_{i\in[N]-1}|\sqrt{\mathcal{P}_i}-\sqrt{\mathcal{Q}_i}|^2}\sqrt{\sum_{i\in[N]-1}|\sqrt{\mathcal{P}_i}+\sqrt{\mathcal{Q}_i}|^2}\\&=d_H(\mathcal{P},\mathcal{Q})\cdot\sqrt{1+\sum_{i\in[N]-1}\sqrt{\mathcal{P}_i\mathcal{Q}_i}}=d_H(\mathcal{P},\mathcal{Q})\cdot\sqrt{2-d_H^2(\mathcal{P},\mathcal{Q})}\leq\sqrt{2}\,d_H(\mathcal{P},\mathcal{Q}).\end{aligned}$$

□

After introducing the above tools, we are ready to prove the following result.

Theorem 6.22. *Let $m \in \mathbb{Z}_+$, $p \in (0, 0.5)$, and $\mathcal{P}, \mathcal{Q}$ be two hypergeometric distributions $Hyp(N, N/2, m)$ and $Hyp(N, N/2+pN, m)$ respectively. Then we have $d_{TV}(\mathcal{P}, \mathcal{Q}) \leq 2(m-1)/(N-1) + p\sqrt{3m}$.*

Proof. Let $\mathcal{P}', \mathcal{Q}'$ be the binomial distributions $\mathrm{Bin}(m, 1/2)$ and $\mathrm{Bin}(m, 1/2+p)$ respectively. By the triangle inequality, Theorem 6.19, and Lemma 6.21, we have

$$\begin{aligned} d_{TV}(\mathcal{P}, \mathcal{Q}) &\leq d_{TV}(\mathcal{P}', \mathcal{P}) + d_{TV}(\mathcal{P}', \mathcal{Q}') + d_{TV}(\mathcal{Q}', \mathcal{Q}) \\ &\leq (m-1)/(N-1) + d_{TV}(\mathcal{P}', \mathcal{Q}') + (m-1)/(N-1) \\ &\leq 2(m-1)/(N-1) + \sqrt{2}\, d_H(\mathcal{P}', \mathcal{Q}'). \end{aligned}$$

Suppose $\mathbf{p}'$ and $\mathbf{q}'$ are Bernoulli distributions with mean $1/2$ and $1/2+p$ respectively, then using Eq. (6.1) we have

$$d_H(\mathcal{P}', \mathcal{Q}') \leq d_H(\mathbf{p}'^{\otimes m}, \mathbf{q}'^{\otimes m}) \leq \sqrt{m} \cdot d_H(\mathbf{p}', \mathbf{q}') \leq \sqrt{m} \cdot \sqrt{1 - \frac{1}{2}(\sqrt{1+2p} + \sqrt{1-2p})} \leq p\sqrt{\frac{3m}{2}},$$

where the last inequality holds because $\sqrt{1+2p} + \sqrt{1-2p} \geq 2 - 3p^2$ for $p \in (0, 0.5)$. Combining the above two results, we have

$$d_{TV}(\mathcal{P}, \mathcal{Q}) \leq 2(m-1)/(N-1) + \sqrt{2} d_H(\mathcal{P}', \mathcal{Q}') \leq 2(m-1)/(N-1) + p\sqrt{3m}.$$

□

Using the above theorem, we can show a classical query lower bound for the *exact set-finding problem* $\mathrm{ESF}_{d,w,p,N}$, which is the following: given a matrix $X \in \{-1, 1\}^{N \times d}$ where each column-sum is either $2pN$ or 0, the goal is to find the set W (of size w or $w-1$) of indices of the columns whose entries add up to $2pN$. The following theorem gives a classical query lower bound for this problem.

Theorem 6.23. *Let $N \in 2\mathbb{Z}_+$, $p \in (1/\sqrt{N}, 0.5)$ be an integer multiple of $1/N$, and $w = 1/(2p)$. Given a matrix $X \in \{-1, 1\}^{N \times d}$ such that there exists a set $W \subset [d] - 1$ with size w or $w-1$ satisfying*

- *for every $j \in W$, $\sum_{i \in [N]-1} X_{ij} = 2pN$;*
- *for every $j' \in \overline{W}$, $\sum_{i \in [N]-1} X_{ij'} = 0$.*

Suppose we have classical query access to X. Then every classical algorithm that computes W with success probability $\geq 1 - 1/100$ uses $\Omega((d-w)/p^2)$ queries.

Proof. Let $W = \{1, \ldots, w-1\}$ and let $\mathcal{D}_W$ be the distribution on the input X where each column of X is chosen uniformly at random subject to the column sums as specified in the theorem: if $j \in W$ then the jth column-sum is $2pN$, and if $j \notin W$ then the jth column-sum is 0.

Let $\mathcal{A}$ be a randomized classical algorithm with worst-case query complexity T that computes the hidden set with error probability $\leq 1/100$ for every input. Let random variable t_j be the number of queries that algorithm $\mathcal{A}$ makes in the jth N-bit string (i.e., to entries in the jth column of X) under $\mathcal{D}_W$, and T_j be the expectation of t_j under $\mathcal{D}_W$. Because $\sum_{j\in[d]-1} T_j \leq T$, there must be a $k \in [d]-1 \setminus W$ such that $T_k \leq T/(d-w+1)$. We use algorithm $\mathcal{A}$ to prove the following claim:

Claim 6.1. There exists a classical randomized algorithm with worst-case query complexity $\leq 100T/(d-w+1) := \tau$ in the k-th column that distinguishes $\mathcal{D}_W$ from $\mathcal{D}_{W\cup\{k\}}$ with success probability ≥ 0.98.

Proof: Let $\mathcal{B}$ be the following randomized classical algorithm: run $\mathcal{A}$ until the number of queries in k-th column is $\geq \tau$. If $\mathcal{A}$ outputs W, then we output that; if $\mathcal{A}$ does not output W or did not terminate within τ queries, then we output $W\cup\{k\}$.

We here prove the correctness of algorithm $\mathcal{B}$. If we run $\mathcal{B}$ on input distribution $\mathcal{D}_W$, then the probability that $\mathcal{A}$ (run all the way until it terminates) does not output W, is $\leq 1/100$. By Markov's inequality, the probability (still under distribution $\mathcal{D}_W$) that $\mathcal{A}$ did not terminate within τ queries, $\leq 1/100$. Hence by the union bound, the probability that $\mathcal{B}$ does not output W is $\leq 2/100$.

If, on the other hand, we run $\mathcal{B}$ on input distribution is $\mathcal{D}_{W\cup\{k\}}$, then the probability that $\mathcal{B}$ outputs the correct set $W\cup\{k\}$ is lower bounded by the probability that $\mathcal{A}$ outputs $W\cup\{k\}$, because we defined $\mathcal{B}$ to output $W\cup\{k\}$ when $\mathcal{A}$ did not already terminate. Since $\mathcal{A}$ has success probability $\geq 99/100$, the probability (under $\mathcal{D}_{W\cup\{k\}}$) that $\mathcal{B}$ outputs $W\cup\{k\}$ is $\geq 99/100$. ■

Because algorithm $\mathcal{B}$ decides with success probability $\geq 98/100$ whether k is in the hidden set or not, it has to distinguish (in the kth column of X) a uniformly random column with column-sum 0 from a uniformly random column with column-sum $2pN$ with success probability $\geq 98/100$. Hence we must have

$$\Omega(1) \leq \frac{2(100T_k-1)}{N-1} + p\sqrt{300T_k} \leq \frac{200T}{(d-w+1)(N-1)} + p\sqrt{\frac{300T}{(d-w+1)}},$$

where the first inequality follows by Theorem 6.22 and the last one follows because $T_k \leq T/(d-w+1)$. Rearranging implies $T = \Omega((d-w)/p^2)$. □

The last step towards our lower bound for classical Lasso-solvers is to show that one can solve the exact set-finding problem using an approximate Lasso-solver, as follows.

Theorem 6.24. *Let $N \in 2\mathbb{Z}_+$, $p \in (1/\sqrt{N}, 0.25)$ be an integer multiple of $1/N$, and $w = 1/(2p)$. Suppose $\mathcal{A}$ is an algorithm that finds a $p/4000$-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ with probability $\geq 1-1/(20000\log d)$ for every $W \subset [d]-1$ with size w or $w-1$. Then Algorithm 7 outputs the correct answer for $ESF_{d,w,p,N}$ with success probability $\geq 99/100$.*

input : Algorithm $\mathcal{A}$ that outputs (with probability $\geq 1-1/(20000\log d)$) a $p/4000$-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ for every possible $W \subset [d]-1$ of size w or $w-1$, using M samples; $X \in \{-1,1\}^{N\times d}$ a valid input for $\mathrm{ESF}_{d,w,p,N}$;
for $u = 0$ *to* $U-1 = \lceil 100\log d\rceil - 1$ **do**
  GENERATE a permutation $\pi \in S_d$ uniformly at random;
  GENERATE an $M \times d$ matrix R such that all its entries are i.i.d. samples from $\mathcal{U}_N$;
  Let $X' \in \{-1,1\}^{M\times d}$ be $X'_{ij} = X_{R_{i\pi(j)}\pi(j)}$; let X'_m denote its mth row;
  RUN Algorithm $\mathcal{A}$ with inputs $(X'_1,1),(X'_2,1),\ldots,(X'_M,1)$ and let $W'_u \subseteq [d]-1$ be the set of indices of entries of the output of Algorithm $\mathcal{A}$ whose absolute value is $\geq 2p/3$;
  STORE $W_u = \pi^{-1}(W'_u)$;
end
output: $\tilde{W} = \{j \in [d]-1 : j$ is included in at least half of the sets $W_0,\ldots,W_{U-1}\}$;

Algorithm 7: Solve $\mathrm{ESF}_{d,w,p,N}$ using a $p/4000$-Lasso solver $\mathcal{A}$

Proof. Let $u \in [U]-1$ and define $p_j = \Pr[j \text{ is in } W_u]$. Note that p_j is independent of u because all iterations do the same thing. First we show that $\forall u \in [U]-1$ and $\forall j_1, j_2 \in W$, $p_{j_1} = p_{j_2}$. Let $p_{j,\pi} = \Pr[j \in \pi^{-1}(W'_u) \mid \pi]$ be the probability of the event that index j is in $\pi^{-1}(W'_u) = W_u$ if the uth iteration used the permutation π. We can see that $p_j = \frac{1}{d!}\sum_{\pi\in S_d} p_{j,\pi}$. Consider a permutation $\sigma \in S_d$ satisfying that

$$\sigma(j) = \begin{cases} j_2, & \text{if } j = j_1, \\ j_1, & \text{if } j = j_2, \\ j, & \text{otherwise.} \end{cases}$$

We have for every $\pi \in S_d$, that $p_{j_1,\pi} = p_{j_2,\sigma\pi}$ because both j_1 and j_2 are in W and each of them is drawn from the corresponding columns with replacement. Therefore, we have

$$p_{j_1} = \frac{1}{d!}\sum_{\pi\in S_d} p_{j_1,\pi} = \frac{1}{d!}\sum_{\pi\in S_d} p_{j_2,\sigma\pi} = p_{j_2},$$

and using a similar argument, we can also show that for arbitrary $j'_1, j'_2 \in \overline{W}$, we have $p_{j'_1} = p_{j'_2}$.

Combining the above argument and Theorem 6.3, we have that if algorithm $\mathcal{A}$ succeeds, then for every $j \in W$, $p_j \geq 0.995$ and for every $j' \in \overline{W}$, $p_{j'} \leq 0.005$. This implies that for every $j \in W$, $\Pr[j \notin \tilde{W}] \leq 1/200d$ and for every $j' \in [d]-1 \setminus W$, $\Pr[j' \in \tilde{W}] \leq 1/200d$ by Hoeffding bound. Also because algorithm $\mathcal{A}$ outputs a $p/4000$-minimizer with error probability at most $1/(20000\log d)$, by union bound the probability that $\mathcal{A}$ fails in at least one of the U inner loops is at most $1/200$. Hence $\tilde{W}$ is the correct answer for $\mathrm{ESF}_{d,w,p,N}$ with probability $\geq 1-1/200-d/200d = 99/100$. □

Similar to the arguments in Section 6.2.3, the above theorem tells us that we can convert an instance of $\mathrm{ESF}_{d,w,p,N}$ to an instance of $\mathrm{DSF}_{\mathcal{D}_{p,W}}$ (and hence an instance for Lasso). Again the matrix R is produced offline and therefore we do not use extra queries in Algorithm 7 apart from the runs of $\mathcal{A}$. Now suppose we have a T-query classical algorithm that outputs (with success probability $\geq 1-1/(20000\log d)$) a $p/4000$-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ for arbitrary $W \subset [d]-1$ of size w or $w-1$, then using $T\cdot 100\log d$ queries, we can solve $\mathrm{ESF}_{d,w,p,N}$ with probability $\geq 99/100$. Note that we can construct such a high-success-probability Lasso solver by applying a Lasso solver with success probability $\geq 2/3$ for $\mathcal{O}(\log\log d)$ times and then outputting the output vector with the smallest objective value (estimating objective values with additive error p with success probability $\geq 1-\delta$ uses only $\tilde{\mathcal{O}}(\log(1/\delta)/p^2)$ queries). Also, Theorem 6.23 gives a $\Omega((d-w)/p^2)$ lower bound for $\mathrm{ESF}_{d,w,p,N}$, and hence we obtain a classical lower bound of $\tilde{\Omega}(d/\varepsilon^2)$ queries for Lasso for $\varepsilon \in (1/\sqrt{d}, 1/200)$ by letting $p=\varepsilon/2$ and by the fact $\lfloor 1/\varepsilon\rfloor \geq w \geq \lfloor 1/\varepsilon\rfloor - 1$:

Corollary 6.25. *Let $\varepsilon \in (1/\sqrt{d}, 1/200)$, w be either $\lfloor 1/\varepsilon\rfloor$ or $\lfloor 1/\varepsilon\rfloor - 1$, $p = 1/(2\lfloor 1/\varepsilon\rfloor)$, and $W \subset [d]-1$ with size w. Every bounded-error classical algorithm that computes an ε-minimizer for Lasso with respect to $\mathcal{D}_{p,W}$ uses $\tilde{\Omega}(d/\varepsilon^2)$ queries.*

6.5 Open problems

We mention a few directions for future work:

- We show both Lasso and Ridge solvers can solve distributional set-finding (with respect to different distribution), but is it possible that we can use a distributional set-finding solver for solving Lasso or Ridge?
- We conjecture the lower bound for Lasso is not tight for the ε-dependency. As for the lower bound for Ridge, $\tilde{\Theta}(d/\varepsilon)$ might be the right bound.
- On the other hand, is it possible to reduce a harder problem to Lasso/Ridge? We tried to encode a bigger subset into the distributional set-finding problem for a better quantum query lower bound for Lasso, but because ℓ_1-regularizer ensures a $\mathcal{O}(1/\varepsilon)$ sparsity for an ε-minimizer, a Lasso ε-minimizer cannot approximate a bigger subset in this case.

CHAPTER 7

Query lower bounds for approximating the top eigenvector

7.1 Introduction

In this chapter we prove essentially tight classical and quantum query lower bounds for approximating the top eigenvector of a matrix whose entries we can query.

7.1.1 Main results and high-level intuition

We show an $\tilde{\Omega}(d^{1.5})$ quantum query lower bound for finding the top eigenvector, implying that Algorithm 6 (Corollary 4.16 with $s = d$ and $q = 1$) in Chapter 4 is essentially optimal. We do this by analyzing a hard instance $A = \frac{1}{d}uu^T + N$, which hides a vector $u \in \{-1, 1\}^d$ using a $d \times d$ matrix N with i.i.d. Gaussian entries of mean 0 and standard deviation $\sim 1/\sqrt{d}$. Note that the entry $A_{ij} = u_i u_j / d$ has magnitude $1/d$, but is "hidden" in the entry A_{ij} by adding noise to it of much larger magnitude $\sim 1/\sqrt{d}$. One can show that (with high probability) this matrix has a constant eigenvalue gap, and its top eigenvector is close to $u/\sqrt{d}$. A lower bound for recovering u thus implies a lower bound for recovering the top eigenvector.

First, this hard instance provides the above-mentioned unsurprising[1] $\Omega(d^2)$ query lower bound for *classical* algorithms, as follows. To approximate the top eigenvector (and hence u), an algorithm has to recover most of the d signs u_i of u. Note that the entries of the ith row and column of A are the only entries that depend on u_i. If the algorithm makes T queries overall, then there is an index $\mathbf{i}$ such that the algorithm makes at most $4T/d$ queries to entries in the $\mathbf{i}$th row and column, while still recovering

[1] A simpler way to see this lower bound is to consider the problem of distinguishing the all-0 matrix from a matrix that has a 1 in one of the d^2 positions and 0s elsewhere. This is just the d^2-bit OR problem, for which we have an easy and well-known $\Omega(d^2)$ classical query bound. However, the quantum analogue of this approach only gives an $\Omega(d)$ lower bound, since the quantum query complexity of d^2-bit OR is $\Theta(d)$. Therefore we present a more complicated argument for the classical lower bound whose quantum analogue *does* provide an essentially tight bound of $\tilde{\Omega}(d^{1.5})$.

$u_{\mathbf{i}}$ with good probability. Slightly simplifying, one can think of each of those entries as a sample from either the distribution $N(1/d, 1/\sqrt{d})$ or the distribution $N(-1/d, 1/\sqrt{d})$, with the sign of the mean corresponding to $u_{\mathbf{i}}$. It is well known that $\Omega(d)$ classical samples are necessary to estimate the mean of the distribution to within $\pm 1/d$ and hence to distinguish between these two distributions. This implies $4T/d = \Omega(d)$, giving us the $T = \Omega(d^2)$ classical query lower bound for approximating the top eigenvector of A (Corollary 7.2).

Second, a similar but more technical argument works to obtain the $\tilde{\Omega}(d^{1.5})$ quantum query lower bound (Corollary 7.4), as follows. A good algorithm recovers most u_i-s with good probability. If it makes T quantum queries overall, then there is an index $\mathbf{i}$ such that the algorithm has at most $4T/d$ "query mass" on the entries of the $\mathbf{i}$th row and column (in expectation over the distribution of A), while still recovering $u_{\mathbf{i}}$ with good probability. It then remains to show that distinguishing between either the distribution $N(1/d, 1/\sqrt{d})$ or the distribution $N(-1/d, 1/\sqrt{d})$, with the ability to query multiple samples from that distribution in quantum superposition, requires $\tilde{\Omega}(\sqrt{d})$ quantum queries. This we prove by a rather technical modification of the adversary bound of Ambainis [Amb02; Amb06], using expectations under a joint distribution μ on pairs of matrices (the two marginal distributions of μ are our hard instance conditioned on $u_{\mathbf{i}} = 1$ and $u_{\mathbf{i}} = -1$, respectively) of Hamming distance roughly $\sqrt{d}$ in the $\mathbf{i}$th row and column.[2]

Roadmap

In Section 7.2 we prove that an approximate top-eigenvector (with additive ℓ_2-norm error at most $1/1000$) can recover $\Omega(d)$ entries of vector u. In Section 7.3 and 7.4, we show that we need to make $\Omega(d^2)$ queries ($\tilde{\Omega}(d^{1.5})$ quantum queries) to entries of the matrix to recover $\Omega(d)$ entries of vector u, and hence $\Omega(d^2)$ ($\tilde{\Omega}(d^{1.5})$ quantum) query lower bound for approximating the top-eigenvector with additive ℓ_2-error at most $1/1000$.

7.2 The hard instance for the lower bound

Consider the following case, which is the "hard instance" for which we prove the lower bounds. Let $u \in \{-1, 1\}^d$ be a vector, and define symmetric random matrix $A = \frac{1}{d} uu^T + N$ where the entries of N are i.i.d. $N_{ij} \sim N(0, \frac{1}{4 \cdot 10^6 d})$ for all $1 \le i \le j \le d$ (and $N_{ij} = N_{ji}$ if $i > j$); the goal is to recover most (say, 99%) of the entries of the vector u. In this problem, the information about the u_i-s is hidden in the matrix A: the entry A_{ij} is clearly a sample from $N(\frac{u_i u_j}{d}, \frac{1}{4 \cdot 10^6 d})$. Hence to learn entries of u, intuitively we should be able to distinguish the distribution $N(\frac{1}{d}, \frac{1}{4 \cdot 10^6 d})$ from the distribution $N(-\frac{1}{d}, \frac{1}{4 \cdot 10^6 d})$. In the classical case (where querying an entry of A_{ij} is the same as obtaining one sample fom the distribution) it requires roughly $\Omega(d)$ queries to the entries of the ith row and col-

[2] We cannot just use the adversary bound "off the shelf", because our inputs are vectors of samples from a continuous distribution rather than fixed binary strings.

umn to learn one u_i, even if all u_j with $j \neq i$ are already known. In the quantum case, it requires $\Omega(\sqrt{d})$ queries. Intuitively, learning $0.99d$ of the u_i-s should then require roughly d times more queries, so $\Omega(d^2)$ and $\Omega(d^{1.5})$ classical and quantum queries in total, respectively. We show in the next two subsections that this is indeed the case.

First we show that (with high probability) A has a large eigenvalue gap. Note that $A = \frac{1}{d}uu^T + \frac{1}{2000\sqrt{d}}G$ where $G_{ij} \sim N(0,1)$ for $1 \leq i \leq j \leq d$ and $G_{ij} = G_{ji}$ for the lower-triangular elements. Since G itself is symmetric and its entries are i.i.d. $\sim N(0,1)$, by Theorem 2.33 with $t = 0.4\sqrt{d}$ (here $b_{max} = \sqrt{d}$ and $b^*_{max} = 1$) we have that the operator norm of G is upper bounded by $2.5\sqrt{d} + (0.4 + o(1))\sqrt{d} \leq 3\sqrt{d}$ with probability at least $1 - \exp(-0.04d)$; below we assume this is indeed the case. Therefore, by triangle inequality, the top eigenvalue of A is upper bounded by $1 + \frac{3}{2000}$ and lower bounded by $\|A\frac{u}{\sqrt{d}}\|_2 \geq 1 - \frac{3}{2000}$, implying that there is a unit top eigenvector $v_1 = v_1(A)$ of A that has inner product nearly 1 with the unit vector $\frac{u}{\sqrt{d}}$:

$$1 - \frac{3}{2000} \leq \|Av_1\|_2 \leq \|\frac{1}{d}uu^T v_1\|_2 + \|\frac{1}{2000\sqrt{d}}Gv_1\|_2 \leq |\frac{1}{\sqrt{d}}u^T v_1| + \frac{3}{2000},$$

hence $|\langle v_1, \frac{u}{\sqrt{d}}\rangle| \geq 1 - \frac{3}{1000}$. We may assume without loss of generality that the eigenvector v_1 has been chosen such that $\langle v_1, \frac{u}{\sqrt{d}}\rangle$ is positive, so we can ignore the absolute value sign. The signs of $v_1(A)$ have to agree with the signs of u in at least 99.4% of the d entries, because each entry where the signs are different contributes at least $1/d$ to the squared distance between $u/\sqrt{d}$ and $v_1(A)$:

$$\frac{1}{d}\#\{j \in [d] \mid u_j \cdot (v_1(A))_j \leq 0\} \leq \|\frac{u}{\sqrt{d}} - v_1(A)\|_2^2 = 2 - 2\langle\frac{u}{\sqrt{d}}, v_1(A)\rangle \leq \frac{3}{500}.$$

Moreover, the second eigenvalue $\lambda_2(A)$ of A is at most 0.08:

$$\lambda_2(A) = \max_{w:\|w\|_2=1, w\perp v_1} \|Aw\|_2 \leq \frac{3}{2000} + \max_{w:\|w\|_2=1, w\perp v_1} \|\frac{uu^T w}{d}\|_2 \leq \frac{3}{2000} + \frac{\sqrt{3\cdot 1997}}{997} < 0.08,$$

where the last inequality holds because $\langle v_1, \frac{u}{\sqrt{d}}\rangle \geq 1 - \frac{3}{1000}$.[3] Hence there is a constant gap between the top and the second eigenvalue of A.

If we have an algorithm that outputs a vector $\tilde{u}$ satisfying $\|\tilde{u} - v_1(A)\|_2 \leq \frac{1}{1000}$, then we can use the signs of $\tilde{u}$ to learn 99% of the u_i-s. Hence a (classical or quantum) query lower bound for recovering (most of) u is also a query lower bound for approximating the top eigenvector of our hard instance.

7.3 A classical lower bound

We first show that every classical algorithm that recovers 99% of the u_i-s needs $\Omega(d^2)$ queries.

[3] Let $v_1 = \alpha_1\frac{u}{\sqrt{d}} + \beta_1 w_1$ and $v_2 = \alpha_2\frac{u}{\sqrt{d}} + \beta_2 w_2$ for some unit vectors $w_1, w_2 \perp \frac{u}{\sqrt{d}}$ and for some $\alpha_1, \alpha_2, \beta_1, \beta_2 \in [-1,1]$ satisfying $\alpha_1^2 + \beta_1^2 = \alpha_2^2 + \beta_2^2 = 1$. Since $v_1 \perp v_2$, we have $\langle v_1, v_2\rangle = \alpha_1\alpha_2 + \beta_1\beta_2\langle w_1, w_2\rangle = 0$, implying $|\alpha_2| = \frac{|\beta_1\beta_2|}{|\alpha_1|}|\langle w_1, w_2\rangle| \leq \frac{|\beta_1|}{|\alpha_1|} = \frac{\sqrt{1-\alpha_1^2}}{|\alpha_1|} \leq \frac{\sqrt{3\cdot 1997}}{997}$.

Theorem 7.1. *Let $u \in \{-1,1\}^d$ be a vector, $e_{11}, e_{12}, \ldots, e_{1d}, e_{22}, \ldots, e_{dd}$ be $d(d+1)/2$ independent samples drawn from $N(0, \frac{1}{4\cdot 10^6 d})$, and $A \in \mathbb{R}^{d\times d}$ be the matrix defined by*

$$A_{ij} = \begin{cases} \frac{1}{d}u_i u_j + e_{ij}, & \text{if } 1 \le i \le j \le d, \\ A_{ji}, & \text{otherwise.} \end{cases}$$

Suppose we have query access to entries of A. Every bounded-error classical algorithm that computes a $\tilde{u} \in \{-1,1\}^d$ at Hamming distance $\le d/100$ from u, uses $\Omega(d^2)$ queries.

Proof. Suppose there exists a T-query bounded-error classical algorithm $\mathcal{A}$ to compute such a $\tilde{u}$, with worst-case error probability $\le 1/20$. Note that the only entries that depend on u_i are in the ith column and row of A. Let random variable T_i be the number of queries that $\mathcal{A}$ makes in the ith column and row (here the randomness comes from the input distribution and from the internal randomness of $\mathcal{A}$). Because every query is counted at most twice among the T_i-s (and only once if the query is to a diagonal entry of A), we have $\mathbb{E}[\sum_{i\in[d]} T_i] \le 2T$. Define index i as "good" if $\Pr[u_i = \tilde{u}_i] \ge 0.8$, and let I_G be the set of good indices. Since $\mathcal{A}$ has error probability at most $1/20$, we can bound the expected Hamming distance by

$$\mathbb{E}[\mathrm{Ham}(u,\tilde{u})] \le \Pr[\mathrm{Ham}(u,\tilde{u}) \le \frac{d}{100}]\cdot\frac{d}{100} + \Pr[\mathrm{Ham}(u,\tilde{u}) > \frac{d}{100}]\cdot d \le 1\cdot\frac{d}{100} + \frac{1}{20}\cdot d \le \frac{d}{10}.$$

By plugging in the definition of I_G, we obtain

$$\frac{d}{10} \ge \mathbb{E}[\mathrm{Ham}(u,\tilde{u})] = \sum_{i\in[d]} \Pr[u_i \ne \tilde{u}_i] \ge \sum_{i\in[d]\setminus I_G} \Pr[u_i \ne \tilde{u}_i] \ge \sum_{i\in[d]\setminus I_G} \frac{1}{5} = \frac{d - |I_G|}{5},$$

implying that $|I_G| \ge d/2$. Because $\mathbb{E}[\sum_{i\in I_G} T_i] \le \mathbb{E}[\sum_{i\in[d]} T_i] \le 2T$, by averaging there exists an index $\mathbf{i} \in I_G$ such that $\mathbb{E}[T_i] \le 4T/d$. This implies that there is a classical algorithm $\mathcal{A}'$ that recovers $u_\mathbf{i}$ with probability at least 0.8 using an expected number of at most $4T/d$ queries to entries in the $\mathbf{i}$th row and column of A.

Now suppose we want to distinguish $u_\mathbf{i} = 1$ from $u_\mathbf{i} = -1$ using samples from $N(\frac{u_\mathbf{i}}{d}, \frac{1}{4\cdot 10^6 d})$. We can use $\mathcal{A}'$ for this task, as follows. Generate $u_1, \ldots, u_{\mathbf{i}-1}, u_{\mathbf{i}+1}, \ldots, u_d$ uniformly at random from ± 1, and generate e'_{ij} from $N(0, \frac{1}{4\cdot 10^6 d})$ for all $1 \le i \le j \le d$. Then we define a $d \times d$ matrix A' as

$$A'_{ij} = \begin{cases} \frac{1}{d}u_i u_j + e'_{ij}, & \text{if } 1 \le i \le j \le d, \mathbf{i} \notin \{i,j\} \\ u_j \cdot \text{sample from } N(\frac{u_\mathbf{i}}{d}, \frac{1}{4\cdot 10^6 d}) & \text{if } i = \mathbf{i} \text{ and } i < j \\ u_i \cdot \text{sample from } N(\frac{u_\mathbf{i}}{d}, \frac{1}{4\cdot 10^6 d}) & \text{if } j = \mathbf{i} \text{ and } i < j \\ A'_{ji}, & \text{otherwise.} \end{cases}$$

Note that for every $i, j \in [d]$, A'_{ij} is a sample drawn from $N(\frac{u_i u_j}{d}, \frac{1}{4\cdot 10^6 d})$. Given that our algorithm knows the values it generated itself (in particular, all u_j with $j \ne \mathbf{i}$), it can implement one query to an entry in the $\mathbf{i}$th row or column of A' by at most one

new sample from $N(\frac{u_{\mathbf{i}}}{d}, \frac{1}{4\cdot 10^6 d})$, and queries to other entries of A' do not cost additional samples.

By running $\mathcal{A}'$ on the input matrix A', we have $\Pr[u_{\mathbf{i}} = \tilde{u}_{\mathbf{i}}] \geq 0.8$. Hence by using an expected number of $4T/d$ samples drawn from $N(\frac{u_{\mathbf{i}}}{d}, \frac{1}{4\cdot 10^6 d})$, we can distinguish $u_{\mathbf{i}} = 1$ from $u_{\mathbf{i}} = -1$ with probability ≥ 0.8. Then by Markov's inequality, worst-case $40T/d$ samples suffice to distinguish $u_{\mathbf{i}} = 1$ from $u_{\mathbf{i}} = -1$ with probability ≥ 0.7.

The KL-divergence between $N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d})$ and $N(-\frac{1}{d}, \frac{1}{4\cdot 10^6 d})$ is $\mathbb{E}_{x\sim N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d})}[8\cdot 10^6 x] = \frac{8\cdot 10^6}{d}$. As a result, by using the well-known Pinsker's inequality ($d_{TV}(P,Q) \leq \sqrt{\frac{1}{2} D_{KL}(P||Q)}$) and the fact that $D_{KL}(P^{\otimes t}||Q^{\otimes t}) = t \cdot D_{KL}(P||Q)$ for any distributions P, Q and natural number t, we obtain

$$\begin{aligned}
\Omega(1) &\leq d_{TV}\Big(N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d})^{\otimes \frac{40T}{d}}, N(-\frac{1}{d}, \frac{1}{4\cdot 10^6 d})^{\otimes \frac{40T}{d}}\Big) \\
&\leq \sqrt{\frac{1}{2} D_{KL}\Big(N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d})^{\otimes \frac{40T}{d}} \Big|\Big| N(-\frac{1}{d}, \frac{1}{4\cdot 10^6 d})^{\otimes \frac{40T}{d}}\Big)} \quad \text{(by Pinsker's inequality)} \\
&= \sqrt{\frac{1}{2} \cdot \frac{40T}{d} \cdot D_{KL}\Big(N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d}) \Big|\Big| N(-\frac{1}{d}, \frac{1}{4\cdot 10^6 d})\Big)} = \mathcal{O}\left(\sqrt{T/d^2}\right),
\end{aligned}$$

implying $T = \Omega(d^2)$. □

By the discussion in Section 7.2, we therefore obtain the following corollary.

Corollary 7.2. *Let A be a $d \times d$ symmetric matrix with $\|A\| = \mathcal{O}(1)$ and an $\Omega(1)$ gap between its top and second eigenvalues. Suppose we have query access to the entries of A. Every classical algorithm that with probability at least $\geq 99/100$, approximates the top eigenvector of A with ℓ_2-error at most $\frac{1}{1000}$ uses $\Omega(d^2)$ queries.*

This query lower bound is tight up to the constant factor, since we can compute the top eigenvector exactly using d^2 queries: just query every entry of A and diagonalize the now fully known matrix A (without any further queries) to find the top eigenvector exactly.

7.4 A quantum lower bound

Now we move to the quantum case, still using the same hard instance. Our proof uses similar ideas as the hybrid method [BBB+97] and adversary method [Amb02; Amb06] for quantum query lower bounds, but adjusted to continuous random variables instead of input bits.

Theorem 7.3. *Let $u \in \{-1,1\}^d$ be a uniformly random vector, $e_{11}, e_{12}, \ldots, e_{1d}, e_{22}, \ldots, e_{dd}$ be $d(d+1)/2$ independent samples drawn from $N(0, \frac{1}{4\cdot 10^6 d})$, and $A \in \mathbb{R}^{d\times d}$ be the random matrix defined by*

$$A_{ij} = \begin{cases} \frac{1}{d} u_i u_j + e_{ij}, & \text{if } 1 \leq i \leq j \leq d, \\ A_{ji}, & \text{otherwise.} \end{cases}$$

Suppose we have quantum query access to entries of A. Every bounded-error quantum algorithm that computes $\tilde{u} \in \{-1,1\}^d$ at Hamming distance $\leq d/100$ from u, uses $\Omega(d^{1.5}/\sqrt{\log d})$ queries.

Proof. Let ν denote the input distribution given in the theorem statement, and $X \sim \nu$ be a random $d \times d$ input matrix according to that distribution (with instantiations of random variable X denoted by lower-case x). Let O_x denote the query oracle to input matrix x. Suppose there exists a T-query quantum algorithm $\mathcal{A} = U_T O_x U_{T-1} \cdots U_1 O_x U_0$, alternating queries and input-independent unitaries on some fixed initial state (say, all-0), to compute such a $\tilde{u}$ with error probability $\leq 1/20$, probability taken over both ν and the internal randomness of $\mathcal{A}$ caused by the measurement of its final state. Our goal is to lower bound T.

For $t \in \{0,\ldots,T-1\}$, let $|\psi_x^t\rangle = \sum_{i,j\in[d]} \alpha_{xij}^t |i,j\rangle|\phi_{xij}^t\rangle$ be the quantum state of algorithm $\mathcal{A}$ just before its $(t+1)$st query on input matrix x. Let $|\psi_x^T\rangle$ be the final state on input x. We define the query mass on $(i,j) \in [d] \times [d]$ on input x as $p_{ijx} = \sum_{t\in[T]} |\alpha_{xij}^t|^2$, and define the query mass on i on input x as $p_{ix} = \sum_{t\in[T],j\in[d]} |\alpha_{xij}^t|^2 + |\alpha_{xji}^t|^2$. Note that $\sum_{i,j} |\alpha_{xij}^t|^2 \leq 1$ for all x and t. Every $|\alpha_{xij}^t|^2$ is counted once in p_{ix} and once in p_{jx} if $i \neq j$, and counted only in p_{ix} if $i = j$. Hence we have $\sum_{i\in[d]} p_{ix} \leq 2T$ for every x. Define $T_i = \mathbb{E}_{x\sim\nu}[p_{ix}]$ as the expected query mass on the ith row and column of the input matrix, then $\sum_{i\in[d]} T_i \leq 2T$.

Call index $i \in [d]$ "good" if $\Pr[u_i = \tilde{u}_i] \geq 0.8$, where the probability is taken over ν and the internal randomness of the algorithm. Let I_G be the set of good indices. Since $\mathcal{A}$ has error probability at most $1/20$, we have

$$\mathbb{E}_x[\mathrm{Ham}(u,\tilde{u})] \leq \Pr[\mathrm{Ham}(u,\tilde{u}) \leq \frac{d}{100}]\cdot\frac{d}{100} + \Pr[\mathrm{Ham}(u,\tilde{u}) > \frac{d}{100}]\cdot d \leq 1\cdot\frac{d}{100} + \frac{1}{20}\cdot d \leq \frac{d}{10}.$$

Using linearity of expectation and the definition of I_G, we have

$$\frac{d}{10} \geq \mathbb{E}_x[\mathrm{Ham}(u,\tilde{u})] = \sum_{i\in[d]} \Pr[u_i \neq \tilde{u}_i] \geq \sum_{i\in[d]\setminus I_G} \Pr[u_i \neq \tilde{u}_i] \geq \frac{d - |I_G|}{5},$$

which implies $|I_G| \geq d/2$. Since $\sum_{i\in I_G} T_i \leq \sum_{i\in[d]} T_i \leq 2T$, by averaging there exists an index $\mathbf{i} \in I_G$ such that $T_{\mathbf{i}} \leq 4T/d$. We fix this $\mathbf{i}$ for the rest of the proof. Note that because $\mathbf{i}$ has $\Pr_\nu[u_{\mathbf{i}} = \tilde{u}_{\mathbf{i}}] \geq 0.8$, we also have $\Pr_{\nu_+}[u_{\mathbf{i}} = \tilde{u}_{\mathbf{i}}] \geq 0.6$ and $\Pr_{\nu_-}[u_{\mathbf{i}} = \tilde{u}_{\mathbf{i}}] \geq 0.6$, where the distributions ν_+ and ν_- are ν conditioned on $u_{\mathbf{i}} = 1$ and $u_{\mathbf{i}} = -1$, respectively.

We now define an (adversarial) joint distribution μ on (X,Y)-pairs of input matrices, such that the marginal distribution of X is ν_+ and the marginal distribution of Y is ν_-. First sample a matrix x (with associated $u \in \{-1,1\}^d$ with $u_{\mathbf{i}} = 1$) according to ν_+. We want to probabilistically modify this into a matrix y by changing only a small number of entries, and only in the $\mathbf{i}$th row and column of x. Let f and g be the pdf of $N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d})$ and $N(-\frac{1}{d}, \frac{1}{4\cdot 10^6 d})$, respectively. Consider an entry $x_{\mathbf{i}j}$ in the $\mathbf{i}$th row of x, with $j \neq \mathbf{i}$. Conditioned on the particular u we sampled, its pdf was f if $u_j = 1$ and g

if $u_j = -1$. If the pdf of $x_{\mathbf{i}j}$ was f, then obtain $y_{\mathbf{i}j}$ from $x_{\mathbf{i}j}$ as follows: if $x_{\mathbf{i}j} > 0$, then negate it with probability $\frac{f(x_{\mathbf{i}j})-g(x_{\mathbf{i}j})}{f(x_{\mathbf{i}j})}$, else leave it unchanged.

Claim 7.1. If $x_{\mathbf{i}j} \sim N(\frac{1}{d}, \frac{1}{4\cdot 10^6 d})$, then $y_{\mathbf{i}j} \sim N(-\frac{1}{d}, \frac{1}{4\cdot 10^6 d})$.

Proof: Let h be the pdf of $y_{\mathbf{i}j}$. For a value $z > 0$, we have $h(z) = f(z) - f(z) \cdot \frac{f(z)-g(z)}{f(z)} = g(z)$.
For $z \leq 0$ we have $h(z) = f(z) + f(-z) \cdot \frac{f(-z)-g(-z)}{f(-z)} = f(z) + f(-z) - g(-z) = f(-z) = g(z)$.
■

If the pdf of $x_{\mathbf{i}j}$ was g instead of f, then we do something analogous: if $x_{\mathbf{i}j} < 0$, then negate it with probability $\frac{g(x_{\mathbf{i}j})-f(x_{\mathbf{i}j})}{g(x_{\mathbf{i}j})}$. This gives the analogous claim: the pdf of $y_{\mathbf{i}j}$ is then f.

Let matrix y be obtained by applying this probabilistic process to all entries in the **i**th row of x, and changing the entries in the **i**th column to equal the new **i**th row (since the resulting y needs to be a symmetric matrix). Outside of the **i**th row and column, x and y are equal. Let μ be the resulting joint distribution on (x, y) pairs. An important property of this distribution that we use below, is that the $d \times d$ matrices x and y typically only differ in roughly $\sqrt{d}$ entries, because the probability with which $x_{\mathbf{i}j}$ is modified (=negated) is $\mathcal{O}(1/\sqrt{d})$. The marginal distribution of Y is ν_-, because the change we made in the X-distribution corresponds exactly to changing $u_{\mathbf{i}}$ from 1 to -1. We could equivalently have defined μ by first sampling $Y \sim \nu_-$, and then choosing x_{ij} by an analogous negating procedure on y_{ij}.

We now use the general template of the adversary method [Amb02] together with our distribution μ to lower bound the total number T of queries that $\mathcal{A}$ makes. Define a progress measure $S_t = \mathbb{E}_{xy\sim\mu}[\langle\psi_x^t|\psi_y^t\rangle]$. As usual in the adversary method, this measure is large at the start of the algorithm and becomes small at the end: $S_0 = 1$ because $\langle\psi_x^0|\psi_y^0\rangle = 1$ for all x, y (since the initial state is fixed, independent of the input); and $S_T \leq 1 - \Omega(1)$ because for $(x, y) \sim \mu$, our algorithm outputs 1 with probability at least 0.6 on x and outputs -1 with probability at least 0.6 on y, meaning that $\langle\psi_x^T|\psi_y^T\rangle$ is typically bounded below 1. Let $\Delta_t = |S_{t+1} - S_t|$ be the change in the progress measure

due to the $(t+1)$st query. We can upper bound that change as follows:

$$\begin{aligned}
\Delta_t &= |\mathbb{E}_{xy\sim\mu}[\langle\psi_x^{t+1}|\psi_y^{t+1}\rangle - \langle\psi_x^t|\psi_y^t\rangle]| \\
&= |\mathbb{E}_{xy\sim\mu}[\langle\psi_x^t|\big(O_x^\dagger O_y - I\big)|\psi_y^t\rangle]| \\
&= |\mathbb{E}_{xy\sim\mu}\left[\sum_{i,j\in[d]} \alpha_{xij}^t \langle i,j|\langle\phi_{xij}^t| \sum_{i,j:x_{ij}\neq y_{ij}} \alpha_{yij}^t |i,j\rangle|\phi_{yij}'^t\rangle\right]| \\
&\leq \mathbb{E}_{xy\sim\mu}\left[\sum_{i,j:x_{ij}\neq y_{ij}} |\alpha_{xij}^t|\cdot|\alpha_{yij}^t|\right] \\
&= \mathbb{E}_{xy\sim\mu}\left[\sum_{j:x_{\mathbf{i}j}\neq y_{\mathbf{i}j}} |\alpha_{x\mathbf{i}j}^t|\cdot|\alpha_{y\mathbf{i}j}^t| + \sum_{j:x_{j\mathbf{i}}\neq y_{j\mathbf{i}}} |\alpha_{xj\mathbf{i}}^t|\cdot|\alpha_{yj\mathbf{i}}^t|\right] \\
&\leq \frac{1}{2}\mathbb{E}_{xy\sim\mu}\left[\sum_{j:x_{\mathbf{i}j}\neq y_{\mathbf{i}j}} \left(|\alpha_{x\mathbf{i}j}^t|^2 + |\alpha_{y\mathbf{i}j}^t|^2\right) + \sum_{j:x_{j\mathbf{i}}\neq y_{j\mathbf{i}}} \left(|\alpha_{xj\mathbf{i}}^t|^2 + |\alpha_{yj\mathbf{i}}^t|^2\right)\right].
\end{aligned}$$

where we use that $|\psi_x^{t+1}\rangle = U_{t+1}O_x|\psi_x^t\rangle$ and $|\psi_y^{t+1}\rangle = U_{t+1}O_y|\psi_y^t\rangle$, that $O_x^\dagger O_y : |i,j,b\rangle \to |i,j\rangle|b - x_{ij} + y_{ij}\rangle$, that x and y only differ in the $\mathbf{i}$th row and column, and the AM-GM inequality ($ab \leq (a^2+b^2)/2$) in the last step.

Now observe that

$$\begin{aligned}
\mathbb{E}_{xy\sim\mu}\left[\sum_{j:x_{\mathbf{i}j}\neq y_{\mathbf{i}j}} |\alpha_{x\mathbf{i}j}^t|^2\right] &= \sum_{j\in[d]} \mathbb{E}_x[\Pr_{y\sim\mu|x}[x_{\mathbf{i}j}\neq y_{\mathbf{i}j}]\cdot|\alpha_{x\mathbf{i}j}^t|^2] \\
&= \sum_j \int_0^\infty \frac{f(x_{\mathbf{i}j}) - g(x_{\mathbf{i}j})}{f(x_{\mathbf{i}j})}\cdot|\alpha_{x\mathbf{i}j}^t|^2\cdot f(x_{\mathbf{i}j})\mathrm{d}x_{\mathbf{i}j} \\
&= \sum_j \Big(\int_0^{\frac{10\sqrt{\log d}}{\sqrt{d}}} \big(1 - \frac{g(x_{\mathbf{i}j})}{f(x_{\mathbf{i}j})}\big)\cdot|\alpha_{x\mathbf{i}j}^t|^2\cdot f(x_{\mathbf{i}j})\mathrm{d}x_{\mathbf{i}j} \\
&\quad + \int_{\frac{10\sqrt{\log d}}{\sqrt{d}}}^\infty \big(f(x_{\mathbf{i}j}) - g(x_{\mathbf{i}j})\big)\cdot|\alpha_{x\mathbf{i}j}^t|^2\mathrm{d}x_{\mathbf{i}j}\Big) \\
&\leq \sum_j \Big(\max_{z\in[0,\frac{10\sqrt{\log d}}{\sqrt{d}}]} |1-\exp(-8\cdot10^6 z)|\cdot\mathbb{E}_x[|\alpha_{x\mathbf{i}j}^t|^2] \\
&\quad + \int_{\frac{10\sqrt{\log d}}{\sqrt{d}}}^\infty \big(f(x_{\mathbf{i}j}) - g(x_{\mathbf{i}j})\big)\mathrm{d}x_{\mathbf{i}j}\Big) \\
&\leq \sum_j \Big(\frac{8\cdot10^7\sqrt{\log d}}{\sqrt{d}}\cdot\mathbb{E}_{x\sim\nu_+}[|\alpha_{x\mathbf{i}j}^t|^2] + 2\exp(-100\log d)\Big),
\end{aligned}$$

where the first part of the first inequality holds because $\frac{g(x_{\mathbf{i}j})}{f(x_{\mathbf{i}j})} = \exp(-8\cdot10^6 x_{\mathbf{i}j})$, the first part of the second inequality holds because for every z, $1-\exp(-z)\leq z$, and the second part of the second inequality holds because both f, g are Gaussians with variance $\frac{1}{4\cdot10^6 d}$ and $\frac{10\sqrt{\log d}}{\sqrt{d}} - \frac{1}{d} \geq 10\cdot\frac{1}{2000\sqrt{d}}$.

We can similarly upper bound $\sum_{t\in[T]} \mathbb{E}_{xy\sim\mu}[\sum_{j:x_{j\mathbf{i}}\neq y_{j\mathbf{i}}} |\alpha^t_{xj\mathbf{i}}|^2]$, $\sum_{t\in[T]} \mathbb{E}_{xy\sim\mu}[\sum_{j:x_{\mathbf{i}j}\neq y_{\mathbf{i}j}} |\alpha^t_{y\mathbf{i}j}|^2]$, and $\sum_{t\in[T]} \mathbb{E}_{xy\sim\mu}[\sum_{j:x_{j\mathbf{i}}\neq y_{j\mathbf{i}}} |\alpha^t_{yj\mathbf{i}}|^2]$. Now we have

$$\begin{aligned}
\Omega(1) &\leq |S_0 - S_T| \\
&\leq \sum_{t\in[T]-1} \Delta_t \\
&\leq \frac{1}{2} \sum_{t\in[T]-1} \mathbb{E}_{xy\sim\mu} \left[\sum_{j:x_{\mathbf{i}j}\neq y_{\mathbf{i}j}} \left(|\alpha^t_{x\mathbf{i}j}|^2 + |\alpha^t_{y\mathbf{i}j}|^2 \right) + \sum_{j:x_{j\mathbf{i}}\neq y_{j\mathbf{i}}} \left(|\alpha^t_{x\mathbf{i}j}|^2 + |\alpha^t_{y\mathbf{i}j}|^2 \right) \right] \\
&\leq \sum_{t\in[T]-1, j\in[d]} \left(\frac{4\cdot 10^7 \sqrt{\log d}}{\sqrt{d}} \left(\mathbb{E}_{x\sim\nu_+}[|\alpha^t_{x\mathbf{i}j}|^2 + |\alpha^t_{x\mathbf{i}j}|^2] + \mathbb{E}_{y\sim\nu_-}[|\alpha^t_{y\mathbf{i}j}|^2 + |\alpha^t_{yj\mathbf{i}}|^2] \right) + 4\exp(-100\log d) \right) \\
&\leq \frac{4\cdot 10^7 \sqrt{\log d}}{\sqrt{d}} \left(\mathbb{E}_{x\sim\nu_+}[p_{\mathbf{i}x}] + \mathbb{E}_{y\sim\nu_-}[p_{\mathbf{i}y}] \right) + 4d^3 \exp(-100\log d) \\
&\leq \frac{8\cdot 10^7 \sqrt{\log d}}{\sqrt{d}} \mathbb{E}_{x\sim\nu}[p_{\mathbf{i}x}] + 4d^3 \exp(-100\log d) \\
&\leq \frac{8\cdot 10^7 \sqrt{\log d}}{\sqrt{d}} \cdot \frac{4T}{d} + 4d^3 \exp(-100\log d),
\end{aligned}$$

where the sixth inequality uses that $\nu_+ + \nu_- = 2\nu$, and that j ranges over d values and t ranges over T values (and $T \leq d^2$ without loss of generality). This implies $T = \Omega(d^{1.5}/\sqrt{\log d})$. □

Again invoking the discussion in Section 7.2, we obtain the following corollary which shows that our second algorithm is close to optimal.

Corollary 7.4. *Let A be a $d \times d$ symmetric matrix with $\|A\| = \mathcal{O}(1)$ and an $\Omega(1)$ gap between its top and second eigenvalues. Suppose we have quantum query access to the entries of A. Every quantum algorithm that with probability at least $\geq 99/100$, approximates the top eigenvector of A with ℓ_2-error at most $\frac{1}{1000}$ uses $\Omega(d^{1.5}/\sqrt{\log d})$ queries.*

7.5 Open problems

We mention a few directions for future work:

- In Chapter 4 we show that to approximate top-q eigenspace (a rank-q subspace), it is necessary and sufficient to use $\tilde{\Theta}(dq)$ applications of controlled projection Π_q onto the top-q subspace (see Lemma 4.12). Then what is the correct quantum query bound for approximating the top-q eigenspace? It is not even clear how to connect the query lower bound to the lower bound of the applications of controlled Π_q (assuming $\lambda_1 = \cdots = \lambda_q = 1, \lambda_{q+1} = \cdots = \lambda_d = 0$).

- What is the correct quantum query bound for approximating the top-q eigenvectors? We conjecture the correct query lower bound should be $\Omega(\sqrt{d^3q})$. We tried to prove this by considering multiple orthonormal vectors $u_1 \ldots, u_q$ and by letting the matrix $A = \sum_{i\in[q]} \frac{q-i+1}{q+1} u_i u_i^T + N$. However, we do not see how the (approximate) top-q eigenvectors of A could recover all u_i-s. Maybe we should involve more structures for those u_i-s to ensure we can always decode them even only given an isometry $d \times q$ matrix for top-q eigenspace (approximately)?

- Is it possible to remove the $\sqrt{\log d}$ factor in the quantum query lower bound for approximating the top-eigenvector?

Part III

Conditional quantum lower bounds

CHAPTER 8

The quantum strong exponential-time hypothesis and its applications

8.1 Introduction

A popular classical hardness conjecture known as the Strong Exponential Time-Hypothesis (SETH) says that determining whether an input CNF formula is satisfiable or not cannot be done in $\mathcal{O}(2^{n(1-\delta)})$ time for any constant $\delta > 0$ [IP01; IPZ01]. Several conditional lower bounds based on SETH have been shown since then; see [Vas15; Vas18] for a summary of many such results (and see the end of Section 1.1.3 for a brief introduction of fine-grained complexity). Some of the SETH-hard problems are building blocks for fine-grained cryptography [BRS+17; LLW19]. Besides finding a satisfying assignment, natural variants of the CNFSAT problem include computing the number or the parity of the number of satisfying assignments to a CNF formula known as #SETH and ⊕SETH. Counting-SETH (#SETH) says that calculating the exact number of satisfying assignments of an input CNF formula cannot be done in $\mathcal{O}(2^{n(1-\delta)})$ time for any constant δ, and parity-SETH (⊕SETH) says calculating the parity of satisfying assignments of an input CNF formula cannot be done in $\mathcal{O}(2^{n(1-\delta)})$ time for any constant δ. These conjectures are weaker (i.e. more believable) than SETH since if one of #SETH and ⊕SETH is false, then SETH is false immediately as well. Nevertheless, those conjectures still play a central role in fine-grained complexity and can still be used to show conditional lower bounds for various problems [CDL+16].

When considering quantum computation, the SETH conjecture is no longer true, because using Grover's algorithm for unstructured search [Gro96] (see Section 2.3 of this thesis) one can solve the CNFSAT problem in $2^{\frac{n}{2}} \cdot \text{poly}(n)$ time. Aaronson, Chia, Lin, Wang, and Zhang assume this Grover-like quadratic speedup is nearly optimal for CNFSAT and initiate the study of quantum fine-grained complexity [ACL+20]. However, conjectures such as #SETH or ⊕SETH are likely to still hold in the quantum setting because a Grover-like quantum speedup is not witnessed when the task is to compute the total number of satisfying assignments or the parity of this number. This situation can in some cases be exploited to give better quantum lower bounds than one would

get from the conjectured quantum lower bound for the vanilla CNFSAT problem. This makes it at least as relevant (if not more) to study variants of CNFSAT and their implications in the quantum setting, as has been done classically. In fact, motivated by this exact observation, Buhrman, Patro, and Speelman [BPS21] introduced a framework of Quantum Strong Exponential-Time Hypotheses (QSETH) as quantum analogues to SETH, with a striking feature that allows one to 'technically' unify conjectures such as quantum analogues of ⊕SETH, #SETH, etc. under one umbrella conjecture.

8.1.1 Main results and high-level intuition

In this chapter, inspired by Buhrman, Patro, and Speelman's QSETH framework, we state a few variants of CNFSAT problems and state conjectures and discuss their corresponding *time* lower bounds. The high-level idea of their QSETH framework (without going into detailed complexity notions): Consider the problem in which one is given a circuit representation of an n-variable Boolean function and is asked whether a property P on the truth table of this circuit evaluates to 1. The authors in [BPS21] conjectured that for *most natural* properties P, the time taken to compute P on the truth table of poly(n)-sized circuits is lower bounded by $Q_{1/3}(P)$, i.e. the bounded-error quantum query complexity of P, on all bit-strings of length $N := 2^n$. In other words, having a description of the truth table in the form of a small circuit shouldn't help very much compared to the black-box situation.

Problem	Variants	Q-time lower bound	Reference
CNFSAT	Vanilla	$2^{\frac{n}{2}-o(n)}$	Conjecture 8.3
	Parity	$2^{n-o(n)}$	Conjecture 8.9
	Majority	$2^{n-o(n)}$	Conjecture 8.10
	Count	$2^{n-o(n)}$	Conjecture 8.8
	Count_q	$2^{n-o(n)}$	Conjecture 8.11
	Δ-Additive error	$\left(\sqrt{\frac{2^n}{\Delta}} + \frac{\sqrt{\hat{h}(2^n-\hat{h})}}{\Delta}\right)^{1-o(1)}$	Conjecture 8.15
	γ-Multi factor	$\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-o(1)}$	Corollary 8.19
k-SAT $k = \Theta(\log(n))$	Vanilla	$2^{\frac{n}{2}-o(n)}$	[ACL+20]
	Parity	$2^{n-o(n)}$	Corollary 8.21
	Count	$2^{n-o(n)}$	Corollary 8.21
	Count_q	$2^{n-o(n)}$	Corollary 8.21
	γ-Multi factor	$\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-o(1)}$	Corollary 8.22

Table 8.1: Overview of conditional quantum time lower bounds for variants of CNFSAT and k-SAT. The variable $\hat{h}$ above is an arbitrary natural number satisfying $\gamma\hat{h} \geq 1$. Our results hold for the multiplicative factor $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$ and the additive error $\Delta \in [1, 2^n)$.

Instead of starting with a bunch of definitions and notions used in the QSETH framework by [BPS21], in this chapter, we avoid those definitions and notions and will directly introduce some natural properties and their corresponding variants of CNFSAT problems, and then state the conjectures and discuss the time complexity. Though those variants of CNFSAT problems are inspired by the QSETH framework, the standalone conjectures themselves are already interesting. The quantum time lower bound for those variants of CNFSAT is then conjecturally lower-bounded by the corresponding quantum query lower bound, and the conjecture will be called P-QSETH in this chapter for the corresponding property P. We list those variants of QSETH in Table 8.1.

We also use the above-mentioned lower bound for CNFSAT to understand the quantum complexity of k-SAT. As a first step we study the classical reduction from CNFSAT to k-SAT given by [CIP06] and observe that the $2^{\frac{n}{2}}$ quantum lower bound for k-SAT, for $k = \Theta(\log n)$, follows from the quantum lower bound of CNFSAT. After that, we observe that this reduction by [CIP06] is count-preserving and can be used to conclude lower bounds for other counting variants of k-SAT. After that, we give conditional quantum time lower bounds for lattice problems, strong simulation, orthogonal vectors, set cover, hitting set problem, and their respective variants (see Table 8.2).

Roadmap

Throughout the whole chapter, we use N to denote 2^n. In Section 8.2, we will start with introducing some popular properties and their corresponding CNFSAT problems and then conjecture the quantum *time* lower bound for those problems according to their quantum query lower bound, as well as the consequences for the time complexity of corresponding variants of k-SAT problems. After that, in Section 8.3 we use those QSETH-based lower bounds (for variants of CNFSAT) to show quantum conditional lower bounds for some (variants of) popular problems that are widely studied in the classical case.

8.2 Conjectures for variants of CNFSAT

In this section, just as in the QSETH framework [BPS21], the corresponding quantum time lower bound is conjectured to be the quantum query lower bound of computing P on all bit-strings of length N. We begin with the definitions for CNFSAT, k-SAT, and some common variants of CNFSAT which are also very well-studied classically [CDL+16], and then proceed with some less popular variants but with interesting consequences.

Problem	Variants	Q-time lower bound	Reference
STRONG SIMULATION	Exact (n + 1-bit precision)	$2^{n-o(n)}$	Theorem 8.24
	Exact (2-bit precision)	$2^{n-o(n)}$	Corollary 8.25
	Δ-Add error	$\left(\sqrt{\frac{1}{\Delta}} + \frac{\sqrt{\hat{h}(2^n-\hat{h})}}{2^n\Delta}\right)^{1-o(1)}$	Corollary 8.27
	γ-Multi factor	$\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-o(1)}$	Theorem 8.29
CVP$_p$ FOR $p \notin 2\mathbb{Z}$		$2^{\frac{n}{2}-o(n)}$	Section 8.3.2
LATTICE COUNTING (non-even norm)	Vanilla	$2^{n-o(n)}$	Corollary 8.36
	q-count	$2^{n-o(n)}$	Corollary 8.36
	γ-Multi factor	$\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-o(1)}$	Corollary 8.37
ORTHOGONAL VECTORS	Vanilla	$n^{1-o(1)}$	[ACL+20; BPS21]
	Parity	$n^{2-o(1)}$	Corollary 8.47
	Count	$n^{2-o(1)}$	Corollary 8.47
	γ-Multi factor	$\left(\frac{1}{\gamma}\sqrt{\frac{n^2-\hat{h}}{\hat{h}}}\right)^{1-o(1)}$	Corollary 8.48
HITTING SET	Vanilla	$2^{\frac{n}{2}-o(n)}$	Corollary 8.43
	Parity	$2^{n-o(n)}$	Corollary 8.43
	Count	$2^{n-o(n)}$	Corollary 8.43
	γ-Multi factor	$\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-o(1)}$	Corollary 8.44
$\oplus$SET COVER		$2^{n-o(n)}$	Corollary 8.50

Table 8.2: Overview of lower bounds based on conditional quantum time lower bounds for variants of CNFSAT. The variable $\hat{h}$ in the above table is an arbitrary natural number satisfying $\gamma\hat{h} \geq 1$. Our results hold for the multiplicative factor $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$ and the additive error $\Delta \in [\frac{1}{2^n}, 1)$.

CNFSAT and k-SAT

A Boolean formula over variables $x_1, \ldots, x_n$ is in CNF form if it is an AND of OR's of variables or their negations. More generally, a CNF formula has the form

$$\bigwedge_i \left(\bigvee_j v_{ij} \right)$$

where v_{ij} is either x_k or $\neg x_k$. The terms v_{ij} are called *literals* of the formula and the disjunctions $\bigvee_j v_{ij}$ are called its clauses. A k-CNF is a CNF formula in which all clauses contain at most k literals (or the clause width is at most k). Note that when $k > n$, then clauses must contain duplicate or trivial literals (for example, $x_k \vee \neg x_k$ and $x_k \vee x_k$), therefore we can assume without loss of generality that k is at most n. A DNF is defined in the exact same way as CNF, except that it is an OR of AND's of variables or their negations, that is, a DNF formula has the form $\bigvee_i \left(\bigwedge_j v_{ij} \right)$. We also define computational problems k-SAT and CNFSAT.

Definition 8.1 (CNFSAT)**.** Given as input a CNF formula ϕ defined on n variables, the goal is to determine if $\exists x \in \{0,1\}^n$ such that $\phi(x) = 1$.

Definition 8.2 (k-SAT)**.** Given as input a k-CNF formula ϕ defined on n variables, the goal is to determine if $\exists x \in \{0,1\}^n$ such that $\phi(x) = 1$.

8.2.1 Quantum complexity of CNFSAT and other related problems

We first restate the (conjectured) quantum hardness of CNFSAT before showing hardness results for its other variants. One can consider CNFSAT as a problem that asks to compute OR on the truth table of the input CNF formula. Since the quantum query lower bound for computing OR on 2^n-bit strings is $\Omega(2^{n/2})$, we therefore conjecture the following quantum time lower bound:

Conjecture 8.3 (BASIC-QSETH [ACL+20; BPS21])**.** For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}\left(2^{\frac{n(1-\delta)}{2}}\right)$ time.

Quantum complexity of #CNFSAT, $\oplus$CNFSAT, $\#_q$CNFSAT and MAJ-CNFSAT

To give conditional quantum lower bounds for variants of CNFSAT, we should understand their corresponding quantum query lower bound (on the 2^n-bit truth table). Here we introduce the properties that correspond to those popular variants of CNFSAT (which will be defined later.)

Definition 8.4. Let $|x| = |\{i : x_i = 1\}|$ denote the Hamming weight of N-bit binary string x. We here list some properties defined on binary strings.

1. COUNT: Let $\mathrm{COUNT} : \{0,1\}^N \to [N] \cup \{0\}$ be the non-Boolean function defined by $\mathrm{COUNT}(x) = |x|$.

2. PARITY: Let $\mathrm{PARITY} : \{0,1\}^N \to \{0,1\}$ be the Boolean function defined by $\mathrm{PARITY}(x) = |x| \bmod 2$.

3. COUNT_q: Let q be an integer and let $\mathrm{COUNT}_q : \{0,1\}^N \to [q]-1$ be the non-Boolean function defined by $\mathrm{COUNT}_q(x) = |x| \mod q$.

4. MAJORITY: Let $\mathrm{MAJORITY} : \{0,1\}^N \to \{0,1\}$ be the Boolean function defined by

$$\mathrm{MAJORITY}(x) = \begin{cases} 1 & \text{if } |x| \geq \frac{N}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

   And, there is also the following function almost similar to MAJORITY.

5. *strict*-MAJORITY: Let $\textit{strict}\text{-}\mathrm{MAJORITY} : \{0,1\}^N \to \{0,1\}$ be the Boolean function with

$$\textit{strict}\text{-}\mathrm{MAJORITY}(x) = \begin{cases} 1 & \text{if } |x| > \frac{N}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Here, we define variants of CNFSAT corresponding to the above-mentioned properties.

Definition 8.5 (variants of CNFSAT)**.** Let $|\phi| = \{y \in \{0,1\}^n : \phi(y) = 1\}$ denote the Hamming weight of the truth table of ϕ. The following lists five variants of CNFSAT:

1. #CNFSAT: Given a CNF formula ϕ on n input variables, output $|\phi|$.

2. $\oplus$CNFSAT: Given a CNF formula ϕ on n input variables, output $|\phi| \bmod 2$.

3. $\#_q$CNFSAT: Given a CNF formula ϕ on n input variables and an integer $q \in [2^n] \setminus \{1\}$, output $|\phi| \bmod q$.

4. MAJ-CNFSAT: Given a CNF formula ϕ on n input variables, output 1 if $|\phi| \geq 2^n/2$ (else output 0).

5. strict-MAJ-CNFSAT: Given a CNF formula ϕ on n input variables, output 1 if $|\phi| > 2^n/2$ (else output 0).

Now again, we use the quantum query lower bound for P whenever we want to discuss the time complexity of P-CNFSAT as in the QSETH framework by [BPS21]. Therefore, immediately after the definitions for variants of CNFSAT (with respect to property P), we will introduce the corresponding bounded-error quantum *query* lower bound for computing P, and then conjecture the *time* lower bound for P-CNFSAT (P variant CNFSAT) using this query lower bound. After that, we can use lower bounds for those variants of CNFSAT to understand the quantum complexity of (variants of) k-SAT. We include the quantum query lower bounds for those properties for completeness.

Lemma 8.6 ([BBC+01])**.** *The bounded-error quantum query complexity for* COUNT, PARITY, MAJORITY *and strict-*MAJORITY *on inputs of N-bit Boolean strings is* $\Omega(N)$.

Proof. [BBC+01] showed that the bounded-error quantum query complexity of a (total) Boolean function $f : \{0,1\}^N \to \{0,1\}$, denoted by $Q(f) := Q_{1/3}(f)$ (see the end of Section 2.2) is lower bounded by 1/2 of the degree of a minimum-degree polynomial p that approximates f on all $X \in \{0,1\}^N$, i.e., $|p(X) - f(X)| \leq 1/3$; let us denote this approximate degree by $\widetilde{deg}(f)$. Another important result by Paturi [Pat92] showed that if f is a non-constant, symmetric[1] and total Boolean function on $\{0,1\}^N$ then $\widetilde{deg}(f) = \Theta(\sqrt{N(N-\Gamma(f))})$ where $\Gamma(f) = \min\{|2k-N+1| : f_k \neq f_{k+1} \text{ and } 0 \leq k \leq N-1\}$ and $f_k = f(X)$ for $|X| = k$.

Using the above two results we can show the following:

1. $\Gamma(\text{PARITY}) = 0$ for odd N and $\Gamma(\text{PARITY}) = 1$ whenever N is even. Hence $Q(\text{PARITY}) = \Omega(N)$.[2]

2. Similar to the above item $\Gamma(\text{MAJORITY}) = \Gamma(\textit{strict}\text{-MAJORITY}) = 0$ for odd N and $\Gamma(\text{MAJORITY}) = \Gamma(\textit{strict}\text{-MAJORITY}) = 1$ otherwise. Hence, $Q(\text{MAJORITY}) = \Omega(N)$ and $Q(\textit{strict}\text{-MAJORITY}) = \Omega(N)$.

3. Any of the above three properties can be computed from COUNT. Hence, $Q(\text{COUNT}) = \Omega(N)$.

□

Lemma 8.7. *Let $q \in [3, \frac{N}{2}]$ be an integer and $\text{COUNT}_q : \{0,1\}^N \to [q]-1$ be the function defined by $\text{COUNT}_q(x) = \text{COUNT}(x) \mod q$. Then $Q(\text{COUNT}_q) = \Omega(\sqrt{N(N-2q+1)})$.*

Proof. Let DEC-COUNT_q be a decision version of the COUNT_q defined for all $x \in \{0,1\}^N$ as

$$\text{DEC-COUNT}_q(x) = \begin{cases} 1, \text{ if } \text{COUNT}_q(x) = q-1, \\ 0, \text{ otherwise.} \end{cases} \tag{8.1}$$

When the function is non-constant and symmetric then one can use Paturi's theorem to characterize the approximate degree of that function [Pat92]. It is easy to see that DEC-COUNT_q is a non-constant symmetric function. Combining both these results we get that $Q(\text{DEC-COUNT}_q) = \Omega(\sqrt{N(N-\Gamma(\text{DEC-COUNT}_q))})$.

We now compute the value of $\Gamma(\text{DEC-COUNT}_q)$. For any symmetric Boolean function $f : \{0,1\}^N \to \{0,1\}$ the quantity $\Gamma(f)$ is defined as $\Gamma(f) = \min_k\{|2k-N+1|\}$ such that $f_k \neq f_{k+1}$ and $f_k = f(x)$ for $|x| = k$ with $1 \leq k \leq N-1$. It is easy to see that $\text{DEC-COUNT}_q(x) = 1$ only for x with Hamming weight $|x| = rq-1$ where r is an integer and $\text{DEC-COUNT}_q(x) = 0$ elsewhere. Let r' be the integer such that $r'q - 1 \leq \frac{N}{2} \leq (r'+1)q-1$ then the k minimizing $\Gamma(\text{DEC-COUNT}_q)$ is either $r'q-1$ or $(r'+1)q-1$. This

[1] A symmetric Boolean function $f : \{0,1\}^N \to \{0,1\}$ implies $f(X) = f(Y)$ for all X, Y whenever $|X| = |Y|$.

[2] One can actually immediately give $Q(\text{PARITY}) \geq N/2$ by an elementary degree lower bound without using Paturi's result.

implies that $\Gamma(\text{DEC-COUNT}_q) \leq 2q-1$, which in turn implies that $N-\Gamma(\text{DEC-COUNT}_q) \geq N-2q+1$. Therefore, $Q(\text{DEC-COUNT}_q) = \Omega(\sqrt{N(N-2q+1)})$.

As one can compute DEC-COUNT_q using an algorithm that computes COUNT_q, we therefore have $Q(\text{COUNT}_q) = \Omega(\sqrt{N(N-2q+1)})$. □

We now conjecture the quantum time lower bound for the above-mentioned variants of CNFSAT, based on their quantum query lower bound on $N = 2^n$-bit strings.

Conjecture 8.8 (#QSETH)**.** For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves #CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}(2^{n(1-\delta)})$ time.

Conjecture 8.9 ($\oplus$QSETH)**.** For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves $\oplus$CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}(2^{n(1-\delta)})$ time.

Conjecture 8.10 (Majority-QSETH)**.** For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves

1. MAJ-CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}(2^{n(1-\delta)})$ time;

2. strict-MAJ-CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}(2^{n(1-\delta)})$ time.

Note that if the first item of the above conjecture holds, then the second holds immediately. It may seem redundant to define both MAJ-CNFSAT and strict-MAJ-CNFSAT, since from a query-complexity perspective these two problems seem equivalent, and in fact essentially the same. However, Akmal and Williams showed that one can actually compute the Majority on the truth table of k-CNF formulas for constant k in polynomial time, while computing the strict-Majority on the truth table of such formulas is NP-hard [AW21]. Therefore, here we define both majority and strict-majority and their variants of CNFSAT problems for clarity (and state the hardness of both problems in one conjecture). Note that for CNFSAT, each clause is allowed to contain n literals (which means k is no longer a constant), and in this case, it is not clear if one can solve MAJ-CNFSAT in polynomial time or not. Therefore Conjecture 8.10 is not immediately false yet. (See also the discussion at the bottom of page 5 in the arXiv version of [AW21] for reductions between MAJ-CNFSAT and strict-MAJ-CNFSAT.)

Conjecture 8.11 ($\#_q$QSETH)**.** Let $q \in [3, \frac{N}{2}]$ be an integer. For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves $\#_q$CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}\left(2^{n(1-\delta)}\right)$ time.

Note that since we can solve $\oplus$CNFSAT, $\#_q$CNFSAT and MAJ-CNFSAT using a #CNFSAT-solver, if one of Conjectures 8.9 to 8.11 holds, then Conjecture 8.8 holds.

Quantum complexity of Δ-ADD-#CNFSAT

Instead of the exact number of satisfying assignments to a formula, one might be interested in an additive-error approximation. Towards that, we define the problem Δ-ADD-#CNFSAT as follows.

Definition 8.12 (Δ-ADD-#CNFSAT)**.** Given a CNF formula ϕ on n variables. The goal of the problem is to output an integer d such that $|d - |\phi|| < \Delta$ where $\Delta \in [1, 2^n]$.

This problem (Definition 8.12) can be viewed as computing the following property on the truth table of the given formula.

Definition 8.13 (Δ-ADDITIVE-COUNT)**.** Given a Boolean string $x \in \{0,1\}^N$, Δ-ADDITIVE-COUNT asks to output an integer w such that $|w - |x|| < \Delta$ where $\Delta \in [1, N)$.

Note that Δ-ADDITIVE-COUNT is a relation instead of a function now because its value is not necessarily uniquely defined. The bounded-error quantum query complexity for computing Δ-ADDITIVE-COUNT was studied in [NW99]. They showed the following result.

Theorem 8.14 (Theorem 1.11 in [NW99])**.** *Let* $\Delta \in [1, N)$*. Every bounded-error quantum algorithm that computes* Δ-ADDITIVE-COUNT *uses* $\Omega\left(\sqrt{\frac{N}{\Delta}} + \frac{\sqrt{t(N-t)}}{\Delta}\right)$ *quantum queries on inputs with* t *ones.*

Then we use the quantum query lower bound to conjecture the corresponding quantum time lower bound for Δ-ADD-#CNFSAT.

Conjecture 8.15 (Δ-ADD-#QSETH)**.** Let $\Delta \in [1, 2^n)$. For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves Δ-ADD-#CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in $\mathcal{O}\left(\left(\sqrt{\frac{N}{\Delta}} + \frac{\sqrt{\hat{h}(N-\hat{h})}}{\Delta}\right)^{1-\delta}\right)$ time, where $\hat{h}$ is the number of satisfying assignments.

Quantum complexity of γ-#CNFSAT and other related problems

One other approximation of the count of satisfying assignments is the multiplicative-factor approximation, defined as follows.

Definition 8.16 (γ-#CNFSAT)**.** Let $\gamma \in (0,1)$. The γ-#CNFSAT problem is defined as follows. Given a CNF formula formula ϕ on n Boolean variables, The goal of the problem is to output an integer d such that $(1-\gamma)|\phi| < d < (1+\gamma)|\phi|$. [3]

[3]The same results hold if the approximation is defined with the equalities, i.e., $(1-\gamma)|\phi| \leq d \leq (1+\gamma)|\phi|$. An additional observation under this changed definition of γ-#CNFSAT is as follows. Given a CNF formula as input, the algorithm for γ-#CNFSAT outputs 0 only when there is no satisfying assignment to that formula. Hence, one can decide satisfiability of a given CNF formula using the algorithm for γ-#CNFSAT. Therefore, the same lower bound holds for this changed definition of γ-#CNFSAT.

The expression $(1-\gamma)|\phi| < d < (1+\gamma)|\phi|$ can be categorized into the following two cases.

- Case 1 is when $\gamma|\phi| \leq 1$: in this regime, the algorithm solving γ-#CNFSAT is expected to return the value $|\phi|$, which is the *exact* count of the number of solutions to the CNFSAT problem. From Conjecture 8.8 we postulate that for each constant $\delta > 0$, there is no $\mathcal{O}(2^{n(1-\delta)})$ time algorithm that can compute the exact number of solutions to input CNF formula; this is a tight lower bound.

- Case 2 is when $\gamma|\phi| > 1$: in this regime, the algorithm solving γ-#CNFSAT is expected to return a value d which is a *γ-approximate relative count* of the number of solutions to the CNFSAT problem.

In order to understand the hardness of γ-#CNFSAT in the second case, we will first try to understand how hard it is to compute the following property. Let $f_{\ell,\ell'} : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subset \{0,1\}^N$ be a partial function defined as follows

$$f_{\ell,\ell'}(x) = \begin{cases} 1, \text{ if } |x| = \ell, \\ 0, \text{ if } |x| = \ell'. \end{cases}$$

Nayak and Wu in [NW99] analyzed the approximate degree of $f_{\ell,\ell'}$. By using the polynomial method [BBC+01] again we have a lower bound on the quantum query complexity of $f_{\ell,\ell'}$ as mentioned in the following statement.

Lemma 8.17. *[NW99, Corollary 1.2] Let $\ell, \ell' \in \mathbb{N}$ be such that $\ell \neq \ell'$, $f_{\ell,\ell'} : \mathcal{D} \to \{0,1\}$ where $\mathcal{D} \subset \{0,1\}^N$, and*

$$f_{\ell,\ell'}(x) = \begin{cases} 1, \text{ if } |x| = \ell, \\ 0, \text{ if } |x| = \ell'. \end{cases}$$

Let $\Delta_\ell = |\ell - \ell'|$ and $p \in \{\ell, \ell'\}$ be such that $|\frac{N}{2} - p|$ is maximized. Then every bounded-error quantum algorithm that computes $f_{\ell,\ell'}$ uses $\Omega\left(\sqrt{\frac{N}{\Delta_\ell}} + \frac{\sqrt{p(N-p)}}{\Delta_\ell}\right)$ queries.

With the lower bound for $f_{\ell,\ell'}$, we conjecture the following:

Conjecture 8.18. Let $\ell \in [2^n] \cup \{0\}$ and $\ell' \in [2^n] \cup \{0\}$ be such that $\ell \neq \ell'$. Then at least one of the following is true:

- For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that computes $f_{\ell,\ell'}$ on the truth table of CNF formulas defined on n variables in $\mathcal{O}\left(\left(\sqrt{\frac{2^n}{\Delta_\ell}} + \frac{\sqrt{p(2^n-p)}}{\Delta_\ell}\right)^{1-\delta}\right)$ time (even restricted to formulas with $m \leq cn^2$ clauses);

- For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that computes $f_{N-\ell,N-\ell'}$ on the truth table of CNF formulas defined on n variables in $\mathcal{O}\left(\left(\sqrt{\frac{2^n}{\Delta_\ell}} + \frac{\sqrt{p(2^n-p)}}{\Delta_\ell}\right)^{1-\delta}\right)$ time (even restricted to formulas with $m \leq cn^2$ clauses);

here $\Delta_\ell = |\ell - \ell'|$ and $p \in \{\ell, \ell'\}$ such that $|2^{n-1} - p|$ is maximized. In particular, when $\ell + \ell' = 2^n$, the above immediately implies the following:

- For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that computes $f_{\ell,\ell'}$ on the truth table of CNF formulas defined on n variables in $\mathcal{O}\left(\left(\sqrt{\frac{2^n}{\Delta_\ell}} + \frac{\sqrt{\ell\ell'}}{\Delta_\ell}\right)^{1-\delta}\right)$ time (even restricted to formulas with $m \leq cn^2$ clauses).

Inspired by the arguments used in the proof of Theorem 1.13 in [NW99], we will now show that Conjecture 8.18 implies the following result. Our result holds for $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$; this range of γ suffices for our reductions presented in the later sections.

Corollary 8.19 (γ-#QSETH)**.** *Let $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$. For each constant $\delta > 0$, there exists $c > 0$ such that there is no bounded-error quantum algorithm that solves γ-#CNFSAT (even restricted to formulas with $m \leq cn^2$ clauses) in time*

1. $\mathcal{O}\left(\left(\frac{1}{\gamma}\sqrt{\frac{2^n - \hat{h}}{\hat{h}}}\right)^{1-\delta}\right)$ *if $\gamma\hat{h} > 1$, where $\hat{h}$ is the number of satisfying assignments;*
2. $\mathcal{O}(2^{n(1-\delta)})$ *otherwise,*

unless Conjecture 8.18 is false.

We show the first part of Corollary 8.19 in the following way and use the result from Conjecture 8.8 for the second part. Given a value of $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$ we will fix values of $\ell \in [2^n] \cup \{0\}$ and $\ell' \in [2^n] \cup \{0\}$ such that we are able to compute $f_{\ell,\ell'}$ on the truth table of an input CNF formulas on n variables using the algorithm that solves γ-#CNFSAT. Hence, we can show a lower bound on γ-#CNFSAT using the lower bound result from Conjecture 8.18.

Proof. Let $N = 2^n$. Let $\ell = \frac{N}{2} + \lceil \gamma t \rceil = \left\lceil \frac{N}{2} + \gamma t \right\rceil$ and $\ell' = \frac{N}{2} - \lceil \gamma t \rceil = \left\lfloor \frac{N}{2} - \gamma t \right\rfloor$; here $t \in [N]$ is a value that we will fix later but in any case, we have $1 \leq \lceil \gamma t \rceil < \frac{N}{2}$. With that, we are ensured that $\gamma\ell > \frac{1}{2}$. We also make sure to choose values ℓ, ℓ' in such a way that $\gamma\ell' = \Omega(1)$. Clearly, $\ell + \ell' = N$ and $\Delta_\ell = |\ell - \ell'| = 2\lceil \gamma t \rceil$. Therefore by invoking the result from Conjecture 8.18 we can say that for these values of ℓ, ℓ' there is no bounded-error quantum algorithm that can solve $f_{\ell,\ell'}$ on the truth table of CNF formulas in $O\left(\left(\sqrt{\frac{N}{\lceil \gamma t \rceil}} + \frac{\sqrt{\ell(N-\ell)}}{\lceil \gamma t \rceil}\right)^{1-\delta}\right)$ time, for each $\delta > 0$; let us denote this claim by (*).

Let $\mathcal{A}$ be an algorithm that computes γ-#CNFSAT, i.e., Algorithm $\mathcal{A}$ returns a value h such that $(1-\gamma)\hat{h} < h < (1+\gamma)\hat{h}$. Given $\ell = \frac{N}{2} + \lceil \gamma t \rceil$ and $\ell' = \frac{N}{2} - \lceil \gamma t \rceil$, there are values of $t \in [N]$ such that we will be able to distinguish whether the number of satisfying assignments to a formula is ℓ or ℓ' using Algorithm $\mathcal{A}$. As $\ell > \ell'$ in our setup, we want t such that $\ell'(1+\gamma) < \ell(1-\gamma)$; it is then necessary that $\gamma N < 2\lceil \gamma t \rceil$; let us denote this as Condition 1.

Now we set the values of ℓ and ℓ'. Given a value of $\gamma \in \left[\frac{1}{N}, \frac{1}{2}\right)$, we set $\ell = \left\lceil \frac{N}{2(1-\gamma)} \right\rceil$ and $\ell' = N - \ell$. This implies $\frac{N}{2(1-\gamma)} \leq \ell < \frac{N}{2(1-\gamma)} + 1$, $\frac{N(1-2\gamma)}{2(1-\gamma)} - 1 < \ell' \leq \frac{N(1-2\gamma)}{2(1-\gamma)}$, and $\frac{\gamma N}{(1-\gamma)} \leq |\ell - \ell'| < \frac{\gamma N}{(1-\gamma)} + 2$. Therefore we obtain $2\gamma\ell - 2\gamma \leq |\ell - \ell'| < 2\gamma\ell + 2$.[4] We know from claim (*) that every quantum algorithm that (for these values of ℓ, ℓ') computes $f_{\ell,\ell'}$ on CNF formulas requires $\Omega(L^{1-\delta})$ time for each $\delta > 0$, where $L = \frac{1}{\gamma}\sqrt{\frac{N-\ell}{\ell+1}} = \Omega\left(\frac{1}{\gamma}\sqrt{\frac{N-\ell}{\ell}}\right)$. Moreover, ℓ' is $(\ell-1)(1-2\gamma) - 1 < \ell' \leq \ell(1-2\gamma)$. Therefore, we can see that $L = \Omega\left(\frac{1}{\gamma}\sqrt{\frac{N-\ell}{\ell}}\right) = \Omega\left(\frac{1-2\gamma}{\gamma}\sqrt{\frac{N-\ell'}{\ell'}}\right) = \Omega\left(\frac{1}{\gamma}\sqrt{\frac{N-\ell'}{\ell'}}\right)$.

It is also easy to see that if $\ell = \lceil \frac{N}{2(1-\gamma)} \rceil$ were to be expressed as $\frac{N}{2} + \lceil \gamma t \rceil$ (i.e. denote ℓ to be $\frac{N}{2} + \lceil \gamma t \rceil$), then for that value of t we have $\lceil \gamma t \rceil = \lceil \frac{N}{2(1-\gamma)} \rceil - \frac{N}{2} \geq \frac{N\gamma}{2(1-\gamma)} > \frac{N\gamma}{2}$, which satisfies Condition 1. Hence here we can use Algorithm $\mathcal{A}$ to distinguish whether the number of satisfying assignments to a formula is ℓ or ℓ'. Hence given a CNF formula as input, we will be able to use Algorithm $\mathcal{A}$ to distinguish whether the number of satisfying assignments is ℓ or ℓ'. Let $T = \frac{1}{\gamma}\sqrt{\frac{N-\ell}{\ell}} + \frac{1}{\gamma}\sqrt{\frac{N-\ell'}{\ell'}} = \mathcal{O}(\frac{1}{\gamma}\sqrt{\frac{N-\ell'}{\ell'}})$. If for some constant $\delta > 0$, $\mathcal{A}$ can solve γ-#CNFSAT on an input CNF formula that has $\hat{h}$ number of satisfying assignments in $\mathcal{O}((\frac{1}{\gamma}\sqrt{\frac{N-\hat{h}}{\hat{h}}})^{1-\delta})$ time, then we are essentially computing $f_{\ell,\ell'}$ in $\mathcal{O}(T^{1-\delta})$ time, which is a contradiction to claim (*). Hence the first part of the statement of Corollary 8.19 proved.

Proof of the second part of this theorem follows from Conjecture 8.8 as the regime $\gamma\hat{h} \leq 1$ translates to exactly counting the number of satisfying assignments. □

8.2.2 Quantum complexity of #k-SAT and other related problems

In the previous subsection, we discussed the quantum complexity of variants of CNF-SAT problems. However, it is not clear how to immediately derive a similar quantum complexity result for variants of k-SAT problems with constant k by using the quantum (conditional) hardness results for variants of CNFSAT problems. Of course we could make a further conjecture about variants of k-SAT problems like we did in the previous subsection, but it would introduce too many conjectures. Moreover, some variants of k-SAT (for constant k) are even shown to be solvable in polynomial time [AW21].

To give the (quantum) complexity of some optimization problems (for example, lattice problems [BGS17]), on the other hand, we might want to have some (quantum) conditional lower bounds for (variants of) k-SAT problems with not too large k. This is because we might make $2^k \cdot \text{poly}(n)$ calls to a solver of those problems to solve k-SAT. This is undesirable for giving the (quantum) complexity of those optimization problems when k approaches n, while it is tolerable for a relatively small k (like $k = \text{polylog}\, n$). Hence in this subsection, we would like to say something interesting about quantum hardness for #k-SAT and $\oplus k$-SAT when $k = \Theta(\log n)$, only using the

[4] To view the calculations in a less cumbersome way one can use the fact that asymptotically $\ell = \frac{N}{2(1-\gamma)}$, $\ell' = \frac{N(1-2\gamma)}{2(1-\gamma)}$ and $|\ell - \ell'| = \frac{\gamma N}{(1-\gamma)} = 2\gamma\ell$.

hardness assumptions on counting-CNFSAT (that is, #QSETH). Here, variants of k-SAT are defined exactly the same way as Definition 8.5, Definition 8.12, and Definition 8.16, except that the input is now a k-CNF formula.

We use the classical algorithm by Schuler [Sch05].[5] This algorithm can be viewed as a Turing reduction from SAT with bounded clause density to SAT with bounded clause width, which was analyzed in [CIP06]. The time complexity of this algorithm is upper bounded by $\binom{m+n/k}{n/k} \cdot \text{poly}(m,n)$, where m is number of clauses.

Input : CNF formula ψ
if ψ *has no clause of width* $> k$ **then**
  output ψ;
else
  let $C' = \{l_1, \ldots, l_{k'}\}$ be a clause of ψ of width $k' > k$;
  $C = \{l_1, \ldots, l_k\}$; `// the first k literals of C'`
  $\psi_0 \leftarrow (\psi - \{C'\}) \cup \{C\}$; `// replace C' by a shorter C`
  $\psi_1 \leftarrow \psi \wedge \neg l_1 \wedge \neg l_2 \wedge \cdots \wedge \neg l_k$;
  $\psi_1 \leftarrow$ Remove variables corresponding to literals $l_1, \ldots, l_k$ from ψ_1 by setting $l_1 = 0, \ldots, l_k = 0$
  ReduceWidth$_k(\psi_0)$; `// left branch, which sets C to be true`
  ReduceWidth$_k(\psi_1)$; `// right branch, which sets C to be false`
end

Algorithm 8: ReduceWidth$_k(\psi)$

Algorithm 8 takes as input a CNF formula of width greater than k, and then outputs a list of k-CNF formulas ψ_i where the solutions of the input formula is the union of the solutions of the output formulas, i.e., $\text{sol}(\psi) = \cup_i \text{sol}(\psi_i)$, where $\text{sol}(\phi)$ denotes the set of satisfying assignments to a formula ϕ. In fact, it is not hard to see that the count of the number of satisfying assignments also is preserved, i.e., $|\text{sol}(\psi)| = \sum_i |\text{sol}(\psi_i)|$.

Lemma 8.20 (Implicit from Section 3.2 in [CIP06])**.** *Algorithm 8 takes as input a CNF formula ψ on n input variables, with m clauses, that is of width strictly greater than k and outputs a number of k-CNF formulas ψ_i each defined on at most n input variables and at most m clauses such that $|sol(\psi)| = \sum_i |sol(\psi_i)|$.*

Proof. Let $\psi = C'_1 \wedge C'_2 \wedge \cdots \wedge C'_m$ be the input CNF formula to Algorithm 8. The algorithm finds the first clause C'_i that has width $k' > k$. Let $C'_i = (l_1 \vee l_2 \vee \cdots \vee l_{k'})$ and $C_i = (l_1 \vee l_2 \vee \cdots \vee l_k)$. The algorithm then constructs two formulas $\psi_0 = (\psi - \{C'_i\}) \cup \{C_i\}$ and $\psi_1 = \psi \wedge \neg l_1 \wedge \neg l_2 \wedge \cdots \wedge \neg l_k$. Then the algorithm recursively calls the subroutine on ψ_0 and ψ_1. We now claim the following.

Claim 8.1. $\text{sol}(\psi_0) \cap \text{sol}(\psi_1) = \emptyset$ and $\text{sol}(\psi) = \text{sol}(\psi_0) \cup \text{sol}(\psi_1)$, i.e., $\text{sol}(\psi) = \text{sol}(\psi_0) \sqcup \text{sol}(\psi_1)$.

[5] This algorithm can also be used to solve CNFSAT on n variables, m clauses in $\mathcal{O}(\text{poly}(n) 2^{n(1-1/(1+\log m))})$ expected time.

Proof: Let $x \in \mathrm{sol}(\psi_0)$. Then the clause $C_i(x) = (l_1(x) \vee \cdots \vee l_k(x))$ should evaluate to 1. Equivalently, $\neg(l_1(x) \vee \cdots \vee l_k(x)) = 0$. Using De Morgan's laws we know $(\neg l_1(x) \wedge \cdots \wedge \neg l_k(x)) = 0$, which means that $\psi_1(x) = \psi(x) \wedge \neg l_1(x) \wedge \cdots \wedge \neg l_k(x) = 0$. A similar argument can be used to show that if $x \in \mathrm{sol}(\psi_1)$, then $x \notin \mathrm{sol}(\psi_0)$. Therefore, $\mathrm{sol}(\psi_0) \cap \mathrm{sol}(\psi_1) = \emptyset$.

What remains to show is $\mathrm{sol}(\psi) = \mathrm{sol}(\psi_0) \cup \mathrm{sol}(\psi_1)$.

- If $x \in \mathrm{sol}(\psi_0)$ then $C_i(x) = 1$, which implies $C'_i(x) = 1$. Therefore $x \in \mathrm{sol}(\psi)$. If $x \in \mathrm{sol}(\psi_1)$ then $\psi_1(x) = 1$, but $\psi_1(x) = \psi(x) \wedge \neg l_1(x) \wedge \neg l_2(x) \wedge \cdots \wedge \neg l_k(x)$ which means $\psi(x) = 1$ as well. Therefore, if $x \in \mathrm{sol}(\psi_0) \cup \mathrm{sol}(\psi_1)$ then $x \in \mathrm{sol}(\psi)$.

- If $x \in \mathrm{sol}(\psi)$ then $\psi(x) = 1$, which means $C'_i(x) = 1$. However, $C'_i(x) = (l_1(x) \vee \cdots \vee l_k(x)) \vee (l_{k+1}(x) \vee \cdots \vee l'_k(x))$. This means either $(l_1(x) \vee \cdots \vee l_k(x)) = C_i(x) = 1$ or $(l_{k+1}(x) \vee \cdots \vee l'_k(x)) = 1$ or both evaluate to 1. If $C_i(x) = 1$ then $\psi_0(x) = 1$, which means $x \in \mathrm{sol}(\psi_0)$. If $C_i(x) = 0$ then $\psi_1(x) = \psi(x) \wedge (\neg C_i(x)) = 1$, which means $x \in \mathrm{sol}(\psi_1)$.

Therefore, $\mathrm{sol}(\psi) = \mathrm{sol}(\psi_0) \sqcup \mathrm{sol}(\psi_1)$. ■

Using Claim 8.1 we conclude that $\mathrm{sol}(\psi) = \sqcup_i \mathrm{sol}(\psi_i)$, hence $|\mathrm{sol}(\psi)| = \sum_i |\mathrm{sol}(\psi_i)|$. □

Using Lemma 8.20 and Lemma 5 in [CIP06] we will now show the hardness of k-SAT and its counting variants when $k = \Theta(\log n)$ without introducing new conjectures.

Corollary 8.21. *For each constant $\delta > 0$, there exists a constant c such that there is no bounded-error quantum algorithm that solves*

1. *$c \log n$-SAT in $\mathcal{O}(2^{(1-\delta)n/2})$ time unless* BASIC-QSETH *(Conjecture 8.3) is false;*
2. *$\#c \log n$-SAT in $\mathcal{O}(2^{(1-\delta)n})$ time unless #QSETH (Conjecture 8.8) is false;*
3. *$\oplus c \log n$-SAT in $\mathcal{O}(2^{(1-\delta)n})$ time unless $\oplus$QSETH (Conjecture 8.9) is false;*
4. *$\oplus_q c \log n$-SAT in $\mathcal{O}(2^{(1-\delta)n})$ time unless $\#_q$QSETH (Conjecture 8.11) is false.*

Proof. We first prove the first item. Suppose that for each constant c, there is an algorithm $\mathcal{A}$ that solves $\#c \log n$-SAT in 2^{ns} for some constant $s := 1 - \delta < 1$. Let $k = c \log n$ for the rest of the proof. Consider the ReduceWidth$_k$ algorithm (Algorithm 8) with input CNF formula ψ. Let p be some path of length t in the tree T of recursive calls to ReduceWidth$_k(\psi)$. Let ψ_p be the output formula of width at most k at the leaf of p. Let l, r be the number of left, right branches respectively on path p. Every left branch in the path reduces the width of exactly 1 clause to k, therefore $l \leq m$. On the other hand, with additional poly(n, m) time, every right branch of path p reduces the number of variables by k, therefore $r \leq n/k$. As a result, the number of paths in tree T with r right branches is at most $\binom{m+r}{r}$ and each outputs a formula with $n - rk$ variables.

Using the same arguments as in [CIP06, Lemma 5], one can see that $\mathcal{A}$ together with the ReduceWidth$_k$ subroutine can be used to solve #CNFSAT (ignoring poly(n) factors) in time at most

$$\begin{aligned}
&\sum_{r=0}^{n/k}\binom{m+r}{r}2^{s(n-rk)}+\binom{m+n/k}{n/k}\cdot\mathrm{poly}(m,n)\\
\leq&\sum_{r=0}^{n/k}\binom{m+\frac{n}{k}}{r}2^{s(n-rk)}\\
=&2^{sn}\sum_{r=0}^{n/k}\binom{m+\frac{n}{k}}{r}\frac{1}{2^{srk}}\\
\leq&2^{sn}(1+\frac{1}{2^{sk}})^{m+\frac{n}{k}}\\
\leq&2^{sn}e^{\frac{1}{2^{sk}}(m+\frac{n}{k})} \qquad \text{since } (1+x)\leq e^x\\
\leq&2^{sn+\frac{4m}{2^{sk}}},
\end{aligned}$$

where the last equality holds because we can assume that $m \geq \frac{n}{k}$ without loss of generality (by appending dummy clauses). Therefore, for each c', there exist a constant c for $k = c\log n$ and δ' such that if $m \leq c'n^2$, then $s+\frac{4m}{n2^{sk}} < 1-\delta'$. As a result, a 2^{ns}-time algorithm for #$c\log n$-SAT implies a $2^{n(1-\delta')}$-time algorithm for #CNFSAT (restricted to formulas with $m \leq c'n^2$), which would refute #-QSETH (Conjecture 8.8). This proves the first item of the corollary. The same arguments hold for k-SAT, $\oplus$k-SAT, and $\oplus_q$k-SAT as well. □

Note that, we *cannot* extend the same arguments for the MAJORITY or *strict*-MAJORITY or additive-error approximation of count because those properties are not count-preserving. However, these arguments do extend to the multiplicative-factor approximation of the count.

Corollary 8.22. *Let $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$. For each constant $\delta > 0$, there exists constant c such that, there is no bounded-error quantum algorithm that solves γ-#$c\log n$-SAT in time*

1. $\mathcal{O}\left(\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-\delta}\right)$ *if $\gamma\hat{h} > 1$, where $\hat{h}$ is the number of satisfying assignments;*
2. $\mathcal{O}(2^{n(1-\delta)})$ *otherwise,*

unless γ-#QSETH (Corollary 8.19, implied by Conjecture 8.18) is false.

8.3 Implications of QSETH

In Section 8.2, we introduced lots of variants of CNFSAT and conjectured (or gave) quantum conditional lower bounds for all of them. Now having somewhat understood the complexities of the above-mentioned variants of CNFSAT, we give conditional quantum time lower bounds for lattice problem, strong simulation, orthogonal

vectors, set cover, hitting set problem, and their variants in this section by using variants of QSETH (see the corresponding subsections for the precise definitions of those variants of problems and see Table 8.2 for the implications of QSETH).

QSETH-based lower bounds for the strong simulation As an application to the bounds that we get from the property variants of CNFSAT, we look at the strong simulation problem. It was already established by [CHM21; Van10] that strongly simulating a quantum circuit is a #P-hard problem. Here we give exact lower bounds for the same. Additionally, using the lower bounds of approximate counts of CNFSAT, we are able to shed light on how hard it is to quantumly solve the strong simulation problem with additive-error and multiplicative-factor approximation. See Section 8.3.1.

QSETH-based lower bounds for lattice problems The quantum $2^{\frac{n}{2}}$-time lower bound we present for CVP_p (for $p \notin 2\mathbb{Z}$) follows from a reduction from k-SAT to CVP_p by [BGS17; ABG+21] and from the hardness result of k-SAT we present. Though such a result would also trivially follow from the version of QSETH by Aaronson, Chia, Lin, Wang, and Zhang, we stress that our hardness result of k-SAT is based on basic-QSETH which is a weaker conjecture.[6] Additionally, we discuss the quantum conditional lower bound of the lattice counting problem (for non-even norm). We present a reduction, using a similar idea of [BGS17], from #k-SAT to the lattice counting problem, and we show a 2^n-time quantum lower bound for the latter when $k = \Theta(\log n)$. See Section 8.3.2.

QSETH-based lower bounds for orthogonal vectors, set cover, and hitting set problems Last but not least, we are also able to use the lower bounds for the property variants of CNF-SAT to give quantum conditional lower bounds for orthogonal vectors, hitting set problem and their respective variants. See Section 8.3.3.

8.3.1 Quantum time complexity for strong simulation of quantum circuits

We use the phrase *strong simulation problem* to mean strong simulation of quantum circuits which is defined as follows.[7]

Definition 8.23 (The strong simulation problem)**.** Let $p \in \mathbb{N}$. Given a quantum circuit C on n qubits and $x \in \{0,1\}^n$, the goal of *strong simulation* with p-bit precision is to output the value of $|\langle x|C|0^n\rangle| := 0.C_1C_2\ldots$ up to p-bit precision. That is, output $C_0.C_1\ldots C_{p-1}$.[8]

[6] If basic-QSETH framework by Buhrman-Patro-Speelman is false then QSETH by Aaronson-Chia-Lin-Wang-Zhang is also false, but the implication in the other direction is not obvious.

[7] Note that this is different from the *weak simulation* problem; a weak simulation *samples* from probability distribution $p(x) := |\langle 0^n|C|x\rangle|^2$.

[8] Though in some papers the strong simulation problem requires that we output $\langle x|C|0^n\rangle$ instead of $|\langle x|C|0^n\rangle|$, we use this definition because it is more comparable to the definition of the weak simulation problem. Also, the lower bound we present holds for both of these definitions.

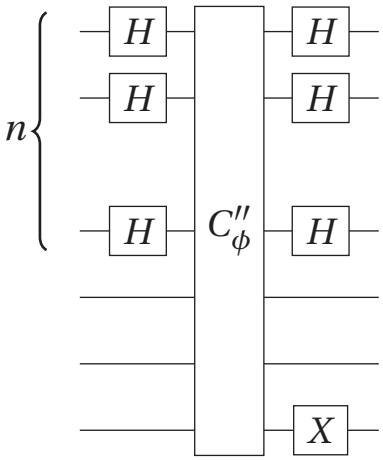

Figure 8.1: The circuit C_ϕ.

For a quantum circuit C, computing $|\langle x|C|0^n\rangle|$ up to a precision of $n+1$ bits, is #P-hard [CHM21; Van10]. This means even a *quantum* computer will likely require exponential time to solve the strong simulation problem. In this subsection, we provide a more *precise* quantum time lower bound for strongly simulating quantum circuits, both exactly and approximately. In the approximate case, we present complexity results for both multiplicative-factor and additive-error approximation. Our results extend the results by [HNS20] in two directions: firstly, we give explicit (conditional) bounds showing that, it is hard to strongly simulate quantum circuits using *quantum* computers as well. Secondly, we also address the open question posed by [HNS20] on the (conditional) hardness of strong simulation with additive error $\Theta(2^{-n/2})$. Our results are, however, based on a hardness assumption on Δ-ADD-#CNFSAT, rather than SETH or Basic-QSETH.

The results presented in this section are based on two main components. Firstly, on the observation that the reduction from CNFSAT to the strong simulation problem given by [HNS20, Theorem 3] encodes the count of the number of satisfying assignments. This fact allows us to use the same reduction to reduce other variants of CNFSAT, such as #CNFSAT or $\oplus$CNFSAT, to the strong simulation problem. Moreover, the same reduction also allows us to reduce γ-#CNFSAT and Δ-ADD-#CNFSAT to analogous variants of the strong simulation problem, respectively. As the second main component, we use the quantum hardness of these variants of CNFSAT problem conjectured in Section 8.2.

We will first state the result of the exact quantum time complexity of the strong simulation problem, and then use that result to show how hard it is for a quantum computer to strongly simulate a quantum circuit with additive-error or multiplicative-factor approximation.

Theorem 8.24. *For each constant $\delta > 0$, there is no bounded-error quantum algorithm*

that solves the strong simulation problem (Definition 8.23) up to a precision of $n+1$ bits in $\mathcal{O}(2^{n(1-\delta)})$ time, unless #QSETH (Conjecture 8.8) is false.

The proof is similar to the proof of [HNS20, Theorem 3]. We restate it here for ease of reading.

Proof. Let ϕ be a CNF formula on n input variables and m clauses. Let C'_ϕ denote the *reversible* classical circuit, using TOFFOLI, CNOT and X gates, that *tidily* computes $\phi(x)$ for all $x \in \{0,1\}^n$.[9] One can construct circuit C'_ϕ of width k with $n \le k \le n + 2(\lceil \log n \rceil + \lceil \log m \rceil)$ and size $s \le 8 \times 3^{\lceil \log n \rceil + \lceil \log m \rceil}$ with sublinear space and polynomial-time overhead; see [HNS20, Section 4.1]. Let C''_ϕ denote the quantum analogue of the classical reversible circuit C'_ϕ, i.e., for the gates in C'_ϕ, TOFFOLI and CNOT remain unchanged. Therefore, the width and size of C''_ϕ remains k and s, respectively.

Clearly, C''_ϕ maps $\mathcal{H}$ to $\mathcal{H}$ where $\mathcal{H}$ denotes a 2^k-dimensional Hilbert space. Then, we see that $C''_\phi |x\rangle |0^{k-n}\rangle = |x\rangle |\phi(x)\rangle |0^{k-1-n}\rangle$. Let C_ϕ denote the quantum circuit in Figure 8.1. Width of this circuit is still k and size is $\mathcal{O}(s)$. One can see that $\langle 0^k | C_\phi | 0^k \rangle$ encodes the fraction of satisfying assignments to formula ϕ (that is, $|\phi|/2^n$).

If there exists a constant $\delta > 0$ such that strong quantum simulation of circuit C_ϕ on basis state $|x\rangle = |0^k\rangle$ can be computed in $T = 2^{k(1-\delta)}$ time up to $n+1$ bits of precision, then this could count the number of satisfying assignments formula of ϕ in time T exactly. Plugging in the values of $k \le n + 2(\lceil \log n \rceil + \lceil \log m \rceil)$ and $s \le 8 \times 3^{\lceil \log n \rceil + \lceil \log m \rceil} = \text{poly}(n,m)$ we get $T \le \mathcal{O}(2^{n(1-\delta)}) \cdot \text{poly}(m)$ time. This would refute #QSETH (Conjecture 8.8). □

Note that even if we only care about the first two bits $C_0.C_1$ of $|\langle x| C |0^n\rangle| = C_0.C_1C_2\ldots$, it is still hard to determine which values C_0 and C_1 are, because it means we determine if the number of satisfying assignments is $\ge 2^n/2$ or not (if $C_0C_1 = 10$ or 01, then $|\phi| \ge 2^n/2$, and if $C_0C_1 = 00$, then $|\phi| < 2^n/2$). Therefore, using exactly the same statement in the proof above, we obtain the following corollary that gives the same lower bound for 2-bit precision, by using a different hardness assumption (which is Majority-QSETH).

Corollary 8.25. *For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves the strong simulation problem (Definition 8.23) up to 2 bits of precision in $\mathcal{O}(2^{n(1-\delta)})$ time, unless Majority-QSETH (Part 1 of Conjecture 8.10) is false.*

[9] A classical circuit $C : \{0,1\}^{n+w(n)+1} \to \{0,1\}^{n+w(n)+1}$ *reversibly and tidily computes* a function $f : \{0,1\}^n \to \{0,1\}$ if the following statements are true.

1. Circuit C *reversibly* computes f if C consists of reversible gates, such as {TOFFOLI, CNOT, NOT}, and
$$\forall x \in \{0,1\}^n, \exists W(x) \in \{0,1\}^{w(n)}, C(x, 0^{w(n)}, b) = (x, W(x), b \oplus f(x)). \tag{8.2}$$
2. Circuit C *tidily* computes f if
$$\forall x \in \{0,1\}^n, \forall b \in \{0,1\}, C(x, 0^{w(n)}, b) = (x, 0^{w(n)}, b \oplus f(x)). \tag{8.3}$$

One can also try to solve the strong simulation problem with additive-error approximation using the following definition.

Definition 8.26. (Strong simulation with additive error Δ') Let $\Delta' \in [\frac{1}{2^{n+1}}, 1)$. Given a quantum circuit C on n qubits and $x \in \{0,1\}^n$, the goal of strong simulation with additive error Δ' is to estimate $d' = |\langle 0^n | C | x \rangle|$ with additive error Δ'.

Using the same reduction in the proof of Theorem 8.24 and the conjectured hardness of the Δ-ADD-#CNFSAT problem we can immediately get the following corollary.[10]

Corollary 8.27. *For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves strong simulation with additive error $\Delta' = \frac{\Delta}{2^n} \in [\frac{1}{2^n}, 1)$ in $\widetilde{\mathcal{O}}\left(\left(\sqrt{\frac{2^n}{\Delta}} + \frac{\sqrt{\hat{h}(2^n - \hat{h})}}{\Delta}\right)^{1-\delta}\right)$ time where $\hat{h} = \langle 0^n | C | x \rangle \cdot 2^n$, unless Δ-ADD-#QSETH (Conjecture 8.15) is false.*

It is beneficial to note that we only get (at best) a poly(n)-time quantum lower bound for the strong simulation problem when $\Delta' = \frac{\Delta}{2^n} = \Theta(1)$. Fortunately, this lower bound matches the poly(n) time quantum upper bound for the strong simulation problem when $\Delta' = \Theta(1)$, see the end of this subsection and Theorem 8.30 for details. In fact, for some values of $\hat{h}$ our lower bounds are also tight in terms of Δ'. Additionally, we can use a similar argument to show strong simulation results with multiplicative factor, defined as follows.

Definition 8.28. (Strong simulation with multiplicative factor γ) Let $\gamma > 0$. Given a quantum circuit C on n qubits and $x \in \{0,1\}^n$, the goal of strong simulation with multiplicative factor γ is to estimate the value $d' = |\langle 0^n | C | x \rangle|$ with multiplicative error γ, i.e., output a value d such that $(1-\gamma)d' < d < (1+\gamma)d'$.

The exact arguments in the proof of Theorem 8.24 can be used to prove the following statement.

Theorem 8.29. *Let $\gamma \in [\frac{1}{2^n}, 0.4999)$. For each constant $\delta > 0$, there is no bounded-error quantum algorithm that can solve the strong simulation problem with multiplicative error γ in time*

1. $\mathcal{O}\left(\left(\frac{1}{\gamma}\sqrt{\frac{2^n - \hat{h}}{\hat{h}}}\right)^{1-\delta}\right)$ *if $\gamma\hat{h} \geq 1$, where $\hat{h} = \langle 0^n | C | x \rangle \cdot 2^n$ for input $x \in \{0,1\}^n$;*
2. $\mathcal{O}(2^{n(1-\delta)})$, *otherwise,*

unless γ-#QSETH (Corollary 8.19, implied by Conjecture 8.18) is false.

[10] Note that the value of k in relation to n is such that $2^k = \widetilde{\mathcal{O}}(2^n)$; here k refers to the k used in the proof of Theorem 8.24 instead of the k of k-SAT.

Proof. Let $\mathcal{A}$ denote an algorithm that for a given s-sized quantum circuit C on k qubits and a given $x \in \{0,1\}^k$, for some constants δ and δ', computes $\langle 0^k | C | x \rangle$ with multiplicative error γ either

1. in $\mathcal{O}\left(\left(\frac{1}{\gamma}\sqrt{\frac{2^k-\hat{h}}{\hat{h}}}\right)^{1-\delta}\right)$ time, whenever $\gamma\hat{h} \geq 1$, or
2. in $\mathcal{O}(2^{k(1-\delta')})$ time, otherwise.

Then, given a CNF formula ϕ on n input variables and m clauses, one can do the following to approximately count the number of satisfying assignments to ϕ: first use the poly(n,m) reduction as in the proof of Theorem 8.24 to construct the quantum circuit C_ϕ of size $s = \text{poly}(n,m)$ and width $k \leq n + 2(\lceil \log n \rceil + \lceil \log m \rceil)$. Then run algorithm $\mathcal{A}$ on quantum circuit C_ϕ and $x = 0^k$ as inputs. The output would then be a γ-multiplicative approximation of the count of the number of satisfying assignments to ϕ. Depending on the value of $\gamma \cdot \hat{h}$, the running time of this entire process is either $\mathcal{O}\left(\text{poly}(n,m) + \left(\frac{1}{\gamma}\sqrt{\frac{2^k-\hat{h}}{\hat{h}}}\right)^{1-\delta}\right) = \mathcal{O}\left(\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-\delta} \cdot \text{poly}(m)\right)$ for some constant $\delta > 0$, or $\mathcal{O}(2^{k(1-\delta')}) = \mathcal{O}(2^{n(1-\delta')} \cdot \text{poly}(m))$ time for some constant $\delta' > 0$. Either way refutes γ-#QSETH (Corollary 8.19). □

Quantum upper bounds for strong simulation

Here, we include a quantum algorithm for strong simulation with additive error Δ' for completeness.

Theorem 8.30. *Let $\Delta' \in [\frac{1}{2^{n+1}}, 1)$. There exists a quantum algorithm that solves strong simulation with additive error Δ' (Definition 8.26) in poly$(n, |C|) \cdot \frac{1}{\Delta'}$ time, where $|C|$ is the size (the number of quantum elementary gates it contains) of input circuit C.*

Proof. Given a quantum circuit C on n input variables and $x \in \{0,1\}^n$, the task (of strong simulation with additive error Δ') is to estimate the value of $|\langle x | C | 0^n \rangle|$ with $\Delta' \in [\frac{1}{2^{n+1}}, 1)$ additive-error approximation.

Let $|\psi\rangle = C|0^n\rangle := \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle$. Let U' denote the unitary $U' : |i\rangle|x\rangle|b\rangle \to |i\rangle|x\rangle|b \oplus (i = x)\rangle$ for $i, x \in \{0,1\}^n$ and $b \in \{0,1\}$. It is easy to verify that combining C and U' we can construct a unitary U on $2n+1$ qubits such that

$$U|0^{2n}\rangle|0\rangle = \alpha_x |x\rangle|x\rangle|1\rangle + \sum_{i \neq x} \alpha_i |i\rangle|x\rangle|0\rangle,$$

and $|\alpha_x| = |\langle x | C | 0^n \rangle|$. Using the amplitude estimation algorithm (Theorem 2.3), we can estimate $|\langle x | C | 0^n \rangle|$ to an additive error of Δ' in poly$(n, |C|) \cdot \frac{1}{\Delta'}$ time. □

8.3.2 Quantum time complexity for lattice counting and q-count problems

In this subsection, we will connect k-SAT to variants of lattice problems. Then, by using the QSETH lower bound we have in Section 8.2.2, we will give quantum conditional lower bounds for those lattice problems. For the definition of a lattice and lattice-related theorems, see Section 2.9.[11]

Conditional lower bounds for lattice problems are popular and widely studied in the classical case [BGS17; ABG+21; AC21; AS18; BP20; BPT22]. Lots of variants of lattice problems have been considered before, and the most well-studied one is the closest vector problem CVP_p (with respect to ℓ_p norm, see Definition 2.41). CVP_p is known to have a 2^n-SETH lower bound for any $p \notin 2\mathbb{Z}$ [BGS17; ABG+21], and for even p, there seems to be a barrier for showing a polynomial-time reduction from k-SAT to CVP [AK23]. Kannan gave an $n^{\mathcal{O}(n)}$-time algorithm for solving CVP_p for arbitrary $p \geq 1$ [Kan83], while the best known algorithm for solving CVP_p with noneven p is still $n^{\mathcal{O}(n)}$-time. To get a conditional quantum lower bound for CVP_p for noneven p, given there is already a classical reduction from k-SAT to CVP_p using $2^k \cdot \mathrm{poly}(n)$ time (for noneven p) [BGS17; ABG+21], either one can directly use the QSETH framework by [ACL+20] to get a $2^{0.5n}$ lower bound, or we can use Corollary 8.21 to get the same lower bound in our QSETH framework.

A natural question arises here: Can we have a $2^{(0.5+\delta)n}$-QSETH lower bound for any (variants of) lattice problems? The answer is yes by using the QSETH framework and the problems introduced in Section 8.2.2 (and by considering the counting variant of lattice problems.) We begin with introducing the (approximate) lattice counting problem and some other related problems as follows:

Definition 8.31 (Lattice counting problem)**.** Let $\gamma \geq 0$ and $1 \leq p \leq \infty$. The γ-approximate Vector Counting Problem γ-VCP_p is the counting problem defined as follows: The input is a basis $\mathbf{B} \in \mathbb{R}^{d \times n}$ of a lattice $\mathcal{L}(\mathbf{B})$, target vector $\boldsymbol{t} \in \mathbb{R}^d$, and radius $r \in \mathbb{R}_+$. The goal of this problem is to output a value C satisfying $|(\mathcal{L} - \boldsymbol{t}) \cap r \cdot B_p^d| \leq C \leq (1+\gamma) \cdot |(\mathcal{L} - \boldsymbol{t}) \cap r \cdot B_p^d|$. If $\gamma = 0$, we simply denote the problem as VCP_p.

The (approximate) lattice counting problem was first introduced by Stephens-Davidowitz as a promise problem [Ste16], and here we slightly modify the definition to make it a counting problem. We also generalize the lattice vector counting problem to the q-count problem, as follows.

Definition 8.32 (Lattice q-count problem)**.** Let $\gamma \geq 1$, $1 \leq p \leq \infty$, and $q \in [2^n] \setminus \{1\}$. The lattice q-count Problem $\#_q$-VCP_p is the lattice q-count problem defined as follows: The input is a basis $\mathbf{B} \in \mathbb{R}^{d \times n}$ of a lattice $\mathcal{L}(\mathbf{B})$, target vector $\boldsymbol{t} \in \mathbb{R}^d$, and radius $r \in \mathbb{R}_+$. The goal of this problem is to output a value $C = |(\mathcal{L} - \boldsymbol{t}) \cap r \cdot B_p^d| \mod q$. If $q = 2$, then we simply denote the problem as $\oplus\mathrm{VCP}_p$.

[11] Note that here lattice problems are defined over the non-Euclidean norm, so unlike Euclidean norm case (see Section 2.9.2), we do not know how to reduce a non-full-rank lattice to a full-rank lattice.

One can consider the above two problems as the counting and the q-counting version of CVP_p, respectively. To connect these problems to the counting k-SAT problem, we should first introduce the following geometric tool introduced by Bennett, Golovnev, and Stephens-Davidowitz [BGS17].

Definition 8.33 (Isolating parallelepiped)**.** Let k be an integer between 3 and n and $1 \le p \le \infty$. We say that $V \in \mathbb{R}^{2^k \times k}$ and $\boldsymbol{u} \in \mathbb{R}^{2^n}$ define a (p,k)-isolating parallelepiped if $\|\boldsymbol{u}\|_p > 1$ and $\|V\boldsymbol{x} - \boldsymbol{u}\|_p = 1$ for all $\boldsymbol{x} \in \{0,1\}^k \setminus \{0_k\}$.

For the sake of completeness, we will explain how to connect a k-CNF formula to an instance of VCP using the above object. The proof is very similar to the proof of [BGS17, Theorem 3.2].

Theorem 8.34. *Let k be an integer between 3 and n. Suppose we have a (p,k)-isolating parallelepiped $(V,\boldsymbol{u})$ for some $p = p(n) \in [1,\infty)$ and can make quantum queries to oracles $O_V : |i\rangle|s\rangle|0\rangle \to |i\rangle|s\rangle|V_{is}\rangle$ and $O_{\boldsymbol{u}} : |i\rangle|0\rangle \to |i\rangle|\boldsymbol{u}_i\rangle$ for $i \in [2^k]$ and $s \in [k]$. Then for every given input oracle $O_\Phi : |j\rangle|w\rangle|0\rangle \to |j\rangle|k\rangle|C_w(j)\rangle$ of k-CNF formula (with m clauses) $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ for $w \in [m]$ and $j \in [n]$, one can output oracles $O_{\mathbf{B}} : |h\rangle|j\rangle|0\rangle \to |h\rangle|j\rangle|\mathbf{B}_j(h)\rangle$ of basis $\mathbf{B} \in \mathbb{R}^{(m\cdot 2^k+n)\times n}$ for each $h \in [m\cdot 2^k+n]$ and $j \in [n]$, $O_{\boldsymbol{t}} : |h\rangle|0\rangle \to |h\rangle|t_h\rangle$ of target vector $\boldsymbol{t} \in \mathbb{R}^{m\cdot 2^k+n}$ for each $h \in [m\cdot 2^k+n]$, and radius r such that $|sol(\psi)| = VCP_p(\mathbf{B},\boldsymbol{t},r)$, using poly$(n,m)$ queries to O_V, $O_V^\dagger$, $O_{\boldsymbol{u}}$, $O_{\boldsymbol{u}}^\dagger$, O_Φ, $O_\Phi^\dagger$, and elementary gates.*

Proof. Let $d = m\cdot 2^k + n$ and $V = \{\boldsymbol{v_1},\ldots,\boldsymbol{v_k}\}$ with $\boldsymbol{v_s} \in \mathbb{R}^{2^k}$ for every $s \in [k]$. The basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ and target vector $\boldsymbol{t} \in \mathbb{R}^d$ in the output instance have the form:

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_m \\ 2\cdot m^{1/p}\cdot I_n \end{pmatrix}, \quad \boldsymbol{t} = \begin{pmatrix} \boldsymbol{t}_1 \\ \vdots \\ \boldsymbol{t}_m \\ m^{1/p}\cdot \mathbf{1}_n \end{pmatrix},$$

with blocks $\mathbf{B}_w \in \mathbb{R}^{2^k\times n}$ that correspond to the clause $C_w = \vee_{s=1}^k \ell_{w,s}$ and $\boldsymbol{t}_w \in \mathbb{R}^{2^n}$ for each $w \in [m]$. For each $w \in [m]$ and $j \in [n]$, the jth column $(\mathbf{B}_w)_j$ of block $\mathbf{B}_w$ is

$$(\mathbf{B}_w)_j = \begin{cases} \boldsymbol{v_w}, \text{ if } x_j \text{ is the } w\text{th literal of clause } w, \\ -\boldsymbol{v_w}, \text{ if } \neg x_j \text{ is the } w\text{th literal of clause } w, \\ 0_{2^d}, \text{ otherwise,} \end{cases}$$

and $\boldsymbol{t}_w = \boldsymbol{u} - \sum_{s\in N_w} \boldsymbol{v}_s$, where $N_w = \{s \in [k] : \ell_{w,s} \text{ is negative}\}$ is the set of the indices of negative literals in C_w. Also set $r = (mn+m)^{1/p}$.

Define the unitary $U_e : |a\rangle|b\rangle|0\rangle \to |a\rangle|b\rangle|\delta_{|a|,|b|}\rangle$ for each $a,b \in [d]$, and the unitary as for each $x_1,\ldots,x_n \in \{0,1\}$,

$$U_{loc} : |x_1\rangle|x_2\rangle\ldots|x_n\rangle|0\rangle \to \begin{cases} |x_1\rangle|x_2\rangle\ldots|x_n\rangle|s\rangle, & \text{if only } x_s = 1 \text{ and all others are 0,} \\ |x_1\rangle|x_2\rangle\ldots|x_n\rangle|0\rangle, & \text{otherwise,} \end{cases}$$

which can both be implemented via poly(n) elementary gates up to negligible error. Also, we define the unitary as follows: for each $w \in [m]$, $j \in [n]$,

$$U_{s\ell} : |w\rangle|j\rangle|0\rangle \to \begin{cases} |w\rangle|j\rangle|s\rangle, & \text{if } x_j \text{ is the } s\text{th literal of clause } C_w, \\ |w\rangle|j\rangle|-s\rangle, & \text{if } \neg x_j \text{ is the } s\text{th literal of clause } C_w, \\ |w\rangle|j\rangle|0\rangle, & \text{otherwise,} \end{cases}$$

which can also be implemented by using poly(m,n) applications of O_Φ and poly(m,n) many elementary gates.

We are now ready to show how to implement $O_{\mathbf{B}}$ and $O_{\boldsymbol{t}}$. For input $|h\rangle|j\rangle|0\rangle|0\rangle|0\rangle|0\rangle$, let $|h\rangle = |h_1\rangle|h_2\rangle$ where h_2 is the last n bits of h and h_1 is the remaining prefix. Then we first apply U_e to the first, third, and fifth registers to obtain $|h_1\rangle|h_2\rangle|j\rangle|0\rangle|\delta_{h_1,0}\rangle|0\rangle|0\rangle$. After that, apply $U_{s\ell}$ to the first, third, and sixth registers, and apply O_V to the second, third, and seventh registers we get

$$|h_1\rangle|h_2\rangle|j\rangle|0\rangle|\delta_{h_1,0}\rangle|s\rangle|V_{js}\rangle,$$

where $V_{js} = (\boldsymbol{v}_s)_j$. Finally, adding another ancilla register, we can store the value $V_{js} \cdot \delta_{h_1,0} + 2m^{1/p}(1-\delta_{h_1,0})\delta_{h-m\cdot 2^k,j}$ in the last register. Uncomputing the fourth to seventh registers, we have

$$|h_1\rangle|h_2\rangle|j\rangle|V_{js}\cdot\delta_{h_1,0} + 2m^{1/p}(1-\delta_{h_1,0})\delta_{h-m\cdot 2^k,j}\rangle = |h\rangle|j\rangle|(\boldsymbol{v}_s)_j\cdot\delta_{h_1,0} + 2m^{1/p}(1-\delta_{h_1,0})\delta_{h-m\cdot 2^k,j}\rangle,$$

and $(\boldsymbol{v}_s)_j\cdot\delta_{h_1,0}+2m^{1/p}(1-\delta_{h_1,0})\delta_{h-m\cdot 2^k,j}$ is exactly the coefficient of $\mathbf{B}_j(h)$. One can see we only use poly(n,m) queries to O_V, $O_V^\dagger$, O_Φ, $O_\Phi^\dagger$ and a similar number of elementary gates. We can also construct $O_{\boldsymbol{t}}$ using a similar strategy, which can also be done using at most poly(n,m) queries to O_V, $O_V^\dagger$, O_Φ, $O_\Phi^\dagger$, O_u, $O_u^\dagger$, and elementary gates.

To see the correctness, consider $\boldsymbol{y} \in \mathbb{Z}^n$. If $\boldsymbol{y} \notin \{0,1\}^n$, then

$$\|\mathbf{B}\boldsymbol{y} - \boldsymbol{t}\|_p^p \geq \|2m^{1/p} I_n \boldsymbol{y} - m^{1/p}\mathbf{1}_n\|_p^p \geq m(n+2).$$

On the other hand, if $\boldsymbol{y} \in \{0,1\}^n$, then for each $\mathbf{B}_w$

$$\begin{aligned} \|\mathbf{B}_w\boldsymbol{y} - \boldsymbol{t}\|_p^p &= \|\sum_{s\in P_w} \boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})}\cdot\boldsymbol{v}_s - \sum_{s\in N_w}\boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})}\cdot\boldsymbol{v}_s - (u - \sum_{s\in N_w}\boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})}\cdot\boldsymbol{v}_s)\|_p^p \\ &= \|\sum_{s\in P_w}\boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})}\cdot\boldsymbol{v}_s + \sum_{s\in N_w}(1-\boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})})\cdot\boldsymbol{v}_s - u\|_p^p \\ &= \|\sum_{s\in S_w(\boldsymbol{y})}\boldsymbol{v}_s - u\|_p^p, \end{aligned}$$

where $\mathrm{ind}(\ell_{w,s})$ is the index of the variable underlying $\ell_{w,s}$ (that is, $\mathrm{ind}(\ell_{w,s}) = j$ if $\ell_{w,s} = x_j$ or $\neg x_j$), $P_w = \{s \in [k] : \ell_{w,s} \text{ is positive}\}$ is the set of the indices of positive literals in C_w, and $S_w(\boldsymbol{y}) = \{s \in P_w : \boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})} = 1\} \cup \{s \in N_w : \boldsymbol{y}_{\mathrm{ind}(\ell_{w,s})} = 0\}$ is the indices of literals in C_w satisfied by $\boldsymbol{y}$. Because $(V,\boldsymbol{u})$ is a (p,k)-isolating parallelepiped, if $|S_w(\boldsymbol{y})| \neq 0$, then $\|\sum_{s\in S_w(\boldsymbol{y})}\boldsymbol{v}_s - \boldsymbol{u}\|_p^p = 1$, and it will be greater than 1 otherwise. Also,

$|S_w(\boldsymbol{y})| \neq 0$ if and only if C_w is satisfied. Therefore, we have that for every satisfying assignment $\boldsymbol{y}$,

$$\|\mathbf{B}\boldsymbol{y} - \boldsymbol{t}\|_p^p = \sum_{k=1}^{m} \|\mathbf{B}_k\boldsymbol{y} - \boldsymbol{t}\|_p^p + mn = m + mn,$$

and for all other unsatisfying assignments $\boldsymbol{y}'$, $\|\mathbf{B}\boldsymbol{y}' - \boldsymbol{t}\|_p^p > m + mn$. As a result, every satisfying assignment of $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is encoded as a lattice point of lattice $\mathcal{L}(\mathbf{B})$ with distance $r = (mn+m)^{1/p}$ to the target vector t, which implies that the answer of counting-SAT with input ψ is equal to $\mathrm{VCP}_p(\mathbf{B}, \boldsymbol{t}, r)$. □

The theorem above shows how to connect the counting k-SAT problem to the vector counting problem given access to a (p,k)-isolating parallelepiped. However, it is not always the case that we can compute such an isolating parallelepiped efficiently. Aggarwal, Bennett, Golovnev, and Stephens-Davidowitz [BGS17; ABG+21] showed the existence of isolating parallelepiped for some p, k and provided an efficient algorithm for computing them.

Theorem 8.35 ([ABG+21])**.** *For $k \in \mathbb{Z}_+$ and computable $p = p(n) \in [1,\infty)$ if p satisfies either (1) $p \notin 2\mathbb{Z}$ or (2) $p \geq k$, the there exists a (p,k)-isolating parallelepiped $V \in \mathbb{R}^{2^k \times k}$, $\boldsymbol{u} \in \mathbb{R}^{2^k}$ and it is computable in time poly$(2^k, n)$.*

Therefore, by choosing $k = \Theta(\log n)$ and combining Corollary 8.21 and Theorems 8.34 and 8.35, we can directly show a 2^n-QSETH lower bound for VCP_p for all non-even p. Also, a similar idea works for the q-count and approximate counting of CNF-SAT: for each CNF-SAT formula ψ, using Algorithm 8 and Lemma 8.20, we can output a number of k-CNF formulas $\psi_1 \ldots, \psi_N$ such that $|\mathrm{sol}(\psi)| = \sum_{i\in[N]} |\mathrm{sol}(\psi_i)|$. Once we have an algorithm that solves γ-#k-SAT (#$_q k$-SAT), we can use it to compute γ-#k-SAT(ψ_i) (#$_q k$-SAT(ψ_i)) for all $i \in [N]$, and then by adding the outputs together, we can get a valid solution to γ-#CNFSAT (#$_q$CNFSAT) with input ψ. Combining the above arguments with Theorem 8.34 (and the proof of Corollary 8.21), we have the following corollaries.

Corollary 8.36. *Let $p \in [1,\infty) \setminus 2\mathbb{Z}$ and $q \in [2^n] \setminus \{1,2\}$. For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves*

1. *VCP$_p$ in $\mathcal{O}(2^{n(1-\delta)})$ time, unless #QSETH(Conjecture 8.8) is false;*
2. *$\oplus$VCP$_p$ in $\mathcal{O}(2^{n(1-\delta)})$ time, unless $\oplus$QSETH(Conjecture 8.9) is false;*
3. *#$_q$VCP$_p$ in $\mathcal{O}(2^{n(1-\delta)})$ time, unless #$_q$QSETH(Conjecture 8.11) is false.*

Corollary 8.37. *Let $\gamma \in [\frac{1}{2^n}, 0.4999)$ and $p \in [1,\infty) \setminus 2\mathbb{Z}$. For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves γ-VCP$_p$ in time*

1. $\mathcal{O}\left(\left(\frac{1}{\gamma}\sqrt{\frac{2^n - \hat{h}}{\hat{h}}}\right)^{1-\delta}\right)$*, if $\gamma\hat{h} > 1$ where $\hat{h}$ is the number of the closest vectors,*
2. $\mathcal{O}(2^{(1-\delta)n})$*, otherwise,*

unless γ-#QSETH (Corollary 8.19, implied by Conjecture 8.18) is false.

The following theorem also shows how to connect the approximate vector counting problem to the closest vector problem. The classical reduction from approximate VCP_p to CVP_p was already shown in [Ste16, Theorem 3.5], while we can easily give a quadratic saving for the number of calls to CVP_p oracle. We include the proof at the end of this subsection for completeness.

Theorem 8.38. *Let $f(n) \geq 20$ be an efficiently computable function and $p \in [1,\infty)$. One can solve $f(n)^{-1}$-VCP_p using $\mathcal{O}(f(n)^2)$ quantum queries to CVP_p.*

To prove Theorem 8.38, we first introduce a gapped version of the lattice counting problem as follows:

Definition 8.39 (Gap-VCP)**.** Let $\gamma \geq 0$ and $1 \leq p \leq \infty$. The problem γ-approximate gap Vector Counting Problem γ-Gap-VCP_p is a promise problem defined as follows: The input is a basis $\mathbf{B} \in \mathbb{R}^{d\times n}$ of a lattice $\mathcal{L}(\mathbf{B})$, target vector $\boldsymbol{t} \in \mathbb{R}^d$, radius $r \in \mathbb{R}_+$, and $N \geq 1$. The goal of this problem is to output "No" if $|(\mathcal{L}-\boldsymbol{t}) \cap r \cdot B_p^d| \leq N$ and "Yes" if $N > (1+\gamma)\cdot|(\mathcal{L}-\boldsymbol{t}) \cap r \cdot B_p^d|$.

One can easily see that if we can solve γ-Gap-VCP_p, then by using $\mathrm{poly}(n)$ calls of it we can solve γ-VCP_p. As a result, it suffices to show how to reduce γ-Gap-VCP_p to CVP_p in the following proof.

Proof of Theorem 8.38. Choose a prime $Q = \Theta(f(n)\cdot N)$ and let O_{CVP} be the quantum CVP_p oracle. Define $U_{Spar} : |\mathbf{B},Q,\boldsymbol{z},\boldsymbol{c}\rangle|0\rangle|0\rangle \to |\mathbf{B},Q,\boldsymbol{z},\boldsymbol{c}\rangle|\mathbf{B}_{Q,\boldsymbol{z}}\rangle|\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}\rangle$, where $\mathbf{B}_{Q,\boldsymbol{z}}, \boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}$ are the output of $Spar(\mathbf{B},Q,\boldsymbol{z},\boldsymbol{c})$. Since given a CVP_p oracle and a basis $\mathbf{B}$, the sparsification process (Theorem 2.44) can be efficiently done according to the construction, we can also implement the unitary U_{Spar} efficiently.

First we prepare the superposition state $\frac{1}{Q^{n/2}}\sum_{\boldsymbol{z},\boldsymbol{c}\in\mathbb{Z}_Q^n}|\mathbf{B},Q,\boldsymbol{z},\boldsymbol{c}\rangle|0\rangle|\boldsymbol{t}\rangle|0\rangle|0\rangle$, apply U_{Spar} on the first six registers, and apply O_{CVP_p} on the fifth, sixth, seventh registers, and then apply the subtraction unitary $U_{sub} : |\boldsymbol{a}\rangle|\boldsymbol{b}\rangle|0\rangle \to |\boldsymbol{a}\rangle|\boldsymbol{b}\rangle|\|\boldsymbol{a}-\boldsymbol{b}\|_p\rangle$ on the last three registers, then we get

$$\frac{1}{Q^{n/2}}\sum_{\boldsymbol{z},\boldsymbol{c}\in\mathbb{Z}_Q^n}|\mathbf{B},Q,\boldsymbol{z},\boldsymbol{c}\rangle|\mathbf{B}_{Q,\boldsymbol{z}}\rangle|\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}\rangle|CVP_p(\mathbf{B}_{Q,\boldsymbol{z}},\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}})\rangle|\|\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}-CVP_p(\mathbf{B}_{Q,\boldsymbol{z}},\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}))\|_p\rangle\,.$$

Let $r_{\boldsymbol{z},\boldsymbol{c}} = \|\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}-CVP_p(\mathbf{B}_{Q,\boldsymbol{z}},\boldsymbol{t}+\boldsymbol{w}_{\boldsymbol{z},\boldsymbol{c}}))\|_p$ and umcompute the first seven registers, then we get

$$\frac{1}{Q^{n/2}}\sum_{\boldsymbol{z},\boldsymbol{c}\in\mathbb{Z}_Q^n}|r_{\boldsymbol{z},\boldsymbol{c}}\rangle\,.$$

Adding another ancilla $|0\rangle$ at the end of the above state, and then applying the r-threshold gate

$$U_r : |R\rangle|0\rangle \to \begin{cases}|R\rangle|1\rangle & \text{if } R \leq r\\ |R\rangle|0\rangle & \text{otherwise,}\end{cases}$$

on it, we get

$$\frac{1}{Q^{n/2}}\Big(\sum_{r_{\boldsymbol{z},\boldsymbol{c}}\le r}|r_{\boldsymbol{z},\boldsymbol{c}}\rangle|1\rangle+\sum_{r_{\boldsymbol{z},\boldsymbol{c}}>r}|r_{\boldsymbol{z},\boldsymbol{c}}\rangle|0\rangle\Big):=\sqrt{a}|\phi_1\rangle|1\rangle+\sqrt{1-a}|\phi_0\rangle|0\rangle,$$

where $a=|\{(\boldsymbol{z},\boldsymbol{c}):r_{\boldsymbol{z},\boldsymbol{c}}\le r\}|/Q^n=\Pr_{\boldsymbol{z},\boldsymbol{c},\in\mathbb{Z}_Q^n}[r_{\boldsymbol{z},\boldsymbol{c}}\le r]$. Note that by Theorem 2.44, if $|\mathcal{L}\cap(rB_p^n+\boldsymbol{t})|\le N$, then

$$\Pr_{\boldsymbol{z},\boldsymbol{c},\in\mathbb{Z}_Q^n}[r_{\boldsymbol{z},\boldsymbol{c}}\le r]\le\frac{N}{Q}+\frac{N}{Q^n},$$

and if $|\mathcal{L}\cap(rB_p^n+\boldsymbol{t})|\ge\gamma N$, then

$$\Pr_{\boldsymbol{z},\boldsymbol{c},\in\mathbb{Z}_Q^n}[r_{\boldsymbol{z},\boldsymbol{c}}\le r]\ge\frac{\gamma N}{Q}-\frac{\gamma^2N^2}{Q^2}-\frac{\gamma^2N^2}{Q^{n-1}}.$$

Observing that $\frac{\gamma N}{Q}-\frac{\gamma^2N^2}{Q^2}-\frac{\gamma^2N^2}{Q^{n-1}}-(\frac{N}{Q}+\frac{N}{Q^n})=\Theta(f(n)^{-1}N/Q)$, we know that to distinguish the above two cases, it suffices to learn $a=\Pr_{\boldsymbol{z},\boldsymbol{c},\in\mathbb{Z}_Q^n}[r_{\boldsymbol{z},\boldsymbol{c}}\le r]$ with additive error $\Theta(f(n)^{-1}N/Q)$. Therefore, by using Theorem 2.3, we can solve γ-Gap-VCP$_p$ with $\gamma=f(n)^{-1}$ using $\mathcal{O}(f(n)Q/N)$ queries to O_{CVP_p} and time. Because $Q=\Theta(f(n)N)$, we finish the proof. □

Note that the QSETH lower bound for $f(n)^{-1}$-VCP$_p$ depends on $f(n)$. Therefore if we can show a reduction from $f(n)^{-1}$-VCP$_p$ to CVP$_p$ using $f(n)^c$ for some constant $c<1$, then we will end up with a better QSETH lower bound for CVP$_p$.

8.3.3 Hardness of Counting/Parity of OV, Hitting Set, and Set-Cover

In this subsection, we will discuss the consequences of Corollary 8.19 and Corollary 8.21 for some well-motivated optimization problems: Orthogonal Vectors, Hitting Set and Set Cover. Following are the definitions of Hitting Set and its variants.

Definition 8.40 (Hitting Set)**.** Let integers $n,m>0$. The Hitting Set problem is defined as follows: The input is a collection of sets $\Sigma=(S_1,\ldots,S_m)$, where $S_i\subset V$ and integer $t>0$, the goal is to output a subset $S'\subset V$ such that $|S'|\le t$ and $\forall i\in[m],|S'\cap S_i|>0$. We call such S' a *hitting set* for Σ.

Definition 8.41 (Variants of Hitting Set)**.** Let integers $n,m>0$ and $\gamma\in\left[\frac{1}{2^n},\frac{1}{2}\right)$. We define the following four variants of Hitting Set. The input for all of them is a collection $\Sigma=(S_1,\ldots,S_m)$, where $S_i\subset V$ and integer $t>0$.

1. In the Count Hitting Set problem, the goal is to output d';
2. in the Parity Hitting Set problem, the goal is to output d' mod 2;
3. in the strict-Majority Hitting Set problem, the goal is to output 1 if $d'>2^{n-1}$, otherwise output 0;

4. in the γ-approximation of count Hitting Set, the goal is to output an integer d such that $(1-\gamma)d' < d < (1+\gamma)d'$;

where $d' = |S' \subset V : \left|S'\right| \leq t, \forall i \in [m], |S' \cap S_i| > 0\Big|$.

In [CDL+16], the authors showed a Parsimonious reduction between CNFSAT and Hitting Set. By Parsimonious reduction, we mean a transformation from a problem to another problem that preserves the number of solutions.

Theorem 8.42 (Theorem 3.4 in [CDL+16])**.** *For each constant $\delta > 0$, there exists a polynomial time Parsimonious reduction from CNFSAT on n variables to Hitting Set on $n(1+\delta)$ size universal set.*

By using the above theorem, we immediately get the following corollaries.

Corollary 8.43. *For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves*

1. *Hitting Set in $\mathcal{O}(2^{\frac{n}{2}(1-\delta)})$ time, unless* BASIC-QSETH *(Conjecture 8.3) is false.*
2. *Count Hitting Set in $\mathcal{O}(2^{n(1-\delta)})$ time, unless #QSETH(Conjecture 8.8) is false.*
3. *Parity Hitting Set in $\mathcal{O}(2^{n(1-\delta)})$ time, unless $\oplus$QSETH(Conjecture 8.9) is false.*
4. *Strict-Majority Hitting Set in $\mathcal{O}(2^{n(1-\delta)})$ time, unless Majority-QSETH(Conjecture 8.10) is false.*

Corollary 8.44. *Let $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$. For each constant $\delta > 0$, there is no bounded-error quantum that solves γ-multiplicative-factor approximation of count Hitting Set in time*

1. $\mathcal{O}\left(\frac{1}{\gamma}\sqrt{\frac{2^n-\hat{h}}{\hat{h}}}\right)^{1-\delta}$, *if $\gamma\hat{h} > 1$ where $\hat{h}$ is the number of hitting sets,*
2. $\mathcal{O}(2^{(1-\delta)n})$, *otherwise,*

unless γ-#QSETH (Corollary 8.19, implied by Conjecture 8.18) is false.

We also use our results from Section 8.2 to show conditional lower bounds for problems in the complexity class P. More specifically, we study Orthogonal Vectors problem and its variants defined as follows.

Definition 8.45 (Orthogonal Vectors (OV))**.** Let d, n be natural numbers. The Orthogonal Vectors problem is defined as follows: The input is two lists A and B, each consisting of n vectors from $\{0,1\}^d$. The goal is to find vectors $a \in A, b \in B$ for which $\langle a, b\rangle = 0$. We call such pair (a, b) a pair of *orthogonal vectors*.

Definition 8.46 (Variants of OV)**.** Let integers $d, n > 0$, $\gamma \in \left[\frac{1}{n^2}, 0.4999\right)$, and the input is two lists A and B each consisting of n vectors from $\{0,1\}^d$.

1. In the Count OV problem, the goal is to output d';
2. in the Parity OV problem, the goal is to output $d' \bmod 2$;
3. in the strict-Majority OV, the goal is to output 1 if $d' > n^2/2$, otherwise output 0;
4. in the γ-approximation of count OV, the goal is to output an integer d such that $(1-\gamma)d' < d < (1+\gamma)d'$;

where $d' = |(a,b) : a \in A, b \in B, \langle a,b\rangle = 0|$.

Orthogonal Vectors is an important computational problem that lies in the complexity class P. It turns out to be one of the central problems to show fine-grained hardness of problems in P [Vas15; ABD+18]. Williams showed a reduction from CNF-SAT to OV [Wil05]. We observe that Williams's reduction is Parsimonious. Therefore we can also show quantum conditional lower bounds for counting versions of OV using our QSETH conjectures. In [BPS21], the authors showed $\mathcal{O}(n)$-hardness for OV under basic-QSETH assumption, and here we give $\mathcal{O}(n^2)$-hardness for counting versions of OV, which might be useful for showing quantum conditional lower bounds for (variants of) other problems (like string problems or dynamic problems, see [Vas15, Figure 1]).

Corollary 8.47. *For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves*

1. *Count OV in $\mathcal{O}(n^{2-\delta})$ time, unless #QSETH (Conjecture 8.8) is false;*
2. *Parity OV in $\mathcal{O}(n^{2-\delta})$ time, unless $\oplus$QSETH (Conjecture 8.9) is false;*
3. *Majority OV in $\mathcal{O}(n^{2-\delta})$ time, unless Majority-QSETH (Conjecture 8.10) is false.*

Corollary 8.48. *Let $\gamma \in \left[\frac{1}{2^n}, 0.4999\right)$. For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves γ-multiplicative-factor approximation of count OV in time*

1. $\mathcal{O}\left(\frac{1}{\gamma}\sqrt{\frac{n^2-\hat{h}}{\hat{h}}}\right)^{1-\delta}$, *if $\gamma\hat{h} > 1$ where $\hat{h}$ is the number of pairs of orthogonal vectors,*
2. $\mathcal{O}(n^{(2-\delta)})$, *otherwise,*

unless γ-#QSETH (Corollary 8.19, implied by Conjecture 8.18) is false.

We can also give a quantum conditional lower bound for the parity Set-Cover problem defined as follows.

Definition 8.49 (parity Set-Cover)**.** For any integers $n, m > 0$, the parity Set-Cover problem is defined as follows: The input is a collection $\Sigma = (S_1, \ldots, S_m)$, where $S_i \subset V$ and integer $t > 0$, the goal is to output $|\{\mathcal{F} \subset \Sigma : \bigcup_{S\in\mathcal{F}} S = V, |\mathcal{F}| \le t\}| \mod 2$.

In [CDL+16], the authors showed an efficient reduction from parity Hitting Set to parity Set-Cover. Using the third item of Corollary 8.43 we get the following corollary:

Corollary 8.50. *For each constant $\delta > 0$, there is no bounded-error quantum algorithm that solves parity Set-Cover in $2^{n(1-\delta)}$ time, unless $\oplus$QSETH (Conjecture 8.9) is false.*

8.4 Open problems

We mention some questions for future work.

- It would be interesting to see if it is possible to use the QSETH framework to give a single-exponential lower bound for CVP in Euclidean norm (CVP_2).
- It's a big open question to get an efficient (quantum) fine-grained reduction from Hitting Set or CNF-SAT to Set-Cover. Note that Hitting Set and Set Cover are dual problems of each other but this reduction does not say anything interesting about the fine-grained hardness of Set-Cover.

Bibliography

[ABD+18] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. *More Consequences of Falsifying SETH and the Orthogonal Vectors Conjecture*. Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. 2018, pp. 253–266. DOI: 10.1145/3188745.3188938.

[ABG+21] Divesh Aggarwal, Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. *Fine-Grained Hardness of CVP(P) - Everything That We Can Prove (and Nothing Else)*. Proceedings of the 34th ACM-SIAM Symposium on Discrete Algorithms. 2021, pp. 1816–1835. DOI: 10.1137/1.9781611976465.10.

[AC21] Divesh Aggarwal and Eldon Chung. *A Note on the Concrete Hardness of the Shortest Independent Vector in Lattices*. Information Processing Letters **167** (2021), p. 106065. DOI: 10.1016/j.ipl.2020.106065.

[ACG+23] Joran van Apeldoorn, Arjan Cornelissen, András Gilyén, and Giacomo Nannicini. *Quantum Tomography Using State-Preparation Unitaries*. Proceedings of the 36th ACM-SIAM Symposium on Discrete Algorithms. 2023, pp. 1265–1318. DOI: 10.1137/1.9781611977554.ch47.

[ACK+21a] Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, Zeyong Li, and Noah Stephens-Davidowitz. *Dimension-Preserving Reductions Between SVP and CVP in Different p-Norms*. Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms. 2021, pp. 2444–2462. DOI: 10.1137/1.9781611976465.145.

[ACK+21b] Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, and Yixin Shen. *Improved (Provable) Algorithms for the Shortest Vector Problem via Bounded Distance Decoding*. Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science. Vol. 187. A version with major updates on arXiv:2002.07955. 2021, 4:1–4:20. DOI: 10.4230/LIPIcs.STACS.2021.4.

[ACL+20] Scott Aaronson, Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang, and Ruizhe Zhang. *On the Quantum Complexity of Closest Pair and Related Problems*. Proceedings of the 35th Computational Complexity Conference. 2020. DOI: 10.4230/LIPIcs.CCC.2020.16.

[ADR+15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. *Solving the Shortest Vector Problem in 2^n Time Using Discrete Gaussian Sampling: Extended Abstract*. Proceedings of the 47th Annual ACM on Symposium on Theory of Computing. 2015, pp. 733–742. DOI: 10.1145/2746539.2746606.

[AGG+20] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. *Quantum SDP-Solvers: Better Upper and Lower Bounds*. Quantum **4** (2020). Earlier version in FOCS'17. arXiv: 1705.01843, p. 230. DOI: 10.22331/q-2020-02-14-230.

[AGN24] Joran van Apeldoorn, Sander Gribling, and Harold Nieuwboer. *Basic Quantum Subroutines: Finding Multiple Marked Elements and Summing Numbers*. Quantum **8** (2024), p. 1284. DOI: 10.22331/Q-2024-03-14-1284.

[AH20] Jonathan Allcock and Chang-Yu Hsieh. *A Quantum Extension of SVM-perf for Training Nonlinear SVMs in Almost Linear Time*. Quantum **4** (2020), p. 342. DOI: 10.22331/Q-2020-10-15-342.

[Ajt96] Miklós Ajtai. *Generating Hard Instances of Lattice Problems (Extended Abstract)*. Proceedings of the 28th Annual ACM Symposium on the Theory of Computing. 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[AK23] Divesh Aggarwal and Rajendra Kumar. *Why We Couldn't Prove SETH Hardness of the Closest Vector Problem for Even Norms!* (2023). The updated version can be found on arXiv:2211.04385, pp. 2213–2230. DOI: 10.1109/FOCS57990.2023.00138.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. *A Sieve Algorithm for the Shortest Lattice Vector Problem*. Proceedings of the 33rd Annual ACM Symposium on Theory of Computing. 2001, pp. 601–610. DOI: 10.1145/380752.380857.

[ALN+20] Divesh Aggarwal, Jianwei Li, Phong Q. Nguyen, and Noah Stephens-Davidowitz. *Slide Reduction, Revisited - Filling the Gaps in SVP Approximation*. Proceedings of the Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference. Vol. 12171. 2020, pp. 274–295. DOI: 10.1007/978-3-030-56880-1.

[ALS21] Divesh Aggarwal, Zeyong Li, and Noah Stephens-Davidowitz. *A $2^{n/2}$-Time Algorithm for $\sqrt{n}$-SVP and $\sqrt{n}$-Hermite SVP, and an Improved Time-Approximation Tradeoff for (H)SVP*. Proceedings of the Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Technique. Vol. 12696. 2021, pp. 467–497. DOI: 10.1007/978-3-030-77870-5.

[Amb02] Andris Ambainis. *Quantum Lower Bounds by Quantum Arguments*. Journal of Computer and System Sciences **64**, 4 (2002), pp. 750–767. DOI: 10.1006/jcss.2002.1826.

[Amb06] Andris Ambainis. *Polynomial Degree vs. Quantum Query Complexity*. Journal of Computer and System Sciences **72**, 2 (2006), pp. 220–238. DOI: 10.1016/j.jcss.2005.06.006.

[ANS18] Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. *Quantum Lattice Enumeration and Tweaking Discrete Pruning*. Proceedings of the Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security. 2018, pp. 405–434. DOI: 10.1007/978-3-030-03326-2.

[AR05] Dorit Aharonov and Oded Regev. *Lattice Problems in NP ∩ coNP*. Journal of the ACM **52**, 5 (2005), pp. 749–765. DOI: 10.1145/1089023.1089025.

[AS18] Divesh Aggarwal and Noah Stephens-Davidowitz. *(Gap/S)ETH hardness of SVP*. Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. ACM, 2018, pp. 228–238. DOI: 10.1145/3188745.3188840.

[AS19] Omer Angel and Yinon Spinka. *Pairwise Optimal Coupling of Multiple Random Variables*. arXiv:1903.00632. 2019.

[AW21] Shyan Akmal and Ryan Williams. *MAJORITY-3SAT (and Related Problems) in Polynomial Time*. 62nd IEEE Annual Symposium on Foundations of Computer Science. Earlier version in arXiv:2107.02748. 2021, pp. 1033–1043. DOI: 10.1109/FOCS52979.2021.00103.

[Ban93] Wojciech Banaszczyk. *New Bounds in some Transference Theorems in the Geometry of Numbers*. Mathematische Annalen **296**, 1 (1993), pp. 625–635. DOI: 10.1007/BF01445125.

[BB84] Charles Bennett and Gilles Brassard. *Quantum Cryptography: Public Key Distribution and Coin Tossing*. Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing. The updated version can be found on DOI:10.1016/j.tcs.2014.05.025. 1984, p. 175.

[BBB+97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. *Strengths and Weaknesses of Quantum Computing*. SIAM Journal on Computing **26**, 5 (1997), pp. 1510–1523. DOI: 10.1137/S0097539796300933.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. *Quantum Lower Bounds by Polynomials*. Journal of the ACM **48**, 4 (2001). Earlier version in FOCS'98. arXiv:quant-ph/9802049, pp. 778–797.

[BBH+98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. *Tight Bounds on Quantum Searching*. Fortschritte der Physik **46**, 4–5 (1998). arXiv: quant-ph/9605034, pp. 493–505. DOI: 10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P.

[BCK+23] Huck Bennett, Yanlin Chen, Rajendra Kumar, Zeyong Li, and Spencer Peters. *Lattice Problems in General Norms: Algorithms with Explicit Constants, Dimension-Preserving Reductions, and More*. Manuscript forthcoming. 2023.

[BCS+23] Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen. *Finding many Collisions via Reusable Quantum Walks*. Proceedings of the Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques. Vol. 14008. 2023, pp. 221–251. DOI: 10.1007/978-3-031-30589-4.

[BCW+99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. *Bounds for Small-Error and Zero-Error Quantum Algorithms*. Proceedings of the 40th IEEE Annual Symposium on Foundations of Computer Science. 1999, pp. 358–368. DOI: 10.1109/SFFCS.1999.814607.

[BDG+16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. *New Directions In Nearest Neighbor Searching with Applications to Lattice Sieving*. Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms. 2016, pp. 10–24. DOI: 10.1137/1.9781611974331.ch2.

[Ben82] Paul Benioff. *Quantum Mechanical Hamiltonian Models of Turing Machines*. Journal of Statistical Physics **29**, 3 (1982), pp. 515–546. DOI: 10.1007/BF01342185.

[BG11] Peter Bühlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer, 2011. DOI: 10.1007/978-3-642-20192-9.

[BGS17] Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. *On the Quantitative Hardness of CVP*. Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science. 2017, pp. 13–24. DOI: 10.1109/FOCS.2017.11.

[BH16] Afonso S. Bandeira and Ramon van Handel. *Sharp Nonasymptotic Bounds on the Norm of Random Matrices with Independent Entries*. The Annals of Probability **44**, 4 (2016). Earlier version in arXiv:1408.6185, pp. 2479–2506. DOI: 10.1214/15-AOP1025.

[Bha97] Rajendra Bhatia. *Matrix Analysis*. Vol. 169. Graduate Texts in Mathematics. Springer, 1997. DOI: 10.1007/978-1-4612-0653-8.

[BHJ26] Max Born, Werner Karl Heisenberg, and Pascual Jordan. *Zur Quantenmechanik. II*. Zeitschrift für Physik **35**, 8-9 (1926), pp. 557–615. DOI: 10.1007/BF01379806.

[BHM+02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum Amplitude Amplification and Estimation*. Quantum Computation and Information (2002). arXiv:quant-ph/0005055, pp. 53–74. DOI: 10.1007/S11128-023-04146-3.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. *Quantum Counting*. Proceedings of the 25th International Colloquium on Automata, Languages and Programming. Vol. 1443. arXiv:quant-ph/9805082. 1998, pp. 820–831.

[BJ25] Max Born and Pascual Jordan. *Zur Quantenmechanik*. Zeitschrift für Physik **34**, 1 (1925), pp. 858–888. DOI: 10.1007/BF01328531.

[BL20] Aleksandrs Belovs and Troy Lee. *The Quantum Query Complexity of Composition with a Relation*. arXiv:2004.06439. 2020.

[BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. ISBN: 9780199535255. DOI: 10.1093/acprof:oso/9780199535255.001.0001.

[Boh13] Niels Bohr. *I. On the Constitution of Atoms and Molecules*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **26**, 151 (1913), pp. 1–25. DOI: 10.1080/14786441308634955.

[BP20] Huck Bennett and Chris Peikert. *Hardness of Bounded Distance Decoding on Lattices in ℓ_p Norms*. Proceedings of the 35th Computational Complexity Conference. Vol. 169. 2020, 36:1–36:21. DOI: 10.4230/LIPIcs.CCC.2020.36.

[BPS21] Harry Buhrman, Subhasree Patro, and Florian Speelman. *A Framework of Quantum Strong Exponential-Time Hypotheses*. Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science. Vol. 187. 2021, 19:1–19:19. DOI: 10.4230/LIPIcs.STACS.2021.19.

[BPT22] Huck Bennett, Chris Peikert, and Yi Tang. *Improved Hardness of BDD and SVP Under Gap-(S)ETH*. 13th Innovations in Theoretical Computer Science Conference. Vol. 215. 2022, 19:1–19:12. DOI: 10.4230/LIPIcs.ITCS.2022.19.

[Bri84] Ernest F. Brickell. *Breaking Iterated Knapsacks*. Proceedings of the Advances in Cryptology - CRYPTO 1984 - 3rd Annual International Conference on the Theory and Applications of Cryptographic Technique. 1984, pp. 342–358. DOI: 10.1007/3-540-39568-7.

[BRS+17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. *Average-Case Fine-Grained Hardness*. Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, 2017, pp. 483–496. DOI: 10.1145/3055399.3055466.

[BŠ06] Harry Buhrman and Robert Špalek. *Quantum Verification of Matrix Products*. Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms. arXiv: quant-ph/0409035. 2006, pp. 880–889. DOI: 10.1145/1109557.1109654.

[Bud89] Rudi de Buda. *Some Optimal Codes Have Structure*. IEEE Journal on Selected Areas in Communications **7**, 6 (1989), pp. 893–899. DOI: 10.1109/49.29612.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. *Lattice-Based FHE as Secure as PKE*. Proceedings of the 5th Conference on Innovations in Theoretical Computer Science. 2014, pp. 1–12. DOI: 10.1145/2554797.2554799.

[CCK+23] Yanlin Chen, Yilei Chen, Rajendra Kumar, Subhasree Patro, and Florian Speelman. *QSETH Strikes Again: Finer Quantum Lower Bounds for Lattice Problem, Strong Simulation, Hitting Set Problem, and More*. arXiv:2309.16431. 2023.

[CCL18] Yanlin Chen, Kai-Min Chung, and Ching-Yi Lai. *Space-Efficient Classical and Quantum Algorithms for the Shortest Vector Problem*. Quantum Information & Computation **18**, 3&4 (2018), pp. 285–306. DOI: 10.26421/QIC18.3-4-7.

[CDL+16] Marek Cygan, Holger Dell, Daniel Lokshtanov, et al. *On Problems as Hard as CNF-SAT*. ACM Transactions on Algorithms **12**, 3 (2016). Earlier version in CCC'12. arXiv: 1112.2275, pp. 1–24. DOI: 10.1145/2925416.

[CEM+98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. *Quantum Algorithms Revisited*. Proceedings of the Royal Society A **454**, 1969 (1998). arXiv: quant-ph/9708016, pp. 339–354. DOI: 10.1098/rspa.1998.0164.

[CGW24] Yanlin Chen, András Gilyén, and Ronald de Wolf. *A Quantum Speed-Up for Approximating the Top Eigenvectors of a Matrix*. Presented as a talk QIP'24, arXiv:2405.14765. 2024.

[Che52] Herman Chernoff. *A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations*. The Annals of Mathematical Statistics **23**, 4 (1952), pp. 493–507. DOI: 10.1214/aoms/1177729330.

[CHM21] Jordan S. Cotler, Hsin-Yuan Huang, and Jarrod R. McClean. *Revisiting Dequantization and Quantum Advantage in Learning Tasks*. arXiv:2112.00811. 2021.

[Chu36] Alonzo Church. *An Unsolvable Problem of Elementary Number Theory*. American Journal of Mathematics, 2 (1936). DOI: 10.2307/2371045.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. *A Duality Between Clause Width and Clause Density for SAT*. Proceedings of the 21st Annual IEEE Conference on Computational Complexity. 2006, 7 pp.–260. DOI: 10.1109/CCC.2006.6.

[CL21] André Chailloux and Johanna Loyer. *Lattice Sieving Via Quantum Random Walks*. Proceedings of the Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security. 2021. DOI: 10.1007/978-3-030-92068-5_3.

[CMP23] Shantanav Chakraborty, Aditya Morolia, and Anurudh Peduri. *Quantum Regularized Least Squares*. Quantum **7** (2023), p. 988. DOI: 10.22331/Q-2023-04-27-988.

[CSS11] Nicolò Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. *Efficient Learning with Partially Observed Attributes*. Journal of Machine Learning Research **12** (2011). Earlier version in ICML'10. arXiv:1004.4421, pp. 2857–2878.

[CW23] Yanlin Chen and Ronald de Wolf. *Quantum Algorithms and Lower Bounds for Linear Regression with Norm Constraints*. Proceedings of the 50th International Colloquium on Automata, Languages, and Programming. Vol. 261. LIPIcs. 2023, 38:1–38:21. DOI: 10.4230/LIPIcs.ICALP.2023.38.

[Deu85] David Deutsch. *Quantum theory, the Church-Turing principle and the universal quantum computer*. Proceedings of the Royal Society of London Series A **400**, 1818 (1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070.

[DH96] Christoph Dürr and Peter Høyer. *A Quantum Algorithm for Finding the Minimum*. arXiv:quant-ph/9607014. 1996.

[DK08] Abhimanyu Das and David Kempe. *Algorithms for Subset Selection in Linear Regression*. Proceedings of the 40th Symposium on Theory of Computing. ACM, 2008, pp. 45–54. DOI: 10.1145/1374376.1374384.

[DRS14] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. *On the Closest Vector Problem with a Distance Guarantee*. Proceedings of the IEEE 29th Conference on Computational Complexity. 2014, pp. 98–109. DOI: 10.1109/CCC.2014.18.

[DS01] Kenneth R. Davidson and Stanisław J. Szarek. *Local Operator Theory, Random Matrices and Banach Spaces. Handbook of the Geometry of Banach Spaces, Vol. I*. Vol. I. Elsevier Science B.V., 2001. Chap. 8, pp. 317–366. ISBN: 9780080532806. DOI: 10.1016/S1874-5849(01)80010-3.

[Edm65] Jack Edmonds. *Paths, Trees, and Flowers*. Canadian Journal of Mathematics **17** (1965), pp. 449–467. DOI: 10.4153/CJM-1965-045-4.

[Ein05] Albert Einstein. *Concerning an Heuristic Point of View Toward the Emission and Transformation of Light*. Annalen der Physik (1905), pp. 132–148. DOI: 10.1002/andp.19053220607.

[Ein24] Albert Einstein. *Quantentheorie des einatomigen idealen Gases*. Königliche Preußische Akademie der Wissenschaften, 261 (1924). DOI: 10.1002/3527608958.ch27.

[Fey82] Richard Feynman. *Simulating Physics with Computers*. International Journal of Theoretical Physics **21**, 6/7 (1982), pp. 467–488. DOI: 10.1007/BF02650179.

[Fey86] Richard Feynman. *Quantum Mechanical Computers*. Foundations of Physics **16**, 6 (1986), pp. 507–531. DOI: 10.1007/bf01886518.

[Fre77] Rūsiņš Freivalds. *Probabilistic Machines Can Use Less Running Time*. Proceedings of the 7th IFIP Congress. 1977, pp. 839–842. URL: https://api.semanticscholar.org/CorpusID:45405368.

[FT87] András Frank and Éva Tardos. *An Application of Simultaneous Diophantine Approximation in Combinatorial optimization*. Combinatorica **7**, 1 (1987), pp. 49–65. DOI: 10.1007/BF02579200.

[FW56] Marguerite Frank and Philip Wolfe. *An Algorithm for Quadratic Programming*. Naval Research Logistics Quarterly **3**, 1-2 (1956), pp. 95–110. DOI: 10.1002/nav.3800030109.

[Gen09] Craig Gentry. *Fully Homomorphic Encryption Using Ideal Lattices*. Proceedings of the 41st Annual ACM Symposium on Theory of Computing. 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[Gil23] András Gilyén. *Iterative Refinement for Improved Tomography and Quantum Linear Equation Solving*. Poster presented at QIP'24, manuscript forthcoming. 2023.

[GKN+21] Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. *No Quantum Speedup over Gradient Descent for Non-Smooth Convex Optimization*. Proceedings of the 12th Innovations in Theoretical Computer Science Conference. Vol. 185. 2021, 53:1–53:20. DOI: 10.4230/LIPICS.ITCS.2021.53.

[GL13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Fourth. Vol. 3. Johns Hopkins Studies in Mathematical Sciences. Johns Hopkins University Press, 2013.

[GN08] Nicolas Gama and Phong Q. Nguyen. *Finding Short Lattice Vectors within Mordell's Inequality*. Proceedings of the 40th Annual ACM Symposium on Theory of Computing. 2008, pp. 207–216. DOI: 10.1145/1374376.1374408.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. *Lattice Enumeration Using Extreme Pruning*. Proceedings of the Advances in Cryptology - EUROCRYPT 2010 - 29nd Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2010, pp. 257–278. DOI: 10.1007/978-3-642-13190-5.

[Gor85] Yehoram Gordon. *Some Inequalities for Gaussian Processes and Applications*. Israel Journal of Mathematics **50**, 4 (1985), pp. 265–289. DOI: 10.1007/BF02759761.

[Gro96] Lov Grover. *A Fast Quantum Mechanical Algorithm for Database Search*. Proceedings of the 28th Annual ACM Symposium on the Theory of Computing. arXiv:quant-ph/9605043. 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[GSL+19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. *Quantum Singular Value Transformation and Beyond: Exponential Improvements for Quantum Matrix Arithmetics*. Proceedings of the 51st ACM Symposium on the Theory of Computing. Full version in arXiv: 1806.01838. 2019, pp. 193–204. DOI: 10.1145/3313276.3316366.

[Hei25] Werner Karl Heisenberg. *Über quantentheoretische Umdeutung kinematischer und mechanischer Beziehungen*. Zeitschrift für Physik **33** (1925), pp. 879–893. DOI: 10.1007/BF01328377.

[Hel85] Bettina Helfrich. *Algorithms to Construct Minkowski Reduced and Hermite Reduced Lattice Bases*. Theor. Comput. Sci. **41**, 2–3 (1985), pp. 125–139. DOI: 10.1016/0304-3975(85)90067-2.

[HHJ+17] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. *Sample-Optimal Tomography of Quantum States*. EEE Transactions on Information Theory **63**, 9 (2017). arXiv: 1508.01797, pp. 5628–5641. DOI: 10.1109/TIT.2017.2719044.

[HK12] Elad Hazan and Tomer Koren. *Linear Regression with Limited Observation*. Proceedings of the 29th International Conference on Machine Learning. arXiv:1206.4678. More extensive version at arXiv:1108.4559. 2012.

[HK70] Arthur Hoerl and Robert Kennard. *Ridge Regression: Biased Estimation for Nonorthogonal Problems*. Technometrics **12**, 1 (1970), pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

[HKO+23] Jeongwan Haah, Robin Kothari, Ryan O'Donnell, and Ewin Tang. *Query-Optimal Estimation of Unitary Channels in Diamond Distance*. Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science. 2023, pp. 363–390. DOI: 10.1109/FOCS57990.2023.00028.

[HNS20] Cupjin Huang, Michael Newman, and Mario Szegedy. *Explicit Lower Bounds on Strong Quantum Simulation*. IEEE Transactions on Information Theory **66**, 9 (2020), pp. 5585–5600. DOI: 10.1109/TIT.2020.3004427.

[Hoe63] Wassily Hoeffding. *Probability Inequalities for Sums of Bounded Random Variables*. Journal of the American Statistical Association **58**, 301 (1963), pp. 13–30. DOI: 10.2307/2282952.

[Hol03] Susan Holmes. *Stein's Method for Birth and Death Chains. Stein's Method: Expository Lectures and Applications*. Ed. by Persi Diaconis and Susan Holmes. 2003. DOI: 10.1214/lnms/1196283799.

[HP14] Moritz Hardt and Eric Price. *The Noisy Power Method: A Meta Algorithm with Applications*. Proceedings of the Advances in Neural Information Processing Systems. Vol. 27. arXiv: 1311.2495. 2014.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. *Analyzing Blockwise Lattice Algorithms Using Dynamical Systems*. Proceedings of the Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference. Ed. by Phillip Rogaway. 2011, pp. 447–464. DOI: 10.1007/978-3-642-22792-9.

[HR12] Ishay Haviv and Oded Regev. *Tensor-Based Hardness of the Shortest Vector Problem to within Almost Polynomial Factors*. Theory of Computing **8**, 1 (2012). Earlier version in STOC'07. arXiv:1806.04087, pp. 513–531. DOI: 10.4086/TOC.2012.V008A023.

[HS07] Guillaume Hanrot and Damien Stehlé. *Improved Analysis of Kannan's Shortest Lattice Vector Algorithm*. Proceedings of the Advances in Cryptology - CRYPTO - 27th Annual International Cryptology Conference. 2007, pp. 170–186. DOI: 10.1007/978-3-540-74143-5.

[HS65] Juris Hartmanis and Richard Edwin Stearns. *On the Computational Complexity of Algorithms*. Transactions of the American Mathematical Society **117** (1965), pp. 285–306. DOI: 10.1090/S0002-9947-1965-0170805-7.

[IP01] Russell Impagliazzo and Ramamohan Paturi. *On the Complexity of k-SAT*. Journal of Computer and System Sciences **62**, 2 (2001), pp. 367–375. DOI: 10.1006/JCSS.2000.1727.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. *Which Problems Have Strongly Exponential Complexity?* Journal of Computer and System Sciences **63**, 4 (2001), pp. 512–530. DOI: 10.1006/JCSS.2001.1774.

[Jag11] Martin Jaggi. *Sparse Convex Optimization Methods for Machine Learning*. Doctoral Thesis. ETH, 2011. DOI: 10.3929/ethz-a-007050453.

[Jag13] Martin Jaggi. *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization*. Proceedings of the 30th International Conference on Machine Learning. Vol. 28. 2013, pp. 427–435. URL: http://proceedings.mlr.press/v28/jaggi13.html.

[Jag14] Martin Jaggi. *An equivalence between the Lasso and Support Vector Machines. Regularization, Optimization, Kernels, and Support Vector Machines*. Ed. by Johan Suykens, Marco Signoretto, and Andreas Argyriou. arXiv:1303.1152. CRC Press, 2014.

[Kan83] Ravi Kannan. *Improved Algorithms for Integer Programming and Related Lattice Problems*. Proceedings of the 15th Annual ACM Symposium on Theory of Computing. 1983, pp. 193–206. DOI: 10.1145/800061.808749.

[Kan87a] Ravi Kannan. *Algorithmic Geometry of Numbers*. Annual Review of Computer Science **2**, Volume 2, 1987 (1987), pp. 231–267. DOI: https://doi.org/10.1146/annurev.cs.02.060187.001311.

[Kan87b] Ravi Kannan. *Minkowski's Convex Body Theorem and Integer Programming*. Mathematics of Operations Research **12**, 3 (1987), pp. 415–440. URL: http://www.jstor.org/stable/3689974.

[Kho05] Subhash Khot. *Hardness of Approximating the Shortest Vector Problem in Lattices*. Journal of the ACM **52**, 5 (2005), pp. 789–808. DOI: 10.1145/1089023.1089027.

[Kit95] Alexei Yu. Kitaev. *Quantum Measurements and the Abelian Stabilizer Problem*. arXiv: quant-ph/9511026. 1995.

[KL78] Grigorii Kabatiansky and Vladimir Iosifovich Levenshtein. *On Bounds for Packings on a Sphere and in Space*. Problemy Peredachi Informatsii **14**, 1 (1978), pp. 3–25.

[KMP+19] Elena Kirshanova, Erik Mårtensson, Eamonn W. Postlethwaite, and Subhayan Roy Moulik. *Quantum Algorithms for the Approximate k-List Problem and Their Application to Lattice Sieving*. Proceedings of the Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security. Vol. 11921. 2019, pp. 521–551. DOI: 10.1007/978-3-030-34578-5_19.

[KP17] Iordanis Kerenidis and Anupam Prakash. *Quantum Recommendation Systems*. Proceedings of the 8th Innovations in Theoretical Computer Science Conference. Vol. 67. 2017, 49:1–49:21. DOI: 10.4230/LIPICS.ITCS.2017.49.

[KP20] Iordanis Kerenidis and Anupam Prakash. *A Quantum Interior Point Method for LPs and SDPs*. ACM Transactions on Quantum Computing **1**, 1 (2020). arXiv: 1808.09266. DOI: 10.1145/3406306.

[LC19] Guang Hao Low and Isaac L. Chuang. *Hamiltonian Simulation by Qubitization*. Quantum **3** (2019). arXiv: 1610.06546, p. 163. DOI: 10.22331/q-2019-07-12-163.

[Led01] Michel Ledoux. *The Concentration of Measure Phenomenon*. Mathematical surveys and monographs. American Mathematical Society, 2001. URL: https://bookstore.ams.org/surv-89-s.

[Len83] Hendrik Lenstra. *Integer Programming with a Fixed Number of Variables*. Mathematics of Operations Research **8**, 4 (1983), pp. 538–548. DOI: 10.1287/moor.8.4.538.

[Li11] Shengqiao Li. *Concise Formulas for the Area and Volume of a Hyperspherical Cap*. Asian Journal of Mathematics and Statistics (2011), pp. 66–70. URL: https://docsdrive.com/pdfs/ansinet/ajms/2011/66-70.pdf.

[LLL82] Arjen Lenstra, Hendrik Lenstra, and László Lovász. *Factoring Polynomials with Rational Coefficients*. Mathematische Annalen **261** (1982), pp. 515–534. DOI: 10.1007/BF01457454.

[LLM06] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. *On bounded Distance Decoding for General Lattices. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2006, pp. 450–461. DOI: 10.1007/11830924_41.

[LLW19] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. *Public-Key Cryptography in the Fine-Grained Setting*. Proceedings of the Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference. Vol. 11694. 2019, pp. 605–635. DOI: 10.1007/978-3-030-26954-8.

[LMP15] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. *Finding Shortest Lattice Vectors Faster Using Quantum Search*. Designs, Codes and Cryptography **77** (2015). DOI: 10.1007/s10623-015-0067-5.

[LO85] Jeffrey C. Lagarias and Andrew M. Odlyzko. *Solving Low-Density Subset Sum Problems*. Journal of the ACM **32**, 1 (1985), pp. 229–246. DOI: 10.1145/2455.2461.

[Low19] Guang Hao Low. *Hamiltonian Simulation with Nearly Optimal Dependence on Spectral Norm*. Proceedings of the 51st ACM Symposium on the Theory of Computing. arXiv: 1807.03967. 2019, pp. 491–502. DOI: 10.1145/3313276.3316386.

[Man80] Yuri Manin. *Vychislimoe i nevychislimoe*. Soviet Radio, 1980. URL: https://books.google.nl/books?id=pAo-zgEACAAJ.

[Man99] Yuri Manin. *Classical Computing, Quantum Computing, and Shor's Factoring Algorithm*. Séminaire Bourbaki **41** (1999), pp. 375–404. URL: http://eudml.org/doc/110265.

[Mic00] Daniele Micciancio. *The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant*. SIAM Journal on Computing **30**, 6 (2000), pp. 2008–2035. DOI: 10.1137/S0097539700373039.

[MP12] Daniele Micciancio and Chris Peikert. *Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller*. Proceedings of the Advances in Cryptology 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques. Vol. 7237. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4.

[MR07] Daniele Micciancio and Oded Regev. *Worst-Case to Average-Case Reductions Based on Gaussian Measures*. SIAM Journal on Computing **37**, 1 (2007), pp. 267–302. DOI: 10.1137/S0097539705447360.

[MR08] Daniele Micciancio and Oded Regev. *Lattice-Based Cryptography*. 2008. URL: https://cims.nyu.edu/~regev/papers/pqc.pdf.

[MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Second. Adaptive Computation and Machine Learning series. MIT Press, 2018. URL: https://mitpress.mit.edu/9780262039406/foundations-of-machine-learning/.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. *Faster Exponential Time Algorithms for the Shortest Vector Problem*. Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms. 2010, pp. 1468–1480. DOI: 10.1137/1.9781611973075.119.

[MW15] Daniele Micciancio and Michael Walter. *Fast Lattice Point Enumeration with Minimal Overhead*. Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms. 2015, pp. 276–294. DOI: 10.1137/1.9781611973730.21.

[Nes83] Yurii Nesterov. *A Method for Solving the Convex Programming Problem with Convergence Rate $\mathcal{O}(1/k^2)$*. Proceedings of the USSR Academy of Sciences **269** (1983), pp. 543–547.

[NV08] Phong Q. Nguyen and Thomas Vidick. *Sieve Algorithms for the Shortest Vector Problem Are Practical*. Journal of Mathematical Cryptology **2**, 2 (2008), pp. 181–207. DOI: 10.1515/JMC.2008.009.

[NW99] Ashwin Nayak and Felix Wu. *The Quantum Query Complexity of Approximating the Median and Related Statistics*. Proceedings of the 31st Annual ACM Symposium on Theory of Computing. 1999, pp. 384–393. DOI: 10.1145/301250.301349.

[Oka59] Masashi Okamoto. *Some Inequalities Relating to the Partial Sum of Binomial Probabilities*. Annals of the Institute of Statistical Mathematics **10**, 1 (1959), pp. 29–35. DOI: 10.1007/BF02883985.

[Par87] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics 20. SIAM, 1987. DOI: 10.1137/1.9781611971163.

[Pat92] Ramamohan Paturi. *On the Degree of Polynomials That Approximate Symmetric Boolean Functions (Preliminary Version)*. Proceedings of the 24th Annual ACM Symposium on Theory of Computing. 1992, pp. 468–474. ISBN: 0897915119. DOI: 10.1145/129712.129758.

[Pau25] Wolfgang Pauli. *Über den Zusammenhang des Abschlusses der Elektronengruppen im Atom mit der Komplexstruktur der Spektren*. Zeitschrift für Physik **31**, 1 (1925), pp. 765–783. DOI: 10.1007/BF02980631.

[Pla00a] Max Planck. *Über eine Verbesserung der Wienschen Spektralgleichung*. at a meeting of the Deutsche Physikalische Gesellschaft (1900), pp. 175–178. DOI: 10.1007/978-3-663-13885-3_15.

[Pla00b] Max Planck. *Zur Theorie des Gesetzes der Energieverteilung im Normalspectrum*. at a meeting of the Deutsche Physikalische Gesellschaft **2** (1900), p. 237. URL: https://cds.cern.ch/record/262745.

[Pra14] Anupam Prakash. *Quantum Algorithms for Linear Algebra and Machine Learning*. PhD thesis. University of California, Berkeley, 2014. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.pdf.

[PS09] Xavier Pujol and Damien Stehlé. *Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$*. ePrint:2009/608. 2009.

[Rav21] Alex Ravsky. *Reply on StackExchange*. StackExchange. 2021. URL: https://math.stackexchange.com/questions/3922615/inner-product-of-unit-vector-with-a-vector-uniformly-distributed-on-a-hyperspher.

[Reg04] Oded Regev. *Lattices in Computer Science, Lecture 8*. 2004. URL: https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/svpalg.pdf.

[Reg06] Oded Regev. *Lattice-Based Cryptography*. Proceedings of the Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference. 2006, pp. 131–141. DOI: 10.1007/11818175_8.

[Reg09] Oded Regev. *On Lattices, Learning with Errors, Random Linear Codes, and Cryptography*. Journal of the ACM **56**, 6 (2009). Earlier version in STOC'05. arXiv:2401.03703, pp. 1–40. DOI: 10.1145/1568318.1568324.

[RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. *Quantum Support Vector Machine for Big Data Classification*. Physical Review Letters **113**, 13 (2014), p. 130503. DOI: 10.1103/PhysRevLett.113.130503.

[Rob55] Herbert Robbins. *A Remark on Stirling's Formula*. The American Mathematical Monthly **62**, 1 (1955), pp. 26–29. DOI: 10.2307/2308012.

[Rud76] Walter Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976. ISBN: 9780070856134. URL: https://books.google.nl/books?id=kwqzPAAACAAJ.

[SA19] Seyran Saeedi and Tom Arodz. *Quantum Sparse Support Vector Machines*. arXiv: 1902.01879. 2019.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019.

[Sch05] Rainer Schuler. *An Algorithm for the Satisfiability Problem of Formulas in Conjunctive Normal Form*. Journal of Algorithms **54**, 1 (2005), pp. 40–44. DOI: 10.1016/j.jalgor.2004.04.012.

[Sch87] Claus-Peter Schnorr. *A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms*. Theoretical Computer Science **53** (1987), pp. 201–224. DOI: 10.1016/0304-3975(87)90064-8.

[SE94] Claus-Peter Schnorr and M. Euchner. *Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems*. Math. Program. **66** (1994), pp. 181–199. DOI: 10.1007/BF01581144.

[Sha84] Adi Shamir. *A Polynomial-Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem*. IEEE Trans. Information Theory **30**, 5 (1984), pp. 699–704. DOI: 10.1109/TIT.1984.1056964.

[Sho97] Peter Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM Journal on Computing **26**, 5 (1997). Earlier version in FOCS'94. arXiv:quant-ph/9508027, pp. 1484–1509. DOI: 10.1137/S0097539795293172.

[SK19] Maria Schuld and Nathan Killoran. *Quantum Machine Learning in Feature Hilbert Spaces*. Physical Review Letters **122**, 13 (2019), p. 040504. DOI: 10.1103/PhysRevLett.122.040504.

[SPA21] Seyran Saeedi, Aliakbar Panahi, and Tom Arodz. *Quantum Semi-Supervised Kernel Learning*. Quantum Machine Intelligence **3** (2021), p. 24. DOI: 10.1007/s42484-021-00053-x.

[Ste16] Noah Stephens-Davidowitz. *Discrete Gaussian Sampling Reduces to CVP and SVP*. Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms. 2016, pp. 1748–1764. DOI: 10.1137/1.9781611974331.CH121.

[Tib96] Robert Tibshirani. *Regression Shrinkage and Selection via the Lasso*. Journal of the Royal Statistical Society **58** (1996), pp. 267–288. URL: https://www.jstor.org/stable/2346178.

[Tur37] Alan Turing. *On Computable Numbers, with an Application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society **s2-42**, 1 (1937), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230.

[Van10] Maarten Van den Nest. *Classical Simulation of Quantum Computation, the Gottesman-Knill Theorem, and Slightly Beyond*. Quantum Information & Computation. **10**, 3 (2010), pp. 258–271. URL: https://doi.org/10.26421/QIC10.3-4-6.

[Vas15] Virginia Vassilevska Williams. *Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk)*. Proceedings of the 10th International Symposium on Parameterized and Exact Computation. 2015. DOI: 10.4230/LIPICS.IPEC.2015.17.

[Vas18] Virginia Vassilevska Williams. *On Some Fine-Grained questions in Algorithms and Complexity*. Proceedings of the international congress of mathematicians: Rio de janeiro 2018. 2018, pp. 3447–3487. URL: https://people.csail.mit.edu/virgi/eccentri.pdf.

[Ver12] Roman Vershynin. *Introduction to the Non-Asymptotic Analysis of Random Matrices. Compressed Sensing: Theory and Applications*. Ed. by Yonina C. Eldar and Gitta Kutyniok. arXiv: 1011.3027. Cambridge University Press, 2012, pp. 210–268. DOI: 10.1017/CBO9780511794308.006.

[Vin78] Hrishikesh Vinod. *A Survey of Ridge Regression and Related Techniques for Improvements over Ordinary Least Squares*. The Review of Economics and Statistics **60**, 1 (1978), pp. 121–131. URL: https://www.jstor.org/stable/1924340.

[Vlă19] Serge Vlăduţ. *Lattices with Exponentially Large Kissing Numbers*. Moscow Journal of Combinatorics and Number Theory **8**, 2 (2019), pp. 163–177. DOI: 10.2140/moscow.2019.8.163.

[Wed72] Per-Åke Wedin. *Perturbation Bounds in Connection with Singular Value Decomposition*. BIT Numerical Mathematics **12**, 1 (1972), pp. 99–111. DOI: 10.1007/BF01932678.

[Wie83] Stephen Wiesner. *Conjugate Coding*. SIGACT News **15**, 1 (1983). Proposed by Stephen Wiesner in 1970 and remained unpublished until 1983. DOI: 10.1145/1008908.1008920.

[Wil05] Ryan Williams. *A New Algorithm for Optimal 2-Constraint Satisfaction and its Implications*. Theoretical Computer Science **348**, 2-3 (2005), pp. 357–365. DOI: 10.1016/J.TCS.2005.09.023.

[YLC14] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. *Fixed-Point Quantum Search with an Optimal Number of Queries*. Physical Review Letters **113**, 21 (2014). arXiv:1409.3305, p. 210501. DOI: 10.1103/PhysRevLett.113.210501.

[ZY06] Peng Zhao and Bin Yu. *On Model Selection Consistency of Lasso*. Journal of Machine Learning Research **7** (2006), pp. 2541–2563. URL: http://jmlr.org/papers/v7/zhao06a.html.

Titles in the ILLC Dissertation Series:

ILLC DS-2020-05: **Tom Bannink**
Quantum and stochastic processes

ILLC DS-2020-06: **Dieuwke Hupkes**
Hierarchy and interpretability in neural models of language processing

ILLC DS-2020-07: **Ana Lucia Vargas Sandoval**
On the Path to the Truth: Logical & Computational Aspects of Learning

ILLC DS-2020-08: **Philip Schulz**
Latent Variable Models for Machine Translation and How to Learn Them

ILLC DS-2020-09: **Jasmijn Bastings**
A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing

ILLC DS-2020-10: **Arnold Kochari**
Perceiving and communicating magnitudes: Behavioral and electrophysiological studies

ILLC DS-2020-11: **Marco Del Tredici**
Linguistic Variation in Online Communities: A Computational Perspective

ILLC DS-2020-12: **Bastiaan van der Weij**
Experienced listeners: Modeling the influence of long-term musical exposure on rhythm perception

ILLC DS-2020-13: **Thom van Gessel**
Questions in Context

ILLC DS-2020-14: **Gianluca Grilletti**
Questions & Quantification: A study of first order inquisitive logic

ILLC DS-2020-15: **Tom Schoonen**
Tales of Similarity and Imagination. A modest epistemology of possibility

ILLC DS-2020-16: **Ilaria Canavotto**
Where Responsibility Takes You: Logics of Agency, Counterfactuals and Norms

ILLC DS-2020-17: **Francesca Zaffora Blando**
Patterns and Probabilities: A Study in Algorithmic Randomness and Computable Learning

ILLC DS-2021-01: **Yfke Dulek**
Delegated and Distributed Quantum Computation

ILLC DS-2021-02: **Elbert J. Booij**
The Things Before Us: On What it Is to Be an Object

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
Modeling Users Interacting with Smart Devices

ILLC DS-2021-04: **Sophie Arnoult**
Adjunction in Hierarchical Phrase-Based Translation

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
A Pragmatic Defense of Logical Pluralism

ILLC DS-2021-06: **Zoi Terzopoulou**
Collective Decisions with Incomplete Individual Opinions

ILLC DS-2021-07: **Anthia Solaki**
Logical Models for Bounded Reasoners

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
Incorporating Structure into Neural Models for Language Processing

ILLC DS-2021-09: **Taichi Uemura**
Abstract and Concrete Type Theories

ILLC DS-2021-10: **Levin Hornischer**
Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation

ILLC DS-2021-11: **Sirin Botan**
Strategyproof Social Choice for Restricted Domains

ILLC DS-2021-12: **Michael Cohen**
Dynamic Introspection

ILLC DS-2021-13: **Dazhu Li**
Formal Threads in the Social Fabric: Studies in the Logical Dynamics of Multi-Agent Interaction

ILLC DS-2021-14: **Álvaro Piedrafita**
On Span Programs and Quantum Algorithms

ILLC DS-2022-01: **Anna Bellomo**
Sums, Numbers and Infinity: Collections in Bolzano's Mathematics and Philosophy

ILLC DS-2022-02: **Jan Czajkowski**
Post-Quantum Security of Hash Functions

ILLC DS-2022-03: **Sonia Ramotowska**
Quantifying quantifier representations: Experimental studies, computational modeling, and individual differences

ILLC DS-2022-04: **Ruben Brokkelkamp**
How Close Does It Get?: From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes

ILLC DS-2022-05: **Lwenn Bussière-Carae**
No means No! Speech Acts in Conflict

ILLC DS-2022-06: **Emma Mojet**
Observing Disciplines: Data Practices In and Between Disciplines in the 19th and Early 20th Centuries

ILLC DS-2022-07: **Freek Gerrit Witteveen**
Quantum information theory and many-body physics

ILLC DS-2023-01: **Subhasree Patro**
Quantum Fine-Grained Complexity

ILLC DS-2023-02: **Arjan Cornelissen**
Quantum multivariate estimation and span program algorithms

ILLC DS-2023-03: **Robert Paßmann**
Logical Structure of Constructive Set Theories

ILLC DS-2023-04: **Samira Abnar**
Inductive Biases for Learning Natural Language

ILLC DS-2023-05: **Dean McHugh**
Causation and Modality: Models and Meanings

ILLC DS-2023-06: **Jialiang Yan**
Monotonicity in Intensional Contexts: Weakening and: Pragmatic Effects under Modals and Attitudes

ILLC DS-2023-07: **Yiyan Wang**
Collective Agency: From Philosophical and Logical Perspectives

ILLC DS-2023-08: **Lei Li**
Games, Boards and Play: A Logical Perspective

ILLC DS-2023-09: **Simon Rey**
Variations on Participatory Budgeting

ILLC DS-2023-10: **Mario Giulianelli**
Neural Models of Language Use: Studies of Language Comprehension and Production in Context

ILLC DS-2023-11: **Guillermo Menéndez Turata**
Cyclic Proof Systems for Modal Fixpoint Logics

ILLC DS-2023-12: **Ned J.H. Wontner**
Views From a Peak: Generalisations and Descriptive Set Theory

ILLC DS-2024-01: **Jan Rooduijn**
Fragments and Frame Classes: Towards a Uniform Proof Theory for Modal Fixed Point Logics

ILLC DS-2024-02: **Bas Cornelissen**
Measuring musics: Notes on modes, motifs, and melodies

ILLC DS-2024-03: **Nicola De Cao**
Entity Centric Neural Models for Natural Language Processing

ILLC DS-2024-04: **Ece Takmaz**
Visual and Linguistic Processes in Deep Neural Networks: A Cognitive Perspective

ILLC DS-2024-05: **Fatemeh Seifan**
Coalgebraic fixpoint logic Expressivity and completeness result

ILLC DS-2024-06: **Jana Sotáková**
Isogenies and Cryptography

ILLC DS-2024-07: **Marco Degano**
Indefinites and their values

ILLC DS-2024-08: **Philip Verduyn Lunel**
Quantum Position Verification: Loss-tolerant Protocols and Fundamental Limits

ILLC DS-2024-09: **Rene Allerstorfer**
Position-based Quantum Cryptography: From Theory towards Practice

ILLC DS-2024-10: **Willem Feijen**
Fast, Right, or Best? Algorithms for Practical Optimization Problems

ILLC DS-2024-11: **Daira Pinto Prieto**
Combining Uncertain Evidence: Logic and Complexity

ILLC DS-2024-12: **Yanlin Chen**
On Quantum Algorithms and Limitations for Convex Optimization and Lattice Problems