ABSTRACT

Title of dissertation: Quantum algorithms for machine learning and optimization

Tongyang Li
Doctor of Philosophy, 2020

Dissertation directed by: Professor Andrew M. Childs
Department of Computer Science

The theories of optimization and machine learning answer foundational questions in computer science and lead to new algorithms for practical applications. While these topics have been extensively studied in the context of classical computing, their quantum counterparts are far from well-understood. In this thesis, we explore algorithms that bridge the gap between the fields of quantum computing and machine learning.

First, we consider general optimization problems with only function evaluations. For two core problems, namely general convex optimization and volume estimation of convex bodies, we give quantum algorithms as well as quantum lower bounds that constitute the quantum speedups of both problems to be polynomial compared to their classical counterparts.

We then consider machine learning and optimization problems with input data stored explicitly as matrices. We first look at semidefinite programs and provide quantum algorithms with polynomial speedup compared to the classical state-of-the-art. We then move to machine learning and give the optimal quantum algorithms

for linear and kernel-based classifications. To complement with our quantum algorithms, we also introduce a framework for quantum-inspired classical algorithms, showing that for low-rank matrix arithmetics there can only be polynomial quantum speedup.

Finally, we study statistical problems on quantum computers, with the focus on testing properties of probability distributions. We show that for testing various properties including ℓ^1-distance, ℓ^2-distance, Shannon and Rényi entropies, etc., there are polynomial quantum speedups compared to their classical counterparts. We also extend these results to testing properties of quantum states.

QUANTUM ALGORITHMS FOR MACHINE LEARNING AND OPTIMIZATION

by

Tongyang Li

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Andrew M. Childs, Chair/Advisor
Professor Edo Waks, Dean's representative
Assistant Professor Furong Huang
Adjunct Associate Professor Yi-Kai Liu
Assistant Professor Xiaodi Wu

Acknowledgements

First and foremost, I would like to thank my advisor, Andrew M. Childs, for his truly exceptional advice, encouragement, and support! Andrew's research and working style is a great standard for me to learn and has deeply influenced my attitude toward academia. His diligence for meeting me at least one hour per week is super helpful for my career: throughout the meetings, his extensive suggestions on research questions and being mature in academia (e.g., paper writing, talk preparation, academic plans, etc.) are probably the dominating reasons for my success during my graduate study.

I would also like to wholeheartedly thank Xiaodi Wu for his enormous and generous help on my career! Among the eight papers that are included in this thesis, I collaborated with him on five of them. In addition, Xiaodi's suggestions on choosing research topics and formulating academic plans have been extremely helpful for me since 2017. I also very enjoy our countless discussions throughout my graduate study.

I would like to especially thank Furong Huang for being in my thesis committee; in addition, when I got into the world of machine learning, I learned a lot from her course on spectral methods and reinforcement learning and her group meetings. I would like to especially thank Aravind Srinivasan for being in my preliminary exam committee, his generous help throughout my career, and for offering two fantastic courses on theoretical machine learning and randomized algorithms. I would also like to especially thank Yi-Kai Liu and Edo Waks for being in my thesis committee.

The work described in this thesis is the product of collaborations with many people. I am grateful to Fernando G.S.L. Brandão, Shouvanik Chakrabarti, Nai-Hui Chia, Andrew M. Childs, Shih-Han Hung, András Gilyén, Amir Kalev, Cedric Yen-Yu Lin, Han-Hsuan Lin, Krysta M. Svore, Ewin Tang, Chunhao Wang, and Xiaodi Wu for their contributions. Special thanks to Shouvanik and Chunhao for the countless time we spent together on various projects from top to bottom.

During my thesis research, I enjoyed my internship at the IBM T. J. Watson Research Center in the summer of 2018. I would like to specially thank David Gosset for his mentorship during the period; I particularly appreciate detailed discussions that helped me to learn state-of-the-art results on various topics. I would like to thank other research staff members at IBM, in particular Sergey Bravyi, Andrew W. Cross, Krzysztof Onak, and Kristan Temme. I would also like to my fellow internship students during that summer: Christopher Chamberland, Andrea Coladangelo, Pranav Mundada, Kanav Setia, and Leonard Wossnig.

I would also like to thank the hosts of my academic visits during my graduate study for their generous support: Richard Cleve, Runyao Duan, Giulio Chiribella, Aram W. Harrow, Elad Hazan, Xiongfeng Ma, Yaoyun Shi, Adam Smith, Xiaoming Sun, Ronald de Wolf, Penghui Yao, and Shengyu Zhang. Special thanks to Giulio for bringing me into quantum information science during my undergraduate study at Tsinghua University.

In addition to those mentioned above, I would also like to thank many others with whom I have had numerous illuminating discussions. I would especially like to thank Mark Bun, Yanjun Han, Cupjin Huang, Michael Newman, Rui Chao, Thomas

Vidick, Adrian Vladu, and Jialin Zhang.

I would like to specially thank Jianxin Chen for his extensive help when he was a postdoctoral scholar at the Joint Center for Quantum Information and Computer Science; his kindness support is one of the most important reason why I can easily get accustomed to the life at University of Maryland when I first came. I would also like to specifically thank Sheng Yang for being my roommate: we had a very enjoyable four years together, and Sheng also provided enormous help in my life at College Park.

I would like to thank postdoctoral scholars at the Joint Center for Quantum Information and Computer Science for their help during my graduate study, especially Jianxin Chen, Cedric Yen-Yu Lin, Guoming Wang, Xin Wang, Xingyao Wu, and Penghui Yao. I would also like to thank my fellow graduate students at QuICS, especially Abhinav Deshpande, Honghao Fu, Shih-Han Hung, Aaron Ostrander, and Yuan Su. In addition, I would like to thank the staff of the QuICS and the CS department, in particular Javiera Caceres, Tom Hurst, Jennifer Story, and Andrea Svejda, for their generous help.

Finally, I would like to thank my mother and father.

My graduate studies were supported by the Lanczos Graduate Fellowship, Dean's Fellowship, Natural Science Foundation CCF-1526380, IBM PhD Fellowship, QISE-NET Triplet Award (Natural Science Foundation DMR-1747426), and U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams program.

Table of Contents

Acknowledgements ii

Table of Contents v

List of Tables ix

List of Figures x

List of Abbreviations xi

1 Introduction 1
- 1.1 Quantum machine learning . . . 2
- 1.2 My contributions . . . 5
- 1.3 Preliminaries . . . 7

2 General Convex Optimization 11
- 2.1 Introduction . . . 11
  - 2.1.1 Convex optimization . . . 13
  - 2.1.2 Contributions . . . 16
  - 2.1.3 Overview of techniques . . . 19
- 2.2 Upper bound . . . 26
  - 2.2.1 Oracle definitions . . . 28
  - 2.2.2 A quantum algorithm for computing subgradients . . . 30
  - 2.2.3 Step 1: from membership to separation . . . 39
  - 2.2.4 Step 2: from separation to optimization . . . 40
- 2.3 Lower bound . . . 41
  - 2.3.1 Membership queries . . . 41
  - 2.3.2 Evaluation queries . . . 46
  - 2.3.3 Complete analysis of Algorithm 2.5 . . . 60
  - 2.3.4 Proof of our quantum lower bound on convex optimization . . . 67
  - 2.3.5 Side points on our quantum lower bound . . . 70
- 2.4 Conclusions and discussion . . . 73

3 Volume Estimation 76
- 3.1 Introduction . . . 76
  - 3.1.1 Contributions . . . 79
  - 3.1.2 Techniques . . . 81

3.1.2.1 Classical volume estimation framework . . . 81
3.1.2.2 Quantum algorithm for volume estimation . . . 84
3.1.3 Related work . . . 92
3.2 Preliminary tools . . . 99
3.2.1 Classical and quantum walks . . . 99
3.2.2 Quantum speedup of MCMC sampling via simulated annealing 101
3.2.3 Quantum Chebyshev inequality . . . 103
3.2.4 Hit-and-run walk . . . 107
3.3 Theory of continuous-space quantum walks . . . 111
3.3.1 Continuous-space quantum walk . . . 112
3.3.2 Stationary distribution . . . 117
3.4 Quantum speedup for volume estimation . . . 120
3.4.1 Review of classical algorithms for volume estimation . . . 120
3.4.2 Quantum algorithm for volume estimation . . . 123
3.4.3 Proof details of the quantum volume estimation algorithm . . 130
3.4.3.1 Pencil construction and the original convex body . . 130
3.4.3.2 Stationary states of consecutive steps . . . 133
3.4.3.3 Nondestructive mean estimation . . . 134
3.4.3.4 Error analysis . . . 137
3.4.4 Quantum algorithms for rounding logconcave densities . . . 150
3.5 Quantum lower bound for volume estimation . . . 155
3.6 Conclusions and discussion . . . 157

4 Semidefinite Programs 160
4.1 Introduction . . . 160
4.1.1 Quantum SDP solvers with optimal dependence on m and n . 163
4.1.2 Quantum SDP solvers with quantum inputs . . . 166
4.1.3 Related works on quantum SDP solvers . . . 172
4.1.4 Techniques . . . 172
4.1.5 Application: Efficient learnability of quantum states . . . 176
4.2 Feasibility of SDPs . . . 182
4.3 Fast quantum OR lemma . . . 187
4.4 Quantum SDP solver in the plain model . . . 190
4.5 Quantum SDP solver with quantum inputs . . . 195
4.5.1 The quantum input model . . . 195
4.5.2 Implementation of Oracle 6 – searching a violated constraint . 197
4.5.3 Quantum SDP solvers with quantum inputs . . . 199
4.5.4 Trace estimation . . . 203
4.5.5 Gibbs state preparation . . . 205
4.5.6 Lower bound for quantum SDP solvers with quantum inputs . 206
4.6 Application: Efficient learnability of quantum states . . . 208
4.6.1 Reduction from shadow tomography to SDP feasibility . . . 208
4.6.2 Finding the violated constraint using $\tilde{O}(\sqrt{m})$ gates . . . 209
4.6.3 Gate complexity of learning quantum states . . . 211
4.7 Gibbs sampling of low-rank Hamiltonians . . . 213

4.7.1 Computing the partition function . . . 214
4.7.2 Sampling from the Gibbs state . . . 218
4.8 Conclusions and discussion . . . 222

5 Linear and Kernel-based Classification 225
5.1 Introduction . . . 225
5.2 Linear classification . . . 233
5.2.1 Techniques . . . 233
5.2.2 Quantum speedup for multiplicative weights . . . 237
5.2.2.1 Quantum state preparation with oracles . . . 238
5.2.2.2 Implementing the quantum oracle for weight vectors 240
5.2.2.3 Proof of Theorem 5.2.1 . . . 241
5.2.3 Quantum speedup for online gradient descent . . . 245
5.3 Applications . . . 252
5.3.1 Kernel methods . . . 252
5.3.2 Quadratic machine learning problems . . . 255
5.3.2.1 Minimum enclosing ball . . . 255
5.3.2.2 ℓ_2-margin SVM . . . 256
5.4 Quantum lower bounds . . . 257
5.4.1 Linear classification . . . 258
5.4.2 Minimum enclosing ball (MEB) . . . 263
5.5 Conclusions and discussion . . . 269

6 Quantum-inspired Classical Machine Learning Algorithms 271
6.1 Revisiting semidefinite programming . . . 272
6.1.1 Motivations and contributions . . . 272
6.1.2 Techniques . . . 277
6.1.3 Related work . . . 282
6.2 Preliminary tools . . . 284
6.2.1 Sampling-based data structure . . . 284
6.2.2 Feasibility testing of SDPs . . . 286
6.3 Two new tools . . . 287
6.3.1 Weighted sampling . . . 287
6.3.2 Symmetric approximation of low-rank Hermitian matrices . . 297
6.4 Gibbs states . . . 301
6.4.1 Estimating matrices inner product . . . 302
6.4.2 Approximating the Gibbs state . . . 303
6.5 Main results: sampling-based SDP and shadow tomography solvers . 305
6.6 Generalization: a framework for quantum-inspired classical algorithms 310
6.7 Conclusions and discussion . . . 314

7 Distribution Property Testing 316
7.1 Introduction . . . 316
7.1.1 Problem statements . . . 318
7.1.2 New results . . . 322

7.1.3 New techniques . . . 324
7.1.4 Related works on distributional property testing . . . 328
7.2 Technical tools . . . 330
7.2.1 Amplitude estimation . . . 330
7.2.2 Quantum singular value transformation . . . 331
7.2.3 Polynomial approximations for singular value transformation . 333
7.2.4 Projected unitary encodings for singular value transformation 335
7.3 Results . . . 336
7.3.1 Shannon entropy estimation . . . 336
7.3.2 Tolerant testers for ℓ^2-closeness with purified query-access . . 339
7.3.3 ℓ^1-closeness testing with purified query-access . . . 344
7.3.4 Independence testing with purified query-access . . . 344
7.4 Rényi entropy estimation . . . 346
7.4.1 Overview . . . 346
7.4.2 Master algorithm . . . 356
7.5 Conclusions and discussion . . . 362

8 Conclusions and Future Work 364

Bibliography 367

List of Tables

1.1 Summary of quantum notations throughout this thesis. 10

2.1 Summary of classical and quantum complexities of convex optimization. 18

3.1 Summary of complexities of volume estimation, where n is the dimension of the convex body, ϵ is the multiplicative precision of volume estimation, and C_{MEM} is the cost of applying the membership oracle once. Total complexity refers to the cost of the of queries plus the number of additional arithmetic operations. 81

3.2 Summary of classical methods for estimating the volume of a convex body $\mathrm{K} \subset \mathbb{R}^n$ when $\epsilon = \Theta(1)$, where R, r are the radii of the balls centered at the origin that contain and are contained by the convex body, respectively. 95

7.1 Summary of sample and query complexity results of distributional property testing. Our new bounds are printed in **bold**. For classical distributions with quantum query-access we prove (almost) matching upper and lower bounds for ℓ^2-testing, and improve the previous best complexity $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^{2.5}\right)$ for ℓ^1-testing by [208] and $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^{2}\right)$ for Shannon entropy estimation by [192]. Note that Ref. [65] imply that in this model quantum speed-ups are at most cubic. The results for Rényi entropy estimation are summarized in Table 7.3 separately. . . 323

7.2 Complexities of Shannon / von Neumann entropy estimation with constant precision. It seems that the n-dependence is roughly quadratically higher for quantum distributions, while coherent quantum access gives a quadratic advantage for both classical and quantum distributions. This suggests that our entropy estimation algorithm has essentially optimal n-dependence for density operators with purified access, however we do not have a matching lower bound yet. 324

7.3 Classical and quantum query complexities of estimating α-Rényi entropy $H_\alpha(p)$, assuming $\epsilon = \Theta(1)$. © 2019 IEEE. 352

List of Figures

2.1 Smoothed hypercube of dimension 3. 70

3.1 The structure of our quantum volume estimation algorithm. The purple frames represent the novel techniques that we propose, the yellow frame represents the known technique from [130], and the green frame at the center represents our quantum algorithm. 92
3.2 The quantum circuit for amplitude estimation. 104
3.3 The quantum circuit for Algorithm 3.3 (assuming well-roundedness). Here $U_{\mathrm{CB},i}$ is the circuit of the quantum Chebyshev inequality (Theorem 3.2.3) in the i^{th} iteration and $U_{i,l}$ is $\pi/3$-amplitude amplification from $|\pi_i\rangle$ to $|\pi_{i+1}\rangle$. 127
3.4 The quantum circuit for nondestructive BasicEst. 138
3.5 The quantum phase estimation circuit. Here W is a unitary operator with eigenvector $|\psi_j\rangle$; in $\pi/3$-amplitude estimation it is the quantum walk operator W_i' in (3.4.45). 147

6.1 Illustration of a data structure that allows for sampling access to a row of $M \in \mathbb{C}^{4\times 4}$. 286

7.1 Visualization of classical and quantum query complexity of $H_\alpha(p)$. The x-axis represents α and the y-axis represents the exponent of n. Red curves and points represent quantum upper bounds. Green curves and points represent classical tight bounds. Blue curve represents quantum lower bounds. Purple points represent quantum tight bounds. © 2019 IEEE. 352

List of Abbreviations and Notations

LP	Linear program
MCMC	Markov chain Monte Carlo
MEB	Minimum enclosing ball
MMW	Matrix multiplicative weights
SDP	Semidefinite program
SVM	Support vector machine
QFT	Quantum Fourier transform
QRAM	Quantum random access memory

$\mathbf{1}_n$	n-dimensional all-one vector
$\mathbb{B}_d$	d-dimensional ℓ_2-norm unit ball, $\{a \in \mathbb{R}^d \mid \sum_{i=1}^d \lvert a_i \rvert^2 \leq 1\}$
$\tilde{O}$	O that suppresses poly-logarithmic factors
$[n]$	$\{1, 2, \ldots, n\}$

Chapter 1: Introduction

Machine learning has been the core of the developments in computer science in the past decade (2010-2019), with ubiquitous applications including computer vision, natural language processing, bioinformatics, etc. Although machine learning techniques have been shown to be extremely powerful in practice, current research suffers from interpretability and end-to-end understanding of trained machine learning models. To address such issue, it has become a main interest to explore the theoretical foundations of machine learning. Recently, significant progress in theory has been made, e.g., see the surveys in convex optimization [61], online learning [143], nonconvex optimization [149], deep learning theory [252], etc. Nevertheless, there are still various challenges in theoretical machine learning that demand future research.

On the other hand, quantum computing is another rapidly advancing technology. In particular, the capability of quantum computers has been dramatically increasing and recently reached "quantum supremacy" [230] by Google in October, 2019 [34]. However, Google's experiment was only performed for a specific sampling task, and at the moment the noise of quantum gates prevents current quantum computers from being useful in practice. Considering this, it is also of

significant interest to understand quantum computing from a theoretical perspective for paving its way to future applications. There have been Shor's algorithm for integer factorization [245] and Grover's algorithm for unstructured search [128] which have provable advantages compared to their classical counterparts, and there are also quite a few other quantum algorithms for algebraic problems, number theory, boolean functions, graph problems, etc.; see [79, 83] for an overview of existing quantum algorithms.

1.1 Quantum machine learning

More recently, "quantum machine learning" (see e.g. [51, 243]) has become a popular research topic. Currently, theoretical research has been conducted in a few different directions:

First, there has been research about quantum computational learning theory. A main framework of computational learning theory is *probably approximately correct learning (PAC learning)* [264]. To be more specific, we are given a concept class $\mathcal{C} \subseteq \{f : \{0,1\}^n \to \{0,1\}\}$ and a concept $c \in \mathcal{C}$, and the goal is to probably approximate c from samples of the form $(x, c(x))$, where x is drawn according to some unknown distribution D over $\{0,1\}^n$. Mathematically, for all concepts $c \in \mathcal{C}$ and distribution D, an (ϵ, δ)-learner outputs a guess c' for c such that with probability at least $1-\delta$, we have $\Pr_{x\sim D}[c'(x) \neq c(x)] \leq \epsilon$. Classically, it is shown by [52, 135] that the sample complexity of such learners is $\Theta(\frac{d}{\epsilon} + \frac{\log(1/\delta)}{\epsilon})$, where d is the *VC-dimension* of $\mathcal{C}$ [268]. Quantumly, given a sample oracle that gives the

state $\sum_{x\in\{0,1\}^n}\sqrt{D(x)}|x,c(x)\rangle$, a series of works [31, 36, 37, 60, 278] studied the quantum sample complexity of PAC learning, which culminates in [33] showing that the quantum sample complexity of PAC learning is also $\Theta(\frac{d}{\epsilon}+\frac{\log(1/\delta)}{\epsilon})$, same as the classical counterpart. Similar result also holds for agnostic learning. See also the survey [32].

Second, Harrow, Hassidim and Lloyd (HHL) proposed a quantum algorithm for linear systems [137]. Given quantum oracles for a matrix $A\in\mathbb{C}^{n\times n}$ and a vector $b\in\mathbb{C}^n$, the goal is to prepare a state $|x\rangle$ (up to normalization) such that $Ax\approx b$. To be more specific, suppose that A is a d-sparse Hermitian matrix with condition number κ; A is modeled by a quantum oracle O_A that, on input (x,j), gives the location of the j^{th} nonzero entry in row x, denoted as y, and the value $A_{x,y}$. We are also given a quantum oracle for preparing $|b\rangle:=\frac{\sum_i b_i|i\rangle}{\|\sum_i b_i|i\rangle\|}$. Let $\vec{x}:=A^{-1}\vec{b}$ and $|x\rangle:=\frac{\sum_i x_i|i\rangle}{\|\sum_i x_i|i\rangle\|}$. Then $|x\rangle$ can be prepared within additive error ϵ with high success probability, using only $\tilde{O}(d\kappa^2/\epsilon)$ queries to O_A and $\tilde{O}(d\kappa)$ copies of $|b\rangle$, and its running time is $\tilde{O}(d^2\kappa^2/\epsilon)$. These bounds were later improved by [18] with $\tilde{O}(\kappa)$ condition number dependence and [81] with $\text{poly}(\log(1/\epsilon))$ error dependence.

However, there are caveats to turn the techniques of [81, 137] into fast quantum algorithms for machine learning (see [2]). On the one hand, it is nontrivial to obtain an efficient oracle for preparing $|b\rangle$. It is assumed in [81, 137] that we can prepare $|b\rangle$ using $\text{poly}(\log N)$ 2-qubit gates; this may assume additional restrictions on b_i, and in such circumstances the classical counterparts may also become easier. An example is the cluster assignment problem [194], where we are given states $|u\rangle,|v_1\rangle,|v_2\rangle,\ldots,|v_M\rangle$ in an N-dimensional Hilbert space and we want

to approximate $\frac{1}{M}\sum_{i=1}^{M}\langle u|v_i\rangle$; [194] showed how to achieve this within error ϵ in $O(\text{poly}(\log MN)/\epsilon)$ steps, exponentially faster than classical inner-product computations. However, in the first place we need to efficiently prepare all these states, whose amplitudes have to be restricted; one possible choice is to make the distributions close to uniform, but in this case we can also estimate $\frac{1}{M}\sum_{i=1}^{M}\langle u|v_i\rangle$ using a classical random sampling algorithm in $O(\log MN/\epsilon^2)$ steps, which is also poly-logarithmic in both M and N. In all, to justify the quantum speedup by [81, 137], a fair comparison is needed under the same assumption for preparing $|b\rangle$.

On the other hand, the output of the algorithms in [81, 137] is a state close to $|x\rangle$, which is different from writing down all the coordinates of $\vec{x}$ that already takes n steps; in particular, some goals are nontrivial to be achieved by constant copies of $|x\rangle$. For instance, learning the value of a specific entry x_i within error ϵ requires $\Omega(\frac{\sqrt{n}}{\epsilon})$ copies of $|x\rangle$, which has at most a polynomial quantum speedup. Therefore, it is not totally clear how quantum machine learning algorithms using HHL, such as quantum data fitting [270], quantum support vector machine [232], etc., are going to provide quantum speedup in practice.

Third, initiated from the quantum algorithm for recommendation systems by Kerenidis and Prakash [167], there have been quantum machine learning algorithms using the quantum random access memory (QRAM). To be more specific, given a matrix $A \in \mathbb{C}^{m\times n}$, they assume the existence of two oracles U and V such that

$$U|0\rangle = \sum_{i\in[m]} \frac{\|A(i,\cdot)\|}{\|A\|_F}|i\rangle, \qquad V|i\rangle|0\rangle = |i\rangle \sum_{j\in[n]} \frac{|A(i,j)|}{\|A(i,\cdot)\|}|j\rangle \quad \forall\, i \in [m]. \tag{1.1.1}$$

Such oracles give the ability to sample from matrices; together with randomized linear algebra tools (e.g. [12]), they find a rank-k approximation of A with constant error in time $O(\text{poly}(k)\,\text{poly}(\log mn))$, an exponential quantum speedup in both m and n. Furthermore, follow-up works have extended the quantum speedups using the QRAM data structure to classification [165], interior-point methods [168], cone programming [169], etc. However, the preprocessing time of the QRAM data structure is $O(w \log^2(mn))$ where w is the number of nonzero elements in A; in general $A = \Omega(m), \Omega(n)$, which ruins the claimed exponential quantum speedup. It is also possible to replace the QRAM data structure by a classical sampling-based data structure; see Section 1.2 below and also Chapter 6.

1.2 My contributions

In spite of the existence of the quantum machine learning algorithms mentioned above, the quantum theories of machine learning and optimization are still far from well-understood compared to their classical counterparts. This thesis delves into quantum algorithms for machine learning and optimization with provable guarantees, under the considerations from the following three perspectives:

Problems with implicit oracles. In many cases, we do not have the detailed structure of the input but some general information, e.g., the function value at a point, whether or not a point is in the definition domain, etc. For instance, the general convex optimization problem contains a convex body $\mathcal{C} \subseteq \mathbb{R}^n$ and a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$, and the goal is to find a $\tilde{x}$ such that $f(\tilde{x}) \leq \min_{x\in\mathcal{C}} f(x) +$

ϵ. The only accesses to the problem are the evaluation oracle of f (i.e., input x, output $f(x)$) and the membership oracle of $\mathcal{C}$ (i.e., input x, output whether $x \in \mathcal{C}$ or not). In Chapter 2, we will give a quantum algorithm for this general convex optimization problem, based on my paper [67] in the conference QIP 2019 and the journal Quantum in 2020. In Chapter 3, we will study the problem of estimating the volume of $\mathcal{C}$ using its membership oracle, based on my paper [66] in QIP 2020.

Problems with explicit data structures. It is also natural to investigate problems where the input data are explicitly stored as matrices, and seek for more fine-grained algorithms and analyses compared to the first part. As mentioned in Section 1.1, there has been progress along this line, but this thesis explores more problems and techniques along this line. To be more specific, we research on two categories of problems: one is semidefinite programming as an instance from optimization (Chapter 4, based on my paper [55] in QIP 2019 and ICALP 2019); it improves upon previous quantum SDP solvers as well as proposing an application to learning quantum states. The other is linear or kernel-based classification as an instance from machine learning (Chapter 5, based on my paper [191] in ICML 2019); it has the advantage of outputting purely (sparse) classical classifiers, hence overcoming the caveats of HHL-type machine learning algorithms as introduced in Section 1.1. In addition, as QRAM has been a main tool in previous quantum machine learning algorithms, we study its limitations by giving quantum-inspired classical algorithms using a classical sampling data structure that resembles (1.1.1). We achieve comparable classical complexities for solving low-rank linear systems,

SDP, recommendation systems, etc. in Chapter 6, based on my papers [77] in submission and [76] in QIP 2020 and to appear in STOC 2020.

Problems with samples. From a statistical perspective, an important question is to infer information from classical distributions or quantum states. Previous works have been focusing on quantum state tomography [132] and learning the spectrum of quantum states [219–221]; in this thesis, we focus on testing properties of discrete probability distributions on quantum computers with speedups compared to their classical counterparts, giving fast quantum algorithms for entropy estimation, ℓ^1-closeness testing, ℓ^2-closeness testing, etc. (see Chapter 7, based on my papers [192] in IEEE Trans. Inf. Theory 2019 and [121] in ITCS 2020).

1.3 Preliminaries

In this section, we define some of the basic notions used in this thesis.

Basic notations in quantum computing. We briefly summarize the definitions and notations of quantum computing. More details can be found at standard textbooks, e.g., [163, 175, 217].

Quantum mechanics can be formulated in terms of linear algebra. Given any complex Euclidean space $\mathbb{C}^d$, we define its computational basis by $\{\vec{e}_0, \ldots, \vec{e}_{d-1}\}$, where $\vec{e}_i = (0, \ldots, 1, \ldots, 0)^\top$ with the $(i+1)^{\text{th}}$ entry being 1 and other entries being 0. These basic vectors are usually written by *Dirac notation*: we write $\vec{e}_i$ as $|i\rangle$ (called a "ket"), and write $\vec{e}_i^\top$ as $\langle i|$ (called a "bra").

Quantum states with dimension d are represented by unit vectors in $\mathbb{C}^d$: i.e., a vector $|v\rangle = (v_0, \ldots, v_{d-1})^\top$ is a quantum state if $\sum_{i=0}^{d-1} |v_i|^2 = 1$. For each i, v_i is called the *amplitude* in $|i\rangle$. If there are at least two non-zero amplitudes, quantum state $|v\rangle$ is in *superposition* of the computational basis, a fundamental feature in quantum mechanics.

Tensor product of quantum states is their Kronecker product: if $|u\rangle \in \mathbb{C}^{d_1}$ and $|v\rangle \in \mathbb{C}^{d_2}$, then $|u\rangle \otimes |v\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$ is

$$|u\rangle \otimes |v\rangle = (u_0 v_0, u_0 v_1, \ldots, u_{d_1-1} v_{d_2-1})^\top. \tag{1.3.1}$$

The basic element in classical computers is one bit; similarly, the basic element in quantum computers is one *qubit*, which is a quantum state in $\mathbb{C}^2$. Mathematically, a qubit state can be written as $a|0\rangle + b|1\rangle$ for some $a, b \in \mathbb{C}$ such that $|a|^2 + |b|^2 = 1$. An n-qubit state can be written as $|v_1\rangle \otimes \cdots \otimes |v_n\rangle$, where each $|v_i\rangle$ ($i \in [n]$) is a qubit state; n-qubit states are in a Hilbert space of dimension 2^n.

Operations in quantum computation are *unitary transformations* and can be stated in the circuit model[1] where a *k-qubit gate* is a unitary matrix in $\mathbb{C}^{2^k}$. It is known that two-qubit gates are *universal*, i.e., every n-qubit gate can be written as composition of a sequence of two-qubit gates. Therefore, one usually refers to the *number of two-qubit gates* as the *gate complexity* of quantum algorithms.

[1] Uniform circuits have equivalent computational power as Turing machines; however, they are more convenient to use in quantum computation.

Quantum oracles. Classically, any boolean computation can be made reversible by replacing any irreversible gate (such as AND or OR) $x \mapsto f(x)$ by the reversible gate $(x, y) \mapsto (x, y \oplus f(x))$; as a result, if the input is $(x, 0)$, the output will be $(x, f(x))$. Quantumly, unitary transformations are invertible, i.e., $U^{-1} = U^\dagger$. In other words, every unitary transformation is reversible. It is hence common to choose the same strategy as the classical counterpart: if we can efficiently compute the function $x \mapsto f(x)$ on a classical computer, on a quantum computer we can efficiently implement the quantum oracle O_f such that

$$O_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle \quad \forall\, x. \tag{1.3.2}$$

The main difference is that quantum computing allows access to different parts of the input data in *superposition*, which is the essence of quantum speedups.

In general, access to other objects can be described as instantiations of (1.3.2). For instance, the function f can not only be boolean but also with domains $f\colon \mathbb{R}^n \to \mathbb{R}$; in this case, the registers are formulated by floating numbers. For matrices, we exploit an oracle O_X such that

$$O_X(|i\rangle \otimes |j\rangle \otimes |0\rangle) = |i\rangle \otimes |j\rangle \otimes |X_{ij}\rangle \tag{1.3.3}$$

for any $z \in \mathbb{R}$ and i, j from the rows and columns of X, respectively. Intuitively, O_X reads the entry X_{ij} and stores it in the third register; it is a natural unitary generalization of classical random access to X, or in cases when any entry of X can

be efficiently read.

We summarize the quantum notations in Table 1.1.

	Classical	Quantum
Ket and bra	$\vec{e}_i$ and $\vec{e}_i^{\top}$	$\|i\rangle$ and $\langle i\|$
Basis	$\{\vec{e}_0, \ldots, \vec{e}_{d-1}\}$	$\{\|0\rangle, \ldots, \|d-1\rangle\}$
State	$\vec{v} = (v_0, \ldots, v_{d-1})^{\top}$	$\|v\rangle = \sum_{i=0}^{d-1} v_i \|i\rangle$
Tensor	$\vec{u} \otimes \vec{v}$	$\|u\rangle \otimes \|v\rangle$ or $\|u\rangle\|v\rangle$
Function oracle	$f(x), f\colon \mathbb{R}^n \to \mathbb{R}$	$O_f\|x\rangle\|0\rangle = \|x\rangle\|f(x)\rangle$
Matrix oracle	$X = (X_{ij})$	$O_X\|i\rangle\|j\rangle\|0\rangle = \|i\rangle\|j\rangle\|X_{ij}\rangle$

Table 1.1: Summary of quantum notations throughout this thesis.

Quantum complexity measure. *Quantum gate complexity* is defined as the total counts of two-qubit gates in a quantum algorithm (a two-qubit gate is a tensor product of a one- or two-qubit operator with the identity operator on the remaining qubits). A quantum algorithm is efficient if it can be described by a quantum circuit with a number of two-qubit gates that is polynomial in the number of qubits needed to write down the input.

Quantum query complexity is defined as the total counts of oracle queries. The main advantage of considering quantum query complexity is that if we have an efficient quantum algorithm for an explicit computational problem in query complexity, then if we are given an explicit circuit realizing the black-box transformation, we will have an efficient quantum algorithm for the problem. Furthermore, there are tools for proving lower bounds on the number of quantum queries needed to solve a given problem.

Chapter 2: General Convex Optimization[1]

First, we study general convex optimization on quantum computers.

2.1 Introduction

Convex optimization has been a central topic in the study of mathematical optimization, theoretical computer science, and operations research over the last several decades. On the one hand, it has been used to develop numerous algorithmic techniques for problems in combinatorial optimization, machine learning, signal processing, and other areas. On the other hand, it is a major class of optimization problems that admits efficient classical algorithms [54, 127]. Approaches to convex optimization include the ellipsoid method [127], interior-point methods [93, 162], cutting-plane methods [164, 261], and random walks [158, 199].

The fastest known classical algorithm for general convex optimization solves an instance of dimension n using $\tilde{O}(n^2)$ queries to oracles for the convex body and the objective function, and runs in time $\tilde{O}(n^3)$ [183].[2] The novel step of [183] is a construction of a separation oracle by a subgradient calculation with $O(n)$ objective

[1] This chapter is based on the paper [67] under the permission of all the authors.

[2] The notation $\tilde{O}$ suppresses poly-logarithmic factors in n, R, r, ϵ, i.e., $\tilde{O}(f(n)) = f(n)\log^{O(1)}(nR/r\epsilon)$.

function calls and $O(n)$ extra time. It then relies on a reduction from optimization to separation that makes $\tilde{O}(n)$ separation oracle calls and runs in time $\tilde{O}(n^3)$ [184]. Although it is unclear whether the query complexity of $\tilde{O}(n^2)$ is optimal for all possible classical algorithms, it is the best possible result using the above framework. This is because it takes $\tilde{\Omega}(n)$ queries to compute the (sub)gradient (see [67, Lemma A.1]) and it also requires $\Omega(n)$ queries to produce an optimization oracle from a separation oracle (see [214] and [213, Section 10.2.2]).

It is natural to ask whether quantum computers can solve convex optimization problems faster. Recently, there has been significant progress on quantum algorithms for solving a special class of convex optimization problems called semidefinite programs (SDPs). SDPs generalize the better-known linear programs (LPs) by allowing positive semidefinite matrices as variables. For an SDP with n-dimensional, s-sparse input matrices and m constraints, the best known classical algorithm [184] finds a solution in time $\tilde{O}(m(m^2 + n^\omega + mns)\operatorname{poly}\log(1/\epsilon))$, where ω is the exponent of matrix multiplication and ϵ is the accuracy of the solution. Brandão and Svore gave the first quantum algorithm for SDPs with worst-case running time $\tilde{O}(\sqrt{mn}s^2(Rr/\varepsilon)^{32})$, where R and r upper bound the norms of the optimal primal and dual solutions, respectively [56]. Compared to the aforementioned classical SDP solver [184], this gives a polynomial speedup in m and n. Van Apeldoorn et al. [24] further improved the running time of a quantum SDP solver to $\tilde{O}(\sqrt{mn}s^2(Rr/\epsilon)^8)$, which was subsequently improved to $\tilde{O}\big((\sqrt{m}+\sqrt{n}(Rr/\epsilon))s(Rr/\epsilon)^4\big)$ [23, 55]. The latter result is tight in the dependence of m and n since there is a quantum lower bound of $\Omega(\sqrt{m}+\sqrt{n})$ for constant R, r, s, ϵ [56].

However, semidefinite programming is a structured form of convex optimization that does not capture the problem in general. In particular, SDPs are specified by positive semidefinite matrices, and their solution is related to well-understood tasks in quantum computation such as solving linear systems (e.g., [81, 137]) and Gibbs sampling (e.g., [23, 55]). General convex optimization need not include such structural information, instead only offering the promise that the objective function and constraints are convex. Currently, little is known about whether quantum computers could provide speedups for general convex optimization. Our goal is to shed light on this question.

2.1.1 Convex optimization

We consider the general minimization problem $\min_{x\in K} f(x)$, where $K \subseteq \mathbb{R}^n$ is a convex set and $f\colon K \to \mathbb{R}$ is a convex function. We assume we are given upper and lower bounds on the function values, namely $m \leq \min_{x\in K} f(x) \leq M$, and inner and outer bounds on the convex set K, namely

$$B_2(0, r) \subseteq K \subseteq B_2(0, R), \tag{2.1.1}$$

where $B_2(x, l)$ is the ball of radius l in L_2 norm centered at $x \in \mathbb{R}^n$. We ask for a solution $\tilde{x} \in K$ with precision ϵ, in the sense that

$$f(\tilde{x}) \leq \min_{x\in K} f(x) + \epsilon. \tag{2.1.2}$$

We consider the very general setting where the convex body K and convex function f are only specified by oracles. In particular, we have:

- A *membership oracle* O_K for K, which determines whether a given $x \in \mathbb{R}^n$ belongs to K;
- An *evaluation oracle* O_f for f, which outputs $f(x)$ for a given $x \in K$.

Convex optimization has been well-studied in the model of membership and evaluation oracles since this provides a reasonable level of abstraction of K and f, and it helps illuminate the algorithmic relationship between the optimization problem and the relatively simpler task of determining membership [127, 183, 184]. The efficiency of convex optimization is then measured by the number of queries to the oracles (i.e., the *query complexity*) and the total number of other elementary gates (i.e., the *gate complexity*).

It is well known that a general bounded convex optimization problem is equivalent to one with a linear objective function over a different bounded convex set. In particular, if promised that $\min_{x \in K} f(x) \leq M$, the convex optimization problem is equivalent to the problem

$$\min_{x' \in \mathbb{R},\, x \in K} x' \quad \text{such that} \quad f(x) \leq x' \leq M. \tag{2.1.3}$$

Observe that a membership query to the new convex set

$$K' := \{(x', x) \in \mathbb{R} \times K \mid f(x) \leq x' \leq M\} \tag{2.1.4}$$

can be implemented with one query to the membership oracle for K and one query to the evaluation oracle for f. Thus the ability to optimize a linear function

$$\min_{x \in K} c^T x \tag{2.1.5}$$

for any $c \in \mathbb{R}^n$ and convex set $K \subseteq \mathbb{R}^n$ is essentially equivalent to solving a general convex optimization problem. A procedure to solve such a problem for any specified c is known as an *optimization oracle*. Thus convex optimization reduces to implementing optimization oracles over general convex sets (Lemma 2.2.1). The related concept of a *separation oracle* takes as input a point $p \notin K$ and outputs a hyperplane separating p from K.

In the quantum setting, we model oracles by unitary operators instead of classical procedures, following (1.3.2). In particular, in the quantum model of membership and evaluation oracles, we are promised to have unitaries O_K and O_f s.t.

- For any $x \in \mathbb{R}^n$, $O_K|x, 0\rangle = |x, \delta[x \in K]\rangle$, where $\delta[P]$ is 1 if P is true and 0 if P is false;
- For any $x \in \mathbb{R}^n$, $O_f|x, 0\rangle = |x, f(x)\rangle$.

In other words, we allow coherent superpositions of queries to both oracles. If the classical oracles can be implemented by explicit circuits, then the corresponding quantum oracles can be implemented by quantum circuits of about the same size, so the quantum query model provides a useful framework for understanding the quantum complexity of convex optimization.

2.1.2 Contributions

We now describe the main contributions of this paper. Our first main result is a quantum algorithm for optimizing a convex function over a convex body. Specifically, we show the following:

Theorem 2.1.1. *There is a quantum algorithm for minimizing a convex function f over a convex set $K \subseteq \mathbb{R}^n$ using $\tilde{O}(n)$ queries to an evaluation oracle for f, $\tilde{O}(n)$ queries to a membership oracle for K, and $\tilde{O}(n^3)$ additional quantum gates.*

Recall that the state-of-the-art classical algorithm [183] for general convex optimization with evaluation and membership oracles uses $\tilde{O}(n^2)$ queries to each. Therefore, our algorithm provides a quadratic improvement over the best known classical result. While the query complexity of [183] is not known to be tight, it is the best possible result that can be achieved using subgradient computation to implement a separation oracle, as discussed above.

The proof of Theorem 2.1.1 follows the aforementioned classical strategy of constructing a separating hyperplane for any given point outside the convex body [183]. We find this hyperplane using a fast quantum algorithm for gradient estimation using $\tilde{O}(1)$ evaluation queries,[3] as first proposed by Jordan [157] and later refined by [120] with more rigorous analysis. However, finding a suitable hyperplane in general requires calculating approximate subgradients of convex functions that may not be differentiable, whereas the algorithms in [157] and [120] both require bounded second derivatives or more stringent conditions. To address this issue, we

[3]Here $\tilde{O}(1)$ has the same definition as Footnote 2, i.e., $\tilde{O}(1) = \log^{O(1)}(nR/r\epsilon)$.

introduce classical randomness into the algorithm to produce a suitable approximate subgradient with $\tilde{O}(1)$ evaluation queries, and show how to use such an approximate subgradient in the separation framework to produce a faster quantum algorithm.

Our new quantum algorithm for subgradient computation is the source of the overall quantum speedup and establishes a separation in query complexity for the subgradient computation between quantum ($\tilde{O}(1)$) and classical ($\tilde{\Omega}(n)$ [67, Lemma A.1]) algorithms. This subroutine is also of independent interest, for instance in quantum algorithms based on gradient descent and its variants (e.g., [166, 233]).

Our techniques for finding an approximate subgradient only require an approximate oracle for the function to be differentiated. Theorem 2.1.1 also applies if the membership and evaluation oracles are given with error that is polynomially related to the required precision in minimizing the convex function (see Theorem 2.2.6). Precise definitions for these oracles with error can be found in Section 2.2.1.

On the other hand, we also aim to establish corresponding quantum lower bounds to understand the potential for quantum speedups for convex optimization. To this end, we prove:

Theorem 2.1.2. *There exists a convex body $K \subseteq \mathbb{R}^n$, a convex function f on K, and a precision $\epsilon > 0$, such that a quantum algorithm needs at least $\Omega(\sqrt{n})$ queries to a membership oracle for K and $\Omega(\sqrt{n}/\log n)$ queries to an evaluation oracle for f to output a point $\tilde{x}$ satisfying*

$$f(\tilde{x}) \leq \min_{x \in K} f(x) + \epsilon \tag{2.1.6}$$

with high success probability (say, at least 0.8).

We establish the query lower bound on the membership oracle by reductions from search with wildcards [20]. The lower bound on evaluation queries uses a similar reduction, but this only works for an evaluation oracle with low precision. To prove a lower bound on precise evaluation queries, we propose a discretization technique that relates the difficulty of the continuous problem to a corresponding discrete one. This approach might be of independent interest since optimization problems naturally have continuous inputs and outputs, whereas most previous work on quantum lower bounds focuses on discrete inputs. Using this technique, we can simulate one perfectly precise query by one low-precision query at discretized points, thereby establishing the evaluation lower bound as claimed in Theorem 2.1.2. As a side point, this evaluation lower bound holds even for an unconstrained convex optimization problem on $\mathbb{R}^n$, which might be of independent interest since this setting has also been well-studied classically [54, 213–215].

We summarize our main results in Table 2.1.

	Classical bounds	Quantum bounds (this paper)
Membership queries	$\tilde{O}(n^2)$ [183], $\Omega(n)$ [182]	$\tilde{O}(n)$, $\Omega(\sqrt{n})$
Evaluation queries	$\tilde{O}(n^2)$ [183], $\Omega(n)$ [182]	$\tilde{O}(n)$, $\tilde{\Omega}(\sqrt{n})$
Time complexity	$\tilde{O}(n^3)$ [183]	$\tilde{O}(n^3)$

Table 2.1: Summary of classical and quantum complexities of convex optimization.

2.1.3 Overview of techniques

Upper bound. To prove our upper bound result in Theorem 2.1.1, we use the well-known reduction from general convex optimization to the case of a linear objective function, which simplifies the problem to implementing an optimization oracle using queries to a membership oracle (Lemma 2.2.1). For the reduction from optimization to membership, we follow the best-known classical result in [183] which implements an optimization oracle using $\tilde{O}(n^2)$ membership queries and $\tilde{O}(n^3)$ arithmetic operations. In [183], the authors first show a reduction from separation oracles to membership oracles that uses $\tilde{O}(n)$ queries and then use a result from [184] to implement an optimization oracle using $\tilde{O}(n)$ queries to a separation oracle, giving an overall query complexity of $\tilde{O}(n^2)$.

The reduction from separation to membership involves the calculation of a *height function* defined by the authors (see Eq. (2.2.27)), whose evaluation oracle can be implemented in terms of the membership oracle of the original set. A separating hyperplane is determined by computing a subgradient, which already takes $\tilde{O}(n)$ queries. In fact, it is not hard to see that any classical algorithm requires $\tilde{\Omega}(n)$ classical queries (see [67, Lemma A.2]), so this part of the algorithm cannot be improved classically. The possibility of using the quantum Fourier transform to compute the gradient of a function using $\tilde{O}(1)$ evaluation queries ([120, 157]) suggests the possibility of replacing the subgradient procedure with a faster quantum algorithm. However, the techniques described in [120, 157] require the function in question to have bounded second (or even higher) derivatives, and the height func-

tion is only guaranteed to be Lipschitz continuous (Definition 2.2.9) and in general is not even differentiable.

To compute subgradients of general (non-differentiable) convex functions, we introduce classical randomness (taking inspiration from [183]) and construct a quantum subgradient algorithm that uses $\tilde{O}(1)$ queries. Our proof of correctness (Section 2.2.2) has three main steps:

1. We analyze the average error incurred when computing the gradient using the quantum Fourier transform. Specifically, we show that this approach succeeds if the function has bounded second derivatives in the vicinity of the point where the gradient is to be calculated (see Algorithm 2.1, Algorithm 2.2, and Lemma 2.2.3). Some of our calculations are inspired by [120].

2. We use the technique of *mollifier functions* (a common tool in functional analysis [146], suggested to us by [182] in the context of [183]) to show that it is sufficient to treat infinitely differentiable functions (the mollified functions) with bounded first derivatives (but possibly large second derivatives). In particular, it is sufficient to output an approximate gradient of the mollified function at a point near the original point where the subgradient is to be calculated (see Lemma 2.2.4).

3. We prove that convex functions with bounded first derivatives have second derivatives that lie below a certain threshold with high probability for a random point in the vicinity of the original point (Lemma 2.2.5). Furthermore, we show that a bound on the second derivatives can be chosen so that the

smooth gradient calculation techniques work on a sufficiently large fraction of the neighborhood of the original point, ensuring that the final subgradient error is small (see Algorithm 2.3 and Theorem 2.2.2).

The new quantum subgradient algorithm is then used to construct a separation oracle as in [183] (and a similar calculation is carried out in Theorem 2.2.3). Finally the reduction from [184] is used to construct the optimization oracle using $\tilde{O}(n)$ separation queries. From Lemma 2.2.1, this shows that the general convex optimization problem can be solved using $\tilde{O}(n)$ membership and evaluation queries and $\tilde{O}(n^3)$ gates.

Lower bound. We prove our quantum lower bounds on membership and evaluation queries separately before showing how to combine them into a single optimization problem. Both lower bounds work over n-dimensional hypercubes.

In particular, we prove both lower bounds by reductions from search with wildcards [20]. In this problem, we are given an n-bit binary string s and the task is to determine all bits of s using wildcard queries that check the correctness of any subset of the bits of s: more formally, the input in the wildcard model is a pair (T, y) where $T \subseteq [n]$ and $y \in \{0,1\}^{|T|}$, and the query returns 1 if $s_{|T} = y$ (here the notation $s_{|T}$ represents the subset of the bits of s restricted to T). Ambainis and Montanaro [20] showed that the quantum query complexity of search with wildcards is $\Omega(\sqrt{n})$.

For our lower bound on membership queries, we consider a simple objective function, the sum of all coordinates $\sum_{i=1}^{n} x_i$. In other words, we take $c = \mathbf{1}^n$ in

(2.1.5). However, the position of the hypercube is unknown, and to solve the optimization problem (formally stated in Definition 2.3.1), one must use the membership oracle to locate it.

Specifically, the hypercube takes the form $\bigtimes_{i=1}^{n}[s_i - 2, s_i + 1]$ ($\bigtimes$ being the Cartesian product) for some offset binary string $s \in \{0,1\}^n$. We prove:

- Any query $x \in \mathbb{R}^n$ to the membership oracle of this problem can be simulated by one query to the search-with-wildcards oracle for s. To achieve this, we divide the n coordinates of x into four sets: $T_{x,0}$ for those in $[-2,-1)$, $T_{x,1}$ for those in $(1,2]$, $T_{x,\text{mid}}$ for those in $[-1,1]$, and $T_{x,\text{out}}$ for the rest. Notice that $T_{x,\text{mid}}$ corresponds to the coordinates that are always in the hypercube and $T_{x,\text{out}}$ corresponds to the coordinates that are always out of the hypercube; $T_{x,0}$ (resp., $T_{x,1}$) includes the coordinates for which $s_i = 0$ (resp., $s_i = 1$) impacts the membership in the hypercube. We prove in Section 2.3.1 that a wildcard query with $T = T_{x,0} \cup T_{x,1}$ can simulate a membership query to x.

- The solution of the sum-of-coordinates optimization problem explicitly gives s, i.e., it solves search with wildcards. This is because this solution must be close to the point $(s_1 - 2, \ldots, s_n - 2)$, and applying integer rounding would recover s.

These two points establish the reduction of search with wildcards to the optimization problem, and hence establishes the $\Omega(\sqrt{n})$ membership quantum lower bound in Theorem 2.1.2 (see Theorem 2.3.2).

For our lower bound on evaluation queries, we assume that membership is

trivial by fixing the hypercube at $\mathcal{C} = [0,1]^n$. We then consider optimizing the max-norm function

$$f(x) = \max_{i \in [n]} |x_i - c_i| \tag{2.1.7}$$

for some unknown $c \in \{0,1\}^n$. Notice that learning c is equivalent to solving the optimization problem; in particular, outputting an $\tilde{x} \in \mathcal{C}$ satisfying (2.1.2) with $\epsilon = 1/3$ would determine the string c. This follows because for all $i \in [n]$, we have $|\tilde{x}_i - c_i| \leq \max_{i \in [n]} |\tilde{x}_i - c_i| \leq 1/3$, and c_i must be the integer rounding of $\tilde{x}_i$, i.e., $c_i = 0$ if $\tilde{x}_i \in [0, 1/2)$ and $c_i = 1$ if $\tilde{x}_i \in [1/2, 1]$. On the other hand, if we know c, then we know the optimum $x = c$.

We prove an $\Omega(\sqrt{n}/\log n)$ lower bound on evaluation queries for learning c. Our proof, which appears in Section 2.3.2, is composed of three steps:

1) We first prove a weaker lower bound with respect to the precision of the evaluation oracle. Specifically, if $f(x)$ is specified with b bits of precision, then using binary search, a query to $f(x)$ can be simulated by b queries to an oracle that inputs $(f(x), t)$ for some $t \in \mathbb{R}$ and returns 1 if $f(x) \leq t$ and returns 0 otherwise. We further without loss of generality assume $x \in [0,1]^n$. If $x \notin [0,1]^n$, we assign a penalty of the L_1 distance between x and its projection $\pi(x)$ onto $[0,1]^n$; by doing so, $f(\pi(x))$ and x fully characterizes $f(x)$ (see (2.3.25)). Therefore, $f(x) \in [0,1]$, and $f(x)$ having b bits of precision is equivalent to having precision 2^{-b}.

   Similar to the interval dividing strategy in the proof of the membership lower

bound, we prove that one query to such an oracle can be simulated by one query to the search-with-wildcards oracle for s. Furthermore, the solution of the max-norm optimization problem explicitly gives s, i.e., it solves the search-with-wildcards problem. This establishes the reduction to search with wildcards, and hence establishes an $\Omega(\sqrt{n}/b)$ lower bound on the number of quantum queries to the evaluation oracle f with precision 2^{-b} (see Lemma 2.3.2).

2) Next, we introduce a technique we call *discretization*, which effectively simulates queries over an (uncountably) infinite set by queries over a discrete set. This technique might be of independent interest since proving lower bounds on functions with an infinite domain can be challenging.

We observe that the problem of optimizing (2.1.7) has the following property: if we are given two strings $x, x' \in [0,1]^n$ such that $x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ and $x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n$ have the same ordering (for instance, strings $x = (0.1, 0.2, 0.7)$ and $x' = (0.1, 0.3, 0.6)$ both have the ordering $x_1 \leq x_2 \leq 1-x_3 \leq x_3 \leq 1-x_2 \leq 1-x_1$), then

$$\arg\max_{i\in[n]} |x_i - c_i| = \arg\max_{i\in[n]} |x'_i - c_i|. \tag{2.1.8}$$

Furthermore, if $x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n$ are $2n$ different numbers, then knowing the value of $f(x')$ implies the value of the arg max in (2.1.8) (denoted i^*) and the corresponding c_{i^*}, and we can subsequently recover $f(x)$ given x since $f(x) = |x_{i^*} - c_{i^*}|$. In other words, $f(x)$ can be computed given x and

$f(x')$.

Therefore, it suffices to consider all possible ways of ordering $2n$ numbers, rendering the problem discrete. Without loss of generality, we focus on x' satisfying $\{x'_1, \ldots, x'_n, 1-x'_1, \ldots, 1-x'_n\} = \{\frac{1}{2n+1}, \ldots, \frac{2n}{2n+1}\}$, and we denote the set of all such x' by D_n (see also (2.3.42)). In Lemma 2.3.5, we prove that one classical (resp., quantum) evaluation query from $[0,1]^n$ can be simulated by one classical evaluation query (resp., two quantum evaluation queries) from D_n using Algorithm 2.5. To illustrate this, we give a concrete example with $n = 3$ in Section 2.3.2.

3) Finally, we use discretization to show that one perfectly precise query to f can be simulated by one query to f with precision $\frac{1}{5n}$; in other words, b in step 1) is at most $\lceil \log_2 5n \rceil = O(\log n)$ (see Lemma 2.3.4). This is because by discretization, the input domain can be limited to the discrete set D_n. Notice that for any $x \in D_n$, $f(x)$ is an integer multiple of $\frac{1}{2n+1}$; even if $f(x)$ can only be computed with precision $\frac{1}{5n}$, we can round it to the closest integer multiple of $\frac{1}{2n+1}$ which is exactly $f(x)$, since the distance $\frac{2n+1}{5n} < \frac{1}{2}$. As a result, we can precisely compute $f(x)$ for all $x \in D_n$, and thus by discretization we can precisely compute $f(x)$ for all $x \in [0,1]^n$.

In all, the three steps above establish an $\Omega(\sqrt{n}/\log n)$ quantum lower bound on evaluation queries to solve the problem in Eq. (2.1.7) (see Theorem 2.3.2). In particular, this lower bound is proved for an unconstrained convex optimization problem on $\mathbb{R}^n$, which might be of independent interest.

As a side result, we prove that our quantum lower bound is optimal for the problem in (2.1.7) (up to poly-logarithmic factors in n), as we can prove a matching $\tilde{O}(\sqrt{n})$ upper bound (Theorem 2.3.5). Therefore, a better quantum lower bound on the number of evaluation queries for convex optimization would require studying an essentially different problem.

Having established lower bounds on both membership and evaluation queries, we combine them to give Theorem 2.1.2. This is achieved by considering an optimization problem of dimension $2n$; the first n coordinates compose the sum-of-coordinates function in Section 2.3.1, and the last n coordinates compose the max-norm function in Section 2.3.2. We then concatenate both parts and prove Theorem 2.1.2 via reductions to the membership and evaluation lower bounds, respectively (see Section 2.3.4).

In addition, all lower bounds described above can be adapted to a convex body that is contained in the unit hypercube and that contains the discrete set D_n to facilitate discretization; we present a "smoothed" hypercube (see Section 2.3.5) as a specific example.

2.2 Upper bound

In this section, we prove:

Theorem 2.2.1. *An optimization oracle for a convex set $K \subseteq \mathbb{R}^n$ can be implemented using $\tilde{O}(n)$ quantum queries to a membership oracle for K, with gate complexity $\tilde{O}(n^3)$.*

The following lemma shows the equivalence of optimization oracles to a general convex optimization problem.

Lemma 2.2.1. *Suppose a reduction from an optimization oracle to a membership oracle for convex sets requires $O(g(n))$ queries to the membership oracle. Then the problem of optimizing a convex function over a convex set can be solved using $O(g(n))$ queries to both the membership oracle and the evaluation oracle.*

Proof. The problem $\min_{x \in K} f(x)$ reduces to the problem $\min_{(x',x) \in K'} x'$ where K' is defined as in (2.1.3). K' is the intersection of convex sets and is therefore convex. A membership oracle of K' can be implemented using 1 query each to the membership oracle of K and the evaluation oracle for f. Since $O(g(n))$ queries to the membership oracle of K' are sufficient to optimize any linear function, the result follows. □

Theorem 2.1.1 directly follows from Theorem 2.2.1 and Lemma 2.2.1.

Overview. We follow the outline listed in Section 2.1.3. Precise definitions of oracles and other relevant terminology appear in Section 2.2.1. Section 2.2.2 develops a fast quantum subgradient procedure that can be used in the classical reduction from optimization to membership. This is done in two parts:

1. First, we present an algorithm based on the quantum Fourier transform that calculates the gradient of a function with bounded second derivatives (i.e., a β-smooth function) with bounded expected one-norm error.

2. Second, we use mollification to restrict the analysis to infinitely differentiable functions without loss of generality, and then uses classical randomness to

eliminate the need for bounded second derivatives.

In Section 2.2.3 we show that the new quantum subgradient algorithm fits into the classical reduction from [183]. Finally, we describe the reduction from optimization to membership in Section 2.2.4.

2.2.1 Oracle definitions

We provide precise definitions for the oracles for convex sets and functions that we use in our algorithm and its analysis. We also provide precise definitions of Lipschitz continuity and β-smoothness which are required in the rest of the section.

Definition 2.2.1 (Ball in ℓ_p-norm)**.** *The ball of radius $r > 0$ in ℓ_p-norm $\|\cdot\|_p$ centered at $x \in \mathbb{R}^n$ is $B_p(x, r) := \{y \in \mathbb{R}_n \mid \|x - y\|_p \leq r\}$.*

Definition 2.2.2 (Interior of a convex set)**.** *For any $\delta > 0$, the δ-interior of a convex set K is defined as $B_2(K, -\delta) := \{x \mid B_2(x, \delta) \subseteq K\}$.*

Definition 2.2.3 (Neighborhood of a convex set)**.** *For any $\delta > 0$, the δ-neighborhood of a convex set K is defined as $B_2(K, \delta) := \{x \mid \exists\, y \in K \text{ s.t. } \|x - y\|_2 \leq \delta\}$.*

Definition 2.2.4 (Evaluation oracle)**.** *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, output α such that $|\alpha - f(x)| \leq \delta$. We use $\mathrm{EVAL}_\delta(f)$ to denote the time complexity. The classical procedure or quantum unitary representing the oracle is denoted by O_f.*

Definition 2.2.5 (Membership oracle)**.** *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, output an assertion that $x \in B_2(K, \delta)$ or $x \notin B_2(K, -\delta)$. The time complexity is denoted by $\mathrm{MEM}_\delta(K)$. The classical procedure or quantum unitary representing the membership oracle is denoted by O_K.*

Definition 2.2.6 (Separation oracle)**.** *When queried with $x \in \mathbb{R}^n$ and $\delta > 0$, with probability $1 - \delta$, either*

- *assert $x \in B_2(K, \delta)$ or*
- *output a unit vector $\hat{c}$ such that $\hat{c}^T x \leq \hat{c}^T y + \delta$ for all $y \in B_2(K, -\delta)$.*

The time complexity is denoted by $\mathrm{SEP}_\delta(K)$.

Definition 2.2.7 (Optimization oracle)**.** *When queried with a unit vector c, find $y \in \mathbb{R}^n$ such that $c^T x \leq c^T y + \delta$ for all $x \in B_2(K, -\delta)$ or asserts that $B_2(K, \delta)$ is empty. The time complexity of the oracle is denoted by $\mathrm{OPT}_\delta(K)$.*

Definition 2.2.8 (Subgradient)**.** *A subgradient of a convex function $f \colon \mathbb{R}^n \to \mathbb{R}$ at x, is a vector g such that*

$$f(y) \geq f(x) + \langle g, y - x \rangle \tag{2.2.1}$$

for all $y \in \mathbb{R}^n$. For a differentiable convex function, the gradient is the only subgradient. The set of subgradients of f at x is called the subdifferential at x and denoted by $\partial f(x)$.

Definition 2.2.9 (L-Lipschitz continuity)**.** *A function $f \colon \mathbb{R}^n \to \mathbb{R}$ is said to be L-Lipschitz continuous (or simply L-Lipschitz) in a set S if for all $x \in S$, $\|g\|_\infty \leq L$ for any $g \in \partial f(x)$. An immediate consequence of this is that for any $x, y \in S$,*

$$|f(y) - f(x)| \leq L\|y - x\|_\infty. \tag{2.2.2}$$

Definition 2.2.10 (β-smoothness)**.** *A function $f \colon \mathbb{R}^n \to \mathbb{R}$ is said to be β-smooth in*

a set S if for all $x \in S$, the magnitudes of the second derivatives of f in all directions are bounded by β. This also means that the largest magnitude of an eigenvalue of the Hessian $\nabla^2 f(x)$ is at most β. Consequently, for any $x, y \in S$, we have

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2} \|y - x\|_\infty^2. \tag{2.2.3}$$

2.2.2 A quantum algorithm for computing subgradients

In this subsection, we give a quantum algorithm that given an evaluation oracle for an L-Lipschitz continuous function $f\colon \mathbb{R}^n \to \mathbb{R}$ with evaluation error at most $\epsilon > 0$, a point $x \in \mathbb{R}^n$, and an "approximation scale" factor $r_1 > 0$, computes an approximate subgradient $\tilde{g}$ of f at x. Specifically, $\tilde{g}$ satisfies

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1 L \tag{2.2.4}$$

for all $q \in \mathbb{R}^n$, where $\mathbb{E}\zeta \leq \xi(r_1, \epsilon)$ and ξ must monotonically increase with ϵ as ϵ^α for some $\alpha > 0$. Here ζ is the error in the subgradient that is bounded in expectation by the function ξ.

Smooth functions. We first describe how to approximate the gradient of a smooth function. Algorithm 2.1 and Algorithm 2.2 use techniques from [120, 157] to evaluate the gradient of a function with bounded second derivatives in the neighborhood of the evaluation point. The following lemma shows that Algorithm 2.1 provides a good estimate of the gradient with bounded failure probability.

Algorithm 2.1: `GradientEstimate`$(f, \epsilon, L, \beta, x_0)$

Input: Function f, evaluation error ϵ, Lipschitz constant L, smoothness parameter β, and point x_0.

Define

- $l = 2\sqrt{\epsilon/n\beta}$ to be the size of the grid used,
- $b \in \mathbb{N}$ such that $\frac{24\pi\sqrt{n\epsilon\beta}}{L} \leq \frac{1}{2^b} = \frac{1}{N} \leq \frac{48\pi\sqrt{n\epsilon\beta}}{L}$,
- $b_0 \in \mathbb{N}$ such that $\frac{N\epsilon}{2Ll} \leq \frac{1}{2^{b_0}} = \frac{1}{N_0} \leq \frac{N\epsilon}{Ll}$,
- $F(x) = \frac{N}{2Ll}[f(x_0 + \frac{l}{N}(x - N/2)) - f(x_0)]$, and
- $\gamma : \{0, \ldots, N-1\} \to G := \{-\frac{N}{2}, -\frac{N}{2}+1, \ldots, \frac{N}{2}-1\}$ s.t. $\gamma(x) = x - \frac{N}{2}$.

Let O_F be a unitary s.t. $O_F |x\rangle = e^{2\pi i \tilde{F}(x)} |x\rangle$ where $|\tilde{F}(x) - F(x)| \leq \frac{1}{N_0}$, with x and $\tilde{F}(x)$ represented by b and b_0 bits, respectively;

1 Start with n b-bit registers set to 0 and Hadamard transform each to obtain

$$\frac{1}{\sqrt{N^n}} \sum_{x_1,\ldots,x_n \in \{0,1,\ldots,N-1\}} |x_1, \ldots, x_n\rangle ; \tag{2.2.5}$$

2 Perform the operation O_F and the map $|x\rangle \mapsto |\gamma(x)\rangle$ to obtain

$$\frac{1}{N^{n/2}} \sum_{g \in G^n} e^{2\pi i \tilde{F}(g)} |g\rangle ; \tag{2.2.6}$$

3 Apply the inverse QFT over G to each of the registers;

4 Measure the final state to get $k_1, \ldots, k_n$ and return $\tilde{g} = \frac{2L}{N}(k_1, \ldots, k_n)$.

Lemma 2.2.2 ([67, Lemma B.2]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz function that is specified by an evaluation oracle with error at most ϵ. Let f be β-smooth in $B_\infty(x, 2\sqrt{\epsilon/\beta})$, and let $\tilde{g}$ be the output of* `GradientEstimate`$(f, \epsilon, L, \beta, x_0)$ *(from Algorithm 2.1). Then*

$$\Pr\left[|\tilde{g}_i - \nabla f(x)_i| > 1500\sqrt{n\epsilon\beta}\right] < \frac{1}{3}, \quad \forall\, i \in [n]. \tag{2.2.7}$$

Next we give Algorithm 2.2, which uses several calls to Algorithm 2.1 to estimate the gradient with small ℓ_1-distance to the true value in expectation.

Lemma 2.2.3 ([67, Lemma 2.3]). *Let f be a convex, L-Lipschitz continuous function*

Algorithm 2.2: `SmoothQuantumGradient`$(f, \epsilon, L, \beta, x)$

Data: Function f, evaluation error ϵ, Lipschitz constant L, smoothness parameter β, and point x.

Set T such that $2e^{-T^2/24} \leq 750\sqrt{n\epsilon\beta}/L$;

for $t = 1, 2, \ldots, T$ **do**

  $e^{(t)} \leftarrow$ `GradientEstimate`$(f, \epsilon, L, \beta, x)$;

for $i = 1, 2, \ldots, n$ **do**

  If more than $T/2$ of $e_i^{(t)}$ lie in an interval of size $3000\sqrt{n\epsilon\beta}$, set $\tilde{g}_i$ to be the median of the points in that interval;

  Otherwise, set $\tilde{g}_i = 0$;

Output $\tilde{g}$.

specified by an evaluation oracle with error at most ϵ. Suppose f is β-smooth in $B_\infty(x, 2\sqrt{\frac{\epsilon}{\beta}})$. Let

$$\tilde{g} = \texttt{SmoothQuantumGradient}(f, \epsilon, L, \beta, x) \tag{2.2.8}$$

(from Algorithm 2.2). Then for any $i \in [n]$, we have $|\tilde{g}_i| \leq L$ and $\mathbb{E}|\tilde{g}_i - \nabla f(x)_i| \leq 3000\sqrt{n\epsilon\beta}$; hence

$$\mathbb{E}\|\tilde{g} - \nabla f(x)\|_1 \leq 3000 n^{3/2}\sqrt{\epsilon\beta}. \tag{2.2.9}$$

If L, $1/\beta$, and $1/\epsilon$ are poly(n)*, the* `SmoothQuantumGradient` *algorithm uses $\tilde{O}(1)$ queries to the evaluation oracle and $\tilde{O}(n)$ gates.*

Extension to nonsmooth functions. Now consider a general L-Lipschitz continuous convex function f. We show that any such function is close to a smooth function, and we consider the relationship between the subgradients of the original function and the gradient of its smooth approximation.

For any $\delta > 0$, let $m_\delta \colon \mathbb{R}^n \to \mathbb{R}$ be the *mollifier function of width* δ, defined as

$$m_\delta(x) := \begin{cases} \frac{1}{I_n} \exp\left(-\frac{1}{1-\|x/\delta\|_2^2}\right) & x \in B_2(0,\delta) \\ 0 & \text{otherwise,} \end{cases} \tag{2.2.10}$$

where I_n is chosen such that $\int_{B_2(0,\delta)} m_\delta(x)\,\mathrm{d}^n x = 1$. The mollification of f, denoted $F_\delta := f * m_\delta$, is obtained by convolving it with the mollifier function, i.e.,

$$F_\delta(x) = (f * m_\delta)(x) = \int_{\mathbb{R}^n} f(x-y) m_\delta(y)\,\mathrm{d}^n x. \tag{2.2.11}$$

The mollification of f has several key properties, as follows:

Proposition 2.2.1 ([67, Lemma A.2])**.** *Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function with mollification F_δ. Then*

(i) *F_δ is infinitely differentiable,*

(ii) *F_δ is convex,*

(iii) *F_δ is L-Lipschitz continuous, and*

(iv) *$|F_\delta(x) - f(x)| \leq L\delta$.*

Furthermore, an approximate gradient of the mollified function gives an approximate subgradient of the original function, as quantified by the following lemma.

Lemma 2.2.4. *Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be an infinitely differentiable L-Lipschitz continuous convex function with mollification F_δ. Then any $\tilde{g}$ satisfying $\|\tilde{g} - \nabla F_\delta(y)\|_1 = \zeta$ for*

some $y \in B_\infty(x, r_1)$ *satisfies*

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1L - 2L\delta. \tag{2.2.12}$$

Here ζ *is the error in the subgradient and* δ *is the parameter used in the mollifier function.*

Proof. For all $q \in \mathbb{R}^n$, convexity of F_δ implies

$$F_\delta(q) \geq F_\delta(y) + \langle \nabla F_\delta(y), q - y \rangle \tag{2.2.13}$$

$$= F_\delta(x) + \langle \nabla F_\delta(y), q - x \rangle + \langle \nabla F_\delta(y), x - y \rangle + (F_\delta(y) - F_\delta(x)) \tag{2.2.14}$$

$$\geq F_\delta(x) + \langle \nabla F_\delta(y), q - x \rangle - 4nr_1L \tag{2.2.15}$$

$$\geq F_\delta(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1L. \tag{2.2.16}$$

Therefore, (2.2.12) follows from Proposition 2.2.1(iv). □

Now consider δ such that $L\delta \ll \epsilon$. Then the evaluation oracle with error ϵ for f is also an evaluation oracle for F_δ with error $\epsilon + L\delta \approx \epsilon$. Thus the given evaluation oracle is also the evaluation oracle for an infinitely differentiable convex function with the same Lipschitz constant and almost the same error, allowing us to analyze infinitely differentiable functions without loss of generality (as long as we make no claim about the second derivatives). This idea is made precise in Theorem 2.2.2. (Note that the mollification of f is never computed or estimated by our algorithm; it is only a tool for analysis.)

Unfortunately, Lemma 2.2.3 cannot be directly used to calculate subgradients

for F_δ as $\delta \to 0$. This is because there exist convex functions (such as $f(x) = |x|$) where if $|f(x) - g(x)| \leq \delta$ and $g(x)$ is β-smooth, then $\beta\delta \geq c$ for some constant c (see [67, Lemma A.3]). Thus using `SmoothQuantumGradient` for this case has an error of $3000n^{3/2}\sqrt{\epsilon\beta} \geq 3000n^{3/2}\sqrt{c}$ in ℓ_1-norm, which is independent of ϵ.

To fix this issue, we take inspiration from [183] and introduce classical randomness into the gradient evaluation. In particular, the following lemma shows that for a Lipschitz continuous function, if we sample at random from the neighborhood of any given point, the probability of having large second derivatives is small. Let $y \sim Y$ indicate that y is sampled uniformly at random from the set Y. Also, let $\lambda(x)$ be the largest eigenvalue of the Hessian matrix $\nabla^2 f(x)$ at x. Since the Hessian is positive semidefinite, we have $\lambda(x) \leq \Delta f(x) := \mathrm{Tr}(\nabla^2 f(x))$. Thus the second derivatives of f are upper bounded by $\Delta f(x)$.

Let $\eta(y)$ denote the area element on the surface $\partial B_\infty(x, r_1)$, defined as

$$\eta(y)_i := \begin{cases} 1 & \text{if } y_i - x_i \geq r_1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.2.17}$$

We have

$$\mathbb{E}_{y \sim B_\infty(x,r_1)} \Delta f(y) = \frac{1}{(2r_1)^n} \int\limits_{B_\infty(x,r_1)} \Delta f(y)\, \mathrm{d}^n y \tag{2.2.18}$$

$$= \frac{1}{(2r_1)^n} \int\limits_{\partial B_\infty(x,r_1)} \langle \nabla f(y), \eta(y) \rangle\, \mathrm{d}^{n-1} y \tag{2.2.19}$$

$$\leq \frac{1}{(2r_1)^n} (2n)(2r_1)^{n-1} L = \frac{nL}{r_1} \tag{2.2.20}$$

where (2.2.19) comes from the divergence theorem (the integral of the divergence of a vector field over a set is equal to the integral of the vector field over the surface of the set). This indicates that while the second derivatives of a Lipschitz continuous function can be unbounded at individual points, its expected value for a point uniformly sampled in an extended region is bounded.

Now, consider a grid of side length l (aligned with the coordinate axes) embedded in $B_\infty(x, r_1)$. We denote this grid by $G_{B_\infty(x,r_1),l}$. For any $i \in [n]$ and a y sampled uniformly from $G_{B_\infty(x,r_1),l}$, the expectation of the integral of the second directional derivative in the i^{th} coordinate direction over a segment from y to the point $y + le_i$ is

$$\mathbb{E}_{y \sim G_{B_\infty(x,r_1),l}} \left[\int_{y_i}^{y_i + le_i} \frac{\mathrm{d}^2 f(z)}{\mathrm{d}z_i^2} \, \mathrm{d}z_i \right] \leq \frac{Ll}{r_1}. \tag{2.2.21}$$

To see this, note that there are $2r_1/l$ segments of length l (corresponding to different points y) inside $B_\infty(y, r_1)$. The total integral of the directional derivative over these segments is upper bounded by the change in the i^{th} component of the gradient, which is in turn bounded by $2L$ due to the Lipschitz property of f.

Let $\Delta \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$\Delta(y, z) := \left| f(z) - f(y) - \langle \nabla f(y), z - y \rangle \right| \tag{2.2.22}$$

be a function that quantifies the deviation from linearity of f between y and z. We now show the following lemma that bounds this deviation in the neighborhood of a

randomly sampled point (with high probability).

Lemma 2.2.5. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz continuous, infinitely differentiable, convex function. Then for a point y chosen uniformly from $G_{B_\infty(x,r_1),l}$, and any $p \in \mathbb{R}$ such that $p \geq n$,*

$$\Delta(y,z) \leq \frac{pnl^2L}{r_1}, \quad \forall\, z \in B_\infty(y,l) \tag{2.2.23}$$

with probability at least $1 - \frac{n}{p}$.

Proof. Note that $\Delta(y,z)$ is a convex function of z and must attain its maximum at one of the extremal points (vertices) of the hypercube $B_\infty(y,l)$, which are the 2^n points of the form

$$\left\{y + ls \mid s \in \{-1,1\}^n\right\}. \tag{2.2.24}$$

This is because every point in the hypercube is a convex combination of the vertices, so having a higher function value at an internal point than at all the vertices would violate convexity.

Consider a path from y to a vertex of $B_\infty(y,l)$ consisting of n segments of length l aligned along the n coordinate axes. For example, the path could move a distance l along $\pm e_1, \pm e_2, \ldots, \pm e_n$ until the vertex is reached. Using Markov's inequality with (2.2.21), we have for every coordinate direction $i \in [n]$,

$$\Pr_{y \sim G_{B_\infty(x,r_1),l}}\left[\int_y^{y+le_i} \frac{\mathrm{d}^2 f(z)}{\mathrm{d}z_i^2}\,\mathrm{d}z > \frac{pLl}{r_1}\right] \leq \frac{1}{p}. \tag{2.2.25}$$

Thus with probability at least $1 - \frac{1}{p}$, the increase in the deviation from linearity along each segment, as quantified by the function Δ, is at most $\frac{pl^2L}{r_1}$. Using the union bound, with probability at least $1 - \frac{n}{p}$, the total deviation from linearity along the path is at most $\frac{pnl^2L}{r_1}$ as claimed. □

Lemma 2.2.5 shows that with high probability a sampled point in $B_\infty(x, r_1)$ has a deviation from linearity in its neighborhood which is the same as that for a function with smoothness parameter $\frac{2pL}{r_1}$. The analysis of the gradient estimation procedure (Lemma 2.2.3) uses the smoothness of the function only to bound its deviation from linearity. Thus, Algorithm 2.2 can be applied as if to a function with smoothness parameter $\frac{2pL}{r_1}$. This observation is applied to find an approximate subgradient in Algorithm 2.3 with the following guarantee:

Algorithm 2.3: `QuantumSubgradient`(f, ϵ, L, x, r_1)

Data: Function f, evaluation error ϵ, Lipschitz constant L, point $x \in \mathbb{R}^n$, length $r_1 > 0$.

1 Sample $y \sim G_{B_\infty(x,r_1),l}$;

2 Output $\tilde{g} =$ `SmoothQuantumGradient`$(f, \epsilon, L, 2n^{1/3}L/r_1^{2/3}\epsilon^{1/3}, y)$.

Theorem 2.2.2 ([67, Theorem 2.2])**.** *Let f be a convex, L-Lipschitz function that is specified by an evaluation oracle with error $\epsilon < \min\{1, r_1/n^2\}$. Let the output of Algorithm 2.3 be $\tilde{g} =$* `QuantumSubgradient`(f, ϵ, L, x, r_1)*. Then for all $q \in \mathbb{R}^n$,*

$$f(q) \geq f(x) + \langle \tilde{g}, q - x \rangle - \zeta \|q - x\|_\infty - 4nr_1L, \tag{2.2.26}$$

where $\mathbb{E}\zeta \leq \frac{5000Ln^{5/3}\epsilon^{1/3}}{r_1^{1/3}}$.

2.2.3 Step 1: from membership to separation

We now give the quantum algorithm for convex optimization as claimed in Theorem 2.1.1. First, we show how the approximate subgradient procedure (Algorithm 2.3) fits into the reduction from separation to membership presented in [183]. We use the *height function* $h_p\colon \mathbb{R}^n \to \mathbb{R}$ defined in [183] for any vector $p \in \mathbb{R}^n$, as

$$h_p(x) = -\max\{t \in \mathbb{R} \mid x + t\hat{p} \in K\}, \tag{2.2.27}$$

where $\hat{p}$ is the unit vector in the direction of p. The height function has the following properties:

Proposition 2.2.2 (Lemmas 11 and 12 of [183]). *Let $K \subset \mathbb{R}^n$ be a convex set with $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ for some $R > r > 0$. Then for any $p \in \mathbb{R}^n$, the height function (2.2.27) satisfies*

(i) *$h_p(x)$ is convex,*

(ii) *$h_p(x) \leq 0$ for all $x \in K$, and*

(iii) *for all $\delta > 0$, $h_p(x)$ is $\frac{R+\delta}{r-\delta}$-Lipschitz continuous for $x \in B_2(0,\delta)$.*

Now, we are ready to give the reduction from separation to membership using the algorithm `SeparatingHalfspace` as follows.

Theorem 2.2.3 ([67, Theorem 2.3]). *Let $K \subset \mathbb{R}^n$ be a convex set such that $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ for some $R > r > 0$. Let $\rho \in (0,1), \kappa = R/r$ and $\delta \in (0, \min\{r/7\kappa, 1/7\kappa\})$. Then* `SeparatingHalfspace`$(K,p,\rho,\delta)$ *outputs a halfspace that contains K and not p with probability at least $1-\rho$.*

Algorithm 2.4: `SeparatingHalfspace`(K, p, ρ, δ)

Data: Convex set K such that $B_2(0,r) \subset K \subset B_2(0,R), \kappa = R/r$, δ-precision membership oracle for K, point p.

if *the membership oracle asserts that* $p \in B_2(K,\delta)$ **then**
  Output: $p \in B_2(K,\delta)$.
else if $p \notin B_2(0,R)$ **then**
  Output: the halfspace $\{x \in \mathbb{R}^n \mid 0 > \langle x-p, p\rangle\}$.
else
  Define $h_p(x)$ as in (2.2.27). The evaluation oracle for $h_p(x)$ for any $x \in B(0,r/2)$ can be implemented to precision $\epsilon = 7\kappa\delta$ using $\log(1/\epsilon)$ queries to the membership oracle for K;
  Compute $\tilde{g} = \texttt{QuantumSubgradient}(h_p, \epsilon, L, 0, n\epsilon^{1/2})$;
  Output: the halfspace $\{x \in \mathbb{R}^n \mid (30000R+25)\, n^3\epsilon^{1/6}\kappa^2/\rho \geq \langle \tilde{g}, x-p\rangle\}$.

Theorem 2.2.4 ([67, Theorem 2.4])**.** *Let $K \subset \mathbb{R}^n$ be a convex set with $B_2(0,r) \subseteq K \subseteq B_2(0,R)$ and $\kappa = R/r$ for some $R > r > 0$, and let $\eta > 0$ be fixed. Further suppose that $R, r, \kappa = \mathrm{poly}(n)$. Then a separating oracle for K with error η can be implemented using $\tilde{O}(1)$ queries to a membership oracle for K and $\tilde{O}(n)$ gates.*

2.2.4 Step 2: from separation to optimization

It is a folklore result to implement an optimization oracle of a convex set by $\tilde{O}(n)$ queries to a separation oracle. Specifically, [183, Theorem 15] proves:

Theorem 2.2.5 (Separation to Optimization)**.** *Let K be a convex set satisfying $B_2(0,r) \subset K \subset B_2(0,R)$ and let $\kappa = 1/r$. For any $0 < \epsilon < 1$, with probability $1-\epsilon$, we can compute $x \in B_2(K,\epsilon)$ such that $c^T x \leq \min_{x\in K} c^T x + \epsilon\|c\|_2$, using $O(n\log(n\kappa/\epsilon))$ queries to $\mathrm{SEP}_\eta(K)$, where $\eta = \mathrm{poly}(\epsilon/n\kappa)$, and $\tilde{O}(n^3)$ arithmetic operations.*

By Theorem 2.2.4 and Theorem 2.2.5, we have:

Theorem 2.2.6 (Membership to Optimization)**.** *Let K be a convex set satisfying $B_2(0,r) \subset K \subset B_2(0,R)$ and let $\kappa = 1/r$. For any $0 < \epsilon < 1$, with probability $1-\epsilon$, we can compute $x \in B_2(K,\epsilon)$ such that $c^T x \leq \min_{x \in K} c^T x + \epsilon$, using $\tilde{O}(n)$ queries to a membership oracle for K with error δ, where $\delta = O(\text{poly}(\epsilon))$, and $\tilde{O}(n^3)$ gates.*

Proof. Using Theorem 2.2.4 with $\eta = \text{poly}(\epsilon/n\kappa)$, each query to the separation oracle requires $\tilde{O}(1)$ queries to a membership oracle with error $\delta = O(\text{poly}(\epsilon))$. We make $\tilde{O}(n)$ separation queries and perform a further $\tilde{O}(n^3)$ arithmetic operations, so the result follows. □

Theorem 2.2.1 follows directly from Theorem 2.2.6.

2.3 Lower bound

In this section, we prove our quantum lower bound on convex optimization (Theorem 2.1.2). We prove separate lower bounds on membership queries (Section 2.3.1) and evaluation queries (Section 2.3.2). We then combine these lower bounds into a single optimization problem in Section 2.3.4, establishing Theorem 2.1.2.

2.3.1 Membership queries

In this subsection, we establish a membership query lower bound using a reduction from the following search-with-wildcards problem:

Theorem 2.3.1 ([20, Theorem 1])**.** *For any $s \in \{0,1\}^n$, let O_s be a wildcard oracle*

satisfying

$$O_s|T\rangle|y\rangle|0\rangle = |T\rangle|y\rangle|Q_s(T,y)\rangle \tag{2.3.1}$$

for all $T \subseteq [n]$ *and* $y \in \{0,1\}^{|T|}$*, where* $Q_s(T,y) = \delta[s_{|T} = y]$*. Then the bounded-error quantum query complexity of determining* s *is* $O(\sqrt{n}\log n)$ *and* $\Omega(\sqrt{n})$*.*

We use Theorem 2.3.1 to give an $\Omega(\sqrt{n})$ lower bound on membership queries for convex optimization. Specifically, we consider the following *sum-of-coordinates optimization problem*:

Definition 2.3.1. *Let*

$$\mathcal{C}_s := \bigtimes_{i=1}^{n} [s_i - 2, s_i + 1], \qquad s_i \in \{0,1\} \quad \forall\, i \in [n], \tag{2.3.2}$$

where $\bigtimes$ *is the Cartesian product on different coordinates. In the* sum-of-coordinates optimization *problem, the goal is to minimize*

$$f(x) = \sum_{i\in[n]} x_i \quad s.t.\ x \in \mathcal{C}_s. \tag{2.3.3}$$

Intuitively, Definition 2.3.1 concerns an optimization problem on a hypercube where the function is simply the sum of the coordinates, but the position of the hypercube is unknown. Note that the function f in (2.3.3) is convex and 1-Lipschitz continuous.

We prove the hardness of solving sum-of-coordinates optimization using its membership oracle:

Theorem 2.3.2. *Given an instance of the sum-of-coordinates optimization problem with membership oracle $O_{\mathcal{C}_s}$, it takes $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s}$ to output an $\tilde{x} \in \mathcal{C}_s$ such that*

$$f(\tilde{x}) \leq \min_{x \in \mathcal{C}_s} f(x) + \frac{1}{3}, \tag{2.3.4}$$

with success probability at least 0.9.

Proof. Assume that we are given an arbitrary string $s \in \{0,1\}^n$ together with the membership oracle $O_{\mathcal{C}_s}$ for the sum-of-coordinates optimization problem.

We prove that a quantum query to $O_{\mathcal{C}_s}$ can be simulated by a quantum query to the oracle O_s in (2.3.1) for search with wildcards. Consider an arbitrary point $x \in \mathbb{R}^n$ in the sum-of-coordinates problem. We partition $[n]$ into four sets:

$$T_{x,0} := \big\{ i \in [n] \mid x_i \in [-2,-1) \big\} \tag{2.3.5}$$

$$T_{x,1} := \big\{ i \in [n] \mid x_i \in (1,2] \big\} \tag{2.3.6}$$

$$T_{x,\text{mid}} := \big\{ i \in [n] \mid x_i \in [-1,1] \big\} \tag{2.3.7}$$

$$T_{x,\text{out}} := \big\{ i \in [n] \mid |x_i| > 2 \big\}, \tag{2.3.8}$$

and denote $T_x := T_{x,0} \cup T_{x,1}$ and $y^{(x)} \in \{0,1\}^{|T_x|}$ such that

$$y_i^{(x)} = \begin{cases} 0 & \text{if } i \in T_{x,0} \\ 1 & \text{if } i \in T_{x,1}. \end{cases} \tag{2.3.9}$$

We prove that $O_{\mathcal{C}_s}(x) = Q_s(T_x, y^{(x)})$ if $T_{x,\text{out}} = \varnothing$, and $O_{\mathcal{C}_s}(x) = 0$ otherwise. On the one hand, if $O_{\mathcal{C}_s}(x) = 1$, we have $x \in \mathcal{C}_s$. Because for all $i \in [n]$, $x_i \in [s_i - 2, s_i + 1] \subset [-2, 2]$ for both $s_i = 0$ and $s_i = 1$, we must have $T_{x,\text{out}} = \varnothing$. Now consider any $i \in T_x$. If $i \in T_{x,0}$, then $x_i \in [-2, -1)$. Because $x_i \in [0 - 2, 0 + 1]$ and $x_i \notin [1 - 2, 1 + 1]$, we must have $s_i = 0$ since $x_i \in [s_i - 2, s_i + 1]$. Similarly, if $i \in T_{x,1}$, then we must have $s_i = 1$. As a result of (2.3.9), for all $i \in T_x$ we have $s_i = y_i^{(x)}$; in other words, $s_{|T_x} = y^{(x)}$ and $Q_s(T_x, y^{(x)}) = 1 = O_{\mathcal{C}_s}(x)$.

On the other hand, if $O_{\mathcal{C}_s}(x) = 0$, there exists an $i_0 \in [n]$ such that $x_{i_0} \notin [s_{i_0} - 2, s_{i_0} + 1]$. We must have $i_0 \notin T_{x,\text{mid}}$ because $[-1, 1] \subset [s_{i_0} - 2, s_{i_0} + 1]$ regardless of whether $s_{i_0} = 0$ or $s_{i_0} = 1$. Next, if $i_0 \in T_{x,\text{out}}$, then $T_{x,\text{out}} \neq \varnothing$ and we correctly obtain $O_{\mathcal{C}_s}(x) = 0$. The remaining cases are $i_0 \in T_{x,0}$ and $i_0 \in T_{x,1}$. If $i_0 \in T_{x,0}$, because $x_{i_0} \in [-2, -1) \subset [0 - 2, 0 + 1]$ and $x_{i_0} \notin [s_{i_0} - 2, s_{i_0} + 1]$, we must have $s_{i_0} = 1$, and thus $s_{|T_x} \neq y^{(x)}$ because $y_{i_0}^{(x)} = 0$ by (2.3.9). If $i_0 \in T_{x,1}$, we similarly have $s_{i_0} = 0$, $y_{i_0}^{(x)} = 1$, and thus $s_{|T_x} \neq y^{(x)}$. In both cases, $s_{|T_x} \neq y^{(x)}$, so $Q_s(T_x, y^{(x)}) = 0 = O_{\mathcal{C}_s}(x)$.

Therefore, we have established that $O_{\mathcal{C}_s}(x) = Q_s(T_x, y^{(x)})$ if $T_{x,\text{out}} = \varnothing$, and $O_{\mathcal{C}_s}(x) = 0$ otherwise. In other words, a quantum query to $O_{\mathcal{C}_s}$ can be simulated by a quantum query to O_s.

We next prove that a solution $\tilde{x}$ of the sum-of-coordinates problem satisfying (2.3.4) solves the search-with-wildcards problem in Theorem 2.3.1. Because

$\min_{x\in\mathcal{C}_s} f(x) = \sum_{i=1}^{n}(s_i - 2)$, we have

$$f(\tilde{x}) = \sum_{i=1}^{n} \tilde{x}_i \leq \frac{1}{3} + \sum_{i=1}^{n}(s_i - 2). \tag{2.3.10}$$

On the one hand, for all $j \in [n]$ we have $\tilde{x}_j \geq s_j - 2$ since $\tilde{x} \in \mathcal{C}_s$; on the other hand, by (2.3.10) we have

$$\frac{1}{3} + \sum_{i=1}^{n}(s_i - 2) \geq \sum_{i=1}^{n} \tilde{x}_i \geq \tilde{x}_j + \sum_{i\in[n],\ i\neq j} (s_i - 2), \tag{2.3.11}$$

which implies $\tilde{x}_j \leq s_j - 2 + \frac{1}{3}$. In all,

$$\tilde{x}_i \in [s_i - 2, s_i - 2 + \tfrac{1}{3}] \quad \forall\, i \in [n]. \tag{2.3.12}$$

Define a rounding function $\mathrm{sgn}_{-3/2}\colon \mathbb{R} \to \{0, 1\}$ as

$$\mathrm{sgn}_{-3/2}(z) = \begin{cases} 0 & \text{if } z < -3/2 \\ 1 & \text{otherwise.} \end{cases} \tag{2.3.13}$$

We prove that $\mathrm{sgn}_{-3/2}(\tilde{x}) = s$ (here $\mathrm{sgn}_{-3/2}$ is applied on all n coordinates, respectively). For all $i \in [n]$, if $s_i = 0$, then $\tilde{x}_i \in [-2, -\frac{5}{3}] \subset (-\infty, -\frac{3}{2})$ by (2.3.12), which implies $\mathrm{sgn}_{-3/2}(\tilde{x}_i) = 0$ by (2.3.13). Similarly, if $s_i = 1$, then $\tilde{x}_i \in [-1, -\frac{2}{3}] \subset (-\frac{3}{2}, +\infty)$ by (2.3.12), which implies $\mathrm{sgn}_{-3/2}(\tilde{x}_i) = 1$ by (2.3.13).

In all, if we can solve the sum-of-coordinates optimization problem with an $\tilde{x}$ satisfying (2.3.4), we can solve the search-with-wildcards problem. By Theo-

rem 2.3.2, the search-with-wildcards problem has quantum query complexity $\Omega(\sqrt{n})$; since a query to the membership oracle $O_{\mathcal{C}_s}$ can be simulated by a query to the wildcard oracle O_s, we have established an $\Omega(\sqrt{n})$ quantum lower bound on membership queries to solve the sum-of-coordinates optimization problem. □

2.3.2 Evaluation queries

In this subsection, we establish an evaluation query lower bound by considering the following *max-norm optimization problem*:

Definition 2.3.2. *In the* max-norm optimization problem*, the goal is to minimize a function* $f_c\colon \mathbb{R}^n \to \mathbb{R}$ *satisfying*

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{2.3.14}$$

for some $c \in \{0,1\}^n$*, where* $\pi\colon \mathbb{R} \to [0,1]$ *is defined as*

$$\pi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 1 \\ 1 & \text{if } x > 1. \end{cases} \tag{2.3.15}$$

Observe that for all $x \in [0,1]^n$, we have $f_c(x) = \max_{i\in[n]} |x_i - c_i|$. Intuitively, Definition 2.3.2 concerns an optimization problem under the max-norm (i.e., L_∞ norm) distance from c for all x in the unit hypercube $[0,1]^n$; for all x not in the unit hypercube, the optimizing function pays a penalty of the L_1 distance between

x and its projection $\pi(x)$ onto the unit hypercube. The function f_c is 2-Lipschitz continuous with a unique minimum at $x = c$; we also have:

Lemma 2.3.1. *The function f_c defined in (2.3.14) is convex on $\mathbb{R}^n$.*

Proof. For convenience, we define $g_i \colon \mathbb{R}^n \to \mathbb{R}$ for $i \in [n]$ as

$$g_i(x) := |\pi(x_i) - x_i| = \begin{cases} -x_i & \text{if } x_i < 0 \\ 0 & \text{if } 0 \le x_i \le 1 \\ x_i - 1 & \text{if } x_i > 1 \end{cases} \tag{2.3.16}$$

where the second equality follows from (2.3.15). It is clear that $g_i(x) = \max\{-x_i, 0, x_i - 1\}$ by (2.3.16). Since the pointwise maximum of convex functions is convex, $g_i(x)$ is convex for all $i \in [n]$.

Moreover, for all $i \in [n]$ we define $h_{c,i} \colon \mathbb{R}^n \to \mathbb{R}$ as $h_{c,i}(x) := |\pi(x_i) - c_i| + |\pi(x_i) - x_i|$. We claim that $h_{c,i}(x) = |x_i - c_i|$, and thus $h_{c,i}$ is convex. If $c_i = 0$, then $|\pi(x_i) - c_i| + |\pi(x_i) - x_i| = \pi(x_i) + |\pi(x_i) - x_i|$; as a result,

$$x_i < 0 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = 0 + |0 - x_i| = -x_i; \tag{2.3.17}$$

$$0 \le x_i \le 1 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = x_i + |x_i - x_i| = x_i; \tag{2.3.18}$$

$$x_i > 1 \quad \Rightarrow \quad \pi(x_i) + |\pi(x_i) - x_i| = 1 + |1 - x_i| = x_i. \tag{2.3.19}$$

Therefore, $\forall\, i \in [n], h_{c,i}(x) = |x_i - c_i|$. The proof is similar when $c_i = 1$.

Now we have

$$f_c(x) = \max_{i\in[n]} \Big(|\pi(x_i) - c_i| + \sum_{j=1}^{n} |\pi(x_j) - x_j| \Big) \tag{2.3.20}$$

$$= \max_{i\in[n]} \Big(\big(|\pi(x_i) - c_i| + |\pi(x_i) - x_i| \big) + \sum_{j\neq i} g_j(x) \Big) \tag{2.3.21}$$

$$= \max_{i\in[n]} \Big(h_{c,i}(x) + \sum_{j\neq i} g_j(x) \Big). \tag{2.3.22}$$

Because $h_{c,i}$ and g_j are both convex functions on $\mathbb{R}^n$ for all $i, j \in [n]$, the function $h_{c,i}(x) + \sum_{j\neq i} g_j(x)$ is convex on $\mathbb{R}^n$. Thus f_c is the pointwise maximum of n convex functions and is therefore itself convex. □

We prove the hardness of solving (2.3.14) using its evaluation oracle:

Theorem 2.3.3. *Given an instance of the max-norm optimization problem with an evaluation oracle O_{f_c}, it takes $\Omega(\sqrt{n}/\log n)$ quantum queries to O_{f_c} to output an $\tilde{x} \in [0,1]^n$ such that*

$$f_c(\tilde{x}) \leq \min_{x\in[0,1]^n} f_c(x) + \frac{1}{3}, \tag{2.3.23}$$

with success probability at least 0.9.

The proof of Theorem 2.3.3 has two steps. First, we prove a weaker lower bound with respect to the precision of the evaluation oracle:

Lemma 2.3.2. *Suppose we are given an instance of the max-norm optimization problem with an evaluation oracle O_{f_c} that has precision $0 < \delta < 0.05$, i.e., f_c is*

provided with $\lceil \log_2(1/\delta) \rceil$ *bits of precision. Then it takes* $\Omega(\sqrt{n}/\log(1/\delta))$ *quantum queries to* O_{f_c} *to output an* $\tilde{x} \in [0,1]^n$ *such that*

$$f_c(\tilde{x}) \leq \min_{x \in [0,1]^n} f_c(x) + \frac{1}{3}, \tag{2.3.24}$$

with success probability at least 0.9.

The second step simulates a perfectly precise query to f_c by a rough query:

Lemma 2.3.3. *One classical (resp., quantum) query to* O_{f_c} *with perfect precision can be simulated by one classical query (resp., two quantum queries) to* O_{f_c} *with precision* $1/5n$.

Theorem 2.3.3 simply follows from the two propositions above: by Lemma 2.3.3, we can assume that the evaluation oracle O_{f_c} has precision $1/5n$, so Lemma 2.3.2 implies that it takes $\Omega(\sqrt{n}/\log 5n) = \Omega(\sqrt{n}/\log n)$ quantum queries to O_{f_c} to output an $\tilde{x} \in [0,1]^n$ satisfying (2.3.23) with success probability 0.9.

The proofs of Lemma 2.3.2 and Lemma 2.3.3 are given in the next paragraphs.

$\tilde{\Omega}(\sqrt{n})$ quantum lower bound on a low-precision evaluation oracle. Similar to the proof of Theorem 2.3.2, we also use Theorem 2.3.1 (the quantum lower bound on search with wildcards) to give a quantum lower bound on the number of evaluation queries required to solve the max-norm optimization problem.

Proof of Lemma 2.3.2. Assume that we are given an arbitrary string $c \in \{0,1\}^n$ together with the evaluation oracle O_{f_c} for the max-norm optimization problem. To

show the lower bound, we reduce the search-with-wildcards problem to the max-norm optimization problem.

We first establish that an evaluation query to O_f can be simulated by wildcard queries on c. Note that if we query an arbitrary $x \in \mathbb{R}^n$, by (2.3.14) we have

$$f_c(x) = \max_{i\in[n]} |\pi(x_i) - c_i| + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{2.3.25}$$

$$= f_c(\pi(x)) + \Big(\sum_{i=1}^{n} |\pi(x_i) - x_i|\Big) \tag{2.3.26}$$

where $\pi(x) := (\pi(x_1), \ldots, \pi(x_n))$. In particular, the difference of $f_c(x)$ and $f_c(\pi(x))$ is an explicit function of x that is independent of c. Thus the query $O_{f_c}(x)$ can be simulated using one query to $O_{f_c}(\pi(x))$ where $\pi(x) \in [0,1]^n$. It follows that we can restrict ourselves without loss of generality to implementing evaluation queries for $x \in [0,1]^n$.

Now we consider a decision version of oracle queries to f_c, denoted $O_{f_{c,\mathrm{dec}}}$, where the function $f_{c,\mathrm{dec}}\colon [0,1]^n \times [0,1] \to \{0,1\}$ satisfies

$$f_{c,\mathrm{dec}}(x,t) = \delta[f_c(x) \le t]. \tag{2.3.27}$$

(We restrict to $t \in [0,1]$ because $f_c(x) \in [0,1]$ always holds for $x \in [0,1]^n$.) Using binary search, a query to O_{f_c} with precision δ can be simulated by at most $\lceil \log_2(1/\delta) \rceil = O(\log 1/\delta)$ queries to the oracle $O_{f_{c,\mathrm{dec}}}$.

Next, we prove that a query to $O_{f_{c,\mathrm{dec}}}$ can be simulated by a query to the search-with-wildcards oracle O_c in (2.3.1). Consider an arbitrary query $(x,t) \in [0,1]^n \times [0,1]$

to $O_{f_{c,\mathrm{dec}}}$. For convenience, we denote $J_{0,t} := [0,t]$, $J_{1,t} := [1-t,1]$, and

$$I_{0,t} := J_{0,t} - (J_{0,t} \cap J_{1,t}) \tag{2.3.28}$$

$$I_{1,t} := J_{1,t} - (J_{0,t} \cap J_{1,t}) \tag{2.3.29}$$

$$I_{\mathrm{mid},t} := J_{0,t} \cap J_{1,t} \tag{2.3.30}$$

$$I_{\mathrm{out},t} := [0,1] - (J_{0,t} \cup J_{1,t}). \tag{2.3.31}$$

We partition $[n]$ into four sets:

$$T_{x,0,t} := \left\{ i \in [n] \mid x_i \in I_{0,t} \right\} \tag{2.3.32}$$

$$T_{x,1,t} := \left\{ i \in [n] \mid x_i \in I_{1,t} \right\} \tag{2.3.33}$$

$$T_{x,\mathrm{mid},t} := \left\{ i \in [n] \mid x_i \in I_{\mathrm{mid},t} \right\} \tag{2.3.34}$$

$$T_{x,\mathrm{out},t} := \left\{ i \in [n] \mid x_i \in I_{\mathrm{out},t} \right\}. \tag{2.3.35}$$

The strategy here is similar to the proof of Theorem 2.3.2: $T_{x,\mathrm{mid},t}$ corresponds to the coordinates such that $|x_i - c_i| \le t$ regardless of whether $c_i = 0$ or 1 (and hence c_i does not influence whether or not $\max_{i\in[n]} |x_i - c_i| \le t$); $T_{x,\mathrm{out},t}$ corresponds to the coordinates such that $|x_i - c_i| > t$ regardless of whether $c_i = 0$ or 1 (so $\max_{i\in[n]} |x_i - c_i| > t$ provided $T_{x,\mathrm{out},t}$ is nonempty); and $T_{x,0,t}$ (resp., $T_{x,1,t}$) corresponds to the coordinates such that $|x_i - c_i| \le t$ only when $c_i = 0$ (resp., $c_i = 1$).

Denote $T_{x,t} := T_{x,0,t} \cup T_{x,1,t}$ and let $y^{(x,t)} \in \{0,1\}^{|T_{x,t}|}$ such that

$$y_i^{(x,t)} = \begin{cases} 0 & \text{if } i \in T_{x,0,t} \\ 1 & \text{if } i \in T_{x,1,t}. \end{cases} \tag{2.3.36}$$

We will prove that $O_{f_{c,\mathrm{dec}}}(x) = Q_c(T_{x,t}, y^{(x,t)})$ if $T_{x,\mathrm{out},t} = \varnothing$, and $O_{f_{c,\mathrm{dec}}}(x) = 0$ otherwise.

On the one hand, if $O_{f_{c,\mathrm{dec}}}(x) = 1$, we have $f_c(x) \le t$. In other words, for all $i \in [n]$ we have $|x_i - c_i| \le t$, which implies

$$x_i \in J_{c_i,t} \quad \forall\, i \in [n]. \tag{2.3.37}$$

Since $J_{c_i,t} \subseteq J_{0,t} \cup J_{1,t}$, we have $x_i \in J_{0,t} \cup J_{1,t}$ for all $i \in [n]$, and thus $T_{x,\mathrm{out},t} = \varnothing$ by (2.3.31) and (2.3.35). Now consider any $i \in T_{x,t}$. If $i \in T_{x,0,t}$, then $x_i \in I_{0,t}$ by (2.3.32). By (2.3.28) we have $x_i \in J_{0,t}$ and $x_i \notin J_{1,t}$, and thus $c_i = 0$ by (2.3.37). Similarly, if $i \in T_{x,1,t}$, then we must have $c_i = 1$. As a result of (2.3.36), for all $i \in T_{x,t}$ we have $c_i = y_i^{(x,t)}$; in other words, $c_{|T_{x,t}} = y^{(x,t)}$ and $Q_c(T_{x,t}, y^{(x,t)}) = 1 = O_{f_{c,\mathrm{dec}}}(x)$.

On the other hand, if $O_{f_{c,\mathrm{dec}}}(x) = 0$, there exists an $i_0 \in [n]$ such that

$$x_{i_0} \notin J_{c_{i_0},t}. \tag{2.3.38}$$

Therefore, we must have $i_0 \notin T_{x,\mathrm{mid},t}$ since (2.3.30) implies $I_{\mathrm{mid},t} = J_{0,t} \cap J_{1,t} \subseteq J_{c_{i_0},t}$. Next, if $i_0 \in T_{x,\mathrm{out},t}$, then $T_{x,\mathrm{out},t} \neq \varnothing$ and we correctly obtain $O_{f_{c,\mathrm{dec}}}(x) = 0$. The remaining cases are $i_0 \in T_{x,0,t}$ and $i_0 \in T_{x,1,t}$.

If $i_0 \in T_{x,0,t}$, then $y_{i_0}^{(x,t)} = 0$ by (2.3.36). By (2.3.32) we have $x_{i_0} \in I_{0,t}$, and by (2.3.28) we have $x_{i_0,t} \in J_{0,t}$ and $x_{i_0} \notin J_{1,t}$; therefore, we must have $c_{i_0} = 1$ by (2.3.38). As a result, $c_{|T_{x,t}} \neq y^{(x,t)}$ at i_0. If $i_0 \in T_{x,1,t}$, we similarly have $c_{i_0} = 0$, $y_{i_0}^{(x,t)} = 1$, and thus $c_{|T_{x,t}} \neq y^{(x,t)}$ at i_0. In either case, $c_{|T_{x,t}} \neq y^{(x,t)}$, and $Q_c(T_{x,t}, y^{(x,t)}) = 0 = O_{f_{c,\mathrm{dec}}}(x)$.

Therefore, we have established that $O_{f_{c,\mathrm{dec}}}(x) = Q_c(T_{x,t}, y^{(x,t)})$ if $T_{x,\mathrm{out},t} = \varnothing$, and $O_{f_{c,\mathrm{dec}}}(x) = 0$ otherwise. In other words, a quantum query to $O_{f_{c,\mathrm{dec}}}$ can be simulated by a quantum query to the search-with-wildcards oracle O_c. Together with the fact that a query to O_{f_c} with precision δ can be simulated by $O(\log 1/\delta)$ queries to $O_{f_{c,\mathrm{dec}}}$, it can also be simulated by $O(\log 1/\delta)$ queries to O_c.

We next prove that a solution $\tilde{x}$ of the max-norm optimization problem satisfying (2.3.24) solves the search-with-wildcards problem in Theorem 2.3.1. Because $\min_{x\in[0,1]^n} f_c(x) = 0$, considering the precision of at most $\delta < 0.05$ we have

$$f_c(\tilde{x}) \leq \tfrac{1}{3} + \delta \leq 0.4. \tag{2.3.39}$$

In other words,

$$\tilde{x}_i \in [c_i - 0.4, c_i + 0.4] \quad \forall\, i \in [n]. \tag{2.3.40}$$

Similar to (2.3.13), we define a rounding function $\mathrm{sgn}_{1/2}\colon \mathbb{R} \to \{0,1\}$ as

$$\mathrm{sgn}_{1/2}(z) = \begin{cases} 0 & \text{if } z < 1/2 \\ 1 & \text{otherwise.} \end{cases} \tag{2.3.41}$$

We prove that $\mathrm{sgn}_{1/2}(\tilde{x}) = c$ (here $\mathrm{sgn}_{1/2}$ is applied coordinate-wise). For all $i \in [n]$, if $c_i = 0$, then $\tilde{x}_i \in [0, 0.4] \subset (-\infty, 1/2)$ by (2.3.40), which implies $\mathrm{sgn}_{1/2}(\tilde{x}_i) = 0$ by (2.3.41). Similarly, if $c_i = 1$, then $\tilde{x}_i \in [0.6, 1] \subset (1/2, +\infty)$ by (2.3.40), which implies $\mathrm{sgn}_{1/2}(\tilde{x}_i) = 1$ by (2.3.41).

We have shown that if we can solve the max-norm optimization problem with an $\tilde{x}$ satisfying (2.3.24), we can solve the search-with-wildcards problem. By Theorem 2.3.2, the search-with-wildcards problem has quantum query complexity $\Omega(\sqrt{n})$; since a query to the evaluation oracle O_{f_c} can be simulated by $O(\log 1/\delta)$ queries to the wildcard oracle O_c, we have established an $\Omega(\sqrt{n}/\log(1/\delta))$ quantum lower bound on the number of evaluation queries needed to solve the max-norm optimization problem. □

Discretization: simulating perfectly precise queries by low-precision queries. In this subsection we prove Lemma 2.3.3, which we rephrase more formally as follows. Throughout this subsection, the function f_c in (2.3.14) is abbreviated by f for notational convenience.

Lemma 2.3.4. *Assume that $\hat{f}\colon [0,1]^n \to [0,1]$ satisfies $|\hat{f}(x) - f(x)| \le \frac{1}{5n}$ $\forall\, x \in [0,1]^n$. Then one classical (resp., quantum) query to O_f can be simulated by one*

classical query (resp., two quantum queries) to $O_{\hat{f}}$.

To achieve this, we present an approach that we call *discretization*. Instead of considering queries on all of $[0,1]^n$, we only consider a discrete subset $D_n \subseteq [0,1]^n$ defined as

$$D_n := \big\{\chi(a,\pi) \mid a \in \{0,1\}^n \text{ and } \pi \in S_n\big\}, \tag{2.3.42}$$

where S_n is the symmetric group on $[n]$ and $\chi\colon \{0,1\}^n \times S_n \to [0,1]^n$ satisfies

$$\chi(a,\pi)_i = (1-a_i)\tfrac{\pi(i)}{2n+1} + a_i(1-\tfrac{\pi(i)}{2n+1}) \quad \forall\, i \in [n]. \tag{2.3.43}$$

Observe that D_n is a subset of $[0,1]^n$.

Since $|S_n| = n!$ and there are 2^n choices for $a \in \{0,1\}^n$, we have $|D_n| = 2^n n!$. For example, when $n = 2$, we have

$$D_2 = \big\{\big(\tfrac{1}{5},\tfrac{2}{5}\big), \big(\tfrac{1}{5},\tfrac{3}{5}\big), \big(\tfrac{4}{5},\tfrac{2}{5}\big), \big(\tfrac{4}{5},\tfrac{3}{5}\big), \big(\tfrac{2}{5},\tfrac{1}{5}\big), \big(\tfrac{2}{5},\tfrac{4}{5}\big), \big(\tfrac{3}{5},\tfrac{1}{5}\big), \big(\tfrac{3}{5},\tfrac{4}{5}\big)\big\} \tag{2.3.44}$$

with $|D_2| = 2^2 \cdot 2! = 8$.

We denote the restriction of the oracle O_f to D_n by $O_{f|D_n}$, i.e.,

$$O_{f|D_n}|x\rangle|0\rangle = |x\rangle|f(x)\rangle \qquad \forall\, x \in D_n. \tag{2.3.45}$$

In fact, this restricted oracle entirely captures the behavior of the unrestricted function.

Lemma 2.3.5 (Discretization). *A classical (resp., quantum) query to O_f can be simulated using one classical query (resp., two quantum queries) to $O_{f|D_n}$.*

Algorithm 2.5: Simulate one query to O_f using one query to $O_{f|D_n}$.

Input: $x \in [0,1]^n$;
Output: $f(x) \in [0,1]$;

1 Compute $b \in \{0,1\}^n$ and $\sigma \in S_n$ such that the $2n$ numbers $x_1, x_2, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ are arranged in decreasing order as

$$\begin{aligned} &b_{\sigma(1)}x_{\sigma(1)} + (1-b_{\sigma(1)})(1-x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1-b_{\sigma(n)})(1-x_{\sigma(n)}) \\ &\geq (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) \geq \cdots \geq (1-b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1-x_{\sigma(1)}); \end{aligned} \tag{2.3.46}$$

2 Compute $x^* \in D_n$ such that $\chi(b, \sigma^{-1}) = x^*$ (where χ is defined in (2.3.43));
3 Query $f(x^*)$ and let $k^* = (2n+1)(1-f(x^*))$;
4 Return

$$f(x) = \begin{cases} (1-b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1-x_{\sigma(n)}) & \text{if } k^* = n+1 \\ b_{\sigma(k^*)}x_{\sigma(k^*)} + (1-b_{\sigma(k^*)})(1-x_{\sigma(k^*)}) & \text{otherwise.} \end{cases} \tag{2.3.47}$$

We prove this proposition by giving an algorithm (Algorithm 2.5) that performs the simulation. The main idea is to compute $f(x)$ only using x and $f(x^*)$ for some $x^* \in D_n$. We observe that max-norm optimization has the following property: if two strings $x \in [0,1]^n$ and $x^* \in D_n$ are such that $x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n$ and $x_1^*, \ldots, x_n^*, 1-x_1^*, \ldots, 1-x_n^*$ have the same ordering, then

$$\arg\max_{i \in [n]} |x_i - c_i| = \arg\max_{i \in [n]} |x_i^* - c_i|. \tag{2.3.48}$$

Furthermore, $x^* \in D_n$ ensures that $\{x_1^*, \ldots, x_n^*, 1-x_1^*, \ldots, 1-x_n^*\} = \{\frac{1}{2n+1}, \ldots, \frac{2n}{2n+1}\}$ are $2n$ distinct numbers, so knowing the value of $f(x^*)$ is sufficient to determine the value of the arg max above (denoted i^*) and the corresponding c_{i^*}. We can then

recover $f(x) = |x_{i^*} - c_{i^*}|$ using the given value of x. Moreover, $f(x^*)$ is an integer multiple of $\frac{1}{2n+1}$; even if $f(x^*)$ can only be computed with precision $\frac{1}{5n}$, we can round it to the closest integer multiple of $\frac{1}{2n+1}$ which is exactly $f(x^*)$, since the distance $\frac{2n+1}{5n} < \frac{1}{2}$. As a result, we can precisely compute $f(x^*)$ for all $x \in D_n$, and thus we can precisely compute $f(x)$.

We illustrate Algorithm 2.5 by a simple example. For convenience, we define an order function $\mathrm{Ord}\colon [0,1]^n \to \{0,1\}^n \times S_n$ by $\mathrm{Ord}(x) = (b, \sigma)$ for all $x \in [0,1]^n$, where b and σ satisfy Eq. (2.3.46).

An example with $n = 3$. Consider the case where the ordering in (2.3.46) is

$$1 - x_3 \geq x_1 \geq x_2 \geq 1 - x_2 \geq 1 - x_1 \geq x_3. \tag{2.3.49}$$

Then Algorithm 2.5 proceeds as follows:

- Line 1: With the ordering (2.3.49), we have $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$; $b_3 = 0$, $b_1 = 1$, $b_2 = 1$.
- Line 2: The point $x^* \in D_3$ that we query given $\mathrm{Ord}(x)$ satisfies $1 - x_3^* = 6/7$, $x_1^* = 5/7$, $x_2^* = 4/7$, $1 - x_2^* = 3/7$, $1 - x_1^* = 2/7$, and $x_3^* = 1/7$; in other words, $x^* = (5/7, 4/7, 1/7)$.
- Line 3: Now we query $f(x^*)$. Since $f(x^*)$ is a multiple of $1/7$ and $f(x^*) \in [1/7, 6/7]$, there are only 6 possibilities: $f(x^*) = 6/7$, $f(x^*) = 5/7$, $f(x^*) = 4/7$, $f(x^*) = 3/7$, $f(x^*) = 2/7$, or $f(x^*) = 1/7$.

After running Line 1, Line 2, and Line 3, we have a point x^* from the discrete set D_3 such that $\mathrm{Ord}(x) = \mathrm{Ord}(x^*)$. Since they have the same ordering and $|x_i - c_i|$ is either x_i or $1 - x_i$ for all $i \in [3]$, the function value $f(x^*)$ should essentially reflect the value of $f(x)$; this is made precise in Line 4.

- Line 4: Depending on the value of $f(x^*)$, we have six cases:
  - $f(x^*) = 6/7$: In this case, we must have $c_3 = 1$, so that $|x_3 - c_3| = |1/7 - 1| = 6/7$ ($|x_1 - c_1|$ can only give $5/7$ or $2/7$, and $|x_2 - c_2|$ can only give $4/7$ or $3/7$). Because $1 - x_3$ is the largest in (2.3.49), we must have $f(x) = 1 - x_3$.
  - $f(x^*) = 5/7$: In this case, we must have $c_1 = 0$, so that $|x_1 - c_1| = |5/7 - 0| = 5/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$). As a result of (2.3.49), we must have $f(x) = x_1$ since $x_1 \geq x_3$ and $x_1 \geq \max\{x_2, 1 - x_2\}$.
  - $f(x^*) = 4/7$: In this case, we must have $c_2 = 0$, so that $|x_2 - c_2| = |4/7 - 0| = 4/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$) and $c_1 = 1$ (otherwise if $c_1 = 0$, $f(x) \geq |x_1 - c_1| = 5/7$). As a result of (2.3.49), we must have $f(x) = x_2$ since $x_2 \geq 1 - x_1 \geq 1 - x_3$.
  - $f(x^*) = 3/7$: In this case, we must have $c_2 = 1$, so that $|x_2 - c_2| = |4/7 - 1| = 3/7$. Furthermore, we must have $c_3 = 1$ (otherwise if $c_3 = 0$, $f(x) \geq |x_3 - c_3| = 6/7$) and $c_1 = 1$ (otherwise if $c_1 = 0$, $f(x) \geq |x_1 - c_1| = 5/7$). As a result of (2.3.49), we must have $f(x) = 1 - x_2$ since since $1 - x_2 \geq 1 - x_1 \geq 1 - x_3$.

– $f(x^*) = 2/7$ or $f(x^*) = 1/7$: This two cases are impossible because $f(x^*) \geq |x_2 - c_2| = |4/7 - c_2| \geq 3/7$, no matter $c_2 = 0$ or $c_2 = 1$.

While Algorithm 2.5 is a classical algorithm for querying O_f using a query to $O_{f|D_n}$, it is straightforward to perform this computation in superposition using standard techniques to obtain a quantum query to O_f. However, note that this requires two queries to a quantum oracle for $O_{f|D_n}$ since we must uncompute $f(x^*)$ after computing $f(x)$.

Having the discretization technique at hand, Lemma 2.3.4 is straightforward.

Proof of Lemma 2.3.4. Recall that $|\hat{f}(x) - f(x)| \leq \frac{1}{5n}\ \forall\, x \in [0,1]^n$. We run Algorithm 2.5 to compute $f(x)$ for the queried value of x, except that in Line 3 we take $k^* = \lceil (2n+1)(1-\hat{f}(x^*)) \rfloor$ (here $\lceil a \rfloor$ is the closest integer to a). Because $|\hat{f}(x^*) - f(x^*)| \leq \frac{1}{5n}$, we have

$$\left|(2n+1)(1-\hat{f}(x^*)) - (2n+1)(1-f(x^*))\right| = (2n+1)|\hat{f}(x^*) - f(x^*)| \leq \tfrac{2n+1}{5n} < \tfrac{1}{2};$$

as a result, $k^* = (2n+1)(1-f(x^*))$ because the latter is an integer (see Lemma 2.3.7). Therefore, due to the correctness of Algorithm 2.5 established in Section 2.3.3, and noticing that the evaluation oracle is only called at Line 3 (with the replacement described above), we successfully simulate one query to O_f by one query to $O_{\hat{f}}$ (actually, to $O_{\hat{f}|D_n}$). □

The proof of Lemma 2.3.5 is given in the next subsection. In particular,

- First, we prove that the discretized vector x^* obtained in Line 2 is a good

approximation of x in the sense that $\mathrm{Ord}(x^*) = \mathrm{Ord}(x)$;

- Second, we prove that the k^* obtained in Line 3 satisfies $k^* \in \{1, \ldots, n+1\}$;
- Third, we prove that the output returned in Line 4 is correct.

2.3.3 Complete analysis of Algorithm 2.5

Correctness of Line 1 and Line 2. In this paragraph, we prove:

Lemma 2.3.6. *Let b and σ be the values computed in Line 1 of Algorithm 2.5, and let $x^* = \chi(b, \sigma^{-1})$. Then $\mathrm{Ord}(x^*) = \mathrm{Ord}(x)$.*

Proof. First, observe that $b \in \{0,1\}^n$ and $\sigma \in S_n$ because

- For all $i \in [n]$, both x_i and $1 - x_i$ can be written as $b_i x_i + (1 - b_i)(1 - x_i)$ for some $b_i \in \{0, 1\}$;
- $\mathrm{Ord}(x)$ is *palindrome*, i.e., if x_{i_1} is the largest in $\{x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n\}$ then $1 - x_{i_1}$ is the smallest in $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$; if $1 - x_{i_2}$ is the second largest in $\{x_1, \ldots, x_n, 1-x_1, \ldots, 1-x_n\}$ then x_{i_2} is the second smallest in $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$; etc.

Recall that in (2.3.46), the decreasing order of $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$ is

$$\begin{aligned} &b_{\sigma(1)}x_{\sigma(1)} + (1 - b_{\sigma(1)})(1 - x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1 - b_{\sigma(n)})(1 - x_{\sigma(n)}) \\ &\geq (1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq \cdots \geq (1 - b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1 - x_{\sigma(1)}). \end{aligned} \tag{2.3.50}$$

On the other hand, by the definition of D_n, we have

$$\{x_1^*, \dots, x_n^*, 1-x_1^*, \dots, 1-x_n^*\} = \Big\{\frac{1}{2n+1}, \frac{2}{2n+1}, \dots, \frac{2n}{2n+1}\Big\}. \tag{2.3.51}$$

Combining (2.3.50) and (2.3.51), it suffices to prove that for any $i \in [n]$,

$$b_{\sigma(i)} x^*_{\sigma(i)} + (1-b_{\sigma(i)})(1-x^*_{\sigma(i)}) = 1 - \frac{i}{2n+1}; \tag{2.3.52}$$

$$(1-b_{\sigma(i)}) x^*_{\sigma(i)} + b_{\sigma(i)}(1-x^*_{\sigma(i)}) = \frac{i}{2n+1}. \tag{2.3.53}$$

We only prove (2.3.52); the proof of (2.3.53) follows symmetrically.

By (2.3.43), we have $x_j^* = (1-b_j)\frac{\sigma^{-1}(j)}{2n+1} + b_j(1-\frac{\sigma^{-1}(j)}{2n+1})$ for all $j \in [n]$; taking $j = \sigma(i)$, we have $x^*_{\sigma(i)} = (1-b_{\sigma(i)})\frac{i}{2n+1} + b_{\sigma(i)}(1-\frac{i}{2n+1})$. Moreover, since $b_{\sigma(i)} \in \{0,1\}$ implies that $b_{\sigma(i)}(1-b_{\sigma(i)}) = 0$ and $b^2_{\sigma(i)} + (1-b_{\sigma(i)})^2 = 1$, we have

$$\begin{aligned}
&b_{\sigma(i)} x^*_{\sigma(i)} + (1-b_{\sigma(i)})(1-x^*_{\sigma(i)}) \\
&\quad = b_{\sigma(i)}\big[(1-b_{\sigma(i)})\tfrac{i}{2n+1} + b_{\sigma(i)}\big(1-\tfrac{i}{2n+1}\big)\big] \\
&\qquad + (1-b_{\sigma(i)})\big[b_{\sigma(i)}\tfrac{i}{2n+1} + (1-b_{\sigma(i)})\big(1-\tfrac{i}{2n+1}\big)\big] && (2.3.54)\\
&\quad = 2b_{\sigma(i)}(1-b_{\sigma(i)})\tfrac{i}{2n+1} + \big(b^2_{\sigma(i)} + (1-b_{\sigma(i)})^2\big)\big(1-\tfrac{i}{2n+1}\big) && (2.3.55)\\
&\quad = 1 - \tfrac{i}{2n+1}, && (2.3.56)
\end{aligned}$$

which is exactly (2.3.52). □

Correctness of Line 3. In this paragraph, we prove:

Lemma 2.3.7. *There is some* $k^* \in \{1, \ldots, n+1\}$ *such that* $f(x^*) = 1 - \frac{k^*}{2n+1}$.

Proof. Because $|x_i^* - c_i|$ is an integer multiple of $\frac{1}{2n+1}$ for all $i \in [n]$, $f(x^*)$ must also be an integer multiple of $\frac{1}{2n+1}$. As a result, $k^* = (2n+1)(1 - f(x^*)) \in \mathbb{Z}$.

It remains to prove that $1 \leq k^* \leq n+1$. By the definition of D_n in (2.3.42), we have

$$x_i^* = (1 - b_i)\frac{\sigma^{-1}(i)}{2n+1} + b_i\Big(1 - \frac{\sigma^{-1}(i)}{2n+1}\Big) \qquad \forall\, i \in [n]; \tag{2.3.57}$$

since $b_i = 0$ or 1, we have $x_i^* \in \{\frac{\sigma^{-1}(i)}{2n+1}, 1 - \frac{\sigma^{-1}(i)}{2n+1}\}$. Because we also have $c_i \in \{0, 1\}$,

$$|x_i^* - c_i| \leq 1 - \frac{\sigma^{-1}(i)}{2n+1} \leq \frac{2n}{2n+1}. \tag{2.3.58}$$

As a result,

$$f(x^*) = \max_{i \in [n]} |x_i^* - c_i| \leq \frac{2n}{2n+1} \;\Rightarrow\; k^* \geq 1. \tag{2.3.59}$$

It remains to prove $k^* \leq n+1$. By (2.3.57), we have

$$x_{\sigma(n)}^* \in \Big\{\frac{n}{2n+1}, \frac{n+1}{2n+1}\Big\}; \tag{2.3.60}$$

because $c_{\sigma(n)} \in \{0, 1\}$, we have

$$|x_{\sigma(n)}^* - c_{\sigma(n)}| \geq \frac{n}{2n+1}. \tag{2.3.61}$$

Therefore, we have

$$f(x^*) = \max_{i\in[n]} |x_i^* - c_i| \geq |x_{\sigma(n)}^* - c_{\sigma(n)}| \geq \frac{n}{2n+1}, \tag{2.3.62}$$

which implies $k^* \leq n+1$. □

Correctness of Line 4. In this paragraph, we prove:

Lemma 2.3.8. *The output of $f(x)$ in Line 4 is correct.*

Proof. A key observation we use in the proof, following directly from (2.3.57), is that

$$|x_{\sigma(i)}^* - c_{\sigma(i)}| = \begin{cases} \frac{i}{2n+1} & \text{if } c_{\sigma(i)} = b_{\sigma(i)}; \\ 1 - \frac{i}{2n+1} & \text{if } c_{\sigma(i)} = 1 - b_{\sigma(i)}. \end{cases} \tag{2.3.63}$$

First, assume that $k^* \in \{1, \ldots, n\}$ (i.e., the "otherwise" case in (2.3.47) happens). By (2.3.63),

$$x_{\sigma(k^*)}^* \in \left\{\frac{k^*}{2n+1}, 1 - \frac{k^*}{2n+1}\right\}; \quad x_{\sigma(i)}^* \notin \left\{\frac{k^*}{2n+1}, 1 - \frac{k^*}{2n+1}\right\} \quad \forall\, i \neq k^*,$$

which implies that for all $i \neq k^*$, $|x_{\sigma(i)}^* - c_{\sigma(i)}| \neq 1 - \frac{k^*}{2n+1}$ since $c_{\sigma(i)} \in \{0, 1\}$. As a result, we must have

$$|x_{\sigma(k^*)}^* - c_{\sigma(k^*)}| = 1 - \frac{k^*}{2n+1}. \tag{2.3.64}$$

Together with (2.3.63), this implies

$$c_{\sigma(k^*)} = 1 - b_{\sigma(k^*)}. \tag{2.3.65}$$

For any $i < k^*$, if $c_{\sigma(i)} = 1 - b_{\sigma(i)}$, then (2.3.63) implies that

$$f(x^*) \geq |x^*_{\sigma(i)} - c_{\sigma(i)}| = 1 - \frac{i}{2n+1} > 1 - \frac{k^*}{2n+1}, \tag{2.3.66}$$

which contradicts with the assumption that $f(x^*) = 1 - \frac{k^*}{2n+1}$. Therefore, we must have

$$c_{\sigma(i)} = b_{\sigma(i)} \quad \forall\, i \in \{1, \ldots, k^* - 1\}. \tag{2.3.67}$$

Recall that the decreasing order of $\{x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n\}$ is

$$b_{\sigma(1)}x_{\sigma(1)} + (1 - b_{\sigma(1)})(1 - x_{\sigma(1)}) \geq \cdots \geq b_{\sigma(n)}x_{\sigma(n)} + (1 - b_{\sigma(n)})(1 - x_{\sigma(n)}) \geq$$
$$(1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq \cdots \geq (1 - b_{\sigma(1)})x_{\sigma(1)} + b_{\sigma(1)}(1 - x_{\sigma(1)}). \tag{2.3.68}$$

Based on (2.3.65), (2.3.67), and (2.3.68), we next prove

$$|x_{\sigma(k^*)} - c_{\sigma(k^*)}| \geq |x_{\sigma(i)} - c_{\sigma(i)}| \quad \forall\, i \in [n]. \tag{2.3.69}$$

If (2.3.69) holds, it implies

$$f(x) = \max_{i\in[n]} |x_i - c_i| = |x_{\sigma(k^*)} - c_{\sigma(k^*)}|. \tag{2.3.70}$$

If $b_{\sigma(k^*)} = 0$, then (2.3.65) implies $c_{\sigma(k^*)} = 1$, (2.3.70) implies $f(x) = 1 - x_{\sigma(k^*)}$, and the output in Line 4 satisfies

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) = 1 - x_{\sigma(k^*)} = f(x); \tag{2.3.71}$$

If $b_{\sigma(k^*)} = 1$, then (2.3.65) implies $c_{\sigma(k^*)} = 0$, (2.3.70) implies $f(x) = x_{\sigma(k^*)}$, and the output in Line 4 satisfies

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) = x_{\sigma(k^*)} = f(x). \tag{2.3.72}$$

The correctness of Line 4 follows.

It remains to prove (2.3.69). We divide its proof into two parts:

- Suppose $i < k^*$. By (2.3.68), we have

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}). \tag{2.3.73}$$

  - If $b_{\sigma(k^*)} = 0$ and $b_{\sigma(i)} = 0$, we have $c_{\sigma(k^*)} = 1$ and $c_{\sigma(i)} = 0$ by (2.3.65) and (2.3.67), respectively; (2.3.73) reduces to $1 - x_{\sigma(k^*)} \geq x_{\sigma(i)}$;
  - If $b_{\sigma(k^*)} = 0$ and $b_{\sigma(i)} = 1$, we have $c_{\sigma(k^*)} = 1$ and $c_{\sigma(i)} = 1$ by (2.3.65) and (2.3.67), respectively; (2.3.73) reduces to $1 - x_{\sigma(k^*)} \geq 1 - x_{\sigma(i)}$;

– If $b_{\sigma(k^*)} = 1$ and $b_{\sigma(i)} = 0$, we have $c_{\sigma(k^*)} = 0$ and $c_{\sigma(i)} = 0$ by (2.3.65) and (2.3.67), respectively; (2.3.73) reduces to $x_{\sigma(k^*)} \geq x_{\sigma(i)}$;

– If $b_{\sigma(k^*)} = 1$ and $b_{\sigma(i)} = 1$, we have $c_{\sigma(k^*)} = 0$ and $c_{\sigma(i)} = 1$ by (2.3.65) and (2.3.67), respectively; (2.3.73) reduces to $x_{\sigma(k^*)} \geq 1 - x_{\sigma(i)}$.

In each case, the resulting expression is exactly (2.3.69). Overall, we see that (2.3.69) is always true when $i < k^*$.

- Suppose $i > k^*$. By (2.3.68), we have

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq b_{\sigma(i)}x_{\sigma(i)} + (1 - b_{\sigma(i)})(1 - x_{\sigma(i)});$$

$$b_{\sigma(k^*)}x_{\sigma(k^*)} + (1 - b_{\sigma(k^*)})(1 - x_{\sigma(k^*)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}).$$

– If $b_{\sigma(k^*)} = 0$, we have $c_{\sigma(k^*)} = 1$ by (2.3.65); the two inequalities above give $1 - x_{\sigma(k^*)} \geq \max\{x_{\sigma(i)}, 1 - x_{\sigma(i)}\}$;

– If $b_{\sigma(k^*)} = 1$, we have $c_{\sigma(k^*)} = 0$ by (2.3.65); the two inequalities above give $x_{\sigma(k^*)} \geq \max\{x_{\sigma(i)}, 1 - x_{\sigma(i)}\}$.

Both cases imply (2.3.69), so we see this also holds for $i > k^*$.

The same proof works when $k^* = n + 1$. In this case, there is no $i \in [n]$ such that $i > k^*$; on the other hand, when $i < k^*$, we replace (2.3.73) by

$$(1 - b_{\sigma(n)})x_{\sigma(n)} + b_{\sigma(n)}(1 - x_{\sigma(n)}) \geq (1 - b_{\sigma(i)})x_{\sigma(i)} + b_{\sigma(i)}(1 - x_{\sigma(i)}), \qquad (2.3.74)$$

and the argument proceeds unchanged. □

2.3.4 Proof of our quantum lower bound on convex optimization

We now prove Theorem 2.1.2 using Theorem 2.3.2 and Theorem 2.3.3. Recall that our lower bounds on membership and evaluation queries are both proved on the n-dimensional hypercube. It remains to combine the two lower bounds to establish them simultaneously.

Theorem 2.3.4. *Let* $\mathcal{C}_s := \bigtimes_{i=1}^{n} [s_i - 2, s_i + 1]$ *for some* $s \in \{0,1\}^n$. *Consider a function* $f \colon \mathcal{C}_s \times [0,1]^n \to \mathbb{R}$ *such that* $f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x)$, *where for any* $x = (x_1, x_2, \ldots, x_{2n}) \in \mathcal{C}_s \times [0,1]^n$,

$$f_{\mathrm{M}}(x) = \sum_{i=1}^{n} x_i, \qquad f_{\mathrm{E},c}(x) = \max_{i \in \{n+1,\ldots,2n\}} |x_i - c_{i-n}| \tag{2.3.75}$$

for some $c \in \{0,1\}^n$. *Then outputting an* $\tilde{x} \in \mathcal{C}_s \times [0,1]^n$ *satisfying*

$$f(\tilde{x}) \leq \min_{x \in \mathcal{C}_s \times [0,1]^n} f(x) + \tfrac{1}{3} \tag{2.3.76}$$

with probability at least 0.8 *requires* $\Omega(\sqrt{n})$ *quantum queries to* $O_{\mathcal{C}_s \times [0,1]^n}$ *and* $\Omega(\sqrt{n}/\log n)$ *quantum queries to* O_f.

Notice that the dimension of the optimization problem above is $2n$ instead of n; however, the constant overhead of 2 does not influence the asymptotic lower bounds.

Proof of Theorem 2.3.4. First, we prove that

$$\min_{x \in \mathcal{C}_s \times [0,1]^n} f(x) = S \qquad \text{and} \qquad \arg\min_{x \in \mathcal{C}_s \times [0,1]^n} f(x) = (s - 2_n, c), \tag{2.3.77}$$

where 2_n is the n-dimensional all-twos vector and $S := \sum_{i=1}^{n}(s_i - 2)$. On the one hand,

$$f_{\mathrm{M}}(x) \geq S \qquad \forall\, x \in \mathcal{C}_s \times [0,1]^n, \tag{2.3.78}$$

with equality if and only if $(x_1, \ldots, x_n) = s - 2_n$. On the other hand,

$$f_{\mathrm{E},c}(x) \geq 0 \qquad \forall\, x \in \mathcal{C}_s \times [0,1]^n, \tag{2.3.79}$$

with equality if and only if $(x_{n+1}, \ldots, x_{2n}) = c$. Thus $f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x) \geq S$ for all $x \in \mathcal{C}_s \times [0,1]^n$, with equality if and only if $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n}) = (s - 2_n, c)$.

If we solve this optimization problem with output $\tilde{x}$ satisfying (2.3.76), then

$$f_{\mathrm{M}}(\tilde{x}) + f_{\mathrm{E},c}(\tilde{x}) = f(\tilde{x}) \leq S + \tfrac{1}{3}. \tag{2.3.80}$$

Eqs. (2.3.78), (2.3.79), and (2.3.80) imply

$$f_{\mathrm{M}}(\tilde{x}) \leq S + \tfrac{1}{3} = \min_{x \in \mathcal{C}_s \times [0,1]^n} f_{\mathrm{M}}(x) + \tfrac{1}{3}; \tag{2.3.81}$$

$$f_{\mathrm{E},c}(\tilde{x}) \leq \tfrac{1}{3} = \min_{x \in \mathcal{C}_s \times [0,1]^n} f_{\mathrm{E},c}(x) + \tfrac{1}{3}. \tag{2.3.82}$$

On the one hand, Eq. (2.3.81) says that $\tilde{x}$ also minimizes f_{M} with approximation error $\epsilon = \frac{1}{3}$. By Theorem 2.3.2, this requires $\Omega(\sqrt{n})$ queries to the membership oracle $O_{\mathcal{C}_s}$. Also notice that one query to $O_{\mathcal{C}_s \times [0,1]^n}$ can be trivially simulated one query to $O_{\mathcal{C}_s}$; therefore, minimizing f with approximation error $\epsilon = \frac{1}{3}$ with success probability 0.9 requires $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$.

On the other hand, Eq. (2.3.82) says that $\tilde{x}$ minimizes $f_{\mathrm{E},c}$ with approximation error $\epsilon = \frac{1}{3}$. By Theorem 2.3.3, it takes $\Omega(\sqrt{n}/\log n)$ queries to $O_{f_{\mathrm{E},c}}$ to output $\tilde{x}$. Also notice that

$$f(x) = f_{\mathrm{M}}(x) + f_{\mathrm{E},c}(x) = \sum_{i=1}^{n} x_i + f_{\mathrm{E},c}(x); \tag{2.3.83}$$

therefore, one query to O_f can be simulated by one query to $O_{f_{\mathrm{E},c}}$. Therefore, approximately minimizing f with success probability 0.9 requires $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f.

In addition, f_{M} is independent of the coordinates $x_{n+1}, \ldots, x_{2n}$ and only depends on the coordinates $x_1, \ldots, x_n$, whereas $f_{\mathrm{E},c}$ is independent of the coordinates $x_1, \ldots, x_n$ and only depends on the coordinates $x_{n+1}, \ldots, x_{2n}$. As a result, the oracle $O_{\mathcal{C}_s \times [0,1]^n}$ reveals no information about c, and O_f reveals no information about s.

Since solving the optimization problem reveals both s and c, the lower bounds on query complexity must hold simultaneously.

Overall, to output an $\tilde{x} \in \mathcal{C}_s \times [0,1]^n$ satisfying (2.3.76) with success probability at least $0.9 \cdot 0.9 > 0.8$, we need $\Omega(\sqrt{n})$ quantum queries to $O_{\mathcal{C}_s \times [0,1]^n}$ and $\Omega(\sqrt{n}/\log n)$ quantum queries to O_f, as claimed. □

2.3.5 Side points on our quantum lower bound

Smoothed hypercube. We want to point out that our quantum lower bound in Theorem 2.3.4 also holds for a smooth convex body. Given an n-dimensional hypercube $\mathcal{C}_{x,l} := \bigtimes_{i=1}^n [x_i - l, x_i]$, we define a smoothed version as

$$\mathcal{SC}_{x,l} := B_2\left(\bigtimes_{i=1}^n \left[x_i - \frac{2n}{2n+1}l, x_i - \frac{1}{2n+1}l\right], \frac{1}{2n+1}l\right) \tag{2.3.84}$$

using Definition 2.2.3. For instance, a smoothed 3-dimensional cube is shown in Figure 2.1.

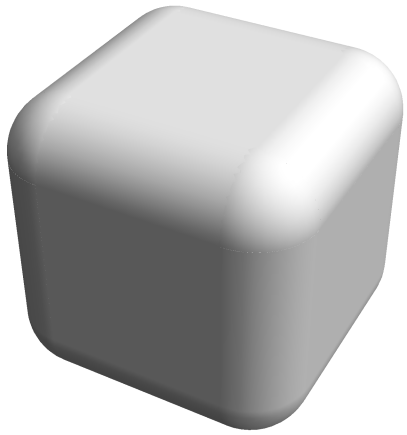

Figure 2.1: Smoothed hypercube of dimension 3.

The smoothed hypercube satisfies

$$\mathcal{C}_{x-\frac{1}{2n+1}l_n,\frac{2n-1}{2n+1}l} \subseteq \mathcal{SC}_{x,l} \subseteq \mathcal{C}_{x,l} \tag{2.3.85}$$

where l_n is l times the n-dimensional all-ones vector; in other words, it is contained in the original (non-smoothed) hypercube, and it contains the hypercube with the same center but edge length $\frac{2n-1}{2n+1}l$. For instance, $\bigtimes_{i=1}^n [\frac{1}{2n+1}, \frac{2n}{2n+1}] \subseteq \mathcal{SC}_{1_n,1} \subseteq \bigtimes_{i=1}^n [0,1]$; by Eq. (2.3.42), $D_n \subseteq \mathcal{SC}_{1_n,1}$. It can be verified that the proof of Theorem 2.3.2 still holds if the hypercube $\bigtimes_{i=1}^n [s_i - 2, s_i + 1] = \mathcal{C}_{s+1_n,3}$ is replaced by $\mathcal{SC}_{s+1_n,3}$, and the proof of Theorem 2.3.3 still holds if the unit hypercube $[0,1]^n$ is replaced by $\mathcal{SC}_{1_n,1}$; consequently Theorem 2.3.4 also holds. More generally, the proofs remain valid as long as the smoothed hypercube is contained in $[0,1]^n$ and contains D_n (for discretization).

Optimality of Theorem 2.3.3. In this paragraph, we prove that the lower bound in Theorem 2.3.3 is optimal (up to poly-logarithmic factors in n) for the max-norm optimization problem:

Theorem 2.3.5. *Let $f_c\colon [0,1]^n \to [0,1]$ be an objective function for the max-norm optimization problem (Definition 2.3.2). Then there exists a quantum algorithm that outputs an $\tilde{x} \in [0,1]^n$ satisfying (2.3.23) with $\epsilon = 1/3$ using $O(\sqrt{n}\log n)$ quantum queries to O_f, with success probability at least 0.9.*

In other words, the quantum query complexity of the max-norm optimization problem is $\tilde{\Theta}(\sqrt{n})$.

We prove Theorem 2.3.5 also using search with wildcards (Theorem 2.3.1).

Proof. It suffices to prove that one query to the wildcard query model O_c in (2.3.1) can be simulated by one query to O_{f_c}, where the c in (2.3.14) is the string c in the wildcard query model.

Assume that we query (T, y) using the wildcard query model. Then we query $O_{f_c}(x^{(T,y)})$ where for all $i \in [n]$,

$$x_i^{(T,y)} = \begin{cases} \frac{1}{2} & \text{if } i \notin T; \\ 0 & \text{if } i \in T \text{ and } y_i = 0; \\ 1 & \text{if } i \in T \text{ and } y_i = 1. \end{cases} \tag{2.3.86}$$

If $c_{|T} = y$, then

- if $|T| = n$ (i.e., $T = [n]$), then

$$f_c(x) = \max_{i \in [n]} |x_i^{(T,y)} - c_i| = 0 \tag{2.3.87}$$

because for any $i \in [n]$, $x_i^{(T,y)} = y_i = c_i$;

- if $|T| \leq n - 1$, then

$$f_c(x) = \max_{i \in [n]} |x_i^{(T,y)} - c_i| + g_i = \frac{1}{2}, \tag{2.3.88}$$

because for all $i \in T$ we have $x_i^{(T,y)} = y_i = c_i$ and hence $|x_i^{(T,y)} - c_i| = 0$, and for all $i \notin T$ we have $|x_i^{(T,y)} - c_i| = |\frac{1}{2} - c_i| = \frac{1}{2}$.

Therefore, if $c_{|T} = y$, then we must have $f_c(x^{(T,y)}) \in \left\{0, \frac{1}{2}\right\}$.

On the other hand, if $c_{|T} \neq y$, then there exists an $i_0 \in T$ such that $c_{i_0} \neq y_{i_0}$. This implies $x_{i_0}^{(T,y)} = 1 - c_{i_0}$; as a result, $f_c(x^{(T,y)}) = 1$ because on the one hand $f_c(x^{(T,y)}) \geq |1 - c_{i_0} - c_{i_0}| = 1$, and on the other hand $f_c(x^{(T,y)}) \leq 1$ as $|x_i^{(T,y)} - c_i| \leq 1$ for all $i \in [n]$.

Notice that the sets $\left\{0, \frac{1}{2}\right\}$ and $\{1\}$ do not intersect. Therefore, after we query $O_{f_c}(x^{(T,y)})$ and obtain the output, we can tell $Q_s(T, y) = 1$ in (2.3.1) if $O_{f_c}(x^{(T,y)}) \in \left\{0, \frac{1}{2}\right\}$, and output $Q_s(T, y) = 0$ if $O_{f_c}(x^{(T,y)}) = 1$. In all, this gives a simulation of one query to the wildcard query model O_c by one query to O_{f_c}.

As a result of Theorem 2.3.1, there is a quantum algorithm that outputs the c in (2.3.14) using $O(\sqrt{n} \log n)$ quantum queries to O_f. If we take $\tilde{x} = c$, then $f_c(\tilde{x}) = \max_i |c_i - c_i| = 0$, which is actually the optimal solution with $\epsilon = 0$ in (2.3.23). This establishes Theorem 2.3.5. □

2.4 Conclusions and discussion

In this chapter, we present a quantum algorithm that can optimize a convex function over an n-dimensional convex body using $\tilde{O}(n)$ queries to oracles that evaluate the objective function and determine membership in the convex body. This represents a quadratic improvement over the best-known classical algorithm. We also study limitations on the power of quantum computers for general convex optimization, showing that it requires $\tilde{\Omega}(\sqrt{n})$ evaluation queries and $\Omega(\sqrt{n})$ membership queries.

Related independent work. In independent simultaneous work, van Apeldoorn, Gilyén, Gribling, and de Wolf [25] establish a similar upper bound, showing that $\tilde{O}(n)$ quantum queries to a membership oracle suffice to optimize a linear function over a convex body (i.e., to implement an optimization oracle). Their proof follows a similar strategy to ours, using a quantum algorithm for evaluating gradients in $\tilde{O}(1)$ queries to implement a separation oracle. As in our approach, they use a randomly sampled point in the neighborhood of the point where the subgradient is to be calculated. The only major difference is that they use finite approximations of the gradient and second derivatives, whereas we use these quantities in their original form and give an argument based on mollifier functions to ensure that they are well defined.

Reference [25] also establishes quantum lower bounds on the query complexity of convex optimization, showing in particular that $\Omega(\sqrt{n})$ quantum queries to a separation oracle are needed to implement an optimization oracle, implying an $\Omega(\sqrt{n})$ quantum lower bound on the number of membership queries required to optimize a convex function. While Ref. [25] does not explicitly focus on evaluation queries, those authors have pointed out to us that an $\Omega(\sqrt{n})$ lower bound on evaluation queries can be obtained from their lower bound on membership queries (although our approach gives a bound with a better Lipschitz parameter).

Open questions. Our work leaves several natural open questions for future investigation. In particular:

- Can we close the gap for both membership and evaluation queries? Our upper

bounds on both oracles in Theorem 2.1.1 uses $\tilde{O}(n)$ queries, whereas the lower bounds of Theorem 2.1.2 are only $\tilde{\Omega}(\sqrt{n})$.

- Can we improve the time complexity of our quantum algorithm? The time complexity $\tilde{O}(n^3)$ of our current quantum algorithm matches that of the classical state-of-the-art algorithm [183] since our second step, the reduction from optimization to separation, is entirely classical. Is it possible to improve this reduction quantumly?

- What is the quantum complexity of convex optimization with a first-order oracle (i.e., with direct access to the gradient of the objective function)? This model has been widely considered in the classical literature (see for example Ref. [215]).

Chapter 3: Volume Estimation[1]

Having studied quantum algorithms for convex optimization in Chapter 2, another closely related question is to estimate the volume of convex bodies, which is the main focus of this chapter.

3.1 Introduction

Volume estimation is a central challenge in theoretical computer science and a basic problem in convex geometry—it can be viewed as a continuous version of counting. Furthermore, algorithms for a generalization of volume estimation—namely log-concave sampling—can be directly used to perform convex optimization, and hence can be widely applied to problems in statistics, machine learning, operations research, etc. See the survey [269] for a more comprehensive introduction.

Volume estimation is a notoriously difficult problem. References [39, 106] proved that any *deterministic algorithm* that approximates the volume of an n-dimensional convex body within a factor of $n^{o(n)}$ necessarily makes exponentially many queries to a membership oracle for the convex body. Furthermore, Refs. [104, 171, 172] showed that estimating the volume exactly (deterministically) is #P-hard,

[1]This chapter is based on the paper [66] under the permission of all the authors.

even for explicitly described polytopes.

Surprisingly, volumes of convex bodies can be approximated efficiently by *randomized algorithms*. Reference [103] gave the first polynomial-time randomized algorithm for estimating the volume of a convex body in $\mathbb{R}^n$. It presents an iterative algorithm that constructs a sequence of convex bodies. The volume of the convex body of interest can be written as the telescoping product of the ratios of the volumes of consecutive convex bodies, and these ratios are estimated by Markov chain Monte Carlo (MCMC) methods via random walks inside these convex bodies. The algorithm in [103] has complexity $\tilde{O}(n^{23})$ with multiplicative error $\epsilon = \Theta(1)$. Subsequent work [26, 102, 160, 196–198, 200] improved the design of the iterative framework and the choice of the random walks. The state-of-the-art algorithm for estimating the volume of a general convex body [201] uses $\tilde{O}(n^4)$ queries to the oracle for the convex body and $\tilde{O}(n^6)$ additional arithmetic operations. It has been an open question for around 15 years to improve this $\tilde{O}(n^4)$ query complexity.[2]

It is natural to ask whether quantum computers can solve volume estimation even faster than classical randomized algorithms. Although there are frameworks with potential quantum speedup for simulated annealing algorithms in general, with volume estimation as a possible application [273], we are not aware of any previous quantum speedup for volume estimation. There are several reasons to develop such a result. First, quantum algorithms for volume estimation can be seen as performing a continuous version of quantum counting [57, 58], a key algorithmic technique

[2]The volume estimation literature mainly focuses on improving the dependence on n, treating ϵ as a constant. To be more explicit, the algorithm in [201] has query complexity $\tilde{O}(n^4/\epsilon^2)$.

with wide applications in quantum computing. Second, quantum algorithms for volume estimation can exploit quantum MCMC methods (e.g., [208, 236, 272]), and a successful quantum volume estimation algorithm may illuminate the application of quantum MCMC methods in other scenarios. Third, there has been recent progress on quantum algorithms for convex optimization [25, 67], so it is natural to understand the closely related task of estimating volumes of convex bodies.

Formulation. Given a convex set $\mathrm{K} \subset \mathbb{R}^n$, we consider the problem of estimating its volume

$$\mathrm{Vol}(\mathrm{K}) := \int_{x \in \mathrm{K}} \mathrm{d}x. \tag{3.1.1}$$

To get a basic sense about the location of K, we assume that it contains the origin. Furthermore, we assume that we are given inner and outer bounds on K, namely

$$\mathrm{B}_2(0, r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, R), \tag{3.1.2}$$

where $\mathrm{B}_2(x, l)$ is the ball of radius l in ℓ_2-norm centered at $x \in \mathbb{R}^n$. Denote $D := R/r$.

We consider the very general setting where the convex body K is only specified by an oracle. In particular, we have a *membership oracle*[3] for K that determines whether a given $x \in \mathbb{R}^n$ belongs to K. The efficiency of volume estimation is then measured by the number of queries to the membership oracle (i.e., the *query*

[3]The membership oracle is commonly used in convex optimization research (see for example [127]). This model is not only general but also of practical interest. For instance, when K is a bounded convex polytope, the membership oracle can be efficiently implemented by checking if all its linear constraints are satisfied; see also [187].

complexity) and the total number of other arithmetic operations.

In the quantum setting, the membership oracle is a unitary operator O_{K} as in (1.3.2). Specifically, we have

$$O_{\mathrm{K}}|x, 0\rangle = |x, \delta[x \in \mathrm{K}]\rangle \qquad \forall x \in \mathbb{R}^n, \tag{3.1.3}$$

where $\delta[P]$ is 1 if P is true and 0 if P is false.[4] In other words, we allow coherent superpositions of queries to the membership oracle. If the classical membership oracle can be implemented by an explicit classical circuit, then the corresponding quantum membership oracle can be implemented by a quantum circuit of about the same size. Therefore, the quantum query model provides a useful framework for understanding the quantum complexity of volume estimation.

3.1.1 Contributions

Our main result is a quantum algorithm for volume estimation:

Theorem 3.1.1 (Main Theorem)**.** *Let* $\mathrm{K} \subset \mathbb{R}^n$ *be a convex set with* $\mathrm{B}_2(0, r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, R)$*. Assume* $0 < \epsilon < 1/2$*. Then there is a quantum algorithm that returns a value* $\widetilde{\mathrm{Vol}(\mathrm{K})}$ *satisfying*

$$\frac{1}{1+\epsilon}\,\mathrm{Vol}(\mathrm{K}) \leq \widetilde{\mathrm{Vol}(\mathrm{K})} \leq (1+\epsilon)\,\mathrm{Vol}(\mathrm{K}) \tag{3.1.4}$$

[4] Here x can be approximated just as in the classical algorithms, such as with floating point numbers. Our algorithmic approach is robust under discretization (see [66, Section 5]), and our quantum lower bound holds even when x is stored with arbitrary precision (Section 3.5). We mostly assume for convenience that O_{K} operates on $x \in \mathbb{R}^n$, since this neither presents a serious obstacle nor conveys significant power.

using $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ *quantum queries to the membership oracle* O_{K} *(defined in (3.1.3)) and* $\tilde{O}(n^5 + n^{4.5}/\epsilon)$ *additional arithmetic operations.*[5]

To the best of our knowledge, this is the first quantum algorithm that achieves quantum speedup for this fundamental problem, compared to the classical state-of-the-art algorithm [88, 201] that uses $\tilde{O}(n^4+n^3/\epsilon^2)$ classical queries and $\tilde{O}(n^6+n^5/\epsilon^2)$ additional arithmetic operations.[6] Furthermore, our quantum algorithm not only achieves a quantum speedup in query complexity, but also in the number of arithmetic operations for executing the algorithm. This differs from previous quantum algorithms for convex optimization [25, 67] where only the query complexity is improved, but the gate complexity is the same as that of the classical state-of-the-art algorithm [183, 184].

On the other hand, we prove that volume estimation with $\epsilon = \Theta(1)$ requires $\Omega(\sqrt{n})$ quantum queries to the membership oracle, ruling out the possibility of achieving superpolynomial quantum speedup for volume estimation (see Theorem 3.5.1).

Technically, we refine a framework for achieving quantum speedups of simulated annealing algorithms, which might be of independent interest. Our framework applies to MCMC algorithms with cooling schedules that ensure each ratio in a tele-

[5] Arithmetic operations (e.g., addition, subtraction, multiplication, and division) can be in principle implemented by a universal set of quantum gates using the Solovay-Kitaev Theorem [95] up to a small overhead. In our quantum algorithm, the number of arithmetic operations is dominated by n-dimensional matrix-vector products computed in superposition for rounding the convex body (see Section 3.4.4).

[6] This is achieved by applying [201] to preprocess the convex body to be well-rounded using $\tilde{O}(n^4)$ queries and then applying [88] using $\tilde{O}(n^3/\epsilon^2)$ queries to estimate the volume of the well-rounded convex body. The number of additional arithmetic operations has an overhead of $O(n^2)$ due to the affine transformation in rounding.

scoping product has bounded variance, an approach known as *Chebyshev cooling*. Furthermore, we propose several novel techniques when implementing this framework, including a theory of continuous-space quantum walks with rigorous bounds on discretization error, a quantum algorithm for nondestructive mean estimation, and a quantum algorithm with interlaced rounding and volume estimation of convex bodies (as described further in Section 3.1.2 below). In principle, our techniques apply not only to the integral of the identity function (as in Theorem 3.1.1), but could also be applied to any log-concave function defined on a convex body, following the approach in [199].

We summarize our main results in Table 3.1.

	Classical bounds	Quantum bounds (this paper)
Query complexity	$\tilde{O}(n^4 + n^3/\epsilon^2)$ [88, 201], $\tilde{\Omega}(n^2)$ [231]	$\tilde{O}(n^3 + n^{2.5}/\epsilon)$, $\Omega(\sqrt{n})$
Total complexity	$\tilde{O}\big((n^2 + C_{\mathrm{MEM}}) \cdot (n^4 + n^3/\epsilon^2)\big)$ [88, 201]	$\tilde{O}\big((n^2 + C_{\mathrm{MEM}}) \cdot (n^3 + n^{2.5}/\epsilon)\big)$

Table 3.1: Summary of complexities of volume estimation, where n is the dimension of the convex body, ϵ is the multiplicative precision of volume estimation, and C_{MEM} is the cost of applying the membership oracle once. Total complexity refers to the cost of the of queries plus the number of additional arithmetic operations.

3.1.2 Techniques

We now summarize the key technical aspects of our work.

3.1.2.1 Classical volume estimation framework

Volume estimation by simulated annealing. The volume of a convex body K can be estimated using simulated annealing. Consider the value

$$Z(a) := \int_{\mathrm{K}} e^{-a\|x\|_2} \, \mathrm{d}x, \tag{3.1.5}$$

where $\|x\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$ is the ℓ_2-norm of x. On the one hand, $Z(0) = \mathrm{Vol(K)}$; on the other hand, because $e^{-\|x\|_2}$ decays exponentially fast with $\|x\|_2$, taking a large enough a ensures that the vast majority of $Z(a)$ concentrates near 0, so it can be well approximated by integrating on a small ball centered at 0. Therefore, a natural strategy is to consider a sequence $a_0 > a_1 > \cdots > a_m$ with a_0 sufficiently large and a_m close to 0. We consider a simulated annealing algorithm that iteratively changes a_i to a_{i+1} and estimates Vol(K) by the telescoping product

$$\mathrm{Vol(K)} \approx Z(a_m) = Z(a_0) \prod_{i=0}^{m-1} \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.1.6}$$

In the i^{th} step, a random walk is used to sample the distribution over K with density proportional to $e^{-a_i\|x\|_2}$. Denote one such sample by X_i, and let $V_i := e^{(a_i - a_{i+1})}\|X_i\|_2$. Then we have

$$\mathbb{E}[V_i] = \int_{\mathrm{K}} e^{(a_i - a_{i+1})\|x\|_2} \frac{e^{-a_i\|x\|_2}}{Z(a_i)}\,\mathrm{d}x = \int_{\mathrm{K}} \frac{e^{-a_{i+1}\|x\|_2}}{Z(a_i)}\,\mathrm{d}x = \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.1.7}$$

Therefore, each ratio $\frac{Z(a_{i+1})}{Z(a_i)}$ can be estimated by taking i.i.d. samples X_i, computing the corresponding V_is, and taking their average.

We can analyze this volume estimation algorithm by considering its behavior at three levels:

1) High level: The algorithm follows the simulated annealing framework described above, where the volume is estimated by a telescoping product as in (3.1.6).

2) Middle level: The number of i.i.d. samples used to estimate $\mathbb{E}[V_i]$ (a ratio in the

telescoping product given by (3.1.7)) is small. Intuitively, the annealing schedule should be slow enough that V_i has small variance.

3) Low level: The random walk converges fast so that we can take each i.i.d. sample of V_i efficiently.

Classical volume estimation algorithm. Our approach follows the classical volume estimation algorithm in [201] (see also Section 3.4.1). At the high level, it is a simulated annealing algorithm that estimates the volume of an alternative convex body K' produced by the *pencil construction*, which intersects a cylinder $[0, 2R/r] \times \mathrm{K}$ and a cone $\mathrm{C} := \{x \in \mathbb{R}^{n+1} : x_0 \geq 0, \|x\|_2 \leq x_0\}$. This construction replaces the integral (3.1.5) by $Z(a) = \int_{\mathrm{K}'} e^{-ax_0} \, \mathrm{d}x$, which is easier to calculate.

Without loss of generality, assume that $r = 1$. Reference [201] proves that if we take the sequence $a_0 > \cdots > a_m$ where $a_0 = 2n$, $a_{i+1} = (1 - \frac{1}{\sqrt{n}})a_i$, and $m = \tilde{O}(\sqrt{n})$, then $Z(a_0) \approx \int_{\mathrm{C}} e^{-a_0 x_0} \, \mathrm{d}x$ and

$$\mathrm{Var}[V_i^2] = O(1) \cdot \mathbb{E}[V_i]^2 \qquad \forall \, i \in [m], \tag{3.1.8}$$

i.e., the variance of V_i is bounded by a constant multiple of the square of its expectation. Such a simulated annealing schedule is known as *Chebyshev cooling* (see also Section 3.4.3.3). This establishes the middle-level requirement of the simulated annealing framework. Furthermore, [201] proves that the product of the average of $\tilde{O}(\sqrt{n}/\epsilon^2)$ i.i.d. samples of V_i for all $i \in [m]$ gives an estimate of $\mathrm{Vol}(\mathrm{K}')$ within multiplicative error ϵ with high success probability.

At the low level, Ref. [201] uses a *hit-and-run walk* to sample X_i. In this walk, starting from a point p, we uniformly sample a line ℓ through p and move to a random point along the chord $\ell \cap \mathrm{K}$ with density proportional to e^{-ax_0} (see Section 3.2.4 for details). Ref. [200] analyzes the convergence of the hit-and-run walk, proving that it converges to the distribution over K with density proportional to e^{-ax_0} within $\tilde{O}(n^3)$ steps, assuming that K is *well-rounded* (i.e., $R/r = O(\sqrt{n})$).

Finally, Ref. [201] constructs an affine transformation that transforms a general K to be well-rounded with $\tilde{O}(n^4)$ classical queries to its membership oracle, hence removing the constraint of the previous steps that K be well-rounded. Because the affine transformation is an n-dimensional matrix-vector product, this introduces an overhead of $O(n^2)$ in the number of arithmetic operations.

Overall, the algorithm has $\tilde{O}(\sqrt{n})$ iterations, where each iteration takes $\tilde{O}(\sqrt{n}/\epsilon^2)$ i.i.d. samples, and each sample takes $\tilde{O}(n^3)$ steps of the hit-and-run walk. In total, the query complexity is

$$\tilde{O}(\sqrt{n}) \cdot \tilde{O}(\sqrt{n}/\epsilon^2) \cdot \tilde{O}(n^3) = \tilde{O}(n^4/\epsilon^2). \tag{3.1.9}$$

The number of additional arithmetic operations is $\tilde{O}(n^4/\epsilon^2) \cdot O(n^2) = \tilde{O}(n^6/\epsilon^2)$ due to the affine transformation for rounding the convex body.

3.1.2.2 Quantum algorithm for volume estimation

It is natural to consider a quantum algorithm for volume estimation following the classical framework in Section 3.1.2.1. A naive attempt might be to develop a

quantum walk that achieves a generic quadratic speedup in mixing time. However, this is unfortunately difficult to achieve in general. Quantum walks are unitary processes that do not converge to stationary distributions in the classical sense. As a result, alternative and indirect quantum analogues of mixing properties of Markov chains have been proposed and studied (see Section 3.1.3 for more detail). None of these methods provide a direct replacement for classical mixing, and we cannot directly apply them in our context.

Instead, we adapt one of the frameworks proposed in [272]. To give a quantum speedup for volume estimation by this method, we address the following additional technical challenges:

- **Quantum walks in continuous space:** Quantum walks are mainly studied in discrete spaces [205, 254], and we need to understand how to define a quantum counterpart of the hit-and-run walk.

- **Quantum mean estimation:** Quantum counting [57] is a general tool for estimating a probability $p \in [0, 1]$ with quadratic speedup compared to classical sampling. However, estimating the mean of an unbounded random variable with a quantum version of Chebyshev concentration requires more advanced tools.

- **Rounding:** Classically, rounding a general convex body takes $\tilde{O}(n^4)$ queries [201], more expensive than volume estimation of a well-rounded body using $\tilde{O}(n^3/\epsilon^2)$ queries [88]. To achieve an overall quantum speedup, we also need to give a fast quantum algorithm for rounding convex bodies.

- **Error analysis of the quantum hit-and-run walk:** We must bound the error

incurred when implementing the quantum walk on a digital quantum computer with finite precision. Existing classical error analyses (e.g., [111]) do not automatically cover the quantum case.

We develop several novel techniques to resolve all these issues, outlined point-by-point as follows.

Theory of continuous-space quantum walks (Section 3.3). Our first technical contribution is to develop a quantum implementation of the low-level framework, i.e., to replace the classical hit-and-run walk by a *quantum hit-and-run walk*. However, although quantum walks in discrete spaces have been well studied (see for example [205, 254]), we are not aware of comparable results that can be used to analyze spectral properties and mixing times of quantum walks in continuous space. Here we describe a framework for continuous-space quantum walks that can be instantiated to give a quantum version of the hit-and-run walk. In particular, we formally define such walks and analyze their spectral properties, generalizing Szegedy's theory [254] to continuous spaces (Section 3.3.1). We also show a direct correspondence between the stationary distribution of a classical walk and a certain eigenvector of the corresponding quantum walk (Section 3.3.2).

Quantum volume estimation algorithm via simulated annealing (Section 3.4.2). Having described a quantum hit-and-run walk, the next step is to understand the high-level simulated annealing framework. As mentioned above, it is nontrivial to directly prepare stationary states of quantum walks. In this paper,

we follow a quantum MCMC framework proposed by [272] that can prepare stationary states of quantum walks by simulated annealing (see Section 3.2.2). In this framework, we have a sequence of slowly-varying Markov chains, and the stationary state of the initial Markov chain can be efficiently prepared. In each iteration, we apply fixed-point amplitude amplification of the quantum walk operator [130] due to Grover to transform the current stationary state to the next one; compared to classical slowly-varying Markov chains, the convergence rate of such quantum procedure is *quadratically better in spectral gap.*

Our **main technical contribution** is to show how to adapt the Chebyshev cooling schedule in [201] to the quantum MCMC framework in [272] using our quantum hit-and-run walk. The conductance lower bound together with the classical $\tilde{O}(n^3)$ mixing time imply that we can perform one step of fixed-point amplitude amplification using $\tilde{O}(n^{1.5})$ queries to O_{K}. Furthermore, the inner product between consecutive stationary states is a constant. These two facts ensure that the stationary state in each iteration can be prepared with $\tilde{O}(n^{1.5})$ queries to the membership oracle O_{K}. The total number of iterations is still $\tilde{O}(\sqrt{n})$, as in the classical case.

Quantum algorithm for nondestructive mean estimation (Section 3.4.3.3). In the next step, we consider how to estimate each ratio in the telescoping product at the middle level. Our main tool is quantum counting [57], which estimates a probability $p \in [0, 1]$ with error ϵ and high success probability using $O(1/\epsilon)$ quantum queries, a quadratic speedup compared to the classical complexity $O(1/\epsilon^2)$ due to Chernoff's bound. In our case, we need to estimate the expectation of a random

variable with bounded variance. This is achieved by truncating the random variable with reasonable upper and lower bounds and reducing to quantum counting, using the "quantum Chebyshev inequality" developed in [134] (see Section 3.2.3). Compared to the classical counterpart, this achieves quadratic speedup in the dependences on both variance and multiplicative error.

There is an additional technical difficulty in quantum simulated annealing: classically, it is implicitly assumed that in the $(i+1)^{\text{th}}$ iteration we have samples to the stationary distribution in the i^{th} iteration. Applying existing quantum mean estimation techniques to the quantum stationary state in the i^{th} iteration would ruin that state and make it hard to use in the subsequent $(i+1)^{\text{th}}$ iteration. To resolve this issue, we show how to estimate the mean *nondestructively* in the quantum Chebyshev inequality while keeping its quadratic speedup in the error dependence.

We achieve this nondestructive property by the following observation. The basic quantum counting algorithm with unitary operation U and state $|\phi\rangle$ [57] is composed of a quantum Fourier transform (QFT), controlled Us on $|\phi\rangle$, and then an inverse QFT, giving an estimate of $\langle 0|\langle 0|U|0\rangle|\phi\rangle$. If we apply a unitary operation that computes a function of $\langle 0|\langle 0|U|0\rangle|\phi\rangle$ in an ancilla register and then uncompute the counting circuit, then we get the state $|\phi\rangle$ back as well as the function value we need. Although the amplitude estimation only succeeds with probability $8/\pi^2$, this can be boosted to $1-\delta$ for any $\delta > 0$ by executing $O(\log 1/\delta)$ copies simultaneously and taking their median. One technical issue is that amplitude estimation can either give positive or negative phase angles (see (3.2.10)), but this can be fixed by applying a sine-square function in superposition on a separate register for each copy

(see (3.4.22)), computing the median of all the $O(\log 1/\delta)$ copies, and applying the inverse of all the sine-square functions and amplitude estimations.[7]

In our quantum volume estimation algorithm, we apply the quantum Chebyshev inequality under the same compute-uncompute procedure. This gives a quadratic speedup in ϵ^{-1} when estimating the $\mathbb{E}[V_i]$ in (3.1.7), so that $\tilde{O}(\sqrt{n}/\epsilon)$ copies of the stationary state suffice (see Lemma 3.4.3).

Quantum algorithm for volume estimation with interlaced rounding (Section 3.4.4). The stationary states of the quantum hit-and-run walk can be prepared with $\tilde{O}(n^{1.5})$ queries to O_{K} only when the corresponding density functions are *well-rounded* (i.e., every level set with probability μ contains a ball of radius μr, and the variance of the density is bounded by R^2, with $R/r = O(\sqrt{n})$). It remains to show how to ensure that the convex body is well-rounded.

Classically, Ref. [201] gave a rounding algorithm that transforms a convex body to ensure that all the densities sampled in the volume estimation algorithm are well-rounded. This algorithm uses $\tilde{O}(n^4)$ queries, via $\tilde{O}(n)$ iterations of simulated annealing. A quantization of this algorithm along the same lines as detailed above gives an algorithm with $\tilde{O}(n^{3.5})$ quantum queries.

To improve over that approach, we instead follow a classical framework for directly rounding logconcave densities [199]. The rounding is interlaced with the volume estimation algorithm, so that in each iteration of the simulated annealing

[7] A recent paper of Harrow and Wei [139] independently showed how to perform nondestructive amplitude estimation using a different approach: they directly apply a state restoration procedure inspired by [257] that first boosts the fidelity to 1/2 and then repeats the projection onto the measured state until it is restored.

framework, we use some of the samples to calculate an affine transformation that makes the next stationary state well-rounded. This ensures that the quantum hit-and-run walk continues to take only $\tilde{O}(n^{1.5})$ queries for each sample. Our algorithm maintains $\tilde{O}(n)$ extra quantum states for rounding, and the quantum hit-and-run walk is used to transform them from one stationary distribution to the next. In each iteration, we use a nondestructive measurement to sample the required affine transformation. With $\tilde{O}(\sqrt{n})$ iterations this results in an additional $\tilde{O}(\sqrt{n}) \cdot \tilde{O}(n) \cdot \tilde{O}(n^{1.5}) = \tilde{O}(n^3)$ cost for rounding.

We also show that this framework can be used as a preprocessing step that puts the convex body itself in well-rounded position (i.e., $\mathrm{B}_2(0, r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, R)$ with $R/r = O(\sqrt{n})$) using $\tilde{O}(n^3)$ quantum queries. Putting a convex body in well-rounded position implies that several random walks used in simulated annealing algorithms (including the hit-and-run walk) mix fast without the need for further rounding. Therefore, as an alternative, we could preprocess the convex body to be well-rounded and then apply the simulated annealing algorithm to obtain a volume estimation algorithm that uses $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ quantum queries.

Summary. Our quantum volume estimation algorithm is summarized as follows.

1) High level: The quantum algorithm follows a simulated annealing framework using a quantum MCMC method [272], where the volume is estimated by a telescoping product (as in (3.1.6)); the number of iterations is $\tilde{O}(\sqrt{n})$.

2) Middle level: We estimate the $\mathbb{E}[V_i]$ in (3.1.7), a ratio in the telescoping product, using the nondestructive version of the quantum Chebyshev inequality [134]. This

takes $\tilde{O}(\sqrt{n}/\epsilon)$ implementations of the quantum hit-and-run walk operators.

3) Low level: If the convex body K is well-rounded (i.e., $R/r = O(\sqrt{n})$), each quantum hit-and-run walk operator can be implemented using $\tilde{O}(n^{1.5})$ queries to the membership oracle O_{K} in (3.1.3).

Finally, we give a quantum algorithm that interlaces rounding and volume estimation of the convex body, using an additional $\tilde{O}(n^{2.5})$ quantum queries to O_{K} in each iteration. Because the affine transformation is an n-dimensional matrix-vector product, it introduces an overhead of $O(n^2)$ in the number of arithmetic operations (just as in the classical rounding algorithm).

Overall, our quantum volume estimation algorithm has $\tilde{O}(\sqrt{n})$ iterations. Each iteration implements $\tilde{O}(\sqrt{n}/\epsilon)$ quantum hit-and-run walks, and each quantum hit-and-run walk uses $\tilde{O}(n^{1.5})$ queries; there is also a cost of $\tilde{O}(n^{2.5})$ for rounding. Thus the quantum query complexity is

$$\tilde{O}(\sqrt{n}) \cdot \left(\tilde{O}(\sqrt{n}/\epsilon) \cdot \tilde{O}(n^{1.5}) + \tilde{O}(n^{2.5})\right) = \tilde{O}(n^3 + n^{2.5}/\epsilon). \tag{3.1.10}$$

The number of additional arithmetic operations is $\tilde{O}(n^3 + n^{2.5}/\epsilon) \cdot O(n^2) = \tilde{O}(n^5 + n^{4.5}/\epsilon)$ due to the affine transformations for interlaced rounding of the convex body.

Figure 3.1 summarizes the techniques in our quantum algorithm. The volume estimation and interlaced rounding algorithms are given as Algorithm 3.3 and Algorithm 3.4, respectively, in Section 3.4.

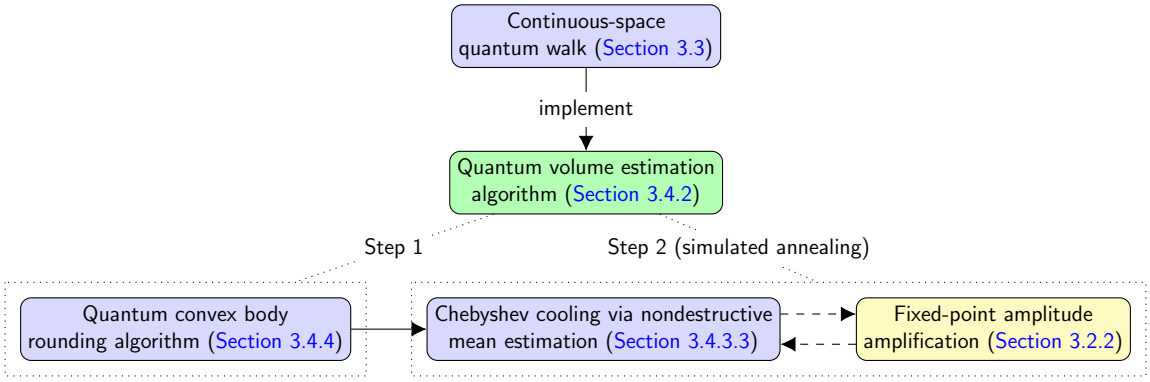

Figure 3.1: The structure of our quantum volume estimation algorithm. The purple frames represent the novel techniques that we propose, the yellow frame represents the known technique from [130], and the green frame at the center represents our quantum algorithm.

Quantum lower bound (Section 3.5). While we do not know whether the query complexity of our algorithm is tight, we prove that volume estimation requires $\Omega(\sqrt{n})$ quantum queries to a membership oracle, ruling out the possibility of exponential quantum speedup. We establish this lower bound by a reduction to search: for a hyper-rectangle $\mathrm{K} = \bigtimes_{i=1}^{n}[0, 2^{s_i}]$ specified by a binary string $s = (s_1, \ldots, s_n) \in \{0,1\}^n$ with $|s| = 0$ or 1, we prove that a membership query to K can be simulated by a query to s. Thus, since $\mathrm{Vol}(\mathrm{K}) = 2$ if and only if $|s| = 1$, the $\Omega(\sqrt{n})$ quantum lower bound on search [46] applies to volume estimation.

3.1.3 Related work

While our paper gives the first quantum algorithm for volume estimation, classical volume estimation algorithms have been well-studied. Quantumly, our quantum algorithm builds upon quantum analogs of Markov chain Monte Carlo methods.

Classical volume estimation algorithms. There is a rich literature on classical algorithms for estimating volumes of convex bodies (e.g., see the surveys [188, 269]). The general approach is to consider a sequence of random walks inside the convex body K whose stationary distributions converge quickly to the uniform distribution on K. Applying simulating annealing to this sequence of walks (as in Section 3.1.2), the volume of K can be approximated by a telescoping product.

The first polynomial-time algorithm for volume estimation was given by [103]. It uses a *grid walk* in which the convex body K is approximated by a grid mesh $\mathrm{K}_{\mathrm{grid}}$ of spacing δ (i.e., $\mathrm{K}_{\mathrm{grid}}$ contains the points in K whose coordinates are integer multiples of δ). The walk proceeds as follows:

1. Pick a grid point y uniformly at random from the neighbors of the current point x.

2. If $y \in \mathrm{K}_{\mathrm{grid}}$, go to y; else stay at x.

Reference [103] proved that for a properly chosen δ, the grid walk converges to the uniform distribution on $\mathrm{K}_{\mathrm{grid}}$ in $\tilde{O}(n^{23})$ steps, and that $\delta^n|\mathrm{K}_{\mathrm{grid}}|$ is a good approximation of Vol(K) (in the sense of (3.1.4)). Subsequently, more refined analysis of the grid walk improved its cost to $\tilde{O}(n^8)$ [26, 102, 197]. However, this is still inefficient in practice.

Intuitively, the grid walk converges slowly because each step only moves locally in K. Subsequent work improved the complexity by considering other types of random walk. These improvements mainly use two types of walk: the *hit-and-run walk* and the *ball walk*. In this paper, we use the hit-and-run walk (see also

Section 3.2.4), which behaves as follows:

1. Pick a uniformly distributed random line ℓ through the current point p.

2. Move to a uniformly random point along the chord $\ell \cap \mathrm{K}$.

Reference [246] proved that the stationary distribution of the hit-and-run walk is the uniform distribution on K. Regarding the convergence of the hit-and-run walk, [196] showed that it mixes in $\tilde{O}(n^3)$ steps from a warm start after appropriate preprocessing, and [200] subsequently proved that the hit-and-run walk mixes rapidly from any interior starting point (see also Theorem 3.2.4). Under the simulated annealing framework, the hit-and-run walk gives the state-of-the-art volume estimation algorithm with query complexity $\tilde{O}(n^4)$ [199, 201]. Our quantum volume estimation algorithm can be viewed as a quantization of this classical hit-and-run algorithm.

Given a radius parameter δ, the ball walk is defined as follows:

1. Pick a uniformly random point y from the ball of radius δ centered at the current point x.

2. If $y \in \mathrm{K}$, go to y; else stay at x.

Lovász and Simonovits [198] proved that the ball walk mixes in $\tilde{O}(n^6)$ steps. Reference [160] subsequently improved the mixing time to $\tilde{O}(n^3)$ starting from a warm start, giving a total query complexity of $\tilde{O}(n^5)$ for the volume estimation problem.

Technically, the analysis of the ball walk relies on a central conjecture in convex geometry, the Kannan-Lovász-Simonovits (KLS) conjecture (see [188]). The KLS

conjecture states that the Cheeger constant of any log-concave density is achieved to within a universal, dimension-independent constant factor by a hyperplane-induced subset, where the Cheeger constant is the minimum ratio between the measure of the boundary of a subset to the measure of the subset or its complement, whichever is smaller. Although this quantity is conjectured to be a constant, the best known upper bound is only $O(n^{1/4})$ [186], which can be used to prove that the ball walk converges in $\tilde{O}(n^{2.5})$ steps from a warm start. If the KLS conjecture were true, the ball walk would converge in $\tilde{O}(n^2)$ steps from a warm start, implying a volume estimation algorithm with query complexity $\tilde{O}(n^3)$ for arbitrary convex bodies.

If $R/r = O(\sqrt{n})$, then we say the body is *well-rounded*. In that special case, a recent breakthrough by Cousins and Vempala [87, 88] proved the KLS conjecture for Gaussian distributions. In other words, they established a volume estimation algorithm with query complexity $\tilde{O}(n^3)$ in the well-rounded case.

Table 3.2 summarizes classical algorithms for volume estimation.

Method	State-of-the-art query complexity	Restriction on the convex body
Grid walk	$\tilde{O}(n^8)$ [102]	General ($R/r = \text{poly}(n)$)
Hit-and-run walk	$\tilde{O}(n^4)$ [199, 201]	General ($R/r = \text{poly}(n)$)
Ball walk	$\tilde{O}(n^3)$ [87, 88]	Well-rounded ($R/r = O(\sqrt{n})$)

Table 3.2: Summary of classical methods for estimating the volume of a convex body $\mathrm{K} \subset \mathbb{R}^n$ when $\epsilon = \Theta(1)$, where R, r are the radii of the balls centered at the origin that contain and are contained by the convex body, respectively.

Quantum Markov chain Monte Carlo methods. The performance of Markov chain Monte Carlo (MCMC) methods is determined by the rate of convergence to their stationary distributions (i.e., the mixing time). Suppose we have a reversible, ergodic Markov chain with unique stationary distribution π. Let π_k denote the

distribution obtained by applying the Markov chain for k steps from some arbitrary initial state. It is well-known (see for example [190]) that $O(\frac{1}{\Delta}\log(1/\epsilon))$ steps suffice to ensure $\|\pi_k - \pi\| \leq \epsilon$, where Δ is the spectral gap of the Markov chain.

Many authors have studied quantum analogs of Markov chains (in both continuous [107] and discrete [13, 19, 254] time) and their mixing properties. While a quantum walk is a unitary process and hence does not converge to a stationary distribution, one can define notions of quantum mixing time by choosing the number of steps at random or by adding decoherence [13, 15, 19, 69, 80, 235, 236], and compare them to the classical mixing time. Note that distribution sampled by such a process may or not be the same as the stationary distribution π of the corresponding classical Markov process, depending on the structure of the process and the notion of mixing. It is also natural to ask how efficiently we can prepare a quantum state close to

$$|\pi\rangle := \sum_x \sqrt{\pi_x}|x\rangle \tag{3.1.11}$$

(which can be viewed as a "quantum sample" from π). However, it is unclear how to do this efficiently in general, even in cases where a corresponding classical Markov process mixes quickly; in particular, a generic quantum algorithm for this task could be used to solve graph isomorphism [14, Section 8.4].

It is also possible to achieve quantum speedup of MCMC methods by not demanding speedup of the mixing time of each separate Markov chain, but only for the procedure as a whole. In particular, MCMC methods are often implemented

by simulated annealing algorithms where the final output is a telescoping product of values at different temperatures. From this perspective, Somma et al. [53, 248, 249] used quantum walks to accelerate classical simulated annealing processes by exploiting the quantum Zeno effect, using measurements implemented by phase estimation of the quantum walk operators of these Markov chains. References [257, 277] also introduced how to implement Metropolis sampling on quantum computers.

Our quantum volume estimation algorithm is most closely related to work of Wocjan and Abeyesinghe [272], which achieves complexity $\tilde{O}(1/\sqrt{\Delta})$ for preparing the final stationary distribution of a sequence of slowly varying Markov chains, where Δ is the minimum of their spectral gaps. Their quantum algorithm transits between the stationary states of consecutive Markov chains by fixed-point amplitude amplification [130], which is implemented by amplitude estimation with $\tilde{O}(1/\sqrt{\Delta})$ implementations of the quantum walk operators of these Markov chains (see Section 3.2.2 for more details).

Our simulated annealing procedure preserves the slowly-varying property, so we adopt the framework of [272] in our algorithm for volume estimation (see Section 3.4.3.2). We develop several novel techniques (described in Section 3.1.2) that allow us to implement the steps of this framework efficiently. Note that the slowly-varying property also facilitates other frameworks that give efficient adiabatic [14] or circuit-based [224] quantum algorithms for generating quantum samples of the stationary state.

Previous work has mainly applied these quantum simulated annealing algorithms to estimating partition functions of discrete systems. Given an inverse tem-

perature $\beta > 0$ and a classical Hamiltonian $H\colon \Omega \to \mathbb{R}$ where Ω is a finite space, the goal is to estimate the partition function

$$Z(\beta) := \sum_{x \in \Omega} e^{-\beta H(x)} \tag{3.1.12}$$

within multiplicative error $\epsilon > 0$. Reference [273] gave a quantum algorithm that achieves quadratic quantum speedup with respect to both mixing time and accuracy.

The classical algorithm that Ref. [273] quantizes uses $\tilde{O}(\log |\Omega|)$ annealing steps to ensure that each ratio $Z(\beta_{i+1})/Z(\beta_i)$ is bounded. In fact, it is possible to relax this requirement and use a cooling schedule with only $\tilde{O}(\sqrt{\log |\Omega|})$ steps such that the variance of each ratio is bounded, so its mean can be well-approximated by Chebyshev's inequality; this is exactly the Chebyshev cooling technique [251] introduced in Section 3.1.2 (see also Section 3.4.3.3). Reference [208] improves upon [273] using Chebyshev cooling; more recently, Harrow and Wei [139] further quadratically improved the spectral gap dependence of the estimation of the partition function.

Organization. We review necessary background in Section 3.2. We describe the theory of continuous-space quantum walks in Section 3.3. In Section 3.4, we first review the classical state-of-the-art volume estimation algorithm in Section 3.4.1, and then give our quantum algorithm for estimating volumes of well-rounded convex bodies in Section 3.4.2. The proofs are given in Section 3.4.3, and the quantum algorithm for rounding convex bodies is given in Section 3.4.4. We conclude with our quantum lower bound on volume estimation in Section 3.5.

3.2 Preliminary tools

3.2.1 Classical and quantum walks

A Markov chain over a finite state space Ω is a sequence of random variables $X_0, X_1, \ldots$ such that for each $i \in \mathbb{N}$, the probability of transition to the next state $y \in \Omega$,

$$\Pr[X_{i+1} = y \mid X_i = x, X_{i-1} = x_{i-1}, \ldots, X_0 = x_0] = \Pr[X_{i+1} = y \mid X_i = x] =: p_{x \to y}$$

only depends on the present state $x \in \Omega$. The Markov chain can be represented by the transition probabilities $p_{x \to y}$ satisfying $\sum_y p_{x \to y} = 1$. For each $i \in \mathbb{N}$, we denote by π_i the distribution over Ω with density $\pi_i(x) = \Pr[X_i = x]$. A *stationary distribution* π satisfies $\sum_{x \in \Omega} p_{x \to y} \pi(x) = \pi(y)$. A Markov chain is *reversible* if $\pi_i(x) p_{x \to y} = \pi_i(y) p_{y \to x}$ for each $i \in \mathbb{N}$ and $x, y \in \Omega$. The *conductance* of a reversible Markov chain is defined as

$$\Phi := \inf_{\mathrm{S} \subseteq \Omega} \frac{\sum_{x \in \mathrm{S}} \sum_{y \in \Omega/\mathrm{S}} \pi(x) p_{x \to y}}{\min\{\sum_{x \in \mathrm{S}} \pi(x), \sum_{x \in \Omega/\mathrm{S}} \pi(x)\}}. \tag{3.2.1}$$

The theory of discrete-time quantum walks has also been well developed. Given a classical reversible Markov chain on Ω with transition probability p, we

define a unitary operator U_p on $\mathbb{C}^{|\Omega|} \otimes \mathbb{C}^{|\Omega|}$ such that

$$U_p|x\rangle|0\rangle = |x\rangle|p_x\rangle, \text{ where } |p_x\rangle := \sum_{y\in\Omega} \sqrt{p_{x\to y}}|y\rangle. \tag{3.2.2}$$

The quantum walk is then defined as [254]

$$W_p := S\big(2U_p(I_\Omega \otimes |0\rangle\langle 0|)U_p^\dagger - I_\Omega \otimes I_\Omega\big), \tag{3.2.3}$$

where I_Ω is the identity map on $\mathbb{C}^{|\Omega|}$ and $S := \sum_{x,y\in\Omega} |x,y\rangle\langle y,x| = S^\dagger$ is the swap gate on $\mathbb{C}^{|\Omega|} \otimes \mathbb{C}^{|\Omega|}$.

To understand the quantum walk, it is essential to analyze the spectrum of W_p. First, observing that $W_p = S(2\Pi - I)$ where $\Pi = U_p(I_\Omega \otimes |0\rangle\langle 0|)U_p^\dagger = \sum_{x\in\Omega} |x\rangle\langle x| \otimes |p_x\rangle\langle p_x|$ projects onto the span of the states $|x\rangle \otimes |p_x\rangle$, we consider the eigenvector $|\lambda\rangle$ of $\Pi S\Pi$ with eigenvalue λ. We have $\Pi S\Pi = \sum_{x\in\Omega} D_{xy} |x\rangle\langle y| \otimes |p_x\rangle\langle p_y|$ where $D_{xy} := \sqrt{p_{x\to y}p_{y\to x}}$. Since $W_p|\lambda\rangle = S|\lambda\rangle$ and $W_pS|\lambda\rangle = 2\lambda S|\lambda\rangle - |\lambda\rangle$, the subspace $\text{span}\{|\lambda\rangle, S|\lambda\rangle\}$ is invariant under W_p. The eigenvalues of W_p within this subspace are $\lambda \pm i\sqrt{1-\lambda^2} = e^{\pm i \arccos\lambda}$. For more details, see [254].

The phase gap $\arccos\lambda \geq \sqrt{2(1-\lambda)} \geq \sqrt{2\delta}$, where δ is the spectral gap of D. Therefore, applying phase estimation using $O(1/\sqrt{\delta})$ calls to W_p suffices to distinguish the state corresponding to the stationary distribution of the classical Markov chain from the other eigenvectors.

3.2.2 Quantum speedup of MCMC sampling via simulated annealing

Consider a Markov chain with spectral gap Δ and stationary distribution π. Classically, it takes $\Theta(\frac{1}{\Delta}\log(1/\epsilon\pi_{\min})))$ steps to sample from a distribution $\tilde{\pi}$ such that $\|\tilde{\pi} - \pi\| \leq \epsilon$, where $\pi_{\min} := \min_i \pi_i$. Quantumly, [272] proved the following result about a sequence of slowly varying Markov chains:

Theorem 3.2.1 ([272, Theorem 2])**.** *Let $p_1, \ldots, p_r$ be the transition probabilities of r Markov chains with stationary distributions $\pi_1, \ldots, \pi_r$, spectral gaps $\delta_1, \ldots, \delta_r$, and quantum walk operators $W_1, \ldots, W_r$, respectively; let $\Delta := \min\{\delta_1, \ldots, \delta_r\}$. Assume that $|\langle\pi_i|\pi_{i+1}\rangle|^2 \geq p$ for some $0 < p < 1$ and all $i \in [r-1]$, and assume that we can efficiently prepare the state $|\pi_1\rangle$ (where each $|\pi_i\rangle$ is a quantum sample defined as in (3.1.11)). Then, for any $0 < \epsilon < 1$, there is a quantum algorithm that produces a quantum state $|\tilde{\pi}_r\rangle$ such that $\||\tilde{\pi}_r\rangle - |\pi_r\rangle\| \leq \epsilon$, using $\tilde{O}(r/p\sqrt{\Delta})$ steps of the quantum walk operators $W_1, \ldots, W_r$.*

Their quantum algorithm produces the states $|\pi_1\rangle, \ldots, |\pi_r\rangle$ sequentially, and can do so rapidly if consecutive states have significant overlap and the walks mix rapidly. Intuitively, this is achieved by amplitude amplification. However, to avoid overshooting, the paper uses a variant of standard amplitude amplification, known as $\pi/3$-*amplitude amplification* [130], that we now review.

Given two states $|\psi\rangle$ and $|\phi\rangle$, we let $\Pi_\psi := |\psi\rangle\langle\psi|$, $\Pi_\psi^\perp := I - \Pi_\psi$, $\Pi_\phi := |\phi\rangle\langle\phi|$,

and $\Pi_\phi^\perp := I - \Pi_\phi$. Define the unitaries

$$R_\psi := \omega\Pi_\psi + \Pi_\psi^\perp, \quad R_\phi := \omega\Pi_\phi + \Pi_\phi^\perp \qquad \text{where} \quad \omega = e^{i\frac{\pi}{3}}. \tag{3.2.4}$$

Given $|\langle\psi|\phi\rangle|^2 \geq p$, it can be shown that $|\langle\phi|R_\psi R_\phi|\psi\rangle|^2 \geq 1-(1-p)^3$. Recursively, one can establish the following:

Lemma 3.2.1 ([272, Lemma 1])**.** *Let $|\psi\rangle$ and $|\phi\rangle$ be two quantum states with $|\langle\psi|\phi\rangle|^2 \geq p$ for some $0 < p \leq 1$. Define the unitaries R_ψ, R_ϕ as in (3.2.4) and the unitaries U_m recursively as follows:*

$$U_0 = I, \qquad U_{m+1} = U_m\, R_\psi\, U_m^\dagger\, R_\phi\, U_m. \tag{3.2.5}$$

Then we have

$$|\langle\phi|U_m|\psi\rangle|^2 \geq 1-(1-p)^{3^m}, \tag{3.2.6}$$

and the unitaries in $\{R_\psi, R_\psi^\dagger, R_\phi, R_\phi^\dagger\}$ are used at most 3^m times in U_m.

Taking $m = \lceil \log_3(\ln(1/\epsilon)/p)\rceil$, the inner product between $|\phi\rangle$ and $U_m|\psi\rangle$ in (3.2.6) is at least $1-\epsilon$, and we use $3^m = O(\log(1/\epsilon)/p)$ unitaries from the set $\{R_\psi, R_\psi^\dagger, R_\phi, R_\phi^\dagger\}$.

To establish Theorem 3.2.1 by Lemma 3.2.1, it remains to construct the unitaries $R_i := \omega|\pi_i\rangle\langle\pi_i| + (I - |\pi_i\rangle\langle\pi_i|)$. In [272], this is achieved by phase estimation of the quantum walk operator W_i with precision $\sqrt{\Delta}/2$. Recall that if a classical

Markov chain has spectral gap δ, then the corresponding quantum walk operator has phase gap of at least $2\sqrt{\delta}$ (see Section 3.2.1). Therefore, phase estimation with precision $\sqrt{\Delta}/2$ suffices to distinguish between $|\pi_i\rangle$ and other eigenvectors of W_i. As a result, we can take

$$R_i = \mathsf{PhaseEst}(W_i)^\dagger \big(I \otimes \big(\omega|0\rangle\langle 0| + (I - |0\rangle\langle 0|)\big)\big)\mathsf{PhaseEst}(W_i). \tag{3.2.7}$$

3.2.3 Quantum Chebyshev inequality

Assume we are given a unitary U such that

$$U|0\rangle|0\rangle = \sqrt{p}|0\rangle|\phi\rangle + |0^\perp\rangle, \tag{3.2.8}$$

where $|\phi\rangle$ is a normalized pure state and $(\langle 0| \otimes I)|0^\perp\rangle = 0$. If we measure the output state, we get 0 in the first register with probability p; by the Chernoff bound, it takes $\Theta(1/\epsilon^2)$ samples to estimate p within ϵ with high success probability. However, there is a more efficient quantum algorithm, called *amplitude estimation* [57], that estimates the value of p using only $O(1/\epsilon)$ calls to U:

Theorem 3.2.2 ([57, Theorem 12])**.** *Given U satisfying (7.2.1), the amplitude estimation algorithm in Figure 3.2 outputs an angle $\tilde{\theta}_p \in [-\pi, \pi]$ such that $\tilde{p} := \sin^2(\tilde{\theta}_p)$ satisfies*

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2} \tag{3.2.9}$$

with success probability at least $8/\pi^2$, using M calls to U and $U^\dagger$.

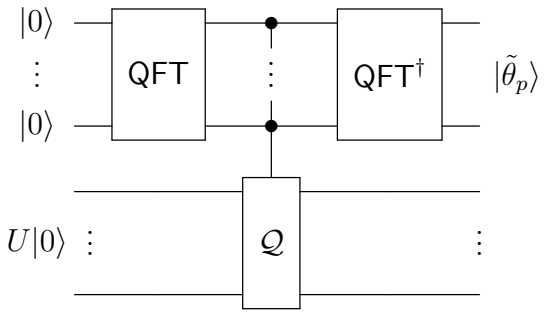

Figure 3.2: The quantum circuit for amplitude estimation.

Here QFT denotes the quantum Fourier transform over $\mathbb{Z}_M$ and $\mathcal{Q} := -US_0U^\dagger S_1$ where S_0 and S_1 are reflections about $|0\rangle$ and the target state, respectively; the controlled-$\mathcal{Q}$ gate denotes the operation $\sum_{j=0}^{M-1} |j\rangle\langle j| \otimes \mathcal{Q}^j$. In fact, it was shown in the proof of [57, Theorem 12] that the state after applying the circuit in Figure 3.2 is

$$\frac{e^{i\theta_p}}{\sqrt{2}}|\tilde{\theta}_p\rangle|0\rangle - \frac{e^{-i\theta_p}}{\sqrt{2}}|-\tilde{\theta}_p\rangle|0^\perp\rangle \tag{3.2.10}$$

where $\theta_p \in [0,\pi]$ such that $p = \sin^2(\theta_p)$, and $\tilde{\theta}_p \in [0,\pi]$ such that $\tilde{p} = \sin^2(\tilde{\theta}_p)$. Measuring the first register either gives $\tilde{\theta}_p$ or $-\tilde{\theta}_p$ with probability $1/2$, but since $\sin^2(\tilde{\theta}_p) = \sin^2(-\tilde{\theta}_p) = \tilde{p}$, this does not influence the success of Theorem 3.2.2.

In (7.2.2), if we take $M = \lceil 2\pi\left(\frac{2\sqrt{p}}{\epsilon} + \frac{1}{\sqrt{\epsilon}}\right)\rceil = O(1/\epsilon)$, we get

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{2\pi}\epsilon + \frac{\pi^2}{4\pi^2}\epsilon^2 \leq \frac{\epsilon}{2} + \frac{\epsilon}{4} \leq \epsilon. \tag{3.2.11}$$

Furthermore, the success probability $8/\pi^2$ can be boosted to $1-\nu$ by executing the algorithm $\Theta(\log 1/\nu)$ times and taking the median of the estimates.

Amplitude estimation can be generalized from estimating a single probability $p \in [0,1]$ to estimating the expectation of a random variable. Assume that U is a unitary acting on $\mathbb{C}^S \otimes \mathbb{C}^{|\Omega|}$ such that

$$U|0\rangle|0\rangle = \sum_{x\in\Omega} \sqrt{p_x}|\psi_x\rangle|x\rangle \tag{3.2.12}$$

where $S \in \mathbb{N}$ and $\{|\psi_x\rangle : x \in \Omega\}$ are unit vectors in $\mathbb{C}^S$. Let

$$\mu_U := \sum_{x\in\Omega} p_x x, \qquad \sigma_U^2 := \sum_{x\in\Omega} p_x (x-\mu_U)^2 \tag{3.2.13}$$

denote the expectation and variance of the random variable, respectively. Several quantum algorithms have given speedups for estimating μ_U. Specifically, Ref. [208] showed how to estimate μ_U within additive error ϵ by $\tilde{O}(\sigma_U/\epsilon)$ calls to U and $U^\dagger$. Given an upper bound H and a lower bound $L > 0$ on the random variable, Ref. [192] showed how to estimate μ_U with multiplicative error ϵ using $\tilde{O}(\sigma_U/\epsilon\mu_U \cdot H/L)$ calls to U and $U^\dagger$. More recently, Ref. [134] mutually generalized these results and proposed a significantly better quantum algorithm:

Theorem 3.2.3 ([134, Theorem 3.5])**.** *There is a quantum algorithm that, given a quantum sampler U as in (3.2.12), an integer Δ_U, a value $H > 0$, and two reals $\epsilon, \delta \in (0,1)$, outputs an estimate $\tilde{\mu}_U$. If $\Delta_U \geq \sqrt{\sigma_U^2 + \mu_U^2}/\mu_U$ and $H > \mu_U$, then $|\tilde{\mu}_U - \mu_U| \leq \epsilon\mu_U$ with probability at least $1-\delta$, and the algorithm uses*

$\tilde{O}(\Delta_U/\epsilon \cdot \log^3(H/\mu_U)\log(1/\delta))$ *calls to* U *and* $U^\dagger$.

The quantum algorithm works as follows. First, assume $\Omega \subseteq [L, H]$ for given real numbers $L, H \geq 0$, there is a basic estimation algorithm (denoted BasicEst) that estimates $H^{-1}\mu_U$ up to ϵ-multiplicative error:

Algorithm 3.1: BasicEst: the basic estimation algorithm.

Input: A quantum sampler U acting on $\mathbb{C}^S \otimes \mathbb{C}^{|\Omega|}$, interval $[L, H]$, precision parameter $\epsilon \in (0, 1)$, failure parameter $\delta \in (0, 1)$.

Output: ϵ-multiplicative approximation of $H^{-1}\mu_U$.

1 Use controlled rotation to implement a unitary $R_{L,H}$ acting on $\mathbb{C}^{|\Omega|} \otimes \mathbb{C}^2$ such that for all $x \in \Omega$,

$$R_{L,H}|x\rangle|0\rangle = \begin{cases} |x\rangle(\sqrt{1-\frac{x}{H}}|0\rangle + \sqrt{\frac{x}{H}}|1\rangle) & \text{if } L \leq x < H \\ |x\rangle|0\rangle & \text{otherwise} \end{cases};$$

2 Let $V = (I_S \otimes R_{L,H})(U \otimes I_2)$ and $\Pi = I_S \otimes I_\Omega \otimes |1\rangle\langle 1|$;

3 **for** $i = 1, \ldots, \Theta(\log(1/\delta))$ **do**

4 Compute $\tilde{p}_i$ by Theorem 3.2.2 with $U \leftarrow V$, $S_1 \leftarrow \Pi$, and $M \leftarrow \Theta(1/\epsilon\sqrt{H^{-1}\mu_U})$;

5 Return $\tilde{p} = \text{median}\{\tilde{p}_1, \ldots, \tilde{p}_{\Theta(\log(1/\delta))}\}$.

However, usually the bounds L and H are not explicitly given. In this case, Ref. [134] considered the truncated mean $\mu_{<b}$ defined by replacing the outcomes larger than b with 0. The paper then runs Algorithm 3.1 (BasicEst) to estimate $\mu_{<b}/b$. A crucial observation is that $\sqrt{b/\mu_{<b}}$ is smaller than Δ_U for large values of b, and it becomes larger than Δ_U when $b \approx \mu_U\Delta_U^2$. As a result, by repeatedly running BasicEst with Δ_U quantum samples, and applying $O(\log(H/L))$ steps of a binary search on the values of b, the first non-zero value is obtained when $b/\Delta_U^2 \approx \mu_U$. In [134], more precise truncation means are used to improve the precision of the result to $\tilde{O}(1/\epsilon)$ and remove the dependence on L.

Note that the quantum algorithm for Theorem 3.2.3 only relies on BasicEst.

This is crucial when we estimate the mean of our simulated annealing algorithm in different iterations *nondestructively* (see Section 3.4.2 for more details).

3.2.4 Hit-and-run walk

As introduced in Section 3.1.3, there are various of random walks that mix fast in a convex body K, such as the grid walk [103] and the ball walk [88, 198]. In this paper, we mainly use the *hit-and-run walk* [196, 200, 246]. It is defined as follows:

1. Pick a uniformly distributed random line ℓ through the current point p.
2. Move to a uniform random point along the chord $\ell \cap \mathrm{K}$.

For any two points $p, q \in \mathrm{K}$, we let $\ell(p, q)$ denote the length of the chord in K through p and q. Then the transition probability of the hit-and-run walk is determined by the following lemma:

Lemma 3.2.2 ([196, Lemma 3])**.** *If the current point of the hit-and-run walk is u, then the density function of the distribution of the next point $x \in \mathrm{K}$ is*

$$p_u(x) = \frac{2}{nv_n} \cdot \frac{1}{\ell(u,x)|x-u|^{n-1}}, \tag{3.2.14}$$

where $v_n := \pi^{\frac{n}{2}}/\Gamma(1+\frac{n}{2})$ is the volume of the n-dimensional unit ball. In other words, the probability that the next point is in a (measurable) set $\mathrm{A} \subseteq \mathrm{K}$ is

$$P_u(\mathrm{A}) = \int_{\mathrm{A}} \frac{2}{nv_n} \cdot \frac{1}{\ell(u,x)|x-u|^{n-1}} \,\mathrm{d}x. \tag{3.2.15}$$

In general, we can also define a hit-and-run walk with a given density. Let f be a density function in $\mathbb{R}^n$. For any point $u, v \in \mathbb{R}^n$, we let

$$\mu_f(u, v) := \int_0^1 f((1-t)u + tv)\,\mathrm{d}t. \tag{3.2.16}$$

For any line ℓ, let ℓ^+ and ℓ^- be the endpoints of the chord $\ell \cap \mathrm{K}$ (with $+$ and $-$ assigned arbitrarily). The density f specifies the following hit-and-run walk:

1. Pick a uniformly distributed random line ℓ through the current point p.
2. Move to a random point x along the chord $\ell \cap \mathrm{K}$ with density $\frac{f(x)}{\mu_f(\ell^-, \ell^+)}$.

Let π_{K} denote the uniform distribution over K. Reference [246] proves that the stationary distribution of the hit-and-run walk with uniform density is π_{K}. Furthermore, Ref. [200] proves that the hit-and-run walk mixes rapidly from any initial distribution:

Theorem 3.2.4 ([200, Theorem 1.1])**.** *Let* K *be a convex body that satisfies (3.1.2):* $B_2(0, r) \subseteq \mathrm{K} \subseteq B_2(0, R)$. *Let* σ *be a starting distribution and let* $\sigma^{(m)}$ *be the distribution of the current point after* m *steps of the hit-and-run walk in* K. *Let* $\epsilon > 0$, *and suppose that the density function* $\mathrm{d}\sigma/\mathrm{d}\pi_{\mathrm{K}}$ *is upper bounded by* M *except on a set* S *with* $\sigma(\mathrm{S}) \leq \epsilon/2$. *Then for any*

$$m > 10^{10} \frac{n^2 R^2}{r^2} \ln \frac{M}{\epsilon}, \tag{3.2.17}$$

the total variation distance between $\sigma^{(m)}$ *and* π_{K} *is less than* ϵ.

Theorem 3.2.4 can also be generalized to exponential distributions on K:

Theorem 3.2.5 ([200, Theorem 1.3]). *Let* $\mathrm{K} \subset \mathbb{R}^n$ *be a convex body and let* f *be a density supported on* K *that is proportional to* $e^{-a^T x}$ *for some vector* $a \in \mathbb{R}^n$. *Assume that the level set of* f *of probability* $1/8$ *contains a ball of radius* r, *and* $\mathbb{E}_f(|x - z_f|^2) \leq R^2$, *where* z_f *is the centroid of* f. *Let* σ *be a starting distribution and let* σ^m *be the distribution for the current point after* m *steps of the hit-and-run walk applied to* f. *Let* $\epsilon > 0$, *and suppose that the density function* $\frac{\mathrm{d}\sigma}{\mathrm{d}\pi_f}$ *is upper bounded by* M *except on a set* S *with* $\sigma(S) \leq \frac{\epsilon}{2}$. *Then for*

$$m > 10^{30} \frac{n^2 R^2}{r^2} \ln^5 \frac{MnR}{r\epsilon},$$

the total variation distance between σ^m *and* π_f *is less than* ϵ.

Roughly speaking, the proofs of Theorem 3.2.4 and Theorem 3.2.5 have two steps. First, for any random walk on a continuous domain Ω with transition probability p, stationary distribution π, and initial distribution σ, we define its conductance (which generalizes the discrete case in Eq. (3.2.1)) as

$$\Phi := \inf_{\mathrm{S} \subseteq \Omega} \frac{\int_{\mathrm{S}} \int_{\Omega/\mathrm{S}} \mathrm{d}x \, \mathrm{d}y \, \pi_x p_{x \to y}}{\min\{\int_{\mathrm{S}} \mathrm{d}x \, \pi_x, \int_{\Omega/\mathrm{S}} \mathrm{d}x \, \pi_x\}}. \tag{3.2.18}$$

It is well-known that the mixing time of this random walk is proportional to $1/\Phi^2$. This is captured by the following proposition:

Proposition 3.2.1 ([198, Corollary 1.5]). *Let* $M := \sup_{\mathrm{S} \subseteq \Omega} \frac{\sigma(\mathrm{S})}{\pi(\mathrm{S})}$. *Then for every*

$\mathrm{S} \subseteq \Omega$,

$$\left|\sigma^{(k)}(\mathrm{S}) - \pi(\mathrm{S})\right| \leq \sqrt{M}\left(1 - \frac{1}{2}\Phi^2\right)^k. \tag{3.2.19}$$

Furthermore, the conductance in Proposition 3.2.1 can be relaxed to that of sets with a fixed small probability p:

Proposition 3.2.2 ([198, Corollary 1.6]). *Let* $M := \sup_{\mathrm{S}\subseteq\Omega} \frac{\sigma(\mathrm{S})}{\pi(\mathrm{S})}$. *If the conductance for all* $\mathrm{A} \subseteq \Omega$ *such that* $\pi(\mathrm{A}) = p \leq 1/2$ *is at least* Φ_p, *then for all* $\mathrm{S} \subseteq \Omega$, *we have*

$$\left|\sigma^{(k)}(\mathrm{S}) - \pi(\mathrm{S})\right| \leq 2Mp + 2M\left(1 - \frac{1}{2}\Phi_p^2\right)^k. \tag{3.2.20}$$

Second, Ref. [200] proved a lower bound on the conductance of the hit-and-run walk with exponential density:

Proposition 3.2.3 ([200, Theorem 6.9]). *Let* f *be a density in* $\mathbb{R}^n$ *proportional to* $e^{-a^T x}$ *whose support is a convex body* K *of diameter* d. *Assume that* $\mathrm{B}_2(0, r) \subseteq \mathrm{K}$. *Then for any subset* S *with* $\pi_f(\mathrm{S}) = p \leq 1/2$, *the conductance of the hit-and-run walk satisfies*

$$\phi(S) \geq \frac{r}{10^{13} nd \ln(nd/rp)}. \tag{3.2.21}$$

Proposition 3.2.1 and Proposition 3.2.3 imply Theorem 3.2.4 and Theorem 3.2.5; complete proofs are given in [200].

For the conductance of the hit-and-run walk with a uniform distribution,

Ref. [200] established a stronger lower bound that is independent of p:

Proposition 3.2.4 ([200, Theorem 4.2])**.** *Assume that* K *has diameter* d *and contains a unit ball. Then the conductance of the hit-and-run in* K *with uniform distribution is at least* $\frac{1}{2^{24}nd}$.

3.3 Theory of continuous-space quantum walks

In this section, we develop the theory of continuous-space, discrete-time quantum walks. Specifically, we generalize the discrete-time quantum walk of Szegedy [254] to continuous space. Let $n \in \mathbb{N}$ and suppose Ω is a continuous[8] subset of $\mathbb{R}^n$. A probability transition density p on Ω is a continuous function $p\colon \Omega \times \Omega \to [0, +\infty)$ such that[9]

$$\int_\Omega \mathrm{d}y\, p(x, y) = 1 \qquad \forall\, x \in \Omega. \tag{3.3.1}$$

We also write $p_{x\to y} := p(x, y)$ for the transition density from x to y. Together, Ω and p specify a continuous-space Markov chain that we denote (Ω, p) throughout the paper.

For background on the mathematical foundations of quantum mechanics over continuous state spaces, see [241, Chapter 1]. In this section, we use $|x\rangle$ (for $x \in \mathbb{R}^n$)

[8] We say that Ω is continuous if for any $x, y \in \Omega$ there is a continuous function $f_{x,y}\colon [0, 1] \to \Omega$ such that $f_{x,y}(0) = x$ and $f_{x,y}(1) = y$.

[9] This setting covers real applications in theoretical computer science, including volume estimation. Because continuous functions on continuous sets are Riemann integrable, the integrals throughout the paper are simply the Riemann integrals.

to denote the computational basis; we have

$$\int_\Omega \mathrm{d}x\, |x\rangle\langle x| = I \quad \text{and} \quad \langle x|x'\rangle = \delta(x - x') \tag{3.3.2}$$

for all $x, x' \in \mathbb{R}^n$, where δ is the Dirac δ-function satisfying $\delta(0) = +\infty$, $\delta(x) = 0$ for all $x \neq 0$, and $\int_{\mathbb{R}^n} \delta(x)\, \mathrm{d}x = 1$.[10] The pure states in Ω correspond to

$$\mathrm{St}(\Omega) := \Big\{ f \colon \Omega \to \mathbb{R} \;\Big|\; \int_\Omega \mathrm{d}x\, |f(x)|^2 = 1 \Big\}. \tag{3.3.3}$$

In general, a function $f \colon \Omega \to \mathbb{R}$ is in $L^2(\Omega)$ if $\int_\Omega \mathrm{d}x\, |f(x)|^2 < \infty$. The inner product $\langle \cdot, \cdot \rangle$ on $L^2(\Omega)$ is defined by

$$\langle f, g \rangle := \int_\Omega \mathrm{d}x\, f(x) g(x) \qquad \forall\, f, g \in L^2(\Omega) \tag{3.3.4}$$

(note that by the Cauchy-Schwarz inequality, $|\langle f, g\rangle|^2 \leq (\int_\Omega \mathrm{d}x\, |f(x)|^2)(\int_\Omega \mathrm{d}x\, |g(x)|^2) < \infty$); the norm of an $f \in L^2(\Omega)$ is subsequently defined as $\|f\| := \sqrt{\langle f, f\rangle}$.

3.3.1 Continuous-space quantum walk

Given a transition density function p, define the following states:

$$|\phi_x\rangle := |x\rangle \otimes \int_\Omega \mathrm{d}y\, \sqrt{p_{x\to y}} |y\rangle \qquad \forall x \in \mathbb{R}^n. \tag{3.3.5}$$

[10] Note that the δ-function here is a generalized function, and rigorously it should be regarded as a (singular) Lebesgue measure (see the textbook [239]). Also note that in the discrete-space case, δ is the Kronecker δ-function defined as $\delta(x) = 0$ for all $x \neq 0$, and $\delta(0) = 1$; this is one main difference between the theory of continuous-space and discrete-space quantum walks.

These states characterize the quantum walk. Now, denote

$$U := \int_\Omega \mathrm{d}x\, |\phi_x\rangle(\langle x| \otimes \langle 0|), \Pi := \int_\Omega \mathrm{d}x\, |\phi_x\rangle\langle\phi_x|, S := \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, |x,y\rangle\langle y,x|. \quad (3.3.6)$$

Notice that Π is the projection onto $\mathrm{span}\{|\phi_x\rangle\}_{x\in\mathbb{R}^n}$ because

$$\Pi^2 = \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}x'\, |\phi_x\rangle\langle\phi_x|\phi_{x'}\rangle\langle\phi_{x'}| = \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}x'\, \delta(x-x')|\phi_x\rangle\langle\phi_{x'}| = \Pi, \quad (3.3.7)$$

and S is the swap operator for the two registers. A single step of the quantum walk is defined as the unitary operator

$$W := S(2\Pi - I). \quad (3.3.8)$$

The first main result of this subsection is the following theorem:

Theorem 3.3.1. *Let*

$$D := \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, \sqrt{p_{x\to y} p_{y\to x}} |x\rangle\langle y| \quad (3.3.9)$$

denote the discriminant operator *of p. Let Λ be the set of eigenvalues of D, so that $D = \int_\Lambda \mathrm{d}\lambda\, \lambda|\lambda\rangle\langle\lambda|$. Then the eigenvalues of the quantum walk operator W in (3.3.8) are ± 1 and $\lambda \pm i\sqrt{1-\lambda^2}$ for all $\lambda \in \Lambda$.*

To prove Theorem 3.3.1, we first prove the following lemma:

Lemma 3.3.1. *For any $\lambda \in \Lambda$, we have $|\lambda| \leq 1$.*

Proof. Since λ is an eigenvalue of D, we have $D|\lambda\rangle = \lambda|\lambda\rangle$. As a result, we have[11]

$$|\lambda|\delta(0) = |\lambda|\langle\lambda|\lambda\rangle = |\langle\lambda|D|\lambda\rangle| \tag{3.3.10}$$

$$= \Big|\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{p_{y\to x}p_{x\to y}}\langle\lambda|x\rangle\langle y|\lambda\rangle\Big| \tag{3.3.11}$$

$$\text{(by Cauchy-Schwarz)} \quad \le \sqrt{\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,p_{y\to x}|\langle y|\lambda\rangle|^2\Big)\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,p_{x\to y}|\langle\lambda|x\rangle|^2\Big)}$$

$$\text{(by } \int_\Omega \mathrm{d}y\,p_{x\to y} = 1) \quad \le \sqrt{\Big(\int_\Omega \mathrm{d}y\,|\langle y|\lambda\rangle|^2\Big)\Big(\int_\Omega \mathrm{d}x\,|\langle\lambda|x\rangle|^2\Big)} \tag{3.3.12}$$

$$= \int_\Omega \mathrm{d}x\,\langle\lambda|x\rangle\langle x|\lambda\rangle \tag{3.3.13}$$

$$\text{(by (3.3.2))} \quad = \langle\lambda|\Big(\int_\Omega \mathrm{d}x\,|x\rangle\langle x|\Big)|\lambda\rangle \tag{3.3.14}$$

$$= \delta(0). \tag{3.3.15}$$

Hence the result follows. □

Proof of Theorem 3.3.1. Define an isometry

$$T := \int_\Omega \mathrm{d}x\,|\phi_x\rangle\langle x| = \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{p_{x\to y}}|x,y\rangle\langle x|. \tag{3.3.16}$$

Then

$$TT^\dagger = \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,|\phi_x\rangle\langle x|y\rangle\langle\phi_y| = \int_\Omega \mathrm{d}x\,|\phi_x\rangle\langle\phi_x| = \Pi, \tag{3.3.17}$$

[11] This proof is not fully rigorous as $\delta(0)$ is ill-defined. However, δ can be regarded as the limit (in the sense of distributions) of the sequence of zero-centered normal distributions $\delta_\sigma(x) = \frac{1}{|\sigma|\sqrt{\pi}}e^{-(x/\sigma)^2}$ as $\sigma \to 0$. Then the LHS of (3.3.10) is replaced by $|\lambda|\cdot\frac{1}{|\sigma|\sqrt{\pi}}$ and the RHS of (3.3.15) is replaced by $\frac{1}{|\sigma|\sqrt{\pi}}$, so $|\lambda| \le 1$, and this also holds in the limit $\sigma \to 0$. For convenience we use similar arguments (regarding $\delta(0)$ as a positive real number) in this section, but keep in mind that rigorous proofs can be given by limit arguments.

and

$$T^\dagger T = \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, |x\rangle\langle\phi_x|\phi_y\rangle\langle y| \tag{3.3.18}$$

$$= \int_\Omega \int_\Omega \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}a\, \mathrm{d}b\, \langle x|y\rangle\langle a|b\rangle \sqrt{p_{x\to a} p_{y\to b}} |x\rangle\langle y| \tag{3.3.19}$$

$$= \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}a\, p_{x\to a} |x\rangle\langle x| \tag{3.3.20}$$

$$= \int_\Omega \mathrm{d}x\, |x\rangle\langle x| \tag{3.3.21}$$

$$= I. \tag{3.3.22}$$

Furthermore,

$$T^\dagger S T = \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, |x\rangle\langle\phi_x|S|\phi_y\rangle\langle y| \tag{3.3.23}$$

$$= \int_\Omega \int_\Omega \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, da\, \mathrm{d}b\, \langle x, a|S|y, b\rangle \sqrt{p_{x\to a} p_{y\to b}} |x\rangle\langle y| \tag{3.3.24}$$

$$= \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}a\, \sqrt{p_{x\to a} p_{a\to x}} |x\rangle\langle a| \tag{3.3.25}$$

$$= D. \tag{3.3.26}$$

As a result, for any $\lambda \in \Lambda$ we have

$$W T|\lambda\rangle = S(2\Pi - I)T|\lambda\rangle = (2STT^\dagger T - ST)|\lambda\rangle = ST|\lambda\rangle. \tag{3.3.27}$$

Similarly, we have

$$WST|\lambda\rangle = S(2\Pi - I)ST|\lambda\rangle = (2STT^\dagger ST - S^2 T)|\lambda\rangle = (2\lambda S - I)T|\lambda\rangle. \tag{3.3.28}$$

By Lemma 3.3.1, $|\lambda| \leq 1$. As a result, we have

$$\begin{aligned} W\big(I - (\lambda + i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle &= WT|\lambda\rangle - (\lambda + i\sqrt{1-\lambda^2})WST|\lambda\rangle \\ &= ST|\lambda\rangle - (\lambda + i\sqrt{1-\lambda^2})(2\lambda S - I)T|\lambda\rangle \\ &= \big(S - (\lambda + i\sqrt{1-\lambda^2})(2\lambda S - I)\big)T|\lambda\rangle \\ &= (\lambda + i\sqrt{1-\lambda^2})\big(I - (\lambda + i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle; \end{aligned}$$

in other words, $\lambda + i\sqrt{1-\lambda^2}$ is an eigenvalue of W with eigenvector $\big(I - (\lambda + i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle$. Similarly, we have

$$W\big(I - (\lambda - i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle = (\lambda - i\sqrt{1-\lambda^2})\big(I - (\lambda - i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle,$$

i.e., $\lambda - i\sqrt{1-\lambda^2}$ is an eigenvalue of W with eigenvector $\big(I - (\lambda - i\sqrt{1-\lambda^2})S\big)T|\lambda\rangle$.

Finally, for any vector $|u\rangle$ in the orthogonal complement of the space $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle, ST|\lambda\rangle\}$, W simply acts as $-S$ because

$$\Pi = TT^\dagger = \int_\Lambda \mathrm{d}\lambda\, T|\lambda\rangle\langle\lambda|T^\dagger, \tag{3.3.29}$$

which projects onto $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle\}$. In this orthogonal complement subspace, the eigenvalues are ± 1 because $S^2 = I$. □

3.3.2 Stationary distribution

Classically, the density $\pi = (\pi_x)_{x\in\Omega}$ corresponding to the stationary distribution of a Markov chain (Ω, p) satisfies

$$\int_\Omega \mathrm{d}x\, \pi_x = 1; \qquad \int_\Omega \mathrm{d}y\, p_{y\to x}\pi_y = \pi_x \qquad \forall\, x \in \Omega. \tag{3.3.30}$$

In other words, we can naturally define a transition operator as

$$P := \int_\Omega \int_\Omega \mathrm{d}x\, \mathrm{d}y\, p_{y\to x} |x\rangle\langle y|, \tag{3.3.31}$$

and the stationary density π satisfies $P\pi = \pi$. The Markov chain (Ω, p) is *reversible* if there exists a classical density $\sigma = (\sigma_x)_{x\in\Omega}$ such that

$$p_{y\to x}\sigma_y = p_{x\to y}\sigma_x \qquad \forall\, x, y \in \Omega. \tag{3.3.32}$$

(This is called the *detailed balance condition.*) Notice that for all $x \in \Omega$,

$$\int_\Omega \mathrm{d}y\, p_{y\to x}\sigma_y = \int_\Omega \mathrm{d}y\, p_{x\to y}\sigma_x = \sigma_x \int_\Omega \mathrm{d}y\, p_{x\to y} = \sigma_x; \tag{3.3.33}$$

therefore, we must have $P\sigma = \sigma$, i.e., σ is a stationary density of P. In this paper, we focus on Markov chains (Ω, p) that are reversible and have a unique stationary distribution (i.e., $\sigma = \pi$). Such assumptions are natural for Markov chains in practice, including the Metropolis-Hastings algorithm, simple random walks on

graphs, etc.

If π is the classical stationary density of a reversible Markov chain (Ω, p), then

$$|\pi_W\rangle := \int_\Omega \mathrm{d}x \, \sqrt{\pi_x}|\phi_x\rangle \tag{3.3.34}$$

is the unique eigenvalue-1 eigenstate of the quantum walk operator W restricted to the subspace $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle, ST|\lambda\rangle\}$. First, a simple calculation shows that

$$\begin{aligned}
W|\pi_W\rangle &= S(2\Pi - I)|\pi_W\rangle && (3.3.35)\\
&= S|\pi_W\rangle && (3.3.36)\\
&= \Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,|x,y\rangle\langle y,x|\Big)\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{\pi_y p_{y\to x}}|y,x\rangle\Big) && (3.3.37)\\
&= \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{\pi_y p_{y\to x}}|x,y\rangle && (3.3.38)\\
&= \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\,\sqrt{\pi_x p_{x\to y}}|x\rangle|y\rangle && (3.3.39)\\
&= \int_\Omega \mathrm{d}x\,\sqrt{\pi_x}|x\rangle\Big(\int_\Omega \mathrm{d}y\,\sqrt{p_{x\to y}}|y\rangle\Big) = \int_\Omega \mathrm{d}x\,\sqrt{\pi_x}|\phi_x\rangle = |\pi_W\rangle, && (3.3.40)
\end{aligned}$$

where (3.3.36) follows from $|\pi_W\rangle \in \mathrm{span}_{x\in\Omega}\{|\phi_x\rangle\}$, (3.3.37) follows from the definition of S in (3.3.6), (3.3.39) follows from (3.3.32), and (3.3.39) follows from the definition of $|\phi_x\rangle$ in (3.3.5). Thus $|\pi_W\rangle$ is an eigenvector of W with eigenvalue 1. On the other hand, since (Ω, p) is reversible, P is similar to D: if we denote

$D_\pi := \int_\Omega \mathrm{d}x \sqrt{\pi_x}|x\rangle\langle x|$, then

$$
\begin{aligned}
D_\pi D D_\pi^{-1} &= \Big(\int_\Omega \mathrm{d}x \sqrt{\pi_x}|x\rangle\langle x|\Big)\Big(\int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y \sqrt{p_{x\to y}p_{y\to x}}|x\rangle\langle y|\Big)\Big(\int_\Omega \mathrm{d}y \sqrt{\pi_y^{-1}}|y\rangle\langle y|\Big) \\
&= \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y \sqrt{\pi_x\pi_y^{-1}p_{x\to y}p_{y\to x}}|x\rangle\langle y| \qquad (3.3.41) \\
&= \int_\Omega\int_\Omega \mathrm{d}x\,\mathrm{d}y\, p_{y\to x}|x\rangle\langle y| \qquad (3.3.42) \\
&= P, \qquad (3.3.43)
\end{aligned}
$$

where (3.3.41) follows from (3.3.32). As a result, D and P have the same set of eigenvalues. Furthermore, Lemma 3.3.1 implies that all eigenvalues of P have norm at most 1, and the proof of Theorem 3.3.1 shows that $|\pi_W\rangle$ is the unique eigenvector with this eigenvalue within $\mathrm{span}_{\lambda\in\Lambda}\{T|\lambda\rangle, ST|\lambda\rangle\}$.

The state

$$
|\pi\rangle := \int_\Omega \mathrm{d}x \sqrt{\pi_x}|x\rangle \qquad (3.3.44)
$$

represents a quantum sample from the density π; in particular, measuring $|\pi\rangle$ in the computational basis gives a classical sample from π. Furthermore, the unitary operator in (3.3.6) satisfies

$$
U^\dagger|\pi_W\rangle = \Big(\int_\Omega \mathrm{d}x\,|x\rangle|0\rangle\langle\phi_x|\Big)\Big(\int_\Omega \mathrm{d}x \sqrt{\pi_x}|\phi_x\rangle\Big) = |\pi\rangle|0\rangle, \qquad (3.3.45)
$$

so we have $U|\pi\rangle|0\rangle = |\pi_W\rangle$.

3.4 Quantum speedup for volume estimation

We now give and analyze our quantum volume estimation algorithm. First, we review the classical state-of-the-art volume estimation algorithm in Section 3.4.1. We then describe our quantum algorithm for estimating the volume of well-rounded convex bodies (i.e., $R/r = O(\sqrt{n})$) with query complexity $\tilde{O}(n^{2.5}/\epsilon)$ in Section 3.4.2, with detailed proofs given in Section 3.4.3. Finally, we remove the well-rounded condition by giving a quantum algorithm with interlaced rounding and volume estimation with additional cost $\tilde{O}(n^{2.5})$ in each iteration in Section 3.4.4.

3.4.1 Review of classical algorithms for volume estimation

The best-known classical volume estimation algorithm uses $\tilde{O}(n^4 + n^3/\epsilon^2)$ queries, where $\tilde{O}(n^4)$ queries are used to construct the affine transformation that makes convex body well-rounded [201] and $\tilde{O}(n^3/\epsilon^2)$ queries are used to estimate the volume of the well-rounded convex body (after the affine transformation) [88].

We review the algorithm of [201] for estimating volumes of well-rounded convex bodies. This algorithm estimates the volume of a convex body obtained by the following *pencil construction*. Define the cone

$$\mathrm{C} := \left\{ x \in \mathbb{R}^{n+1} : x_0 \geq 0, \sum_{i=1}^{n} x_i^2 \leq x_0^2 \right\}. \tag{3.4.1}$$

Let K' be the intersection of the cone C and a cylinder $[0, 2D] \times \mathrm{K}$, i.e.,

$$\mathrm{K}' := ([0, 2D] \times \mathrm{K}) \cap \mathrm{C} \tag{3.4.2}$$

(recall $D = R/r$). Without loss of generality we renormalize to $r = 1$, so that $\mathrm{B}_2(0, 1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, D)$. Since $D \operatorname{Vol}(\mathrm{K}) \leq \operatorname{Vol}(\mathrm{K}') \leq 2D \operatorname{Vol}(\mathrm{K})$, we can estimate $\operatorname{Vol}(\mathrm{K})$ with multiplicative error ϵ by generating $O(1/\epsilon^2)$ sample points from the uniform distribution on $[0, 2D] \times \mathrm{K}$ and then counting how many of them fall into K'. Such an approximation succeeds with high probability by a Chernoff-type argument (see Section 3.4.3.1 for a formal proof).

Reference [201] considers simulated annealing under the pencil construction. For any $a > 0$, define

$$Z(a) := \int_{\mathrm{K}'} e^{-ax_0} \, \mathrm{d}x. \tag{3.4.3}$$

It can be shown that for any $a \leq \epsilon/D$,

$$(1 - \epsilon) \operatorname{Vol}(\mathrm{K}') \leq Z(a) \leq \operatorname{Vol}(\mathrm{K}'). \tag{3.4.4}$$

On the other hand, for any $a \geq 2n$,

$$(1 - \epsilon) \int_{\mathrm{C}} e^{-ax_0} \, \mathrm{d}x \leq Z(a) \leq \int_{\mathrm{C}} e^{-ax_0} \, \mathrm{d}x. \tag{3.4.5}$$

This suggests using a simulated annealing procedure for estimating $\operatorname{Vol}(\mathrm{K}')$. Specif-

ically, if we select a sequence $a_0 > a_1 > \cdots > a_m$ for which $a_0 = 2n$ and $a_m \leq \epsilon/D$, then we can estimate Vol(K′) by

$$Z(a_m) = Z(a_0) \prod_{i=0}^{m-1} \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.4.6}$$

(Note that this procedure uses an increasing sequence of temperatures $1/a_i$, unlike standard simulated annealing in which temperature is decreased.)

Let π_i be the probability distribution over K′ with density proportional to $e^{-a_i x_0}$, i.e., $\mathrm{d}\pi_i(x) = \frac{e^{-a_i x_0}}{Z(a_i)}\mathrm{d}x$. Let X_i be a random sample from π_i, and let $(X_i)_0$ be its first coordinate; define $V_i := e^{(a_i - a_{i+1})(X_i)_0}$. We have

$$\mathbb{E}_{\pi_i}[V_i] = \int_{\mathrm{K}'} e^{(a_i - a_{i+1})x_0}\, \mathrm{d}\pi_i(x) = \int_{\mathrm{K}'} e^{(a_i - a_{i+1})x_0} \frac{e^{-a_i x_0}}{Z(a_i)}\, \mathrm{d}x = \frac{Z(a_{i+1})}{Z(a_i)}. \tag{3.4.7}$$

Furthermore, if the simulated annealing schedule satisfies $a_{i+1} \geq (1 - \frac{1}{\sqrt{n}})a_i$, then V_i satisfies (see [201, Lemma 4.1])

$$\frac{\mathbb{E}_{\pi_i}[V_i^2]}{\mathbb{E}_{\pi_i}[V_i]^2} \leq \left(\frac{a_{i+1}^2}{a_i(2a_{i+1} - a_i)}\right)^{n+1} < 8 \qquad \forall\, i \in [m], \tag{3.4.8}$$

i.e., the variance of V_i is bounded by a constant multiple of the square of its expectation. Thus, this simulated annealing procedure constitutes *Chebyshev cooling* (see also Section 3.4.3.3), ensuring its correctness (see Proposition 3.4.1). Details are given in Algorithm 3.2.

[12] Sampling from π_0 in Line 2 can be achieved by selecting a random positive real number X_0 from the distribution with density e^{-2nx} and choose a uniformly random point $(V_1, \ldots, V_n)$ from the unit ball. If $X = (X_0, X_0V_1, \ldots, X_0V_n) \notin \mathrm{K}'$, try again; else return X. Equation (3.4.5) ensures that we succeed with probability at least $1 - \epsilon$ for each sample.

Algorithm 3.2: Volume estimation with $\tilde{O}(n^4/\epsilon^2)$ classical queries [201].

Input: Membership oracle O_{K} of K; R such that $\mathrm{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0,R)$; $R = O(\sqrt{n})$, i.e., K is well-rounded.

Output: ϵ-multiplicative approximation of Vol(K).

Set $m = 2\lceil \sqrt{n}\ln(n/\epsilon) \rceil$, $k = \frac{512}{\epsilon^2}\sqrt{n}\ln(n/\epsilon)$, $\delta = \epsilon^2 n^{-10}$, and $a_i = 2n(1 - \frac{1}{\sqrt{n}})^i$ for $i \in [m]$;

Take k samples $X_0^{(1)}, \ldots, X_0^{(k)}$ from π_0[12];

for $i \in [m]$ **do**

  Take k samples from π_i with error parameter δ and starting points $X_{i-1}^{(1)}, \ldots, X_{i-1}^{(k)}$, giving points $X_i^{(1)}, \ldots, X_i^{(k)}$;

  Compute $V_i = \frac{1}{k}\sum_{j=1}^{k} e^{(a_i - a_{i+1})(X_i^{(j)})_0}$;

Return $n! v_n (2n)^{-(n+1)} V_1 \cdots V_m$ as the estimate of the volume of K′, where $v_n := \pi^{\frac{n}{2}}/\Gamma(1 + \frac{n}{2})$ is the volume of the n-dimensional unit ball;

3.4.2 Quantum algorithm for volume estimation

As introduced in Section 3.1.2, our quantum algorithm has four main improvements that contribute to the quantum speedup of Algorithm 3.2:

1. We replace the classical hit-and-run walk in Section 3.2.4 by a quantum hit-and-run walk, defined using the framework of Section 3.3. Classically, the hit-and-run walk mixes in $\tilde{O}(n^3)$ steps in a well-rounded convex body given a warm start (see Theorem 3.2.4). Quantumly, we can use the quantum hit-and-run walk operator to prepare its stationary state given a warm start state using only $\tilde{O}(n^{1.5})$ queries to the membership oracle fora the well-rounded convex body.

2. We replace the simulated annealing framework in Algorithm 3.2 by the quantum MCMC framework described in Section 3.2.2. Classically, we sample from π_i in the i^{th} iteration by running the classical hit-and-run walk starting from the samples taken in the $(i-1)^{\text{st}}$ iteration. Quantumly, we prepare the quantum

sample $|\pi_i\rangle$ in the i^{th} iteration by applying $\pi/3$-amplitude amplification to a quantum sample produced in the $(i-1)^{\text{st}}$ iteration, where the unitaries in the $\pi/3$-amplitude amplification are implemented by phase estimation of the quantum hit-and-run walk operators as in (3.2.7).

3. We use the quantum Chebyshev inequality (see Section 3.2.3) to give a quadratic quantum speedup in ϵ^{-1} when taking the average $e^{(a_i-a_{i+1})(\bar{X}_i)_0}$ in Line 5 of Algorithm 3.2. However, we must be cautious because the resulting points $X_i^{(1)}, \ldots, X_i^{(k)}$ in Line 4 follow the distribution π_i, which varies in different iterations of simulated annealing. Instead, our quantum algorithm must be *nondestructive*: it must still have a copy of $|\pi_i\rangle$ after estimating the average $e^{(a_i-a_{i+1})(\bar{X}_i)_0}$, so that we can map this state to $|\pi_{i+1}\rangle$ by $\pi/3$-amplitude amplification for the next iteration. This is achieved in Section 3.4.3.3.

4. In Section 3.4.4, we show how the densities can be transformed to be well-rounded by an affine transformation at each stage of the algorithm. This is to ensure that the hit-and-run walk mixes fast assuming the densities π_i to be sampled from are well-rounded (see Theorem 3.2.5). The high-level idea is to sample points from density π_i and compute an affine transformation S_{i+1} that rounds π_i and the next density π_{i+1} (see Lemma 3.4.11). To sample these points, we use $\pi/3$-amplitude amplification to map the states corresponding to the uniform distributions for one stage to those for the next. The affine transformation can be computed coherently using nondestructive mean estimation, with $\tilde{O}(n^{2.5})$ quantum queries in each iteration.

Algorithm 3.3 is our quantum volume estimation algorithm that satisfies our main theorem:

Theorem 3.1.1 (Main Theorem)**.** *Let* $\mathrm{K} \subset \mathbb{R}^n$ *be a convex set with* $\mathrm{B}_2(0, r) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0, R)$*. Assume* $0 < \epsilon < 1/2$*. Then there is a quantum algorithm that returns a value* $\widetilde{\mathrm{Vol}(\mathrm{K})}$ *satisfying*

$$\frac{1}{1+\epsilon}\,\mathrm{Vol}(\mathrm{K}) \leq \widetilde{\mathrm{Vol}(\mathrm{K})} \leq (1+\epsilon)\,\mathrm{Vol}(\mathrm{K}) \tag{3.1.4}$$

using $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ *quantum queries to the membership oracle* O_{K} *(defined in (3.1.3)) and* $\tilde{O}(n^5 + n^{4.5}/\epsilon)$ *additional arithmetic operations.*[13]

More generally, our framework could be used to provide quantum speedup for any classical simulated annealing algorithm based on Chebyshev cooling, which might be of independent interest.

The proof of Theorem 3.1.1 is organized as follows. We first assume that in each iteration, S_{i+1} puts π_{i+1} in isotropic position, i.e., the densities are promised to be well-rounded. The rest of this subsection presents an overview of the proof of Theorem 3.1.1 (including a quantum circuit in Figure 3.3), and proofs details are given in Section 3.4.3. In Section 3.4.4, we show how the well-roundedness be maintained at an additional cost of $\tilde{O}(n^{2.5})$ quantum queries in each iteration.

Following the discussion in Section 3.1.2, our proof has three levels:

[13] Arithmetic operations (e.g., addition, subtraction, multiplication, and division) can be in principle implemented by a universal set of quantum gates using the Solovay-Kitaev Theorem [95] up to a small overhead. In our quantum algorithm, the number of arithmetic operations is dominated by n-dimensional matrix-vector products computed in superposition for rounding the convex body (see Section 3.4.4).

Algorithm 3.3: Volume estimation with $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ quantum queries.

Input: Membership oracle O_{K} for K; $R = O(\sqrt{n})$ s.t. $\mathrm{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0,R)$.
Output: ϵ-multiplicative approximation of Vol(K).

Set $m = \Theta(\sqrt{n}\log(n/\epsilon))$ to be the number of iterations of simulated annealing and $a_i = 2n(1 - \frac{1}{\sqrt{n}})^i$ for $i \in [m]$. Let π_i be the probability distribution over K' with density proportional to $e^{-a_i x_0}$;

Set error parameters $\delta, \epsilon' = \Theta(\epsilon/m^2), \epsilon_1 = \epsilon/2m$; let $k = \tilde{\Theta}(\sqrt{n}/\epsilon)$ be the number of copies of stationary states in the quantum Chebyshev inequality; let $l = \tilde{\Theta}(n)$ be the number of copies of stationary states needed to obtain the affine transformation S_i;

Prepare $k + l$ (approximate) copies of $|\pi_0\rangle$, denoted $|\tilde{\pi}_0^{(1)}\rangle, \ldots, |\tilde{\pi}_0^{(k+l)}\rangle$;

for $i \in [m]$ **do**

  Use the quantum Chebyshev inequality on the k copies of the state $|\tilde{\pi}_{i-1}\rangle$ with parameters ϵ_1, δ to estimate the expectation $\mathbb{E}_{\pi_i}[V_i]$ (in Eq. (3.4.7)) as $\tilde{V}_i$ (Lemma 3.4.9 and Figure 3.4). The post-measurement states are denoted $|\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k)}\rangle$;

  Use the l copies of the state $|\pi_{i-1}\rangle$ to nondestructively obtain the affine transformation S_i that rounds π_{i-1} and π_i (Section 3.4.4). The post-measurement states are denoted $|\hat{\pi}_{i-1}^{(k+1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k+l)}\rangle$;

  Apply $\pi/3$-amplitude amplification with error ϵ' (Section 3.2.2 and Lemma 3.4.8) and affine transformation S_i to map $|S_i\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |S_i\hat{\pi}_{i-1}^{(k+l)}\rangle$ to $|S_i\tilde{\pi}_i^{(1)}\rangle, \ldots, |S_i\tilde{\pi}_i^{(k+l)}\rangle$, using the quantum hit-and-run walk ;

  Invert S_i to get $k + l$ (approximate) copies of the stationary distribution $|\pi_i\rangle$ for use in the next iteration;

Compute an estimate $\widetilde{\mathrm{Vol}(\mathrm{K}')} = n! v_n (2n)^{-(n+1)} \tilde{V}_1 \cdots \tilde{V}_m$ of the volume of K', where v_n is the volume of the n-dimensional unit ball;

Use $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ to estimate the volume of K as $\widetilde{\mathrm{Vol}(\mathrm{K})}$ (Section 3.4.3.1).

High level (the simulated annealing framework). In Section 3.4.3.1, we show how to estimate Vol(K) given an estimate of the volume of the pencil construction, Vol(K′):

Lemma 3.4.1. *If we have access to* $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ *such that*

$$\frac{1}{1+\epsilon/2}\,\mathrm{Vol}(\mathrm{K}') \le \widetilde{\mathrm{Vol}(\mathrm{K}')} \le (1+\epsilon/2)\,\mathrm{Vol}(\mathrm{K}') \tag{3.4.9}$$

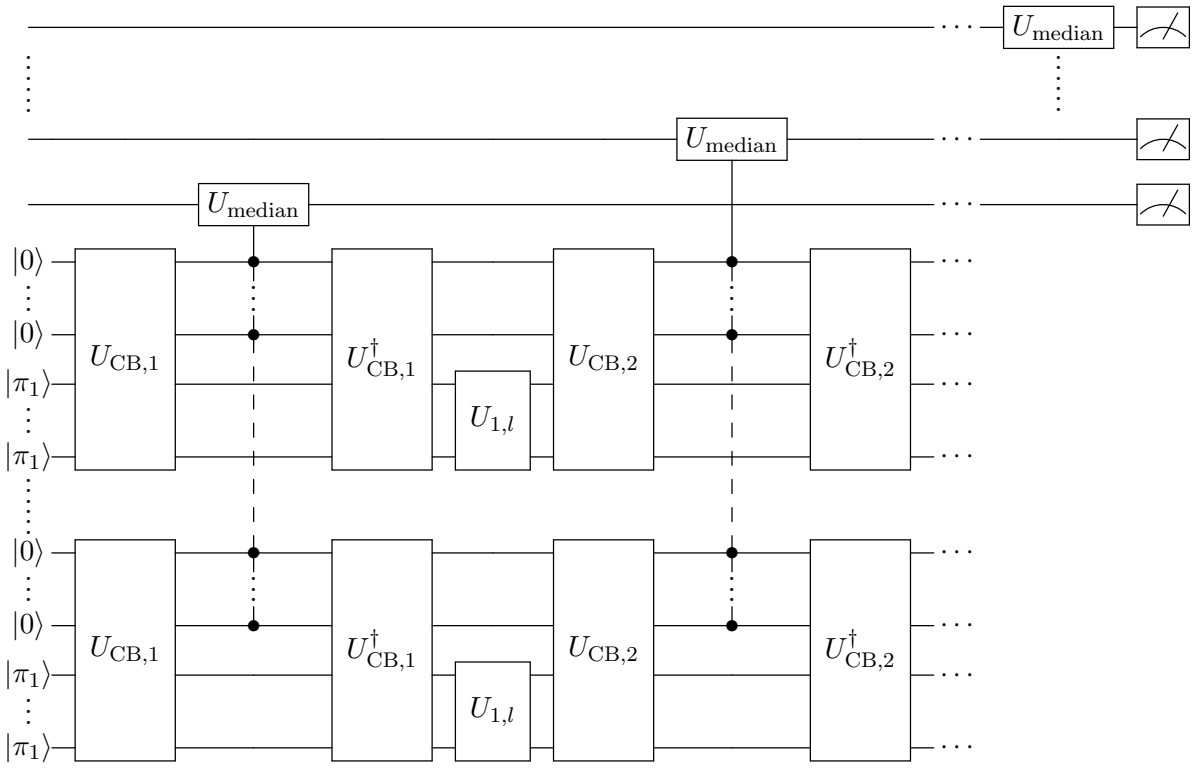

Figure 3.3: The quantum circuit for Algorithm 3.3 (assuming well-roundedness). Here $U_{\text{CB},i}$ is the circuit of the quantum Chebyshev inequality (Theorem 3.2.3) in the i^{th} iteration and $U_{i,l}$ is $\pi/3$-amplitude amplification from $|\pi_i\rangle$ to $|\pi_{i+1}\rangle$.

with probability at least 0.7, then we can return a value $\widetilde{\text{Vol}(\text{K})}$ *such that*

$$\frac{1}{1+\epsilon}\,\text{Vol}(\text{K}) \leq \widetilde{\text{Vol}(\text{K})} \leq (1+\epsilon)\,\text{Vol}(\text{K}) \tag{3.4.10}$$

holds with probability at least $2/3$*, using* $\tilde{O}(n^{2.5}/\epsilon)$ *quantum queries to the membership oracle* O_{K}*.*

In Section 3.4.3.2, we prove that the inner product between stationary states of consecutive simulated annealing steps is at least a constant:

Lemma 3.4.2. *Let* $|\pi_i\rangle$ *be the stationary distribution state of the quantum walk* W_i

for $i \in [m]$ *defined in (3.3.44). For* $n \geq 2$*, we have* $\langle \pi_i | \pi_{i+1} \rangle > 1/3$ *for* $i \in [m-1]$.

This allows $\pi/3$-amplitude amplification to transform the stationary state of one Markov chain to that of the next. The total number of iterations of $\pi/3$-amplitude amplification is thus $\tilde{O}(\sqrt{n})$, just as in the classical volume estimation algorithm of [201].

Middle level (each telescoping ratio). In Section 3.4.3.3, we describe how we apply the quantum Chebyshev inequality (Theorem 3.2.3) to the Chebyshev cooling schedule.

Lemma 3.4.3. *Given* $\tilde{O}(\log(1/\delta)/\epsilon)$ *copies of the quantum states* $|\pi_{i-1}\rangle$*, there exists a quantum algorithm that outputs an estimate of* $\mathbb{E}_{\pi_i}[V_i]$ *(in Eq. (3.4.7)) with relative error less than* ϵ *with probability at least* $1 - O(\delta)$.

Furthermore, we show how to make Lemma 3.4.3 nondestructive on the stationary states. Because the relative error for estimating the volume via Chebyshev cooling is $\Theta(\epsilon/m) = \tilde{\Theta}(\epsilon/\sqrt{n})$, Lemma 3.4.3 implies that $O(\log(1/\delta)/(\epsilon/\sqrt{n})) = \tilde{O}(\sqrt{n}/\epsilon)$ copies of the stationary state suffice for the simulated annealing framework.

Low level (the quantum hit-and-run walk). In Section 3.4.3.4, we give a careful analysis of the errors coming from the quantum Chebyshev inequality as well as the $\pi/3$-amplitude amplification:

Lemma 3.4.4. *Given* $\tilde{O}(\log(1/\delta)/\epsilon_1)$ *copies of a state* $|\tilde{\pi}_{i-1}\rangle$ *such that* $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq \epsilon_1$*, there exists a quantum procedure (using* $\pi/3$*-amplitude amplification*

and the quantum Chebyshev inequality) that outputs a $\tilde{V}_i$ *such that* $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq \epsilon_1 \mathbb{E}_{\pi_i}[V_i]$ *(where* $\mathbb{E}_{\pi_i}[V_i]$ *is defined in Eq. (3.4.7)) with success probability* $1-\delta^4$ *using* $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon_1 + n^{3/2}\log(1/\epsilon'))$ *calls to the membership oracle and returns a final state* $|\tilde{\pi}_i\rangle$ *such that* $\||\tilde{\pi}_i\rangle - |\pi_i\rangle\| = O(\epsilon_1 + \delta + \epsilon')$.

Having the four lemmas above from all the three levels, we establish Theorem 3.1.1 as follows.

Proof of Theorem 3.1.1. We prove the correctness and analyze the cost separately.

Correctness. By Lemma 3.4.1, it suffices to compute the volume of the pencil construction K′ to relative error $\epsilon/2$ with probability at least 0.7 in order to compute the volume of the well-rounded convex body K. This is computed as a telescoping sum of $m = O(\sqrt{n}\log n/\epsilon)$ products of the form $Z(a_{i+1})/Z(a_i)$. The random variable V_i is an unbiased estimator for $Z(a_{i+1})/Z(a_i)$, i.e., $\mathbb{E}_{\pi_i}[V_i] = Z(a_{i+1})/Z(a_i)$. Consider applying Lemma 3.4.4 m times with $\delta, \epsilon' = \Theta(\epsilon/8m^2)$ and $\epsilon_1 = \epsilon/2m$. At each iteration i we have a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq O(\epsilon/4m)$. Thus each telescoping sum can be computed with a relative error of $\epsilon/2m$, resulting in a relative error of less than $\epsilon/2$ for the final volume. The probability of success for each iteration is at least $1 - \delta^4 = 1 - \Theta(\epsilon^4/4m^8)$. Thus the probability of success for the whole algorithm is at least $1 - \Theta(\epsilon^4/4m^7) = 1 - \tilde{O}(\epsilon^{11}/n^{3.5})$, which is greater than 0.7 for a large enough n.

Cost. Ignoring the cost of obtaining the affine transformation to round the log-concave densities to be sampled (assuming that all the relevant densities are well

rounded), we have from Lemma 3.4.4, the number of calls to the membership oracle in each iteration is $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon_1 + n^{3/2}\log(1/\epsilon')) = \tilde{O}(n^2/\epsilon)$. The total number of oracle calls is thus $\tilde{O}(n^{2.5}/\epsilon)$. The argument for correctness above applies for well-rounded logconcave densities. This is ensured by maintaining $\tilde{\Theta}(n)$ states that are used to round the densities in each iteration (Algorithm 3.4). By Proposition 3.4.5, this procedure uses $\tilde{O}(n^{2.5})$ calls to the membership oracle in each iteration, resulting in a final query complexity of $\tilde{O}(n^3 + n^{2.5}/\epsilon)$. Since the affine transformation is an n-dimensional matrix-vector product, the number of additional arithmetic operations is hence $O(n^2) \cdot \tilde{O}(n^3 + n^{2.5}/\epsilon) = \tilde{O}(n^5 + n^{4.5}/\epsilon)$. □

3.4.3 Proof details of the quantum volume estimation algorithm

We now prove the lemmas in Section 3.4.2 that establish Theorem 3.1.1.

3.4.3.1 Pencil construction and the original convex body

Here we prove

Lemma 3.4.1. *If we have access to* $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ *such that*

$$\frac{1}{1+\epsilon/2}\,\mathrm{Vol}(\mathrm{K}') \le \widetilde{\mathrm{Vol}(\mathrm{K}')} \le (1+\epsilon/2)\,\mathrm{Vol}(\mathrm{K}') \tag{3.4.9}$$

with probability at least 0.7, then we can return a value $\widetilde{\mathrm{Vol}(\mathrm{K})}$ *such that*

$$\frac{1}{1+\epsilon}\,\mathrm{Vol}(\mathrm{K}) \le \widetilde{\mathrm{Vol}(\mathrm{K})} \le (1+\epsilon)\,\mathrm{Vol}(\mathrm{K}) \tag{3.4.10}$$

holds with probability at least $2/3$*, using* $\tilde{O}(n^{2.5}/\epsilon)$ *quantum queries to the membership oracle* O_{K}*.*

Proof. We follow the same notation in Section 3.4.1, i.e., without loss of generality we assume that $r = 1$ and denote $D = R/r = R$. In other words, the pencil construction is

$$\mathrm{K}' := ([0, 2D] \times \mathrm{K}) \cap \left\{ x \in \mathbb{R}^{n+1} : x_0 \geq 0, \sum_{i=1}^{n} x_i^2 \leq x_0^2 \right\}. \tag{3.4.11}$$

By the definition of D, for any $(x_1, \ldots, x_n) \in \mathrm{K}$ we have $\sum_{i=1}^{n} x_i^2 \leq D^2$, so $[D, 2D] \times \mathrm{K} \subseteq \mathrm{K}'$. This implies that $D \operatorname{Vol}(\mathrm{K}) \leq \operatorname{Vol}(\mathrm{K}') \leq 2D \operatorname{Vol}(\mathrm{K})$. In other words, letting $\xi_{\mathrm{K}} := \frac{\operatorname{Vol}(\mathrm{K}')}{2D \operatorname{Vol}(\mathrm{K})}$, we have $0.5 \leq \xi_{\mathrm{K}} \leq 1$.

Classically, we consider a Monte Carlo approach to approximating $\operatorname{Vol}(\mathrm{K})$: we take k (approximately) uniform samples $x_1, \ldots, x_k$ from $[0, 2D] \times \mathrm{K}$, and if k' of them are in K', we return $\frac{k'}{k} \cdot \widetilde{\operatorname{Vol}(\mathrm{K}')}$. For each $i \in [k]$, $\delta[x_i \in \mathrm{K}']$ is a boolean random variable with expectation $\xi_{\mathrm{K}} = \Theta(1)$. Any boolean random variable has variance $O(1)$. Therefore, by Chebyshev's inequality, taking $k = O(1/\epsilon^2)$ suffices to ensure that

$$\Pr\left[\left|\frac{k'}{k} - \xi_{\mathrm{K}}\right| \leq \frac{\epsilon \xi_{\mathrm{K}}}{2}\right] \geq 0.99. \tag{3.4.12}$$

Quantumly, we adopt the same Monte Carlo approach but we implement two steps using quantum techniques:

- We take an approximately uniform sample from $K' = [0, 2D] \times \mathrm{K}$ via the quantum

hit-and-run walk. To obtain a quantum stationary state, we use a similar idea as in [103] to construct a sequence of $m = \lceil n \log_2(2D) \rceil$ convex bodies. Let $\hat{\mathrm{K}}_0 := \mathrm{B}$ and $\hat{\mathrm{K}}_i := 2^{i/n}\mathrm{B}_n \cap \mathrm{K}'$ for $i \in [m]$. As the length of the pencil is $2D$, $\hat{\mathrm{K}}_m = \mathrm{K}'$. The state $|\pi_0\rangle$ corresponding to $\hat{\mathrm{K}}_0$ is easy to prepare. It is straightforward to verify that $\langle \pi_i | \pi_{i+1} \rangle \geq c$ for some constant c, as $\mathrm{Vol}(\hat{\mathrm{K}}_{i+1}) \leq 2\,\mathrm{Vol}(\hat{\mathrm{K}}_i)$. To utilize the quantum speedup for MCMC framework (Theorem 3.2.1), it remains to lower bound the phase gap of the quantum walk operator for $\hat{\mathrm{K}}_i$. It can be shown that the mixing property of the hit-and-run walk in Theorem 3.2.5 implies that the phase gap of the quantum walk operator is $\tilde{\Omega}(n^{-1.5})$; see the proof of Lemma 3.4.8. Thus, by Theorem 3.2.1, $|\pi_m\rangle$ can be prepared using $\tilde{O}(n) \cdot \tilde{O}(n^{1.5}) = \tilde{O}(n^{2.5})$ quantum queries to O_{K}.

- We estimate ξ_{K} with multiplicative error $\epsilon/2$ using the quantum Chebyshev inequality (Theorem 3.2.3) instead of its classical counterpart. This means that $O(1/\epsilon)$ executions of quantum sampling in the first step suffice.

Overall, $\tilde{O}(n^{2.5}/\epsilon)$ quantum queries to O_{K} suffice to ensure that we obtain an estimate of ξ_{K} within multiplicative error $\epsilon/2$ with success probability at least 0.99. Since (3.4.9) ensures that $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ estimates $\mathrm{Vol}(\mathrm{K}')$ up to multiplicative error $\epsilon/2$ with probability at least 0.7, $\frac{\widetilde{\mathrm{Vol}(\mathrm{K}')}}{2D\xi_{\mathrm{K}}}$ estimates $\mathrm{Vol}(\mathrm{K})$ up to multiplicative error $\epsilon/2 + \epsilon/2 = \epsilon$ with success probability $0.99 \cdot 0.7 > 2/3$. □

3.4.3.2 Stationary states of consecutive steps

We now show that the inner product between stationary states of consecutive steps is at least a constant. More precisely, we have the following:

Lemma 3.4.2. *Let $|\pi_i\rangle$ be the stationary distribution state of the quantum walk W_i for $i \in [m]$ defined in (3.3.44). For $n \geq 2$, we have $\langle \pi_i | \pi_{i+1} \rangle > 1/3$ for $i \in [m-1]$.*

Proof. Recall that the stationary distribution π_i of step i has density proportional to $e^{-a_i x_0}$ as discussed in Section 3.4.1. The corresponding stationary distribution state is $|\pi_i\rangle = \int_{\mathrm{K}'} \mathrm{d}x \sqrt{\frac{e^{-a_i x_0}}{Z(a_i)}} |x\rangle$. Reference [201, Lemma 3.2] proves that $a^{n+1} Z(a)$ is log-concave (noting that the dimension of K' is $n+1$). This implies that

$$\sqrt{a_i^{n+1} Z(a_i)} \sqrt{a_{i+1}^{n+1} Z(a_{i+1})} \leq \left(\frac{a_i + a_{i+1}}{2} \right)^{n+1} Z\left(\frac{a_i + a_{i+1}}{2} \right). \tag{3.4.13}$$

Now, we have

$$\begin{aligned}
\langle \pi_i | \pi_{i+1} \rangle &= \int_{\mathrm{K}'} \mathrm{d}x \frac{\exp\left(-\frac{a_i + a_{i+1}}{2} x_0\right)}{\sqrt{Z(a_i)} \sqrt{Z(a_{i+1})}} && (3.4.14) \\
&\geq \left(\frac{2\sqrt{a_i}\sqrt{a_{i+1}}}{a_i + a_{i+1}} \right)^{n+1} \frac{\int_{\mathrm{K}'} \mathrm{d}x \exp\left(-\frac{a_i + a_{i+1}}{2} x_0\right)}{Z\left(\frac{a_i + a_{i+1}}{2}\right)} && (3.4.15) \\
&= \left(\frac{2\sqrt{a_i}\sqrt{a_{i+1}}}{a_i + a_{i+1}} \right)^{n+1} && (3.4.16) \\
&= \left(\frac{2\sqrt{a_i}\sqrt{a_i(1 - \frac{1}{\sqrt{n}})}}{a_i + a_i(1 - \frac{1}{\sqrt{n}})} \right)^{n+1} = \left(\frac{2\sqrt{1 - \frac{1}{\sqrt{n}}}}{2 - \frac{1}{\sqrt{n}}} \right)^{n+1}, && (3.4.17)
\end{aligned}$$

where the inequality follows from (3.4.13). To lower bound the above quantity, we

use the fact that $\sqrt{1-1/\sqrt{n}} \geq 1 - \frac{1}{2\sqrt{n}} - \frac{1}{2n}$. Hence, for $n \geq 2$ we have

$$\langle \pi_i | \pi_{i+1} \rangle \geq \left(\frac{2 - \frac{1}{\sqrt{n}} - \frac{1}{n}}{2 - \frac{1}{\sqrt{n}}} \right)^{n+1} = \left(1 - \frac{\frac{1}{n}}{2 - \frac{1}{\sqrt{n}}} \right)^{n+1} \geq \left(1 - \frac{1}{(2 - \frac{1}{\sqrt{2}})n} \right)^{n+1} > \frac{1}{3}$$

as claimed. □

3.4.3.3 Nondestructive mean estimation

Now we briefly review the classical framework for Chebyshev cooling and discuss how to adapt it to quantum algorithms. Suppose we want to compute the expectation of a product

$$V = \prod_{i=1}^{m} V_i \tag{3.4.18}$$

of independent random variables. The following theorem of Dyer and Frieze [102] upper bounds the number of samples from the V_i that suffices to estimate $\mathbb{E}[V]$ with bounded relative error.

Proposition 3.4.1 ([102, Section 4.1])**.** *Let $V_1, \ldots, V_m$ be independent random variables such that $\frac{\mathbb{E}[V_i^2]}{\mathbb{E}[V_i]^2} \leq B$ for all $i \in [m]$. Let $X_j^{(1)}, \ldots, X_j^{(k)}$ be k samples of V_j for $j \in [m]$, and define $\overline{X}_j = \frac{1}{k} \sum_{\ell=1}^{k} X_j^{(\ell)}$. Let $V = \prod_{j=1}^{m} V_j$ and $X = \prod_{j=1}^{m} \overline{X}_j$. Then, taking $k = 16Bm/\epsilon^2$ ensures that*

$$\Pr\left[(1-\epsilon)\mathbb{E}[V] \leq X \leq (1+\epsilon)\mathbb{E}[V] \right] \geq \frac{3}{4}. \tag{3.4.19}$$

With standard techniques, the probability can be boosted to $1-\delta$ with a $\log(1/\delta)$ overhead.

In applications such as volume estimation [200] and estimating partition functions [251], the samples are produced by a random walk. Let the mixing time for each random walk be at most T. Then the total complexity for estimating $\mathbb{E}[V]$ with success probability $1-\delta$ is $O(TBm\log(1/\delta)/\epsilon^2)$. Replacing the random walk with a quantum walk can potentially improve the mixing time; see Section 3.1.3 for relevant literature. In particular, Montanaro [208] proposed a quantum algorithm for the simulated annealing framework with complexity $\tilde{O}(TBm\log(1/\delta)/\epsilon)$, which has a quadratic improvement in precision. Note that the dependence on T was not improved, as multiple copies of quantum states were prepared for the mean estimation (which uses measurements). In this paper, we use the quantum Chebyshev inequality (see Theorem 3.2.3) to estimate the expectation of V_i in a nondestructive manner which, together with Theorem 3.2.1, achieves complexity $\tilde{O}(\sqrt{T}Bm\log(1/\delta)/\epsilon)$.

The random variables V_i (determined by the cooling schedule) satisfy:

Proposition 3.4.2 (Eq. (3.4.8))**.** *Let X be a random sample from π_i and let $V_i = e^{(a_i-a_{i+1})X_0}$. Then*

$$\frac{\mathbb{E}_{\pi_i}[V_i^2]}{\mathbb{E}_{\pi_i}[V_i]^2} \leq 8. \tag{3.4.20}$$

The following lemma uses this property of the simulated annealing procedure to show that the quantum Chebyshev inequality can be used to estimate the mean of V_i on the distribution π_i, which gives an estimate of the ratio $\frac{Z(a_{i+1})}{Z(a_i)}$ in the volume

estimation algorithm. We first show that our random variables can be made to satisfy the conditions of Theorem 3.2.3, and then we outline how the corresponding circuit can be implemented. A detailed error analysis is deferred to Section 3.4.3.4.

Lemma 3.4.3. *Given $\tilde{O}(\log(1/\delta)/\epsilon)$ copies of the quantum states $|\pi_{i-1}\rangle$, there exists a quantum algorithm that outputs an estimate of $\mathbb{E}_{\pi_i}[V_i]$ (in Eq. (3.4.7)) with relative error less than ϵ with probability at least $1 - O(\delta)$.*

Proof. We achieve nondestructive mean estimation by the quantum Chebyshev inequality (Theorem 3.2.3). For the random variables V_i, we let μ_i denote their mean and σ_i^2 their variance. From Proposition 3.4.2, $\sqrt{\sigma_i^2 - \mu_i^2}/\mu_i \leq \sqrt{8} < 3$. For a small constant c, we use $\log(1/\delta)/c^2$ copies of $|\pi_{i-1}\rangle$ to create copies of $|\pi_i\rangle$ using $\pi/3$-amplitude amplification. We now use a quantum circuit that given $|x\rangle|0\rangle$ computes $|x\rangle|e^{a_i x_0 - a_{i-1} x_0}\rangle$, and then apply a circuit U_{median} that computes the median of all the ancilla registers:

$$U_{\text{median}}|0\rangle|a_1\rangle \cdots |a_s\rangle = |\text{median}\{a_1, \ldots, a_s\}\rangle|a_1\rangle \cdots |a_s\rangle. \tag{3.4.21}$$

By the classical Chebyshev inequality, we measure $\hat{\mu}_i$ such that $|\hat{\mu}_i - \mu_i| \leq c\mu_i$ with probability at least $1 - \delta$. Thus the probability that $\hat{\mu}_i/(1-c) < \mu$ is less than δ. Taking $H = \hat{\mu}_i/(1-c)$, our variables satisfy the conditions of the quantum Chebyshev inequality. In order to output an estimate of the mean with relative error at most ϵ, the quantum Chebyshev inequality now requires $\tilde{O}(\log(1/\delta)/\epsilon)$ calls to a sampler for the state $|\pi_i\rangle$, which we construct using $\pi/3$-amplitude amplification on copies of $|\pi_{i-1}\rangle$. By the union bound, the probability of failure of the whole

procedure is $O(\delta)$.

To be more specific, we replace $U|0\rangle$ in BasicEst (Algorithm 3.1) by $U_{i-1,l}|\pi_{i-1}\rangle$, and replace $\mathcal{Q}$ by $-U_{i-1,l}(\Pi_{i-1} - \Pi_{i-1}^{\perp})U_{i-1,l}^{\dagger}(\Pi_i - \Pi_i^{\perp})$ (where $\Pi_i = |\pi_i\rangle\langle\pi_i|$ and $\Pi_i^{\perp} = I - \Pi_i$ for all $i \in [m]$). The quantum circuit for nondestructive BasicEst is shown in Figure 3.4. Here, we run $\Theta(\log(1/\delta))$ executions of amplitude estimation (Figure 3.2) in parallel. Note that by (3.2.10), each amplitude estimation returns a state $\frac{e^{i\theta_p}}{\sqrt{2}}|\tilde{\theta}_p\rangle - \frac{e^{-i\theta_p}}{\sqrt{2}}|-\tilde{\theta}_p\rangle$. We use an ancilla register and apply the unitary

$$U_{\sin^2}|\theta\rangle|0\rangle := |\theta\rangle|\sin^2\theta\rangle; \tag{3.4.22}$$

because $\sin^2(\tilde{\theta}_p) = \sin^2(-\tilde{\theta}_p) = \tilde{p}$, the ancilla register becomes $|\tilde{p}\rangle$, where $\tilde{p}$ estimates p well as claimed in Theorem 3.2.2. We then take the median of such $\Theta(\log(1/\delta))$ executions using (3.4.21), and finally run the inverse of $U_{\sin^2}$ gates and amplitude estimations. This circuit is nondestructive because the states $|\pi_i\rangle$ are recovered after implementing the inverse amplitude amplifications, and a measurement that has a high probability of a single outcome does not disturb the input quantum state by much. The correctness follows from the proof of Theorem 3.2.3 in [134]. A detailed error analysis is given in the next subsection (see Lemma 3.4.9). □

3.4.3.4 Error analysis

In this section, we analyze the error incurred by both the quantum Chebyshev inequality (Line 3) and $\pi/3$-amplitude amplification (Line 5) in Algorithm 3.3.

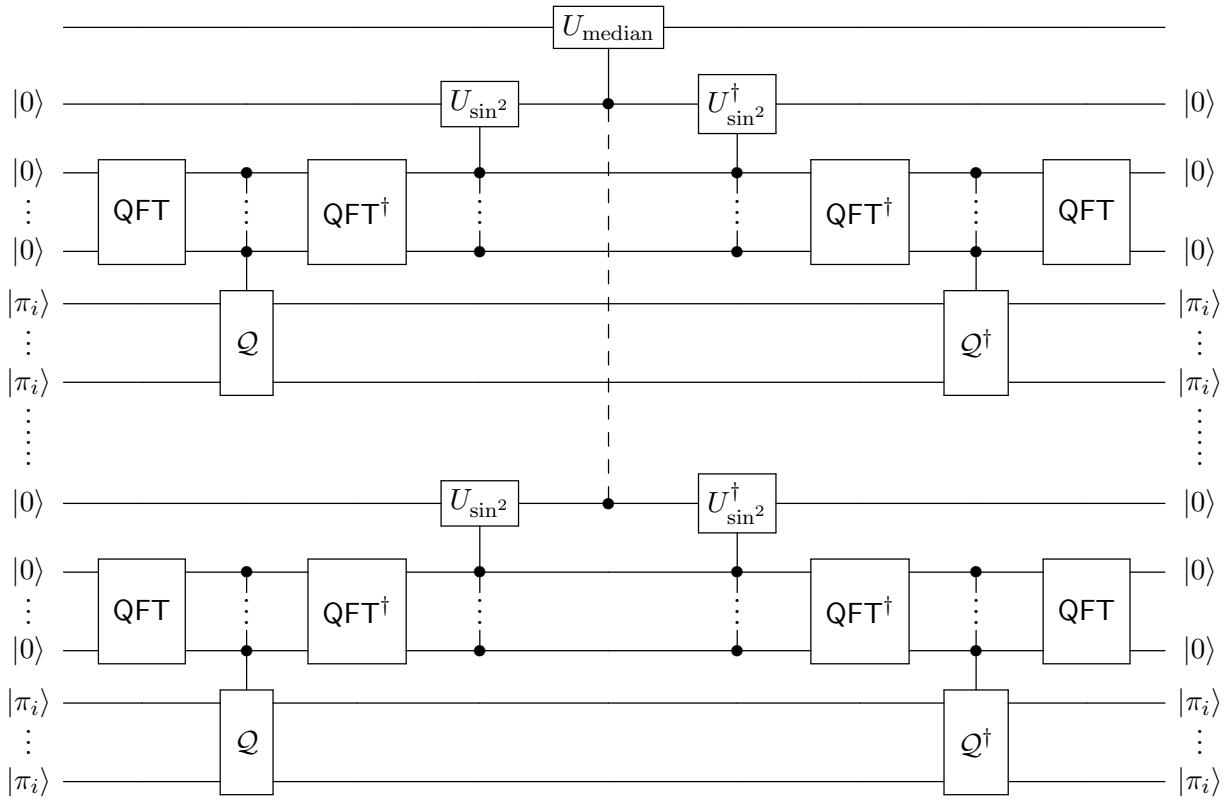

Figure 3.4: The quantum circuit for nondestructive BasicEst.

Lemma 3.4.4. *Given $\tilde{O}(\log(1/\delta)/\epsilon_1)$ copies of a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq \epsilon_1$, there exists a quantum procedure (using $\pi/3$-amplitude amplification and the quantum Chebyshev inequality) that outputs a $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq \epsilon_1 \mathbb{E}_{\pi_i}[V_i]$ (where $\mathbb{E}_{\pi_i}[V_i]$ is defined in Eq. (3.4.7)) with success probability $1-\delta^4$ using $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon_1 + n^{3/2}\log(1/\epsilon'))$ calls to the membership oracle and returns a final state $|\tilde{\pi}_i\rangle$ such that $\||\tilde{\pi}_i\rangle - |\pi_i\rangle\| = O(\epsilon_1 + \delta + \epsilon')$.*

We first show that $\pi/3$-amplitude amplification can be used to rotate $|\pi_i\rangle$ into $|\pi_{i-1}\rangle$ with error ϵ' using $\tilde{O}(\log(1/\epsilon))$ oracle calls. This procedure is used as a subroutine in a mean estimation circuit that estimates the mean of the random variable V_i using multiple approximate copies of $|\pi_{i-1}\rangle$. We ensure that the measurement

probabilities are highly peaked so that the state is not disturbed very much. Finally $\pi/3$-amplitude estimation is used again to rotate the approximate copies of the state $|\pi_{i-1}\rangle$ to approximate copies of the state $|\pi_i\rangle$.

Large effective spectral gap. Consider an ergodic, reversible Markov chain (Ω, p) with transition matrix P and a unique stationary distribution with density π. Let $a(x)$ be a probability measure over Ω such that the Markov chain mixes to its stationary distribution with a corresponding probability density $\pi(x)$ within a total variation distance of ϵ within t steps. From the definition of the transition matrix $P(x, y) = \langle x|P|y\rangle = p_{y\to x}$.

The discriminant matrix D defined in (3.3.9) is related to the transition matrix as $P = D_\pi D D_\pi^{-1}$, as shown in (3.3.43). For a hit-and-run walk, the transition matrix P maps a density concentrated at one point to a uniform density over a compact subset of $\mathbb{R}^n$. Thus a convergent sequence of distributions is still convergent after one step of the walk, and P is a compact linear operator. Since D is similar to P, D is a compact Hermitian operator over $L_2(\Omega)$ and thus has a countable set of real eigenvalues λ_i and corresponding orthonormal eigenvectors (eigenfunctions) $v_i \in L_2(\Omega)$. Orthonormality implies that $\int_\Omega v_i(x)v_j(x)\,\mathrm{d}x = \delta_{i,j}$. Notice that

$$PD_\pi v_i = D_\pi D(v_i) = \lambda_j D_\pi v_i; \tag{3.4.23}$$

thus $f_i = D_\pi v_i$ is an eigenvector of P' with eigenvalue λ_i. The eigenvectors f_i may not be orthogonal under the standard inner product on $L_2(\Omega)$. However, we can

define an inner product

$$\langle f, g\rangle_\pi := \langle D_\pi^{-1} f, D_\pi^{-1} g\rangle = \int_\Omega \frac{f(x)g(x)}{\pi(x)}\,\mathrm{d}x \tag{3.4.24}$$

over the space $L_2(\Omega)$. It is easy to see that $\langle f_i, f_j\rangle_\pi = \langle \phi_i, \phi_j\rangle = \delta_{i,j}$.

It can be verified that $\sqrt{\pi}(x)$ is an eigenfunction of D with eigenvalue 1. Thus the stationary state $\pi(x)$ is an eigenfunction of the transition operator P with eigenvalue 1. Since P is stochastic, this is the leading eigenvalue. The eigenfunctions of P are thus $1, \lambda_1, \lambda_2, \ldots$ with corresponding eigenfunctions $\pi(x), f_2(x), f_3(x), \ldots$. From the orthonormality of the f under $\langle\cdot,\cdot\rangle_\pi$, for any function g in $L_2(\Omega)$ we have

$$g = \sum_{i=1}^{\infty} \langle g, f_i\rangle_\pi f_i = \langle g, \pi\rangle + \sum_{i=2}^{\infty} \langle g, f_i\rangle_\pi f_i \tag{3.4.25}$$

$$= \left(\int_\Omega \frac{g(x)\pi(x)}{\pi(x)}\,\mathrm{d}x\right)\pi + \sum_{i=2}^{\infty} \langle g, f_i\rangle_\pi f_i \tag{3.4.26}$$

$$= \left(\int_\Omega g(x)\,\mathrm{d}x\right)\pi + \sum_{i=2}^{\infty} \langle g, f_i\rangle_\pi f_i. \tag{3.4.27}$$

Since a is a probability density, $a = \pi + \sum_{i=2}^{\infty}\langle a, f_i\rangle_\pi f_i$. After t steps of the Markov chain M on a we obtain the state $P^t a = \pi + \sum_{i=2}^{\infty} \lambda_i^t \langle a, f_i\rangle_\pi f_i$. Since $\|P^t a - \pi\|_2 \le \epsilon$, we have $\|\sum_{i=2}^{\infty} \lambda_i^t \langle a, f_i\rangle_\pi f_i\| \le \epsilon$ and from the orthonormality of f, $\langle a, f_i\rangle_\pi \lambda_i^t \le \epsilon$. If $1 > \lambda_i \ge 1 - \frac{1}{O(t)}$ then $\lambda_i^t = \Omega(1)$ and $\langle a, f_i\rangle_\pi = O(\epsilon)$.

The above analysis indicates that if a probability density a mixes in t steps under a Markov chain (Ω, p), then it has small overlap with each of the "bad" eigenfunctions (with spectral gap less than $\frac{1}{O(t)}$). Thus P effectively has a large

spectral gap when it acts on a.

Corresponding to a, consider the quantum states

$$|a\rangle := \int_\Omega \sqrt{a(x)}|x\rangle \,\mathrm{d}x, \qquad |\phi_a\rangle := \int_\Omega \int_\Omega \sqrt{a_x p_{x\to y}}|x\rangle|y\rangle \,\mathrm{d}x \,\mathrm{d}y. \tag{3.4.28}$$

For an eigenvector v_i of D (with eigenvalue λ_i), define the state $|v_i\rangle := \int_\Omega v_i(x)\,\mathrm{d}x = \int_\Omega \frac{f_i(x)}{\sqrt{\pi(x)}}\,\mathrm{d}x$. Then the walk operator W has the corresponding eigenvector $|u_i\rangle = \big(I - (\lambda_i - i\sqrt{1-\lambda_i^2})S\big)T|v_i\rangle$. Let $C_i := \lambda_i - i\sqrt{1-\lambda^2}$; then $\langle\phi_a|u_i\rangle = \langle\phi_a|T|v_i\rangle - C_i\langle\phi_a|ST|u_i\rangle$. Furthermore,

$$\langle\phi_a|T|v_i\rangle = \langle a|v_i\rangle = \int_\Omega \frac{\sqrt{a(x)}f_i(x)}{\sqrt{\pi(x)}}\,\mathrm{d}x, \tag{3.4.29}$$

and

$$\begin{aligned}
\langle\phi_a|ST|v_i\rangle &= \left(\int_\Omega \sqrt{a_x p_{x\to y}}\langle y|\langle x|\right)\left(\int_\Omega \sqrt{v_{x'}p_{x'\to y'}}|x'\rangle|y'\rangle\right) && (3.4.30)\\
&= \int_\Omega \sqrt{a_x}\Big(\int_\Omega \sqrt{p_{x\to y}p_{y\to x}v_i(y)}\,\mathrm{d}y\Big)\,\mathrm{d}x && (3.4.31)\\
&= \int_\Omega \sqrt{a_x}(Dv_i)(x)\,\mathrm{d}x && (3.4.32)\\
&= \lambda_i \int_\Omega \sqrt{a_x}v_i(x) && (3.4.33)\\
&= \lambda_i\langle a|v_i\rangle. && (3.4.34)
\end{aligned}$$

We have $\langle\phi_a|u_i\rangle = (1-\lambda_i C_i)\langle a|v_i\rangle$ and therefore

$$|\langle\phi_a|u_i\rangle| = |(1-\lambda_i|C_i|)|\langle a|v_i\rangle = \sqrt{(1-\lambda_i^2)^2 + (1-\lambda_i)^2}\langle a|v_i\rangle \le 2\langle a|v_i\rangle. \tag{3.4.35}$$

In addition,

$$\langle a|v_i\rangle = \int_\Omega \frac{\sqrt{a(x)}f_i(x)}{\sqrt{\pi(x)}}\,\mathrm{d}x = \int_\Omega \sqrt{\frac{\pi(x)}{a(x)}}\frac{a(x)f_i(x)}{\pi(x)}\,\mathrm{d}x. \tag{3.4.36}$$

The above discussion establishes the following proposition indicating that if a distribution with density $a(x)$ mixes fast and the stationary distribution with density $\pi(x)$ has a bounded L_2-norm with respect to $a(x)$, then the quantum walk operator W acting on the subspace spanned by $|\pi\rangle$ and $|a\rangle$ has a large spectral gap.

Proposition 3.4.3 ([66, Proposition 4.3])**.** *Let $M = (\Omega, p)$ be an ergodic reversible Markov chain with a transition operator P and unique stationary state with a corresponding density $\pi \in L_2(\Omega)$. Let $\{(\lambda_i, f_i)\}$ be the set of eigenvalues and eigenfunctions of P, and $|u_i\rangle$ be the eigenvectors of the corresponding quantum walk operator W. Let $a \in L_2(\Omega)$ be a probability density that mixes up to total variation distance ϵ in t steps of M. Furthermore, assume that $\int_\Omega \frac{\pi(x)}{a(x)}\,\pi(x)\mathrm{d}x \leq c$ for some constant c. Define*

$$|a\rangle = \int_\Omega \sqrt{a(x)}|x\rangle\,\mathrm{d}x; \tag{3.4.37}$$

$$|\phi_a\rangle = \int_\Omega \sqrt{a(x)}\int_\Omega \sqrt{p_{x\to y}}|x\rangle|y\rangle\,\mathrm{d}x\,\mathrm{d}y. \tag{3.4.38}$$

Then $\langle\phi_a|u_i\rangle = O(\epsilon^{1/2})$ for all i such that $1 > \lambda_i \geq 1 - \frac{1}{O(t)}$.

Warmness of π_{i+1} with respect to π_i. We show that density π_i mixes to π_{i+1} under the walk W_{i+1} and vice versa. To apply Theorem 3.2.5, we show that the two

distributions are warm with respect to each other.

The L_2-norm of a distribution with density $\pi_1 \in L_2(\Omega)$ with respect to another with density $\pi_2 \in L_2(\Omega)$ is defined as

$$\|\pi_1/\pi_2\| = \mathbb{E}_{X\sim\pi_1}\left[\frac{\pi_1(X)}{\pi_2(X)}\right] = \int_\Omega \frac{\pi_1(x)}{\pi_2(x)}\,\pi_1(x)\,\mathrm{d}x. \tag{3.4.39}$$

A density $\pi_1 \in L_2(\Omega)$ is said to be a warm start for $\pi_2 \in L_2(\Omega)$ if the L_2-norm $\|\pi_1/\pi_2\|$ is bounded by a constant.

Lemma 3.4.5 ([201, Lemma 4.4])**.** *The L_2-norm of the probability distribution with density $\pi_i = \frac{e^{-a_i x_0}}{Z(a_i)}$ with respect to that with density $\pi_{i+1} = \frac{e^{-a_{i+1} x_0}}{Z(a_{i+1})}$ is at most 8.*

Lemma 3.4.6. *The L_2-norm of the probability distribution with density $\pi_{i+1} = \frac{e^{-a_{i+1} x_0}}{Z(a_{i+1})}$ with respect to that with density $\pi_i = \frac{e^{-a_i x_0}}{Z(a_i)}$ is at most 1.*

Proof. Since $a^n Z(a)$ is a log-concave function [201, Lemma 3.2], we have

$$\begin{aligned}
\mathbb{E}_{X\sim\pi_{i+1}}\left[\frac{\pi_{i+1}(X)}{\pi_i(X)}\right] &= \frac{\int_{\mathrm{K}'} e^{(a_i-a_{i+1})x_0} e^{-a_{i+1}x_0}\mathrm{d}x \int_{\mathrm{K}'} e^{-a_i x_0}\mathrm{d}x}{\int_{\mathrm{K}'} e^{-a_{i+1}x_0}\mathrm{d}x \int_{\mathrm{K}'} e^{-a_{i+1}x_0}\mathrm{d}x} && (3.4.40)\\
&= \frac{Z(2a_{i+i}-a_i)Z(a_i)}{Z(a_{i+1})^2} && (3.4.41)\\
&\le \left(\frac{a_{i+1}^2}{a_i(2a_{i+1}-a_i)}\right)^{n+1} && (3.4.42)\\
&\le \left(\frac{\left(1-\frac{1}{\sqrt{n}}\right)^2}{1-\frac{2}{\sqrt{n}}}\right)^{n+1} && (3.4.43)\\
&\le \left(1-\frac{1}{n}\right)^{n+1} < 1 && (3.4.44)
\end{aligned}$$

as claimed, where (3.4.41) follows from the definition of Z, (3.4.42) follows from logconcavity of $a^n Z(a)$, and (3.4.43) follows from the definition of a_i. □

Error analysis of $\pi/3$-amplitude amplification. Consider a simulated annealing procedure that follows a sequence of Markov chains $M_1, M_2, \ldots$ with steady states $\mu_1, \mu_2, \ldots$. Consider an alternate walk operator (used in [272]) of the form

$$W_i' = U_i^\dagger S U_i R_{\mathcal{A}} U_i^\dagger S U_i R_{\mathcal{A}} \tag{3.4.45}$$

where $R_{\mathcal{A}}$ denotes the reflection about the subspace $\mathcal{A} := \text{span}\{|x\rangle|0\rangle : x \in \text{K}\}$ and S is the swap operator. We have $U_i|x\rangle|0\rangle = \int_{y\in\text{K}} \sqrt{p^{(i)}_{x\to y}}|x\rangle|y\rangle \, \mathrm{d}y$ where $p^{(i)}$ is the transition probability corresponding to the i^{th} chain.

The W_i' operator is related to the walk operator $W_i = S(2\Pi_i - I)$ via conjugation by U_i, i.e., $W_i = U_i W_i' U_i^\dagger$. Thus W_i' has the same eigenvalues as W_i, and if $|u_j\rangle$ is an eigenvector of W_j then $|v\rangle = U_i^\dagger|u\rangle$ is an eigenvector of W_i with the same eigenvalue λ_j. For any classical distribution f, we define $|f\rangle = \int_\Omega \sqrt{f(x)}|x\rangle \, \mathrm{d}x$ and $|\phi_f^{(i)}\rangle = \int_\Omega \sqrt{f(x)}|x\rangle \int_\Omega \sqrt{p^{(i)}_{x\to y}}|y\rangle \, \mathrm{d}y \, \mathrm{d}x$. Since $|\phi^{(i)}_{\pi_i}\rangle$ is a stationary state of W_i with eigenvalue 1, it follows that $|\pi_i\rangle|0\rangle$ is an eigenvalue of W_i with eigenvalue 1.

In each stage of the volume estimation algorithm, we sample from a state with density $\pi_i(x) = \frac{e^{-a_i x_0}}{Z(a_i)}$. Each such distribution is the stationary state of a hit-and-run walk with the corresponding target density. Thus the corresponding states $|\pi_i\rangle$ are the stationary states of the corresponding walk operators W_i and W_i'. Both W_i and $W_{i'}$ can be implemented using a constant number of U_i gates.

From Lemma 3.4.2, we know that the inner product $\langle\pi_i|\pi_{i+1}\rangle$ between the states at any stage of the algorithm is at least $\frac{1}{3}$. This implies that the inner product between $|\pi_i\rangle|0\rangle$ and $|\pi_{i+1}\rangle|0\rangle$ is also at least $\frac{1}{3}$. In the following we abuse

notation by sometimes writing only $|\pi_i\rangle$ to denote $|\pi_i\rangle|0\rangle$, as it is easy to tell from context whether the ancilla register should be present.

Lemma 3.2.1 in Section 3.2.2 indicates that $\pi/3$-amplitude amplification can be used to rotate the state $|\pi_i\rangle$ to $|\pi_{i+1}\rangle$ if we can implement the rotation unitaries

$$R_i = \omega|\pi_i\rangle\langle\pi_i| + (I - |\pi_i\rangle\langle\pi_i|) \quad \text{and} \quad R_{i+1} = \omega|\pi_{1+1}\rangle\langle\pi_{i+1}| + (I - |\pi_{i+1}\rangle\langle\pi_{i+1}|)\,.$$

To implement these rotations we use the fact that π_i and π_{i+1} are the leading eigenvectors of the operators W_i' and W_{i+1}', respectively. We show the following lemmas which are adapted variants of Lemma 2 and Corollary 2 in [272]:

Lemma 3.4.7. *Let W be a unitary operator with a unique leading eigenvector $|\psi_0\rangle$ with eigenvalue 1. Denote the remaining eigenvectors by $|\psi_j\rangle$ with corresponding eigenvalues $e^{2\pi i\xi_j}$. For any $\Delta \in (0,1]$ and $\epsilon_2 < 1/2$, define $a := \log(1/\Delta)$ and $c := \log(1/\sqrt{\epsilon})$. There exists a quantum circuit V that uses ac ancilla qubits and invokes the controlled-W gate $2^a c$ times such that*

$$V|\psi_0\rangle|0\rangle^{\otimes ac} = |\psi_0\rangle|0\rangle^{\otimes ac} \tag{3.4.46}$$

and

$$V|\psi_j\rangle|0\rangle^{\otimes ac} = \sqrt{1-\epsilon_2}|\psi_j\rangle|\chi_j\rangle + \sqrt{\epsilon_2}|\psi_j\rangle|0\rangle^{\otimes ac} \tag{3.4.47}$$

where $|\chi_j\rangle$ is orthogonal to $|0\rangle^{\otimes ac}$ for all $|\psi_j\rangle$ such that $\xi_j \geq \Delta$.

Proof. Consider a quantum phase estimation circuit U with a ancilla qubits that invokes the controlled-W gate 2^a times (see Figure 3.5). The phase estimation circuit first creates an equal superposition over a ancilla qubits using Hadamard gates. For $k = 0, \ldots, a-1$ we apply a controlled-W^k operator to the input register, controlled by the $(a-k)^{\text{th}}$ register. Finally the inverse quantum Fourier transform is applied on the ancilla registers. Then

$$U|\psi_j\rangle|0\rangle^{\otimes a} = |\psi_j\rangle \otimes \mathsf{QFT}^\dagger \left(\frac{1}{\sqrt{2^a}} \sum_{m=0}^{2^a-1} e^{2\pi i m \xi_j} |m\rangle \right) \tag{3.4.48}$$

$$= |\psi_j\rangle \otimes \frac{1}{2^a} \sum_{m,m'=0}^{2^a-1} e^{2\pi i m(\xi_j - m'/2^a)} |m'\rangle. \tag{3.4.49}$$

The amplitude corresponding to $|0\rangle$ on the ancilla registers is

$$a_{j,0} := \frac{1}{2^a} \sum_{m=0}^{2^a-1} e^{2\pi i m \xi_j} = \frac{1 - e^{2\pi i 2^a \xi_j}}{2^a (1 - e^{2\pi i \xi_j})} \tag{3.4.50}$$

for $j \neq 0$, and $a_{0,0} = 1$. If $j \neq 0$ then

$$|a_{j,0}| = \left| \frac{1 - e^{2\pi i 2^a \xi_j}}{2^a (1 - e^{2\pi i \xi_j})} \right| \leq \left| \frac{1}{2^{a-1}(1 - e^{2\pi i \xi_j})} \right| \leq \frac{1}{2^{a+1}|\xi_j|}. \tag{3.4.51}$$

Thus $|a_{j,0}| \leq \frac{1}{2}$ if $\xi_j \geq \delta$. Using c copies of the circuit (resulting in ac ancilla registers and $2^a c$ controlled-W gates), the amplitude for 0 in all the ancilla registers if $\xi_j \geq \delta$ is $\frac{1}{2^c} = \sqrt{\epsilon}$. □

Corollary 3.4.1. *Let W be a unitary operator with a unique leading eigenvector $|\psi_0\rangle$ with eigenvalue 1. Denote the remaining eigenvectors by $|\psi_j\rangle$ with corresponding*

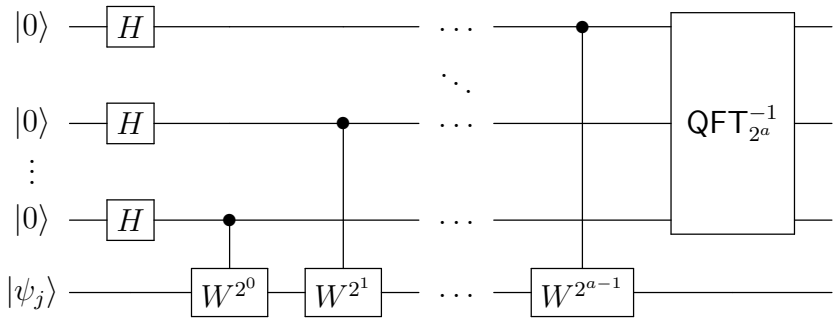

Figure 3.5: The quantum phase estimation circuit. Here W is a unitary operator with eigenvector $|\psi_j\rangle$; in $\pi/3$-amplitude estimation it is the quantum walk operator W_i' in (3.4.45).

eigenvalues $e^{2\pi i\xi_j}$. *For any* $\Delta \in (0,1]$ *and* $\epsilon_2 < 1/2$, *define* $a := \log(1/\Delta)$ *and* $c := \log\bigl(1/\sqrt{\epsilon_2}\bigr)$. *There exists a quantum circuit* $\tilde{R}$ *that uses* ac *ancilla qubits and invokes the controlled-*W *gate* $2^{a+1}c$ *times such that*

$$\tilde{R}|\psi_0\rangle|0\rangle^{\otimes ac} = (R|\psi_0\rangle)|0\rangle^{\otimes ac} \tag{3.4.52}$$

(where $R = \omega|\psi_0\rangle\langle\psi_0| - (I - |\psi_0\rangle\langle\psi_0|)$*) and*

$$\|\tilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} - (R|\psi_j\rangle)|0\rangle^{\otimes ac}\| \leq 2\sqrt{\epsilon_2} \tag{3.4.53}$$

for $j \neq 0$ *such that* $\xi_j \geq \Delta$.

Proof. Let $\tilde{R} := V^\dagger(I \otimes Q)V$ where V is the quantum circuit in Lemma 3.4.7 and $Q := \omega|0\rangle\langle 0|^{\otimes ac} + (I - |0\rangle\langle 0|^{\otimes ac})$. Then we have

$$\tilde{R}|\psi_0\rangle|0\rangle^{\otimes ac} = V^\dagger(I \otimes Q)|\psi_0\rangle|0\rangle^{\otimes ac} = \omega|\psi_0\rangle|0\rangle^{\otimes ac} = R|\psi_0\rangle|0\rangle^{\otimes ac}. \tag{3.4.54}$$

For $j \neq 0$ such that $\xi_j \geq \delta$,

$$\begin{aligned}
\tilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} &= V^\dagger(I \otimes Q)(\sqrt{1-\epsilon_2}|\psi_j\rangle|\chi_j\rangle + \sqrt{\epsilon_2}|\psi_j\rangle|0\rangle^{\otimes ac}) \\
&= V^\dagger(\sqrt{1-\epsilon_2}|\psi_j\rangle|\chi_j\rangle + \sqrt{\epsilon_2}\omega|\psi_j\rangle|0\rangle^{\otimes ac} \\
&= V^\dagger(|\psi_j\rangle \otimes (\sqrt{1-\epsilon_2}|\chi_j\rangle + \sqrt{\epsilon_2}|0\rangle^{\otimes ac}) + \sqrt{\epsilon_2}(\omega-1)|\psi_j\rangle|0\rangle^{\otimes ac}) \\
&= |\psi_j\rangle|0_j\rangle + V^\dagger\sqrt{\epsilon_2}(\omega-1)|\psi_j\rangle|0\rangle^{\otimes ac}. \qquad (3.4.55)
\end{aligned}$$

Thus $\|\tilde{R}|\psi_j\rangle|0\rangle^{\otimes ac} - (R|\psi_j\rangle)|0\rangle^{\otimes ac}\| \leq \|V^\dagger\sqrt{\epsilon_2}(\omega-1)|\psi_j\rangle|0\rangle^{\otimes ac}\| \leq 2\sqrt{\epsilon_2}$. □

Finally, we can prove the following lemma for analyzing the error incurred by $\pi/3$-amplitude amplification in our quantum volume estimation algorithm:

Lemma 3.4.8 ([66, Lemma 4.8])**.** *Starting from $|\pi_i\rangle$, we can obtain a state $|\tilde{\pi}_{i+1}\rangle$ such that $\||\pi_{i+1}\rangle - |\tilde{\pi}_{i+1}\rangle\| \leq \epsilon$ using $\tilde{O}(n^{3/2}\log(1/\epsilon))$ calls to the controlled walk operators W_i', W_{i+1}'. This results in $\tilde{O}(n^{3/2}\log(1/\epsilon))$ calls to the membership oracle O_{K}.*

Error analysis for the quantum Chebyshev inequality. We also analyze the error from the quantum Chebyshev inequality (Theorem 3.2.3), giving a robust version of Lemma 3.4.3.

Lemma 3.4.9. *Suppose we have $\tilde{O}(\log(1/\delta)/\epsilon)$ copies of a state $|\tilde{\pi}_{i-1}\rangle$ such that $\||\tilde{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| \leq \epsilon$. Then the quantum Chebyshev inequality can be used to output $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq O(\epsilon)\mathbb{E}_{\pi_i}[V_i]$ with success probability $1-\delta^4$ using $\tilde{O}(n^{3/2}\log(1/\delta)/\epsilon)$ calls to the membership oracle. The output state $|\hat{\pi}_{i-1}\rangle$ satisfies*

$\||\hat{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| = O(\epsilon + \delta)$.

Proof. The quantum Chebyshev inequality uses an implementation of $US_0U^\dagger S_i$ where U is a unitary operator satisfying $U|\pi_{i-1}\rangle = |\pi_i\rangle$. From Lemma 3.4.8, using $\ln(1/\epsilon_2)$ iterations of $\pi/3$-amplitude amplification ($U_{\log 1/\epsilon_2}$ in (3.2.5)) instead of U induces an error of ϵ_2 and uses $O(n^{3/2}\log(1/\epsilon_2))$ oracle calls. Using approximate phase estimation as in Corollary 3.4.1 and Lemma 3.4.8, Π_{i-1} and Π_i can be implemented up to error ϵ_3 using $O(n^{3/2}\log(1/\epsilon_3))$ oracle calls. Thus each block corresponding to Theorem 3.2.2 induces an error of $O(\epsilon_2+\epsilon_3)$, and the final state before the median is measured has an error of $O(\epsilon+\epsilon_2+\epsilon_3)$. Therefore, using $O(\log(1/\delta_1)/\epsilon)$ copies of $|\tilde{\pi}_{i-1}\rangle$ returns a sample $\tilde{V}_i$ such that $|\tilde{V}_i - \mathbb{E}_{\pi_i}[V_i]| \leq O(\epsilon_2+\epsilon_3+\epsilon)\mathbb{E}_{\pi_i}[V_i]$ with success probability $1-\delta_1$. Performing a measurement with success probability $1-\delta_1$ implies that the posterior state has an overlap $\sqrt{1-\delta_1}$ with the initial state. This induces an error of magnitude at most $\sqrt{2(1-\sqrt{1-\delta_1})} = O(\delta_1^{1/4})$.

The measurement on the $\log(1/\delta)/c$ copies of $|\tilde{\pi}_{i-1}\rangle$ used to estimate $\hat{\mu}$ has relative error at most c with probability $1-\delta$. This causes an error $O(\delta_1^{1/4})$ in addition to the error ϵ_2 from $\pi/3$-amplitude amplification.

Finally, note that the basic amplitude estimation circuit (analyzed in Theorem 3.2.2) is a subroutine of the quantum Chebyshev inequality (Theorem 3.2.3), and uncomputing the block corresponding to Theorem 3.2.2 induces an error of $O(\epsilon_2+\epsilon_3)$, giving an overall error of $O(\epsilon_2+\epsilon_3+\epsilon+\delta^{1/4})$. The result follows by taking $\epsilon_2 = \epsilon_3 = \epsilon$ and $\delta_1 = \delta^4$. □

Proof of Lemma 3.4.4. We finally prove Lemma 3.4.4:

Proof. Lemma 3.4.9 is used to estimate the mean with $\epsilon = \epsilon_1$ and leaves a posterior state $|\hat{\pi}_{i-1}\rangle$ such that $\||\hat{\pi}_{i-1}\rangle - |\pi_{i-1}\rangle\| = O(\epsilon_1 + \delta)$. We can then use $\pi/3$-amplitude amplification to rotate this state into $|\tilde{\pi}_i\rangle$, adding error $O(\epsilon')$ at the cost of $O(n^{3/2}\log(1/\epsilon'))$. This completes the proof. □

3.4.4 Quantum algorithms for rounding logconcave densities

We first define roundedness of logconcave density functions as follows:

Definition 3.4.1. *A logconcave density function f is said to be c-rounded if*

1. *The level set of f of probability $1/8$ contains a ball of radius r;*
2. *$\mathbb{E}_f(|x - z_f|) \leq R^2$, where z_f is the centroid of π_f;*

and $R/r \leq c\sqrt{n}$.

In the previous section we assumed that the distributions π_i sampled during the hit-and-run walk are $O(1)$-rounded (i.e., well-rounded). From Theorem 3.2.5, this implies that the hit-and-run walk for the distribution π_i mixes from a warm start in time $\tilde{O}(n^3)$. In this subsection we show how the distributions π_i can be transformed to satisfy this condition.

Following the classical discussion in [199], we actually show a stronger condition: the distributions are transformed to be in "near-isotropic" position. A density function f is said to be in *isotropic* position if

$$\mathbb{E}_f[x] = 0 \quad \text{and} \quad \mathbb{E}_f[xx^T] = I. \tag{3.4.56}$$

In other words, $\int_{\mathbb{R}^n}(u^T x)^2 f(x)\,\mathrm{d}x = |u|^2$ for every vector $u \in \mathbb{R}^n$. We say that K is *near-isotropic* up to a factor of c if

$$\frac{1}{c} \leq \int_{\mathbb{R}^n}(u^T(x - z_f))^2 f(x)\,\mathrm{d}x \leq c \tag{3.4.57}$$

for every unit vector $u \in \mathbb{R}^n$.

The following lemma shows that logconcave density functions in isotropic position are also $O(1)$-rounded:

Lemma 3.4.10 ([202, Lemma 5.13])**.** *Every isotropic logconcave density is* $(1/e)$*-rounded.*

The following lemma shows that any logconcave density function can be put into isotropic position by applying an affine transformation, generalizing the same result for uniform distributions by Rudelson [238]:

Lemma 3.4.11 ([199, Lemma 2.2])**.** *Let* f *be a logconcave function in* $\mathbb{R}^n$ *that is not concentrated on a subspace, and let* $X^1, \ldots, X^k$ *be independent random points from the corresponding distribution. There is a constant* C_0 *such that if* $k > C_0 t^3 \ln n$*, then the transformation* $g(x) = T^{-1/2}x$ *where*

$$\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X^i, \qquad T = \frac{1}{k}\sum_{i=1}^{k}(X^i - \bar{X})(X^i - \bar{X})^T \tag{3.4.58}$$

puts f *in* 2*-isotropic position with probability at least* $1 - 1/2^t$*.*

From Lemma 3.4.11, $k = \lceil C_0 n \ln^5 n \rceil = \tilde{\Theta}(n)$ samples from a logconcave density f suffice to put it into near-isotropic position. However, efficiently obtaining

samples from a density π_i requires it to be well-rounded to start with. To overcome this difficulty, we interlace the rounding with the stages of the volume estimation algorithm where in each stage, we obtain an affine transformation that puts the density to be sampled in the next stage into isotropic position. The density π_0 is very close to an exponential distribution (since it is concentrated inside the convex body) and can hence be sampled without resorting to a random walk.

To show that samples from π_i can be used to transform π_{i+1} into isotropic position, we use the following lemma:

Lemma 3.4.12 ([158, Lemma 4.3])**.** *Let f and g be logconcave densities over K with centroids z_f and z_g respectively. Then for any $u \in \mathbb{R}^n$,*

$$\mathbb{E}_f[(u \cdot (x - z_f))^2] \leq 16\mathbb{E}_f\left[\frac{f}{g}\right]\mathbb{E}_g[(u \cdot (x - z_g))^2]. \tag{3.4.59}$$

We now have the following proposition:

Proposition 3.4.4. *If affine transformation S_i puts π_i in near-isotropic position then it also puts π_{i+1} in near-isotropic position.*

Proof. Let S_i put π_i in 2-isotropic position. Applying Lemma 3.4.12 with $f = \pi_{i+1}, g = \pi_i$, we have that for any unit vector $u \in \mathbb{R}^n$,

$$\mathbb{E}_{\pi_{i+1}}[(u \cdot (x - z_{\pi_{i+1}}))^2] \leq 16\mathbb{E}_{\pi_{i+1}}\left[\frac{\pi_{i+1}}{\pi_i}\right]\mathbb{E}_{\pi_i}[(u \cdot (x - z_{\pi_i}))^2] \leq 32 \tag{3.4.60}$$

from Lemma 3.4.6, and

$$\frac{1}{2} \leq \mathbb{E}_{\pi_i}[(u \cdot (x - z_{\pi_i}))^2] \leq 144\mathbb{E}_{\pi_{i+1}}[(u \cdot (x - z_{\pi_{i+1}}))^2] \tag{3.4.61}$$

from Lemma 3.4.5. Thus $E_{\pi_{i+1}}$ is also put in near-isotropic position. □

We finally have the main result of this section:

Proposition 3.4.5. *At each stage i of Algorithm 3.4, the affine transformation puts the distribution π_{i+1} in near-isotropic position using an additional $\tilde{O}(n^{2.5})$ quantum queries to O_K.*

Proof. Since π_1 is nearly an exponential distribution, it can be sampled without using a random walk and thus the proposition is true for $i = 0$. Assume that the proposition is true for $1, 2, \ldots, k$. Then an affine transformation can be found to put π_k in near-isotropic position. Thus a classical hit-and-run walk starting from π_{k-1} converges to π_k in $\tilde{O}(n^3)$ steps. By the analysis in Section 3.4.3.4, a quantum sample $|\pi_{k-1}\rangle$ can be rotated to $|\pi_k\rangle$ using $\tilde{O}(n^{1.5})$ quantum queries. $\tilde{O}(n)$ such samples suffice to compute the covariance matrix T in (3.4.58), which puts π_k in 2-isotropic position. By Proposition 3.4.4, this also puts π_{k+1} in near-isotropic position. This concludes the proof. □

Rounding the convex body as a preprocessing step. Consider applying only the rounding part of Algorithm 3.4. By Proposition 3.4.5, the final affine transformation puts the density $\pi_m \propto e^{-a_m x_0}$ in near-isotropic position. Since $a_m \leq$

Algorithm 3.4: Volume estimation of convex K with interlaced rounding.

Input: Membership oracle O_{K} for K; $R = O(\sqrt{n})$ s.t. $\mathrm{B}_2(0,1) \subseteq \mathrm{K} \subseteq \mathrm{B}_2(0,R)$.

Output: ϵ-multiplicative approximation of Vol(K).

Set $m = \Theta(\sqrt{n}\log(n/\epsilon))$ to be the number of iterations of simulated annealing and $a_i = 2n(1-\frac{1}{\sqrt{n}})^i$ for $i \in [m]$. Let π_i be the probability distribution over K′ with density proportional to $e^{-a_i x_0}$;

Set error parameters $\delta, \epsilon' = \Theta(\epsilon/m^2), \epsilon_1 = \epsilon/2m$; let $k = \tilde{\Theta}(\sqrt{n}/\epsilon)$ be the number of copies of stationary states in the quantum Chebyshev inequality; let $l = \tilde{\Theta}(n)$ be the number of copies of stationary states needed to obtain the affine transformation S_i;

Prepare $k+l$ (approximate) copies of $|\pi_0\rangle$, denoted $|\tilde{\pi}_0^{(1)}\rangle, \ldots, |\tilde{\pi}_0^{(k+l)}\rangle$;

for $i \in [m]$ **do**

- Use the quantum Chebyshev inequality on the k copies of the state $|\tilde{\pi}_{i-1}\rangle$ with parameters ϵ_1, δ to estimate the expectation $\mathbb{E}_{\pi_i}[V_i]$ (in Eq. (3.4.7)) as $\tilde{V}_i$ (Lemma 3.4.9 and Figure 3.4). The post-measurement states are denoted $|\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k)}\rangle$;
- Use the l copies of the state $|\pi_{i-1}\rangle$ to nondestructively[14] obtain the affine transformation $S_i = T = \frac{1}{l}\sum_{q=1}^{l}(X^q - \bar{X})(X^q - \bar{X})^T$ where the X_q are samples from the density π_{i-1} and $\bar{X} = \frac{1}{l}\sum_{q=1}^{l} X^q$. The post-measurement states are denoted $|\hat{\pi}_{i-1}^{(k+1)}\rangle, \ldots, |\hat{\pi}_{i-1}^{(k+l)}\rangle$;
- Apply $\pi/3$-amplitude amplification with error ϵ' (Section 3.2.2 and Lemma 3.4.8) and affine transformation S_i to map $|S_i\hat{\pi}_{i-1}^{(1)}\rangle, \ldots, |S_i\hat{\pi}_{i-1}^{(k+l)}\rangle$ to $|S_i\tilde{\pi}_i^{(1)}\rangle, \ldots, |S_i\tilde{\pi}_i^{(k+l)}\rangle$, using the quantum hit-and-run walk ;
- Invert S_i to get $k+l$ (approximate) copies of the stationary distribution $|\pi_i\rangle$ for use in the next iteration;

Compute an estimate $\widetilde{\mathrm{Vol}(\mathrm{K}')} = n! v_n (2n)^{-(n+1)} \tilde{V}_1 \cdots \tilde{V}_m$ of the volume of K′, where v_n is the volume of the n-dimensional unit ball;

Use $\widetilde{\mathrm{Vol}(\mathrm{K}')}$ to estimate the volume of K as $\widetilde{\mathrm{Vol}(\mathrm{K})}$ (Section 3.4.3.1).

ϵ^2/n, we have

$$(1-\epsilon^2)\mathbb{E}_{\mathrm{K}'}[|X-\bar{X}|]^2 \leq \int_{\mathrm{K}'} \frac{e^{-a_m x_0}|x-\bar{x}|^2}{Z(a_m)}\,\mathrm{d}x \leq 2n; \tag{3.4.62}$$

thus $\mathbb{E}_{\mathrm{K}'}[|X-\bar{X}|]^2 \leq \frac{2n}{1-\epsilon^2}$. From [199, Lemma 3.3], all but an ϵ-fraction of the body is contained in a ball of radius $O(\sqrt{n})$. Combined with our assumption $B_2(0,1) \subseteq \mathrm{K}'$,

this shows that S_{m+1} puts the convex body K′ in well-rounded position.

3.5 Quantum lower bound for volume estimation

In this section, we prove a quantum query lower bound on volume estimation.

Theorem 3.5.1. *Suppose* $0 < \epsilon < \sqrt{2} - 1$. *Estimating the volume of* K *with multiplicative precision* ϵ *requires* $\Omega(\sqrt{n})$ *quantum queries to the membership oracle* O_{K} *defined in (3.1.3).*

Proof. We prove Theorem 3.5.1 by reduction from search. In the search problem, we are given an oracle $O_s\colon\ |i,b\rangle \mapsto |i,b\oplus s_i\rangle$ for an input n-bit string $s = (s_1,\ldots,s_n) \in \{0,1\}^n$, and the task is to find an index i such that $s_i = 1$. It is well known that the bounded-error quantum query complexity of this problem is $\Omega(\sqrt{n})$ [46].

To establish an $\Omega(\sqrt{n})$ lower bound for volume estimation, for an n-bit string $s \in \{0,1\}^n$ with Hamming weight $|s|_{\mathrm{Ham}} \le 1$, we consider $\mathrm{K} = \bigtimes_{i=1}^{n}[0, 2^{s_i}]$. The volume of K is $2^{|s|_{\mathrm{Ham}}} \in \{1,2\}$, and membership in K is determined by

$$\mathrm{MEM}_s(x) := \begin{cases} 1 & \text{if for each } i \in [n], 0 \le x_i \le 2^{s_i}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.1}$$

The corresponding membership oracle O_{K} (defined in (3.1.3)) can be simulated by querying O_s using Algorithm 3.5.

[14] Similar to Lemma 3.4.3, we do not directly measure the states; instead we use a quantum circuit to (classically) compute the affine transformation S_i and apply it to the convex body coherently for the next iteration. Note that the quantum register holding the affine transformation will be in some superposition, but by using $O(\log n)$ copies and taking the mean (as in Lemma 3.4.3), the amplitude of the correct affine transformation will be arbitrarily close to 1.

Algorithm 3.5: Simulating $\textsc{mem}_s$ with one query to O_s.

Input: A vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$.
Output: $\textsc{mem}_s(x)$.

1 **for** $i = 1, \ldots, n$ **do**
2 **if** $x_i > 2$ or $x_i < 0$ **then** Return 0;
3 Set $y_i = 1$ if $x_i > 1$ and 0 otherwise;
4 **if** $|y|_{\text{Ham}} \geq 1$ **then** Return 0;
5 **else**
6 **if** $|y|_{\text{Ham}} = 1$ **then** Find i such that $y_i = 1$. Return $O_s(i)$;
7 **else** Return 1;
8 Return $O_s(i)$;

We now prove that for any positive integer k and $s \in \{0,1\}^n$ with $|s|_{\text{Ham}} \leq 1$, if there is a k-query algorithm that computes the volume with access to $\textsc{mem}_s$, then there is a k-query algorithm for deciding whether $|s|_{\text{Ham}} > 0$ with access to O_s. We first show that Algorithm 3.5 simulates the oracle $\textsc{mem}_s$. In the for loop of Line 1, we know that $y_i = 1$ if and only if $1 < x_i \leq 2$, which is inside the convex body if $s_i = 1$. The case $|y|_{\text{Ham}} > 1$ implies that there exist two distinct coordinates i, j such that $x_i, x_j > 1$, which implies that x lies outside the convex body. Now we are left with the cases $|y|_{\text{Ham}} = 1$ or 0. In Line 6, $y_i = 1$ implies $1 < x_i \leq 2$, which lies in the convex body if and only if $s_i = O_s(i) = 1$. Also, $|y| = 0$ implies that for every coordinate i, $0 \leq x_i \leq 1$, which lies in the body for all s.

Finally, if there is a k-query algorithm that computes an estimate $\widetilde{\text{Vol}(\text{K})}$ of the volume of K up to multiplicative precision $0 < \epsilon < \sqrt{2} - 1$, then $s = \lceil \log_2 \widetilde{\text{Vol}(\text{K})} \rfloor$ where $\lceil \cdot \rfloor$ returns the nearest integer. This immediately gives a k-query algorithm that decides whether $|s|_{\text{Ham}} > 0$. Since there is an $\Omega(\sqrt{n})$ quantum query lower bound for this task, the $\Omega(\sqrt{n})$ lower bound on volume estimation follows. □

3.6 Conclusions and discussion

In this chapter, we present a quantum algorithm that estimates the volume of an n-dimensional convex body within multiplicative error ϵ using $\tilde{O}(n^3 + n^{2.5}/\epsilon)$ queries to a membership oracle and $\tilde{O}(n^5 + n^{4.5}/\epsilon)$ additional arithmetic operations. For comparison, the best known classical algorithm [88, 201] uses $\tilde{O}(n^4 + n^3/\epsilon^2)$ queries and $\tilde{O}(n^6 + n^5/\epsilon^2)$ additional arithmetic operations. To the best of our knowledge, this is the first quantum speedup for volume estimation. Our algorithm is based on a refined framework for speeding up simulated annealing algorithms that might be of independent interest. This framework applies in the setting of "Chebyshev cooling", where the solution is expressed as a telescoping product of ratios, each having bounded variance. We develop several novel techniques when implementing our framework, including a theory of continuous-space quantum walks with rigorous bounds on discretization error.

Error analysis of discretized hit-and-run walks. To implement the quantum hit-and-run walk on a digital quantum computer, we also propose a *discretized hit-and-run walk* and provide rigorous bounds on the discretization error in Section 5 of [66]. Specifically, we prove a lower bound on the conductance of a discrete hit-and-run walk that approximates the continuous hit-and-run walk. Our proof addresses a gap in previous (classical) studies by giving a rigorous analysis of discretization, which might be of independent interest to the classical algorithm design community.

The basic idea of the discretization is to represent the coordinates with ra-

tional numbers, discretizing the space $\mathbb{R}^n$. We approximate K by the set of discretized points that lie within it and define the Markov chain on these points. For the hit-and-run walk, we use a two-level discretization: the hit-and-run process is performed with a coarser discretization and then a point in a finer discretization of the coarse grid is chosen uniformly at random as the actual point to jump to. This ensures that the starting and ending points (in the coarser discretization) of one jump are far from the boundary so that a small perturbation does not change the length of the chord induced by the two points significantly. Then the discrete conductance can be bounded by bounding the distance between the discrete and continuous transition probabilities as well as the distance between the discrete and continuous subset measures. We further prove that the quantum gate complexity of implementing the discretized quantum hit-and-run walk is $\tilde{O}(n)$, the same overhead as for implementing classical hit-and-run walks.

For the details of the discretized hit-and-run walk, please refer to Section 5 of [66].

Open questions. Our work leaves several natural open questions for future investigation. In particular:

- Can we improve the complexity of our quantum volume estimation algorithm? The current gap between the upper bound $\tilde{O}(n^3+n^{2.5}/\epsilon)$ and the lower bound $\Omega(\sqrt{n})$ is large; possible improvements might result from designing a shorter simulated annealing schedule, giving better analysis of the conductance of the hit-and-run walk, or even using other types of walks.

- Can we prove better quantum query lower bounds on volume estimation? Note that classically there is an $\tilde{\Omega}(n^2)$ query lower bound [231].

- Can we give faster quantum algorithms for volume estimation in some special circumstances? For instance, volume estimation of well-rounded convex bodies only takes $\tilde{O}(n^3)$ classical queries [88] (see Section 3.1.3), and the volume of polytopes with m faces can be estimated with only $\tilde{O}(mn^{2/3})$ classical queries [187]. Specifically, it is a natural question to ask whether the ball walk in [88] or the Riemannian Hamiltonian Monte Carlo (RHMC) method in [187] can be implemented by continuous-space quantum walks.

- Can we apply our simulated annealing framework to solve other problems? As a concrete example, it may be of interest to check whether our framework can recover the results of Ref. [139] on estimating the partition functions in counting problems.

- In general, can we have a better understanding about the convergence of quantum dynamics? It is well-known that the stationary distributions of many common dynamics are log-concave, including Metropolis sampling, Langevin dynamics, etc. But only until recently rigorous analysis of their convergence rate was given, starting from the convergence analysis of Langevin dynamics by Dalalyan [92]. Quantumly, such analysis might rely on better understanding of open quantum systems, for instance our previous work [82] showed how to efficiently simulate sparse Markovian open systems.

Chapter 4: Semidefinite Programs[1]

Chapter 2 and Chapter 3 have been devoted to quantum algorithms for convex problems with *implicit oracle inputs*. In this chapter, we focus on an important class of convex optimization problems with *matrices as explicit inputs*: semidefinite programs.

4.1 Introduction

Motivation. Semidefinite programming has been a central topic in the study of mathematical optimization, theoretical computer science, and operations research in the last decades. It has become an important tool for designing efficient optimization and approximation algorithms. The power of semidefinite programs (SDPs) lies in their generality (that extends the better-known linear programs (LPs)) and the fact that they admit polynomial-time solvers.

There is a rich classical literature on solving SDPs. Ellipsoid methods gave the first polynomial-time SDP solvers [126, 170], and the complexities of the SDP solvers had been subsequently improved by the interior-point method [216] and the cutting-plane method [22, 207]; see also the survey paper [267]. The current state-

[1]This chapter is based on the paper [55] under the permission of all the authors.

of-the-art SDP solver [184] runs in time $\tilde{O}(m(m^2+n^\omega+mn^2)\,\mathrm{poly}(\log 1/\varepsilon))$, where n and s are the dimension and row sparsity of the input matrices respectively, m is the number of constraints, ε is the accuracy of the solution, and $\omega < 2.373$ is the exponent of matrix multiplication. On the other hand, if we tolerate polynomial dependence in $1/\varepsilon$, Arora and Kale [29] gave an SDP solver with better complexities in m and n: $\tilde{O}(mn^2(R\tilde{R}/\varepsilon)^4 + n^2(R\tilde{R}/\varepsilon)^7)$, where R and $\tilde{R}$ are upper bounds on the ℓ_1-norm of the optimal primal and dual solutions, respectively (see more details in [24]). This was subsequently improved to $\tilde{O}(m/\varepsilon^2+n^2/\varepsilon^{2.5})$ by Garber and Hazan [114, 115] when $R, \tilde{R} = 1$ and $b_i = 0$ in (6.1.2) for all $i \in [m]$; as a complement, [114] also established a lower bound $\Omega(m/\varepsilon^2 + n^2/\varepsilon^2)$ under the same assumption.

It is natural to ask whether quantum computers can have advantage in solving this important optimization problem. In Ref. [56], Brandão and Svore provided an affirmative answer, giving a quantum algorithm with worst-case running time $\tilde{O}(\sqrt{mn}s^2(R\tilde{R}/\varepsilon)^{32})$. This is a *polynomial* speed-up in m and n comparing to the two state-of-the-art classical SDP-solvers [29, 184], and beating the classical lower bound of $\Omega(m+n)$ [56]. The follow-up work by van Apeldoorn et al. [24] improved the running time giving a quantum SDP solver with complexity $\tilde{O}(\sqrt{mn}s^2(R\tilde{R}/\epsilon)^8)$. In terms of limitations, Ref. [56] proved a quantum lower bound $\Omega(\sqrt{m}+\sqrt{n})$ when $R, \tilde{R}, s, \epsilon$ are constants; stronger lower bounds can be proven if R and/or $\tilde{R}$ scale with m and n [24]. We note all these results are shown in an input model in which there is an oracle for the entry of each of the input matrices (see Oracle 1 below for a formal definition).

In this paper, we investigate quantum algorithms for SDPs (i.e., quantum

SDP solvers) further in the following two perspectives: (1) the best dependence of parameters, especially the dimension n and the number of constraints m; (2) whether there is any reasonable alternative input model for quantum SDP solvers and what is its associated complexity. To that end, let us first formulate the precise SDP instance in our discussion.

The SDP approximate feasibility problem. We work with the SDP approximate feasibility problem formulated as follows (see Section 4.2 for details): Given an $\epsilon > 0$, m real numbers $a_1, \ldots, a_m \in \mathbb{R}$, and Hermitian $n \times n$ matrices $A_1, \ldots, A_m$ where $-I \preceq A_i \preceq I, \forall\, j \in [m]$, define the convex region $\mathcal{S}_\epsilon$ as all X such that

$$\mathrm{Tr}[A_i X] \leq a_i + \epsilon \quad \forall\, i \in [m];\ X \succeq 0, \mathrm{Tr}[X] = 1. \tag{4.1.1}$$

For approximate feasibility testing, it is required that either (1) If $\mathcal{S}_0 = \varnothing$, output fail; or (2) If $\mathcal{S}_\epsilon \neq \varnothing$, output an $X \in \mathcal{S}_\epsilon$. Throughout the paper, we denote by n the the dimension of the matrices, m the number of constraints, and ϵ the (additive) error of the solution. For Hermitian matrices A and B, we denote $A \preceq B$ if $B - A$ is positive semidefinite, and $A \succeq B$ if $A - B$ is positive semidefinite. We denote I_n to be the $n \times n$ identity matrix.

There are a few reasons that guarantee our choice of approximate SDP feasibility problem do not lose generality: (1) first, it is a routine[2] to reduce general

[2]To see why this is the case, for any general SDP problem, one can guess a candidate value (e.g., c_0) for the objective function (e.g., $\mathrm{Tr}(CX)$ and assume one wants to maximize $\mathrm{Tr}(CX)$) and convert it into a constraint (e.g., $\mathrm{Tr}(CX) \geq c_0$). Hence one ends up with a feasibility problem and the candidate value c_0 can then be found via binary search with $O(\log(1/\epsilon))$ overhead when $\mathrm{Tr}(CX) \in [-1, 1]$.

optimization SDP problems to the feasibility problem; (2) second, for general feasible solution $X \succeq 0$ with width bound $\mathrm{Tr}(X) \leq R$, there is a procedure[3] to derive an equivalent SDP feasibility instance with variable $\hat{X}$ s.t. $\mathrm{Tr}(\hat{X}) = 1$. Note, however, the change of ϵ to ϵ/R in this conversion. Also note one can use an approximate feasibility solver to find a strictly feasible solution, by changing ϵ to $\epsilon/R\tilde{R}$ (see Lemma 18 of [56]). The benefit of our choice of (4.1.1) is its simplicity in presentation, which provides a better intuition behind our techniques and an easy adoption of our SDP solver in learning quantum states. In contrast to Ref. [24], we do not need to formulate the dual program of Eq. (4.1.1) since our techniques do not rely on it. We will elaborate more on these points in Section 4.1.4.

4.1.1 Quantum SDP solvers with optimal dependence on m and n

Existing quantum SDP solvers [24, 56] have close-to-optimal dependence on some key parameters but poor dependence on others. Seeking optimal parameter dependence has been an important problem in the development of classical SDP solvers and has inspired many new techniques. It is thus well motivated to investigate the optimal parameter dependence in the quantum setting. Our first contribution is the construction of a quantum SDP solver with the optimal dependence on m and n in the (plain) input model (1.3.3) as used by [24, 56], given as follows:

Oracle 1 (Plain model for A_j)**.** *A quantum oracle, denoted $\mathcal{P}_A$, such that given the indices $j \in [m]$, $k \in [n]$ and $l \in [s]$, computes a bit string representation of the l-th*

[3]The procedure goes as follows: (a) scale down every constraint by a factor R and let $X' = X/R$ (thus $\mathrm{Tr}(X') \leq 1$) (b) let $\hat{X} = \mathrm{diag}\{X, w\}$ be a block-diagonal matrix with X in the upper-left corner and a scaler w in the bottom-right corner. It is easy to see that $\mathrm{Tr}(\hat{X}) = 1 \iff \mathrm{Tr}(X) \leq 1$.

non-zero element of the k-th row of A_j, i.e. the oracle performs the following map:

$$|j,k,l,z\rangle \to |j,k,l,z \oplus (A_j)_{kf_{jk}(l)}\rangle, \tag{4.1.2}$$

with $f_{jk} : [r] \to [N]$ a function (parametrized by the matrix index j and the row index k) which given $l \in [s]$ computes the column index of the l-th nonzero entry.

Before we move on to our main result, we will define two primitives which will appear in our quantum SDP solvers. Our main result will also be written in terms of the cost for each primitive.

Definition 4.1.1 (trace estimation)**.** *Assume that we have an s-sparse $n \times n$ Hermitian matrix H with $\|H\| \leq \Gamma$ and a density matrix ρ. Then we define $\mathcal{S}_{\mathrm{Tr}}(s,\Gamma,\epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(s,\Gamma,\epsilon)$ as the number of copies of ρ and the time complexity (in terms of oracle call and number of gates) of using the plain model (Oracle 1) for H, respectively, such that one can compute $\mathrm{Tr}[H\rho]$ with additive error ϵ with success probability at least $2/3$.*

Definition 4.1.2 (Gibbs sampling)**.** *Assume that we have an s-sparse $n \times n$ Hermitian matrix H with $\|H\| \leq \Gamma$. Then we define $\mathcal{T}_{Gibbs}(s,\Gamma,\epsilon)$ as the complexity of preparing the Gibbs state $\frac{e^{-H}}{\mathrm{Tr}[e^{-H}]}$ with additive error ϵ using the plain model (Oracle 1) for H.*

Our main result is as follows.

Theorem 4.1.1 (informal; see Theorem 4.4.1)**.** *In the plain input model (Oracle 1), for any $0 < \epsilon < 1$, there is a quantum SDP solver for the feasibility problem (4.1.1)*

using $\frac{s}{\epsilon^4}\tilde{O}\big(\mathcal{S}_{\mathrm{Tr}}\big(\frac{s}{\epsilon^2},\frac{1}{\epsilon},\epsilon\big)\mathcal{T}_{Gibbs}\big(\frac{s}{\epsilon^2},\frac{1}{\epsilon},\epsilon\big)+\sqrt{m}\mathcal{T}_{\mathrm{Tr}}\big(\frac{s}{\epsilon^2},\frac{1}{\epsilon},\epsilon\big)\big)$ *quantum gates and queries to Oracle 1, where* s *is the sparsity of* $A_j, j\in[m]$.

When combined with specific instantiation of these primitives (i.e., in our case, we directly make use of results on $\mathcal{S}_{\mathrm{Tr}}(s,\Gamma,\epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(s,\Gamma,\epsilon)$ from [56], and results on $\mathcal{T}_{\mathrm{Gibbs}}(s,\Gamma,\epsilon)$ from [229]), we end up with the following concrete parameters:

Corollary 4.1.1 (informal; see Corollary 4.4.1)**.** *In the plain input model (Oracle 1), for any* $0<\epsilon<1$*, there is a quantum SDP solver for the feasibility problem (4.1.1) using* $\tilde{O}(s^2(\frac{\sqrt{m}}{\epsilon^{10}}+\frac{\sqrt{n}}{\epsilon^{12}}))$ *quantum gates and queries to Oracle 1, where* s *is the sparsity of* $A_j, j\in[m]$.

Comparing to prior art, our main contribution is to decouple the dependence on m and n, which was $O(\sqrt{mn})$ and now becomes $O(\sqrt{m}+\sqrt{n})$. Note that the $(\sqrt{m}+\sqrt{n})$ dependence is optimal due to the quantum lower bound proven in [56].

Remark 4.1.1. *Even though our result achieves the optimal dependence on* m *and* n*, it is nontrivial to obtain quantum speed-ups by directly applying our quantum SDP solvers to SDP instances from classical combinatorial problems. The major obstacle is the poly-dependence on* $1/\epsilon$*, whereas, for interesting SDP instances such as Max-Cut,* $1/\epsilon$ *is linear in* n*. In fact, the general framework of the classical Arora-Kale SDP solver also suffers from the poly-dependence on* $1/\epsilon$ *and cannot be applied directly either. Instead, one needs to specialize the design of SDP solvers for each instance to achieve better time complexity.*

Extending this idea to quantum seems challenging. One difficulty is that known classical approaches require explicit information of intermediate states, which re-

quires $\Omega(n)$ time and space even to store. It is not clear how one can directly adapt classical approaches on intermediate states when stored as amplitudes in quantum states, which is the case for our current SDP solvers. It seems to us that a resolution of the problem might require an independent tool beyond the scope of this paper. We view this as an important direction for future work.

However, our quantum SDP solvers are sufficient for instances with mild $1/\epsilon$, which are natural in the context of quantum information, such as learnability of the quantum state problem (elaborated in Section 4.1.5) as well as examples in [23]. For those cases, we do establish a quantum speed-up as any classical algorithm needs at least linear time in n and/or m.

4.1.2 Quantum SDP solvers with quantum inputs

Given the optimality of the algorithm presented before (in terms of m and n), a natural question is to ask about the existence of alternative input models, *which can be justified for specific applications, and at the same time allows more efficient quantum SDP solvers.* This is certainly a challenging question, but we can get inspiration from the application of SDPs in quantum complexity theory (e.g., Refs. [131, 150]) and quantum information (e.g., Refs. [1, 3]). In these settings, input matrices of SDP instances, with dimension 2^ℓ, are typically quantum states and/or measurements generated by poly(ℓ)-size circuits on ℓ qubits. For the sake of these applications, it might be reasonable to equip quantum SDP solvers with the ability to leverage these circuit information, rather than merely allowing access to

the entries of the input matrices.

In this paper, we propose a *truly* quantum input model in which we can construct quantum SDP solvers with running time only *poly-logarithmic* in the dimension. We note that such proposal was mentioned in an earlier version of Ref. [56], whose precise mathematical form and construction of quantum SDP solvers were unfortunately incorrect, and later removed. Note that since we consider a non-standard input model in this section, our results are incomparable to those in the plain input model. We argue for the relevance of our quantum input model, by considering an applications of the framework to the problem of learning quantum states in Section 4.1.5.

Quantum input model. Consider a specific setting in which we are given decompositions of each A_j: $A_j = A_j^+ - A_j^-$, where $A_j^+, A_j^- \succeq 0$. (For instance, a natural choice is to let A_j^+ (resp. A_j^-) be the positive (resp. negative) part of A.)

Oracle 2 (Oracle for traces of A_j)**.** *A quantum oracle (unitary), denoted O_{Tr} (and its inverse $O_{\mathrm{Tr}}^\dagger$), such that for any $j \in [m]$,*

$$O_{\mathrm{Tr}}|j\rangle|0\rangle|0\rangle = |j\rangle|\operatorname{Tr}[A_j^+]\rangle|\operatorname{Tr}[A_j^-]\rangle, \tag{4.1.3}$$

where the real values $\operatorname{Tr}[A_j^+]$ and $\operatorname{Tr}[A_j^-]$ are encoded into their binary representations.

Oracle 3 (Oracle for preparing A_j)**.** *A quantum oracle (unitary), denoted O (and*

its inverse $O^\dagger$), which acts on $\mathbb{C}^m \otimes (\mathbb{C}^n \otimes \mathbb{C}^n) \otimes (\mathbb{C}^n \otimes \mathbb{C}^n)$ such that for any $j \in [m]$,

$$O|j\rangle|0\rangle|0\rangle = |j\rangle|\psi_j^+\rangle|\psi_j^-\rangle, \tag{4.1.4}$$

where $|\psi_j^+\rangle, |\psi_j^-\rangle \in \mathbb{C}^n \otimes \mathbb{C}^n$ are any purifications of $\frac{A_j^+}{\mathrm{Tr}[A_j^+]}, \frac{A_j^-}{\mathrm{Tr}[A_j^-]}$, respectively.

Oracle 4 (Oracle for a_j)**.** *A quantum oracle (unitary), denoted O_a (and its inverse $O_a^\dagger$), such that for any $j \in [m]$,*

$$O_a|j\rangle|0\rangle = |j\rangle|a_j\rangle, \tag{4.1.5}$$

where the real value a_j is encoded into its binary representation.

Throughout the paper, let us assume that A_j has rank at most r for all $j \in [m]$ and $\mathrm{Tr}[A_j^+] + \mathrm{Tr}[A_j^-] \leq B$. The parameter B is therefore an upper bound to the trace-norm of all input matrices which we assume is given as an input of the problem. Similar to the plain input model, we will define the same two primitives and their associated costs in the quantum input model.

Definition 4.1.3 (trace estimation)**.** *We define $\mathcal{S}_{\mathrm{Tr}}(B, \epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(B, \epsilon)$ as the sample complexity of a state $\rho \in \mathbb{C}^{n \times n}$ and the gate complexity of using the quantum input oracles (Oracle 2, Oracle 3, Oracle 4), respectively, for the fastest quantum algorithm that distinguishes with success probability at least $1 - O(1/m)$ whether for a fixed $j \in [m]$, $\mathrm{Tr}(A_j\rho) > a_j + \epsilon$ or $\mathrm{Tr}(A_j\rho) \leq a_j$.*

Definition 4.1.4 (Gibbs sampling)**.** *Assume that $K = K^+ - K^-$, where $K^\pm = \sum_{j \in S} c_j A_j^\pm$, $c_j > 0$, $S \subseteq [m]$ and $|S| \leq \Phi$, and that K^+, K^- have rank at*

most r_K. *Moreover, assume that* $\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-) \leq B_K$ *for some* B_K. *Then we define* $\mathcal{T}_{Gibbs}(r_K, \Phi, B_K, \epsilon)$ *as the gate complexity of preparing the Gibbs state* $\rho_G = \exp(-K)/\mathrm{Tr}(\exp(-K))$ *to* ϵ *precision in trace distance using Oracle 2, Oracle 3, and Oracle 4.*

Our main result in the quantum input model is as follows.

Theorem 4.1.2 (informal; see Theorem 4.5.1)**.** *For any* $\epsilon > 0$*, there is a quantum algorithm using at most* $\frac{1}{\epsilon^2}\tilde{O}\big(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)\mathcal{T}_{Gibbs}\big(\frac{r}{\epsilon^2}, \frac{1}{\epsilon^2}, \frac{B}{\epsilon}, \epsilon\big) + \sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)\big)$ *quantum gates and queries to Oracle 2, Oracle 3, and Oracle 4 for the approximate SDP feasibility problem.*

Contrary to the plain model setting, the quantum input model is a completely new setting so that we have to construct these two primitive by ourselves. In particular, we give a construction of trace estimation in Lemma 4.5.2 with $\mathcal{S}_{\mathrm{Tr}}(B,\epsilon) = \mathcal{T}_{\mathrm{Tr}}(B,\epsilon) = O(B^2 \log m/\epsilon^2)$ and a construction of Gibbs sampling in Lemma 4.5.4 with $\mathcal{T}_{\mathrm{Gibbs}}(r_K, \Phi, B_K, \epsilon) = O(\Phi \cdot \mathrm{poly}(\log n, r_K, B_K, \epsilon^{-1}))$. As a result,

Corollary 4.1.2 (informal; see Corollary 4.5.1)**.** *For any* $\epsilon > 0$*, there is a quantum algorithm using at most* $(\sqrt{m} + \mathrm{poly}(r)) \cdot \mathrm{poly}(\log m, \log n, B, \epsilon^{-1})$ *quantum gates and queries to Oracle 2, Oracle 3, and Oracle 4 for the SDP feasibility problem.*

We also show the square-root dependence on m is also optimal by establishing the following result:

Theorem 4.1.3 (lower bound on Corollary 4.1.2)**.** *There exists an SDP feasibility testing problem such that* $B, r, \epsilon = \Theta(1)$*, and solving the problem requires* $\Omega(\sqrt{m})$ *calls to Oracle 2, Oracle 3, and Oracle 4.*

Comparison between the plain model and the quantum input model. In the quantum input model (Oracle 2, Oracle 3, and Oracle 4), our quantum SDP solver has a *poly-logarithmic* dependence on n (but polynomial in r) and a *square-root* dependence on m, while in the plain input model (Oracle 1), the dependence on n needs to be $\Omega(\sqrt{n})$ [56]. It is also worth mentioning that our quantum SDP solver in Corollary 4.1.2 does *not* assume the *sparsity* of A_i's, which are crucial for the quantum SDP solvers with the plain model (such as Corollary 4.1.1 and Refs. [24, 56]). This is because the quantum input models provide an alternative way to address the technical difficulty that was resolved by the sparsity condition (namely efficient algorithms for Hamiltonian evolution associated with the input matrices of the SDP).

Comparison between quantum and classical input models. The poly-log dependence on n in Corollary 4.1.2 is intriguing and suggests that quantum computers might offer exponential speed-ups for some SDP instances. However, one has to be cautious as the input model we consider is inherently quantum, so it is incomparable to classical SDP solvers. As suggested to us by Aram Harrow (personal communication), we could consider a classical setting in which we get as input all inner products between all eigenvectors of the input matrices. Then in that case one could solve the problem classically in time $\mathrm{poly}(r, m, 1/\epsilon)$ (essentially using Jaynes's principle which will be discussed in Section 4.1.5 to reduce the problem to a SDP of dimension $\mathrm{poly}(r)$). We have not formalized this approach, and there seems to be some technical problems doing so when the input matrices have close-by eigenvalues.

However, Harrow's observation shows the importance of justifying the input model in terms of natural applications to argue for the relevance of the run time obtained. We present one of its application in Section 4.1.5; more applications are given in [23].

Furthermore, several quantum-inspired classical algorithms were recently proposed originated from Tang [256]. Such classical algorithms assume the following sampling access:

Definition 4.1.5 (Sampling access)**.** *Let $A \in \mathbb{C}^{n\times n}$ be a matrix. We say that we have the* sampling access *to A if we can*

1. *sample a row index $i \in [n]$ of A such that $\Pr[i] = \frac{\|A_{i\cdot}\|^2}{\|A\|_F^2}$, and*[4]

2. *for all $i \in [n]$, sample an index $j \in [n]$ such that $\Pr[j] = \frac{|A_{ij}|^2}{\|A_{i\cdot}\|^2}$,*

with time and query complexity $O(\mathrm{poly}(\log n))$ for each sampling.

In particular, we notice that Ref. [77] recently gave a classical SDP solver for (4.1.1) with complexity $O(m \cdot \mathrm{poly}(\log n, r, \epsilon^{-1}))$, given the above sampling access to $A_1, \ldots, A_m$. We point out that this result is incomparable to Corollary 4.1.2 because the sampling access (Definition 4.1.5) and our quantum state model (Oracle 2, Oracle 3, and Oracle 4) are incomparable. Nevertheless, it reminds us that under various input models, the speedup of quantum SDP solvers (compared to their classical counterparts) can also vary.

[4]Here $\|A\|_F$ is the Frobenius norm of A and $\|A_{i\cdot}\|$ is the ℓ_2 norm of the i^{th} row of A.

4.1.3 Related works on quantum SDP solvers

Previous quantum SDP solvers [24, 56] focused on the plain input model. A major contribution of ours is to improve the dependence $\tilde{O}(\sqrt{mn})$ to $\tilde{O}(\sqrt{m}+\sqrt{n})$ (ignoring dependence on other parameters) which is optimal given the lower bound $\Omega(\sqrt{m}+\sqrt{n})$ in [56]. To that end, we have also made a few technical contributions, including bringing in a new SDP solving framework and a fast version of quantum OR lemma (Lemma 4.3.2), which will be elaborated in Section 4.1.4.

The quantum input model was briefly mentioned in an earlier version of [56]. The construction of quantum SDP solvers under the quantum input model therein was unfortunately incorrect. We provide the first rigorous mathematical formulation of the quantum input model and its justification in the context of learning quantum states (see Section 4.1.5). We also provide a construction of quantum SDP solvers in this model with a rigorous analysis. Moreover, we construct the first Gibbs state sampler with quantum inputs (Lemma 4.5.4).

4.1.4 Techniques

At a high level, and in similarity to Refs. [24, 56], our quantum SDP solver can be seen as a "quantized" version of classical SDP solvers based on the matrix multiplicative weight (MMW) method [28]. In particular, we will leverage quantum Gibbs samplers as the main source of quantum speed-ups. In Refs. [24, 56], quantum Gibbs samplers with quadratic speed-ups (e.g., [84, 229]) have been exploited to replace the classical Gibbs state calculation step in [28]. Because the number of

iterations in MMW is poly-logarithmic in terms of the input size, the use of quantum Gibbs samplers, together with a few other tricks, leads to the overall quadratic quantum speed-up.

However, there are a few key differences (our major technical contributions) which are essential for our improvements.

Zero-sum game approach for MMW. Our quantum SDP solvers do not follow the primal-dual approach in Arora-Kale's SDP solver [29] which is the classical counterpart of previous quantum SDP solvers [24, 56]. Instead, we follow a zero-sum game framework to solve SDP feasibility problems, which is also based on the MMW method (details in Section 4.2). This framework has appeared in the classical literature (e.g., [142]) and has already been used to in semidefinite programs of relevance in quantum complexity theory (e.g., [131, 181, 274]). Let us briefly describe how the zero-sum game framework works when solving the SDP feasibility problem (4.1.1).

Assume there are two players. Player 1 wants to provide a feasible $X \in \mathcal{S}_\epsilon$. Player 2, on the other side, wants to find any violation of any proposed X, which can be formulated as follows.

Oracle 5 (Search for violation)**.** *Inputs a density matrix X, outputs an $i \in [m]$ such that $\mathrm{Tr}(A_i X) > a_i + \epsilon$. If no such i exists, output "FEASIBLE".*

If the original problem is feasible, there exists a feasible point X_0 (provided by Player 1) such that there is no violation of X_0 that can be found by Player 2 (i.e., Oracle 5). This actually refers to an *equilibrium* point of the zero-sum game, which

can also be approximated by the matrix multiplicative weight update method [28].

We argue that there are a few advantages of adopting this framework. One prominent example is its simplicity, which perhaps provides more intuition than the primal-dual approach. Together with our choice of the approximate feasibility problem, our presentation is simple both conceptually and technically (indeed, the simplicity of this framework has led to the development of the fast quantum OR lemma, another main technical contribution of ours.) Another example is that the zero-sum game approach does not make use of the dual program of SDPs and thus there is no dependence on the size of any dual solution. The game approach also admits an intuitive application of our SDP solvers to learning quantum states Section 4.1.5, which coincides with the approach adopted by [181] in a similar context.

One might wonder whether the simplicity of this framework will restrict the efficiency of SDP solvers. As indicated by the independent work of van Apeldoorn and Gilyén [23] which has achieved the same complexity of quantum SDP solvers following both the primal-dual approach and the zero-sum approach, we conclude that it is not the case at least up to our current knowledge.

Fast quantum OR lemma. We now outlines what is the main idea to find a solution to Oracle 5 efficiently. Roughly speaking, the idea behind previous quantum SDP solvers [24, 56] when applied to this context was to generate a new copy of a quantum state X for each time one would query the expectation value of one of the input matrices on it. The cost of generating X (i.e., Gibbs sampling) is $O(\sqrt{n})$ (ignoring the dependence on other parameters) and one can use a Grover-search-

like approach to test for m constraints with $O(\sqrt{m})$ iterations. The resultant cost is then $O(\sqrt{mn})$. Our key observation is to leverage the quantum OR lemma [138] to detect a single violation with only a single copy of X.

At a high level, given a single copy of any state ρ and m projections $\Lambda_1, \ldots, \Lambda_m$, the quantum OR lemma describes a procedure to distinguish between the case that $\exists\, i \in [m]$ s.t. $\mathrm{Tr}[\rho\Lambda_i]$ is very large, or $\frac{1}{m}\sum_{i=1}^{m} \mathrm{Tr}[\rho\Lambda_i]$ is very small. It is not hard to see that with some gap-amplification step and a search-to-decision reduction, the above procedure will output a violation i^* if any. By using quantum OR lemma, one can already decouple the cost of generating X and the number of iterations in violation-detection.

Unfortunately, Ref. [138] has only been focusing on the use of a single copy of ρ, while its gate complexity is $O(m)$ for m projections. To optimize the gate complexity, we develop the following fast implementation of the quantum OR lemma with gate complexity $O(\sqrt{m})$, using ideas from the fast amplification technique in [211]. Overall, this leads to a complexity of $O(\sqrt{m} + \sqrt{n})$.

Lemma 4.1.1 (informal; see Lemma 4.3.2)**.** *Let $\Lambda_1, \ldots, \Lambda_m$ be projections, and fix parameters $0 < \varepsilon \leq 1/2$ and $\varphi, \xi > 0$. Let ρ be a state such that either $\exists\, j \in [m]$ $\mathrm{Tr}[\rho\Lambda_j] \geq 1 - \varepsilon$, or $\frac{1}{m}\sum_{j=1}^{m} \mathrm{Tr}[\rho\Lambda_j] \leq \varphi$. There is a test using one copy of ρ and $O(\xi^{-1}\sqrt{m}(p + \mathrm{poly}(\log m)))$ operations such that: in the former case, accepts with probability at least $(1-\varepsilon)^2/4 - \xi$; in the latter case, accepts with probability at most $3\varphi m + \xi$.*

The dependence on m is also tight, as one can easily embed Grover search into

this problem.

Gibbs sampler with quantum inputs. To work with the quantum input model, as our main technical contribution, we construct the first quantum Gibbs sampler of low-rank Hamiltonians when given Oracle 2 and Oracle 3:

Theorem 4.1.4 (informal; see Theorem 4.7.1)**.** *Assume the $n \times n$ matrix $K = K^+ - K^-$ and K^+, K^- are PSD matrices with rank at most r_K and $\mathrm{Tr}[K^+] + \mathrm{Tr}[K^-] \leq B$. Given quantum oracles that prepare copies of $\rho^+ = K^+/\mathrm{Tr}(K^+)$, $\rho^- = K^-/\mathrm{Tr}(K^-)$ and estimates of $\mathrm{Tr}(K^+)$, $\mathrm{Tr}(K^-)$, there is a quantum Gibbs sampler that prepares the Gibbs state $\rho_G = \exp(-K)/\mathrm{Tr}(\exp(-K))$ to precision ϵ in trace distance, using $\mathrm{poly}(\log n, r_K, B, \epsilon^{-1})$ quantum gates.*

Our quantum Gibbs sampler has a poly-logarithmic dependence on n and polynomial dependence on the maximum rank of the input matrices, while in the plain input model the dependence of n is $\Theta(\sqrt{n})$ [84, 229]. Our construction deviates significantly from [84, 229]. Because of the existence of copies of ρ^+ and ρ^-, we rely on efficient Hamiltonian simulation techniques developed in quantum principle component analysis (PCA) [195] and its follow-up work in [173]. As a result, we can also get rid of the sparsity assumption which is crucial for evoking results about efficient Hamiltonian simulation into the Gibbs sampling used in [84, 229].

4.1.5 Application: Efficient learnability of quantum states

Problem description. Given many realizations of an experiment producing a quantum state with density matrix ρ, learning an approximate description of ρ is

a fundamental task in quantum information and experimental physics. It refers to *quantum state tomography*, which has been widely used to identify quantum systems. However, to tomograph an ℓ-qubit state ρ (with dimension $n = 2^\ell$), the optimal procedure [132, 220] requires n^2 number of copies of ρ, which is impractical already for relatively small ℓ.

An interesting alternative is to find a description of the unknown quantum state ρ which approximates $\mathrm{Tr}[\rho E_i]$ up to error ϵ for a specific collection of POVM elements $E_1, \ldots, E_m$, where $0 \preceq E_i \preceq I$ and $E_i \in \mathbb{C}^{n\times n}, \forall i \in [m]$. This is an old problem, dating back at least to the work of Jaynes on statistical mechanics [152]. Jaynes's principle (also known as the principle of maximum entropy) shows that there is always a state of the form

$$\frac{\exp\left(\sum_i \lambda_i E_i\right)}{\mathrm{Tr}\left(\exp\left(\sum_i \lambda_i E_i\right)\right)}, \tag{4.1.6}$$

which has the same expectation values on the E_i's as the original state ρ, where the λ_i's are real numbers. In words, there is always a Gibbs state with Hamiltonian given by a linear combination of the E_i's which gives the same expectation values as the state described by ρ. Therefore one can solve the learning problem by finding the right λ_i's (or finding a quantum circuit creating the state in Eq. (4.1.6)).

Applying quantum SDP solvers. By formulating the learning problem in terms of the SDP feasibility problem (with each A_i replaced by E_i) where one looks for a trace unit PSD σ matching the measurement statistics, i.e., $\mathrm{Tr}(\sigma E_i) \approx \mathrm{Tr}(\rho E_i)\ \forall i \in$

$[m]$, we observe that our quantum SDP solvers actually provides a solution to the learning problem with associated speed-ups on m and n.

In fact, our algorithm also outputs each of the λ_i's (only $\mathrm{poly}(\log(mn))/\varepsilon^2$ of them are nonzero, but it suffices for a solution with error ε), as well as a circuit description of the Gibbs state in Eq. (4.1.6) achieving the same expectation values as ρ up to error ε. (This is mainly because the similarity between the matrix multiplicative update method and Jaynes's principle. Compare (4.1.6) and Algorithm 4.1.) In this sense our result can be seen as an *algorithmic* version of Jaynes's principle. We note that a similar idea was adopted by [181] in learning quantum states, although for a totally different purpose (namely proving lower bounds on the size of SDP approximations to constraint satisfaction problems).

It is worthwhile noting that our quantum SDP solvers when applied in this context will output a description of the state ρ in the form of Eq. (4.1.6) which has the same expectation values as ρ on measurements $E_1, \ldots, E_m$ up to error ϵ. This is slightly different from directly outputting estimates of $\mathrm{Tr}(E_i\rho)$ for each $i \in [m]$, which by itself will take $\Omega(m)$ time.

Relevance of the quantum input model. More importantly, we argue that our quantum input model is *relevant* in this setting for low-rank measurements E_i's. Since all $E_i \succeq 0$ by definition, we can consider the following (slightly simplified version of) oracles:

Oracle 2 for traces of E_i: A unitary O_{Tr} such that for any $i \in [m]$, $O_{\mathrm{Tr}}|i\rangle|0\rangle = |i\rangle|\operatorname{Tr}[E_i]\rangle$.

Oracle 3 for preparing E_i: A unitary O such that for any $i \in [m]$, $O|i\rangle\langle i| \otimes |0\rangle\langle 0|O^\dagger = |i\rangle\langle i| \otimes |\psi_i\rangle\langle\psi_i|$, where $|\psi_i\rangle\langle\psi_i|$ is any purification of $E_i/\operatorname{Tr}[E_i]$.

We now show how one can implement this oracle in the case where each E_i is a low rank projector and we have an efficient (with $\operatorname{poly}\log(n)$ many gates) implementation of the measurement. Let the rank of E_i's bounded by r and suppose the measurement operators E_i's are of the form

$$E_i = V_i P_i V_i^\dagger \tag{4.1.7}$$

for polynomial (in $\log(n)$) time circuits V_i, and projectors P_i of the form

$$P_i := \sum_{i=1}^{r_i} |i\rangle \langle i| \tag{4.1.8}$$

with $|i\rangle$ the computational basis and $r_i \leq r$. Then for Oracle 2 we just need to output the r_i's. Oracle 3 can be implemented efficiently (in time $r \operatorname{poly}\log(n)$) by first creating a maximally entangled state between the subspace spanned by P_i and a purification and applying V_i to one half of it. In more detail, consider the following purification of $E_i/\operatorname{Tr}(E_i)$:

$$|\psi_i\rangle := \frac{1}{\sqrt{r_i}} \sum_{i=1}^{r_i} (V_i \otimes I) |i, i\rangle \tag{4.1.9}$$

This can be constructed first by preparing the state $\frac{1}{\sqrt{r_i}}\sum_{i=1}^{r_i}|i,i\rangle$ in time r_i and then applying $V_i \otimes I$ to it (which can be done in time $\operatorname{poly}\log(n)$).

Efficient learning for low rank measurements. By applying our SDP solver in the quantum input model, we obtain that

Theorem 4.1.5 (informal; see Corollary 4.6.1)**.** *For any $\epsilon > 0$, there is a quantum procedure that outputs a description of the state ρ in the form of Eq. (4.1.6) (namely the λ_i's parameters) using at most $\operatorname{poly}(\log m, \log n, r, \epsilon^{-1})$ copies of ρ and at most $\sqrt{m} \cdot \operatorname{poly}(\log m, \log n, r, \epsilon^{-1})$ quantum gates and queries to Oracle 2 and Oracle 3.*

Let us briefly sketch how our SDP solver applies to this setting. Note first that we do not aim to estimate $\operatorname{Tr}(E_i\rho)$ for each $i \in [m]$, which helps us circumvent the $\Omega(m)$ lower bound. What we really want is to generate a state $\tilde{\rho}$ such that $\operatorname{Tr}(E_i\tilde{\rho}) \approx \operatorname{Tr}(E_i\rho)$ for each i. Our SDP solver will maintain and update a description of $\tilde{\rho}$ per iteration. In each iteration, given copies of $\tilde{\rho}$ and the actual unknown state ρ, we want to know whether $\operatorname{Tr}(E_i\tilde{\rho}) \approx \operatorname{Tr}(E_i\rho)\ \forall i \in [m]$ or there is at least a violation i^*. To that end, we design for each i a projection for the following procedure: (1) perform multiple independent SWAP tests between $E_i/\operatorname{Tr}[E_i]$ (from Oracle 3) and ρ, $\tilde{\rho}$ respectively; (2) accept when the statistics of both SWAP tests (one with ρ, the other with $\tilde{\rho}$) are close. Hence, one can apply our fast quantum OR lemma on these projections to find such i^* if it exists.

Note that both the sample and complexities of the above procedure have a poly-log dependence on n (i.e., the dimension of the quantum state to learn).

Shadow tomography. In a sequence of works [1, 3], Aaronson asked whether one can predict information about a dimension-n quantum state with $\mathrm{poly}(\log n)$ copies. In Ref. [1], he showed that a linear number of copies is sufficient to predict the outcomes of "most" measurements according to some (arbitrary) distribution over a class of measurements. Very recently, in Ref. [3], he referred the following problem as the "shadow tomography" problem: for any n-dimensional state ρ and two-outcome measurements $E_1, \ldots, E_m$, estimate $\mathrm{Tr}[\rho E_i]$ up to error ϵ, $\forall i \in [m]$. He has further designed a quantum procedure for the shadow tomography problem with $\tilde{O}(\ell \cdot \log^4 m/\epsilon^5)$ copies of ρ.

Noting that the shadow tomography problem is essentially the same problem considered by Jaynes [152], and one can apply Jaynes's principle and its algorithmic version we discussed before. Although this can be used to give a version of the result of Ref. [3], Aaronson obtained his result [3] through a different route, based on a post-selection argument. A drawback of this approach is that its gate complexity is high, scaling linearly in m and as $n^{O(\log\log n)}$ (for fixed error).

Our Theorem 4.1.5 can be applied here to improve the time complexity. It gives a quantum procedure with a *square-root* dependence on m and $n^{O(1)}$ dependence on n for arbitrary E_i's.

When we assume r is small, say $r = O(\mathrm{poly}\log n)$, the gate complexity of the entire procedure becomes $\tilde{O}(\sqrt{m}\,\mathrm{poly}\log(n))$. This gives a class of measurement (namely any set of low-rank measurements which can be efficiently implemented) for which the learning problem is efficient both in the number of samples and the computational complexity. This solves an open problem proposed in Ref. [1].

Although we have not worked out an explicit bound of the sample complexity of our procedure, the authors of [23] followed our approach with more sophisticated techniques and obtained a sample complexity of $\tilde{O}(\ell \cdot \log^4 m/\epsilon^4)$, improving on the bound from [3]. We also note that very recently, Aaronson et al. claimed the same sample complexity (i.e., $\tilde{O}(\ell \cdot \log^4 m/\epsilon^4)$) in [5].

Organization. We will formulate the SDP feasibility problem and prove the correctness of the basic framework in Section 4.2. Our implementation of the fast quantum OR lemma is given in Section 4.3. We describe our main results the constructions of quantum SDP solvers in the plain input model and the quantum input model in Section 4.4, Section 4.5, respectively. The application to learning quantum states is illustrated in Section 4.6. In Section 4.7 we describe how to sample from the Gibbs state of low-rank Hamiltonians.

4.2 Feasibility of SDPs

In this section, we formulate the feasibility problem of SDPs. It is a standard fact that one can use binary search to reduce any optimization problem to a feasibility one. The high-level idea is to first guess a candidate value for the objective function, and add that as a constraint to the optimization problem. It converts the optimization problem into a feasibility problem. One can then use binary search on the candidate value to find a good approximation to the optimal one.

Definition 4.2.1 (Feasibility)**.** *Given an $\epsilon > 0$, m real numbers $a_1, \ldots, a_m \in \mathbb{R}$, and Hermitian $n \times n$ matrices $A_1, \ldots, A_m$ where $-I \preceq A_i \preceq I, \forall\, j \in [m]$, define the*

convex region $\mathcal{S}_\epsilon$ *as all* X *such that*

$$\mathrm{Tr}(A_i X) \leq a_i + \epsilon \quad \forall\, i \in [m]; \tag{4.2.1}$$

$$X \succeq 0; \tag{4.2.2}$$

$$\mathrm{Tr}[X] = 1. \tag{4.2.3}$$

For approximate feasibility testing, it is required that:

- *If* $\mathcal{S}_0 = \varnothing$*, output fail;*
- *If* $\mathcal{S}_\epsilon \neq \varnothing$*, output an* $X \in \mathcal{S}_\epsilon$*.*

Zero-sum game approach for SDPs. We adopt the zero-sum game approach to solve SDPs. Note that it is different from [24, 56] which follow the primal-dual approach of [29] to solve SDPs. Instead of leveraging the dual program, we rely on the following oracle:

Oracle 6 (Search for violation)**.** *Input a density matrix* X*, output an* $i \in [m]$ *such that Eq. (6.1.6) is violated. If no such* i *exists, output "FEASIBLE".*

This oracle helps establish a game view to solve any SDP feasibility problem. Imagine Player 1 who wants to provide a feasible $X \in \mathcal{S}_\epsilon$. Player 2, on the other side, wants to find any violation of any proposed X. (This is exactly the function of Oracle 6.) If the original problem is feasible, there exists a feasible point X_0 (provided by Player 1) such that there is no violation of X_0 that can be found by Player 2 (i.e., Oracle 6). This actually refers to an *equilibrium* point of the zero-

sum game, which can be approximated by the matrix multiplicative weight update method [28].

This game view of solving the SDP feasibility problem has appeared in the classical literature (e.g., [142]) and has already been used in solving semidefinite programs in the context of quantum complexity theory (e.g., [131, 274]). We observe that many techniques to quantize Arora-Kale's primal-dual approach [29] for solving SDPs in Refs. [24, 56] readily extends to the zero-sum game approach, e.g., using quantum Gibbs samplers to generate candidate solution states.

The main difference, however, lies in the way one make use of the matrix multiplicative weight update method [159], which is a meta algorithm behind both the Arora-Kale's primal-dual approach [29] and the game view approach (e.g., [142]). As we have elaborated in Section 4.1.4, there are a few advantages of adopting this game view approach.

Master algorithm. We present a master algorithm that solves the SDP feasibility problem with the help of Oracle 6. It should be understood that the master algorithm is *not* the final quantum algorithm, where a few steps will be replaced by their quantum counterparts. However, the master algorithm helps demonstrate the correctness of the algorithm and the number of oracle queries.

Our algorithm heavily relies on the matrix multiplicative weight method given in Algorithm 4.1.

Proposition 4.2.1 ([159], Corollary 4)**.** *Assume that for all* $t \in [T]$*, either* $M^{(t)} \preceq 0$ *or* $M^{(t)} \succeq 0$*. Then Algorithm 4.1 guarantees that after* T *rounds, for any density*

Algorithm 4.1: Matrix multiplicative weights algorithm [159, Figure 3.1].

Initialization: Fix a $\delta \leq 1/2$. Initialize the weight matrix $W^{(1)} = I_n$;
for $t = 1, 2, \ldots, T$ **do**
  Set the density matrix $\rho^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}[W^{(t)}]}$;
  Observe the gain matrix $M^{(t)}$;
  Define the new weight matrix: $W^{(t+1)} = \exp\left[\delta \sum_{\tau=1}^{t} M^{(\tau)}\right]$;

matrix ρ, we have

$$(1-\delta) \sum_{t:\, M^{(t)} \preceq 0} \mathrm{Tr}(M^{(t)}\rho^{(t)}) + (1+\delta) \sum_{t:\, M^{(t)} \succeq 0} \mathrm{Tr}(M^{(t)}\rho^{(t)}) \geq \sum_{t=1}^{T} \mathrm{Tr}(M^{(t)}\rho) - \frac{\ln n}{\delta}.$$

We use Algorithm 4.1 and Proposition 4.2.1 to test the feasibility of SDPs.

Theorem 4.2.1 (Master Algorithm)**.** *Assume we are given Oracle 6. Then for any $\epsilon > 0$, feasibility of the SDP in (6.1.6), (6.1.7), and (6.1.8) can be tested by Algorithm 4.2 with at most $\frac{16 \ln n}{\epsilon^2}$ queries to the oracle.*

Algorithm 4.2: The MMW algorithm for testing the feasibility of SDPs.

Initialize the weight matrix $W^{(1)} = I_n$, and $T = \frac{16 \ln n}{\epsilon^2}$;
for $t = 1, 2, \ldots, T$ **do**
  Prepare the Gibbs state $\rho^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}[W^{(t)}]}$;
  Find a $j^{(t)} \in \{1, 2, \ldots, m\}$ such that $\mathrm{Tr}(A_{j^{(t)}}\rho^{(t)}) > a_{j^{(t)}} + \epsilon$ by Oracle 6. Take $M^{(t)} = \frac{1}{2}(I_n - A_{j^{(t)}})$ if such $j^{(t)}$ can be found; otherwise, claim that $\mathcal{S}_\epsilon \neq \varnothing$, output $\rho^{(t)}$ as a feasible solution, and terminate the algorithm;
  Define the new weight matrix: $W^{(t+1)} = \exp\left[\frac{\epsilon}{2} \sum_{\tau=1}^{t} M^{(\tau)}\right]$;
Claim that $\mathcal{S}_0 = \varnothing$ and terminate the algorithm;

Proof. For all $j \in [m]$, denote $M_j = \frac{1}{2}(I_n - A_j)$; note that $0 \preceq M_j \preceq I \ \forall j \in [m]$. In round t, after computing the density matrix $\rho^{(t)}$, equivalently speaking, Oracle 6

checks whether there exists a $j \in [m]$ such that $\mathrm{Tr}(M_j \rho^{(t)}) < \frac{1}{2} - \frac{a_j + \epsilon}{2}$. If not, then $\mathrm{Tr}(M_j \rho^{(t)}) \geq \frac{1}{2} - \frac{a_j + \epsilon}{2}$ $\forall j \in [m]$, $\mathrm{Tr}(A_j \rho^{(t)}) \leq a_j + \epsilon$ $\forall j \in [m]$, and hence $\rho^{(t)} \in \mathcal{S}_\epsilon$.

Otherwise, the oracle outputs an $M_{j^{(t)}} \in \{M_j\}_{j=1}^m$ such that $\mathrm{Tr}(M_{j^{(t)}} \rho^{(t)}) < \frac{1}{2} - \frac{a_{j^{(t)}} + \epsilon}{2}$. After $T = \frac{16 \ln n}{\epsilon^2}$ iterations, by Proposition 4.2.1 (taking $\delta = \epsilon/4$ therein), this matrix multiplicative weights algorithm promises that for any density matrix ρ, we have

$$\left(1 + \frac{\epsilon}{4}\right) \sum_{t=1}^{T} \mathrm{Tr}(M_{j^{(t)}} \rho^{(t)}) \geq \sum_{t=1}^{T} \mathrm{Tr}(M_{j^{(t)}} \rho) - \frac{4 \ln n}{\epsilon}. \tag{4.2.4}$$

If $\mathcal{S}_0 \neq \varnothing$, there exists a $\rho^* \in \mathcal{S}_0$ such that $\mathrm{Tr}(M_{j^{(t)}} \rho^*) \geq \frac{1}{2} - \frac{a_{j^{(t)}}}{2}$ for all $t \in [T]$. On the other hand, $\mathrm{Tr}(M_{j^{(t)}} \rho^{(t)}) < \frac{1}{2} - \frac{a_{j^{(t)}} + \epsilon}{2}$ for all $t \in [T]$. Plugging these two inequalities into (4.2.4), we have

$$\left(1 + \frac{\epsilon}{4}\right) \sum_{t=1}^{T} \left(\frac{1}{2} - \frac{a_{j^{(t)}} + \epsilon}{2}\right) > \sum_{t=1}^{T} \left(\frac{1}{2} - \frac{a_{j^{(t)}}}{2}\right) - \frac{4 \ln n}{\epsilon}, \tag{4.2.5}$$

which is equivalent to

$$\frac{16 \ln n}{\epsilon^2} > \frac{3 + \epsilon}{2} T + \frac{1}{2} \sum_{t=1}^{T} a_{j^{(t)}}. \tag{4.2.6}$$

Furthermore, since $\frac{1}{2} - \frac{a_{j^{(t)}}}{2} \leq \mathrm{Tr}(M_{j^{(t)}} \rho^*) \leq 1$, we have $a_{j^{(t)}} \geq -1$ for all $t \in [T]$. Plugging this into (4.2.6), we have $\frac{16 \ln n}{\epsilon^2} > (1 + \frac{\epsilon}{2}) T$, and hence

$$T < \frac{16 \ln n}{\epsilon^2 (1 + \epsilon/2)} < \frac{16 \ln n}{\epsilon^2}, \tag{4.2.7}$$

contradiction! Therefore, if $\mathrm{Tr}(M_{j^{(t)}}\rho^{(t)}) < \frac{1}{2} - \frac{a_{j^{(t)}}+\epsilon}{2}$ happens for at least $\frac{16\ln n}{\epsilon^2}$ times, it must be the case that $\mathcal{S}_0 = \varnothing$. □

4.3 Fast quantum OR lemma

A key step in our master algorithm (Algorithm 4.2) is to implement Oracle 6 that finds a violated constraint in the SDP. This is basically to search among m measurements, which motivates us to use the quantum OR lemma from [138].

Lemma 4.3.1 ([138], Corollary 11)**.** *Let $\Lambda_1, \ldots, \Lambda_m$ be projectors, and fix parameters $0 < \epsilon \leq 1/2$, $0 < \delta < 1/4m$. Let ρ be a state such that either $\exists\, j \in [m]$ such that $\mathrm{Tr}[\rho\Lambda_j] \geq 1-\epsilon$, or $\frac{1}{m}\sum_{j=1}^{m}\mathrm{Tr}[\rho\Lambda_j] \leq \delta$. Then there is a test that uses one copy of ρ and: in the former case, accepts with probability at least $(1-\epsilon)^2/7$; in the latter case, accepts with probability at most $4\delta m$.*

However, the focus of Lemma 4.3.1 was on the single copy of ρ and its proof in [138] leads to a poor gate complexity. As a result, we prove the "fast" quantum OR lemma below (Lemma 4.3.2). This new version basically follows the analysis of the original quantum OR lemma; however, the projections are implemented with a quadratic speed-up in m by the fast amplification technique in [211]. This speed-up enables us to decouple the cost of $\sqrt{m}\cdot\sqrt{n}$ in [24, 56] to $(\sqrt{m}+\sqrt{n})$ (see Section 4.4 and Section 4.5 for more details); in particular, it leads to the optimal bound for solving SDPs when other parameters are constants.

Lemma 4.3.2. *Let $\Lambda_1, \ldots, \Lambda_m$ be projections, and fix parameters $0 < \varepsilon \leq 1/2$ and φ. Let ρ be a state such that either $\exists\, i \in [m]$ such that $\mathrm{Tr}[\rho\Lambda_i] \geq 1-\varepsilon$, or*

$\frac{1}{m}\sum_{j=1}^{m}\mathrm{Tr}[\rho\Lambda_j] \leq \varphi$. *Then there is a test that uses one copy of* ρ *and: in the former case, accepts with probability at least* $(1-\varepsilon)^2/4 - \xi$*; in the latter case, accepts with probability at most* $3\varphi m + \xi$*; here* ξ *satisfies* $\xi > 0$ *and* $(1-\varepsilon)^2/4 - \xi > 3\varphi m + \xi$*. Furthermore, as long as the controlled reflection* $\mathrm{ctrl} - (I - 2\sum_{i=0}^{m-1}\Lambda_{i+1}\otimes|i\rangle\langle i|)$ *can be performed in at most* p *operations, this test requires only* $O(\xi^{-1}\sqrt{m}(p + \mathrm{poly}(\log m)))$ *operations to complete.*

Proof. Similar to [138], we will reduce the task of distinguishing the two cases to estimating the eigenvalues of

$$\Lambda := \frac{1}{m}\sum_{i=1}^{m}\Lambda_i, \tag{4.3.1}$$

the average of these POVM operators. Write $P_{\geq\lambda}$ for the projector onto $\mathrm{span}\{|\lambda'\rangle : \Lambda|\lambda'\rangle = \lambda'|\lambda'\rangle, \lambda' \geq \lambda\}$. Then the following was shown in [138]:

Lemma 4.3.3 ([138, Corollary 11])**.** *For any state* ρ *and* $\lambda \leq \max_i \mathrm{Tr}(\Lambda_i\rho)/m$*,*

$$\mathrm{Tr}(P_{\geq\lambda}\rho) \geq [\max_i \mathrm{Tr}(\Lambda_i\rho) - m\lambda]^2. \tag{4.3.2}$$

Choose $\lambda = (1-\varepsilon)/(2m)$. Then we want to distinguish between the following two cases:

1. $\mathrm{Tr}(P_{\geq\lambda}\rho) \geq (1-\varepsilon-m\lambda)^2 = (1-\varepsilon)^2/4$;
2. $\mathrm{Tr}(\Lambda\rho) \leq \varphi$. This implies $\mathrm{Tr}(P_{\geq 0.8\lambda}\rho) \leq \varphi/(0.8\lambda) \leq 3m\varphi$.

We can explicitly decompose Λ as follows (see also [138, Section 2]): Let Q be the

quantum Fourier transform on $\mathbb{Z}_m$, and define the projectors $\Pi = \sum_{i=0}^{m-1} \Lambda_{i+1} \otimes (Q |i\rangle \langle i| Q^\dagger)$, $\Delta = I \otimes |0\rangle \langle 0|$. Then

$$\Delta\Pi\Delta = \frac{1}{m} \sum_{i=1}^{m} \Lambda_i \otimes |0\rangle \langle 0| = \Lambda \otimes |0\rangle \langle 0| . \tag{4.3.3}$$

where $|0\rangle \langle 0|$ in the above equation is shorthand for $|0\rangle \langle 0|^\ell$ for $\ell = \lceil \log m \rceil$.

Let $a = \arccos\left(\sqrt{\lambda}\right)$ and $b = \arccos\left(\sqrt{0.8\lambda}\right)$. Consider the following algorithm, essentially based on the fast amplification algorithm of [211]:

Algorithm 4.3: The fast amplification algorithm in [211].

---

1. Create the state $\rho \otimes |0\rangle \langle 0|^{\otimes \ell}$.
2. Perform phase estimation of the rotation $(I - 2\Pi)(I - 2\Delta)$ on the state, with precision $(b - a)/2$ and error probability ξ. Let the measured eigenvalue be ϕ.
3. Accept iff $|\phi| \leq (a + b)/2$.

The following lemma follows from a direct application of Jordan's lemma:

Lemma 4.3.4 ([211, Section 2.1])**.** *If $|\psi\rangle \otimes |0\rangle^{\otimes \ell}$ is an eigenvector of $\Delta\Pi\Delta$ with eigenvalue $\cos^2 \phi$, then*

$$|\psi\rangle \otimes |0\rangle^{\otimes \ell} = \frac{1}{\sqrt{2}}(|\phi\rangle + |-\phi\rangle) \tag{4.3.4}$$

where $|\phi\rangle$ and $|-\phi\rangle$ are some eigenvectors of $(I - 2\Pi)(I - 2\Delta)$ with eigenvalues ϕ and $-\phi$, respectively.

In Case 1, we have $\mathrm{Tr}(P_{\geq\lambda}\rho) \geq (1-\varepsilon)^2/4$, and therefore Algorithm 4.3 accepts with probability at least $(1-\varepsilon)^2/4 - \xi$. In Case 2, we have $\mathrm{Tr}(P_{\geq 0.8\lambda}\rho) \leq 3m\varphi$, and

therefore Algorithm 4.3 accepts with probability at most $3m\varphi + \xi$.

Algorithm 4.3 requires applying the controlled version of the Grover iterate $(I-2\Pi)(I-2\Delta)$ $O(((b-a)\xi)^{-1}) = O(\sqrt{m}\xi^{-1})$ times. Furthermore, the controlled reflection ctrl-$(I-2\Delta)$ is implementable by $O(\log m)$ gates since $\Delta = I \otimes |0\rangle\langle 0|^{\otimes \lceil \log m \rceil}$, and the controlled reflection ctrl-$(I-2\Pi)$ is implementable using $O(p+\mathrm{poly}(\log m))$ gates by assumption. □

Remark 4.3.1. *The gate complexity in Lemma 4.3.2 is optimal in $\sqrt{m}$, i.e., there exists projections $\Lambda_1, \ldots, \Lambda_m$ and a state ρ such that distinguishing whether $\exists\, i \in [m]$ $\mathrm{Tr}[\rho\Lambda_i] \geq 2/3$ or $\frac{1}{m}\sum_{j=1}^{m} \mathrm{Tr}[\rho\Lambda_j] \leq 1/8m$ requires at least $\Omega(\sqrt{m})$ gates. In particular, assume that $\Lambda_i = |i\rangle\langle i|$ for all $i \in [m]$ and $\rho = |k\rangle\langle k|$ where $k \in [m+1]$. Then to distinguish whether $\exists\, i \in [m]$ $\mathrm{Tr}[\rho\Lambda_i] \geq 2/3$ or $\frac{1}{m}\sum_{j=1}^{m} \mathrm{Tr}[\rho\Lambda_j] \leq 1/8m$, it is equivalent to searching whether $k \in [m]$ or not; deciding this requires at least $\Omega(\sqrt{m})$ gates due to the hardness of Grover search [46].*

4.4 Quantum SDP solver in the plain model

Before we get into the quantum SDP solver in the plain model, we first modularize the cost of two important blocks as follows.

Definition 4.4.1 (trace estimation)**.** *Assume that we have an s-sparse $n \times n$ Hermitian matrix H with $\|H\| \leq \Gamma$ and a density matrix ρ. Then we define $\mathcal{S}_{\mathrm{Tr}}(s, \Gamma, \epsilon)$ and $\mathcal{T}_{\mathrm{Tr}}(s, \Gamma, \epsilon)$ as the sample complexity of ρ and the time complexity of using the plain model (Oracle 1) of H and two-qubit gates, respectively, such that one can compute $\mathrm{Tr}[H\rho]$ with additive error ϵ with success probability at least $2/3$.*

Definition 4.4.2 (Gibbs sampling)**.** *Assume that we have an s-sparse $n \times n$ Hermitian matrix H with $\|H\| \leq \Gamma$. Then we define $\mathcal{T}_{Gibbs}(s, \Gamma, \epsilon)$ as the complexity of preparing the Gibbs state $\frac{e^{-H}}{\mathrm{Tr}[e^{-H}]}$ with additive error ϵ using the plain model (Oracle 1) of H and two-qubit gates.*

As a subsequence of Lemma 4.3.2, Definition 4.4.1, and Definition 4.4.2, we prove the following theorem under the plain model:

Theorem 4.4.1. *Assume we are given Oracle 1. Furthermore, assume that A_j is s-sparse for all $j \in [m]$. Then for any $\epsilon > 0$, feasibility of the SDP in (6.1.6), (6.1.7), and (6.1.8) can be tested by Algorithm 4.4 with success probability at least 0.96 and $\frac{s}{\epsilon^4}\tilde{O}\big(\mathcal{S}_{\mathrm{Tr}}\big(\frac{s}{\epsilon^2}, \frac{1}{\epsilon}, \epsilon\big)\mathcal{T}_{Gibbs}\big(\frac{s}{\epsilon^2}, \frac{1}{\epsilon}, \epsilon\big) + \sqrt{m}\mathcal{T}_{\mathrm{Tr}}\big(\frac{s}{\epsilon^2}, \frac{1}{\epsilon}, \epsilon\big)\big)$ quantum gates and queries to Oracle 1.*

Algorithm 4.4: Efficiently testing the feasibility of SDPs: Plain model.

1 Initialize the weight matrix $W^{(1)} = I_n$, and $T = \frac{16 \ln n}{\epsilon^2}$;
2 **for** $t = 1, 2, \ldots, T$ **do**
3 Prepare $\log m \cdot \mathcal{S}_{\mathrm{Tr}}(\frac{s \log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ samples of Gibbs state $\rho^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}[W^{(t)}]}$ by Definition 4.4.2;
4 Using these $\log m \cdot \mathcal{S}_{\mathrm{Tr}}(\frac{s \log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ copies of $\rho^{(t)}$, search for a $j^{(t)} \in [m]$ such that $\mathrm{Tr}[A_{j^{(t)}}\rho^{(t)}] > a_{j^{(t)}} + \epsilon$ by Lemma 4.3.2 (for each j, we use Definition 4.4.1 to compute $\mathrm{Tr}[A_j\rho]$). If such $j^{(t)}$ is found, take $M^{(t)} = \frac{1}{2}(I_n - A_{j^{(t)}})$; otherwise, claim that $\mathcal{S}_\epsilon \neq \varnothing$ (the SDP is feasible);
5 Define the new weight matrix: $W^{(t+1)} = \exp\big[-\frac{\epsilon}{4}\sum_{\tau=1}^{t} M^{(\tau)}\big]$;
6 Claim that $\mathcal{S}_0 = \varnothing$ and terminate the algorithm.

Proof. The correctness of Algorithm 4.4 is automatically established by Theorem 4.2.1; it suffices to analyze the gate cost of Algorithm 4.4.

In Line 3 of Algorithm 4.4, we apply Definition 4.4.2 to compute the Gibbs state $\rho^{(t)}$. In round t, because $t \leq \frac{16 \ln n}{\epsilon^2}$, $\frac{\epsilon}{4}\sum_{\tau=1}^{t} M^{(\tau)}$ has sparsity at most $s' \leq t \cdot s =$

$O(\frac{s\log n}{\epsilon^2})$, and $\|\frac{\epsilon}{4}\sum_{\tau=1}^{t} M^{(\tau)}\| \leq \frac{\epsilon}{4}\cdot t = O(\frac{\log n}{\epsilon})$. As a result, $\mathcal{T}_{\text{Gibbs}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ quantum gates and queries to Oracle 1 suffice to prepare a copy of the Gibbs state $\rho^{(t)}$. In addition, since to query an element of $\frac{\epsilon}{4}\sum_{\tau=1}^{t} M^{(\tau)}$ we need to query each of the $A_{j^{(\tau)}}$, we have an overhead of $s\cdot\frac{16\ln n}{\epsilon^2}$ for constructing Oracle 1 for $\frac{\epsilon}{4}\sum_{\tau=1}^{t} M^{(\tau)}$ (in particular, Appendix D of the full version of [24] showed that this overhead $\Theta(\frac{s\ln n}{\epsilon^2})$ is necessary and sufficient for constructing the plain oracle for $\frac{\epsilon}{4}\sum_{\tau=1}^{t} M^{(\tau)}$). In total, Line 3 of Algorithm 4.4 costs

$$\frac{16s\ln n}{\epsilon^2}\cdot\log m\cdot\mathcal{S}_{\text{Tr}}\Big(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon\Big)\cdot\mathcal{T}_{\text{Gibbs}}\Big(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon\Big) \tag{4.4.1}$$

quantum gates and queries to Oracle 1.

Next, using these $\log m\cdot\mathcal{S}_{\text{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ copies of $\rho^{(t)}$, we apply Definition 4.4.1 for $O(\log m)$ times to create two-outcome POVMs M_j for any $j\in[m]$ such that M_j decides whether $\text{Tr}(A_j\rho) - a_j > \epsilon$ with success probability boosted to $1-O(1/m)$. The gate complexity of each M_j is $\mathcal{T}_{\text{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ by Definition 4.4.1. Furthermore, because Oracle 1 is reversible, we can assume an explicit decomposition $M_j\otimes|0\rangle\langle 0|^{\otimes a} = P\Lambda_j P$ for some integer a, $P = I\otimes|0\rangle\langle 0|^{\otimes a}$, and some orthogonal projector Λ_j. Let $\widetilde{\rho} = \rho^{\otimes C}\otimes|0\rangle\langle 0|^{a}$ where $C = \log m\cdot\mathcal{S}_{\text{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ with a large enough constant in $\tilde{O}$. We therefore need to decide between the cases

1. $\text{Tr}[\Lambda_j\widetilde{\rho}] \geq 1 - \frac{0.01}{m}$ for some $j\in[m]$; or
2. $\text{Tr}[\Lambda_j\widetilde{\rho}] \leq \frac{0.01}{m}$ for all $j\in[m]$.

This corresponds to the two cases of Lemma 4.3.2, where $\varepsilon = \varphi = \frac{0.01}{m}$. Because each

M_j can be implemented with $\mathcal{T}_{\mathrm{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$ two-qubit gates, the total gate complexity of implementing the reflection $I - 2\sum_{j=0}^{m-1} \Lambda_j \otimes |j\rangle\langle j|$ is also $\mathcal{T}_{\mathrm{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)$. As a result, the total cost of applying Lemma 4.3.2 is $\tilde{O}\big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)\big)$.

In Lemma 4.3.2, we choose $\xi = \frac{1}{3}(\frac{(1-\varepsilon)^2}{4} - 3m\varphi)$ – this is a positive constant. We can thus tell the two cases apart with constant probability. Then, we repeat the call of Lemma 4.3.2 for $L = \Theta(\log \frac{\log n}{\epsilon^2})$ times and accept if and only if Lemma 4.3.2 accepts for at least $\frac{L}{2} \cdot (\frac{(1-\epsilon)^2}{4} + 3m\varphi)$ times. By Chernoff's bound, this can enhance the success probability to at least $1 - \frac{\epsilon^2}{400 \ln n}$.

In all, we have a quantum algorithm that determines whether there exists a $j \in [m]$ such that $\mathrm{Tr}[A_j\rho] \geq a_j + \epsilon$ with success probability at least $1 - \frac{\epsilon^2}{400\ln n}$, using $\tilde{O}\big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)\big)$ quantum gates and queries to Oracle 1. To find this j, we apply binary search on $j \in \{1, 2, \ldots, m\}$, i.e., apply the algorithm to $j \in \{1, \ldots, \lfloor m/2 \rfloor\}$ and $j \in \{\lceil m/2 \rceil, \ldots, m\}$ respectively, and if the output is yes then call the algorithm recursively. This gives an extra $\mathrm{poly}(\log m)$ overhead on the queries to Oracle 1, which is still $\tilde{O}\big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon)\big)$. In addition, similar to the analysis of Line 3, there is an overhead of $s \cdot \frac{16\ln n}{\epsilon^2}$ for constructing Oracle 1 of the Gibbs state using Oracle 1 of each of the $A_{j^{(\tau)}}$. Therefore, the total cost of executing Line 4 of Algorithm 4.4 is

$$\frac{s}{\epsilon^2}\tilde{O}\Big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}\Big(\frac{s\log n}{\epsilon^2}, \frac{\log n}{\epsilon}, \epsilon\Big)\Big). \tag{4.4.2}$$

Because Algorithm 4.4 has at most $\frac{16\ln n}{\epsilon^2}$ iterations, with success probability at least $1 - \frac{16\ln n}{\epsilon^2} \cdot \frac{\epsilon^2}{400\ln n} = 0.96$ Algorithm 4.4 works correctly, and its execution

takes

$$\frac{16s\ln n}{\epsilon^4}\Big(\log m\,\mathcal{S}_{\mathrm{Tr}}\Big(\frac{s\log n}{\epsilon^2},\frac{\log n}{\epsilon},\epsilon\Big)\mathcal{T}_{\mathrm{Gibbs}}\Big(\frac{s\log n}{\epsilon^2},\frac{\log n}{\epsilon},\epsilon\Big)+\tilde{O}\Big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}\big(\frac{s\log n}{\epsilon^2},\frac{\log n}{\epsilon},\epsilon\big)\Big)\Big)$$
$$=\frac{s}{\epsilon^4}\tilde{O}\Big(\mathcal{S}_{\mathrm{Tr}}\Big(\frac{s}{\epsilon^2},\frac{1}{\epsilon},\epsilon\Big)\mathcal{T}_{\mathrm{Gibbs}}\Big(\frac{s}{\epsilon^2},\frac{1}{\epsilon},\epsilon\Big)+\sqrt{m}\mathcal{T}_{\mathrm{Tr}}\Big(\frac{s}{\epsilon^2},\frac{1}{\epsilon},\epsilon\Big)\Big) \tag{4.4.3}$$

two-qubit gates and queries to Oracle 1. □

To be more explicit, the complexities of $\mathcal{S}_{\mathrm{Tr}}$, $\mathcal{T}_{\mathrm{Tr}}$, and $\mathcal{T}_{\mathrm{Gibbs}}$ are given in previous literatures:

Lemma 4.4.1 ([56], Lemma 12)**.** *Given an s-sparse $n\times n$ Hermitian matrix H with $\|H\|\leq 1$ and a density matrix ρ, with probability larger than $1-p_e$, one can compute $\mathrm{Tr}[H\rho]$ with additive error ϵ in time $O(s\epsilon^{-2}\log^4(ns/p_e\epsilon))$ using $O(\epsilon^{-2}\log(1/p_e))$ copies of ρ. In other words, $\mathcal{S}_{\mathrm{Tr}}(s,1,\epsilon)=O(1/\epsilon^2)$ and $\mathcal{T}_{\mathrm{Tr}}(s,1,\epsilon)=O(s/\epsilon^2)$.*

Lemma 4.4.2 ([229])**.** *Given an s'-sparse $n\times n$ Hermitian matrix H with $\|H\|\leq\beta$ for some $\beta>0$, one can prepare the Gibbs state $\frac{e^{-H}}{\mathrm{Tr}[e^{-H}]}$ with additive error ϵ using $\tilde{O}(\frac{\sqrt{\dim(H)}\beta s'}{\epsilon})$ calls to Oracle 1 of H and two-qubit gates. In other words, $\mathcal{T}_{Gibbs}(s,\Gamma,\epsilon)=\tilde{O}(s\Gamma\sqrt{n}/\epsilon)$.*

As a consequence of Theorem 4.4.1, Lemma 4.4.1, and Lemma 4.4.2, we have the following complexity result for solving SDPs under the plain model:

Corollary 4.4.1. *Assume we are given Oracle 1. Furthermore, assume that A_j is s-sparse for all $j\in[m]$. Then for any $\epsilon>0$, feasibility of the SDP in (6.1.6), (6.1.7), and (6.1.8) can be tested by Algorithm 4.4 with success probability at least 0.96 and $\tilde{O}(s^2(\frac{\sqrt{m}}{\epsilon^{10}}+\frac{\sqrt{n}}{\epsilon^{12}}))$ quantum gates and queries to Oracle 1.*

Proof. Note that $\mathcal{S}_{\mathrm{Tr}}(s,\Gamma,\epsilon) = \mathcal{S}_{\mathrm{Tr}}(s,1,\frac{\epsilon}{\Gamma})$ and $\mathcal{T}_{\mathrm{Tr}}(s,\Gamma,\epsilon) = \mathcal{T}_{\mathrm{Tr}}(s,1,\frac{\epsilon}{\Gamma})$ by renormalizing the Hamiltonian H to H/Γ. As a result, plugging Lemma 4.4.1 and Lemma 4.4.2 into Theorem 4.4.1, the complexity of solving the SDP becomes

$$\frac{s}{\epsilon^4}\cdot\tilde{O}\Big(\frac{1}{\epsilon^4}\cdot\frac{s\sqrt{n}}{\epsilon^4}+\frac{s\sqrt{m}}{\epsilon^6}\Big)=\tilde{O}\Big(s^2\Big(\frac{\sqrt{m}}{\epsilon^{10}}+\frac{\sqrt{n}}{\epsilon^{12}}\Big)\Big). \tag{4.4.4}$$

□

Remark 4.4.1. *The $(\sqrt{m}+\sqrt{n})$ dependence is optimal compared to [24, 56].*

Remark 4.4.2. *Using more elaborated techniques and analyses, Ref. [23] improved the complexity of Corollary 4.4.1 to $\tilde{O}(s(\frac{\sqrt{m}}{\epsilon^4}+\frac{\sqrt{n}}{\epsilon^5}))$.*

4.5 Quantum SDP solver with quantum inputs

In this section, we illustrate our quantum SDP solver in the quantum input model. To that end, we first provide a precise formulation of the quantum input model, and then demonstrate how to implement Oracle 6 in such scenario and how the actual quantum algorithm works.

4.5.1 The quantum input model

As mentioned in the introduction, we would like to equip the quantum SDP solver with some extra power beyond only accessing the entries of the input matrices (i.e., A_j, $j=1,\ldots,m$, each of $n\times n$ size). We imagine the setting where these A_j's are nice so that the following oracles, representing various means to access A_j's, can

be efficiently implemented.

Oracle 7 (Oracle for traces of A_j)**.** *A quantum oracle (unitary), denoted O_{Tr} (and its inverse $O_{\mathrm{Tr}}^\dagger$), such that for any $j \in [m]$,*

$$O_{\mathrm{Tr}}|j\rangle|0\rangle|0\rangle = |j\rangle|\operatorname{Tr}[A_j^+]\rangle|\operatorname{Tr}[A_j^-]\rangle, \tag{4.5.1}$$

where A_j^+ and A_j^- are two PSD matrices such that $A_j = A_j^+ - A_j^-$ (the real values $\operatorname{Tr}[A_j^+]$ and $\operatorname{Tr}[A_j^-]$ are encoded into their binary representations).

Oracle 8 (Oracle for preparing A_j)**.** *A quantum oracle (unitary), denoted O (and its inverse $O^\dagger$), which acts on $\mathbb{C}^m \otimes (\mathbb{C}^n \otimes \mathbb{C}^n) \otimes (\mathbb{C}^n \otimes \mathbb{C}^n)$ such that for any $j \in [m]$,*

$$O|j\rangle\langle j| \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0|O^\dagger = |j\rangle\langle j| \otimes |\psi_j^+\rangle\langle\psi_j^+| \otimes |\psi_j^-\rangle\langle\psi_j^-|, \tag{4.5.2}$$

where $|\psi_j^+\rangle, |\psi_j^-\rangle \in \mathbb{C}^n \otimes \mathbb{C}^n$ are any purifications of $\frac{A_j^+}{\operatorname{Tr}[A_j^+]}, \frac{A_j^-}{\operatorname{Tr}[A_j^-]}$, respectively.[5]

Oracle 9 (Oracle for a_j)**.** *A quantum oracle (unitary), denoted O_a (and its inverse $O_a^\dagger$), such that for any $j \in [m]$,*

$$O_a|j\rangle\langle j| \otimes |0\rangle\langle 0|O_a^\dagger = |j\rangle\langle j| \otimes |a_j\rangle\langle a_j|, \tag{4.5.3}$$

where the real value a_j is encoded into its binary representation.

Similar to Section 4.4, we also modularize the cost of two important blocks as follows.

[5] By tracing out the extra space, one can easily obtain states $A_j^+ / \operatorname{Tr}[A_j^+], A_j^- / \operatorname{Tr}[A_j^-]$.

Definition 4.5.1 (trace estimation)**.** *Assume that* $\mathrm{Tr}(A_j^+) + \mathrm{Tr}(A_j^-) \leq B$ *for some bound* B *for all* $j \in [m]$*. Then we define* $\mathcal{S}_{\mathrm{Tr}}(B, \epsilon)$ *and* $\mathcal{T}_{\mathrm{Tr}}(B, \epsilon)$ *as the sample complexity of a state* $\rho \in \mathbb{C}^{n \times n}$ *and the gate complexity of using the quantum input oracles (Oracle 7, Oracle 8, Oracle 9), and two-qubit gates, respectively, such that there exists a quantum algorithm which distinguishes with success probability at least* $1 - O(1/m)$ *whether for a fixed* $j \in [m]$, $\mathrm{Tr}(A_j \rho) > a_j + \epsilon$ *or* $\mathrm{Tr}(A_j \rho) \leq a_j$*.*

Definition 4.5.2 (Gibbs sampling)**.** *Assume that* $K = K^+ - K^-$*, where* $K^\pm = \sum_{j \in S} c_j A_j^\pm$*,* $S \subseteq [m]$ *and* $|S| \leq \Phi$*,* $c_j > 0$*, and* $A_j^\pm$ *refers to either* A_j^+ *or* A_j^- *for all* $j \in [m]$*. Moreover, assume that* $\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-) \leq B_K$ *for some bound* B_K*, and that* K^+*,* K^- *have rank at most* r_K*. Then we define* $\mathcal{T}_{Gibbs}(r_K, \Phi, B_K, \epsilon)$ *as the complexity of preparing the Gibbs state* $\rho_G = \exp(-K)/\mathrm{Tr}(\exp(-K))$ *to* ϵ *precision in trace distance using Oracle 7, Oracle 8, Oracle 9, and two-qubit gates.*

4.5.2 Implementation of Oracle 6 – searching a violated constraint

Using Oracle 7, Oracle 8, and Oracle 9, Oracle 6 can be implemented by the following lemma, using our fast quantum OR lemma (Lemma 4.3.2):

Lemma 4.5.1. *Given* $\epsilon, \delta \in (0, 1)$*. Assume we have Oracle 7, Oracle 8, Oracle 9, and* $(\log 1/\delta) \cdot \tilde{O}(\mathcal{S}_{\mathrm{Tr}}(B, \epsilon))$ *copies of a state* ρ*. Assume either* $\exists\, j \in [m]$ *such that* $\mathrm{Tr}(A_j \rho) \geq a_j + \epsilon$*, or* $\mathrm{Tr}(A_j \rho) \leq a_j$ *for all* $j \in [m]$*. Then there is an algorithm that in the former case, finds such a* j*; and in the latter case, returns "FEASIBLE". This algorithm has success probability* $1 - \delta$ *and uses in total* $\log 1/\delta \cdot \tilde{O}(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B, \epsilon))$ *quantum gates and queries to Oracle 7, Oracle 8, and Oracle 9.*

Proof. First, we use Definition 4.5.1 to create two-outcome POVMs M_j, acting on ρ, $|\psi_j^+\rangle\langle\psi_j^+|$, and $|\psi_j^-\rangle\langle\psi_j^-|$ with $C = \mathcal{S}_{\mathrm{Tr}}(B,\epsilon)$ copies, such that M_j decides with probability $1 - O(1/\operatorname{poly}(m))$ whether $\mathrm{Tr}(A_j\rho) - a_j > \epsilon$.

Because we are given purifications of all A_j^+ and A_j^- in Oracle 8, for all $j \in \{1,\ldots,m\}$ we can assume an explicit decomposition $M_j \otimes |0\rangle\langle 0|^{\otimes a} = P\Lambda_j P$, for some integer a, $P = I \otimes |0\rangle\langle 0|^{\otimes a}$, and some orthogonal projector Λ_j. Let $\widetilde{\rho} = \rho^{\otimes C} \otimes (|\psi_j^+\rangle\langle\psi_j^+|)^{\otimes C} \otimes (|\psi_j^-\rangle\langle\psi_j^-|)^{\otimes C} \otimes |0\rangle\langle 0|^a$. We therefore need to decide between the cases

1. $\mathrm{Tr}[\Lambda_j\widetilde{\rho}] \geq 1 - O(1/\operatorname{poly}(m))$ for some j; or
2. $\mathrm{Tr}[\Lambda_j\widetilde{\rho}] \leq O(1/\operatorname{poly}(m))$ for all j.

This corresponds to the two cases of Lemma 4.3.2, where both ε and δ are $O(1/\operatorname{poly}(m))$. To implement the the projection $I - 2\sum_{j=1}^m \Lambda_j \otimes |j\rangle\langle j|$ in Lemma 4.3.2, we use Oracle 8 to obtain purifications $|\psi_j^+\rangle\langle\psi_j^+|$ and $|\psi_j^-\rangle\langle\psi_j^-|$ of $\frac{A_j^+}{\mathrm{Tr}[A_j^+]}$ and $\frac{A_j^-}{\mathrm{Tr}[A_j^-]}$, and apply the reflection with respect to $|\psi_j^+\rangle$ and $|\psi_j^-\rangle$; note that we can obtain the numbers $\mathrm{Tr}[A_j^+]$ and $\mathrm{Tr}[A_j^-]$ in superposition by Oracle 7. Including the controlling ancilla $|j\rangle\langle j|$, the p in Lemma 4.3.2 is at most $O(\log m)$.

In Lemma 4.3.2, choose $\xi = \frac{1}{3}(\frac{(1-\varepsilon)^2}{4} - 3m\varphi)$ – this is a positive constant. We can thus tell the two cases apart with constant probability, using $\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)$ samples of ρ and $\tilde{O}(\sqrt{m})\cdot\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)$ other operations. Then, we repeat the call of Lemma 4.3.2 for $L = \Theta(\log \delta^{-1})$ times and accept if and only if Lemma 4.3.2 accepts for at least $\frac{L}{2}\cdot(\frac{(1-\epsilon)^2}{4} + 3m\varphi)$ times. By Chernoff's bound, this enhances the success probability to at least $1-\delta$.

In all, we have a quantum algorithm that determines whether there exists a $j \in [m]$ such that $\mathrm{Tr}(A_j\rho) \geq a_j + \epsilon$ (or $\mathrm{Tr}(A_j\rho) \leq a_j$ for all $j \in [m]$) with success probability at least $1-\delta$, using $\log 1/\delta \cdot \tilde{O}(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon))$ quantum gates and queries to Oracle 7, Oracle 8, and Oracle 9. To find this j, we take $\delta \leftarrow \delta/\log m$, and apply binary search on $j \in \{1,2,\ldots,m\}$, i.e., apply the algorithm to $j \in \{1,\ldots,\lfloor m/2 \rfloor\}$ and $j \in \{\lceil m/2 \rceil,\ldots,m\}$ respectively, and if the output is yes then call the algorithm recursively. This gives an extra $\mathrm{poly}(\log m)$ overhead on both sample complexity and gate complexity, which are still $(\log 1/\delta)\cdot\tilde{O}(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon))$ and $\log 1/\delta \cdot \tilde{O}(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon))$, respectively. □

4.5.3 Quantum SDP solvers with quantum inputs

We now instantiate Algorithm 4.2 to the fully quantum version (Algorithm 4.5). A key difference is that we use Definition 4.5.2 to generate (many copies) of the Gibbs state $\rho^{(t)}$ and rely on Lemma 4.5.1 to implement Oracle 6. At a high-level, the correctness of Algorithm 4.5 still roughly comes from Theorem 4.2.1, as well as Lemma 4.5.1. However, its gate complexity will be efficient because of the help of Oracle 7, Oracle 8, and Oracle 9.

Theorem 4.5.1. *Assume we are given Oracle 7, Oracle 8, and Oracle 9. Furthermore, assume $\mathrm{Tr}[A_j^+] + \mathrm{Tr}[A_j^-] \leq B$ for some bound B, and A_j have rank at most r for all $j \in [m]$. Then for any $\epsilon > 0$, feasibility of the SDP in (6.1.6), (6.1.7), and (6.1.8) can be tested by Algorithm 4.5 with success probability at least 0.96 and at most $\frac{1}{\epsilon^2}\tilde{O}\big(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)\mathcal{T}_{Gibbs}\big(\frac{r}{\epsilon^2},\frac{1}{\epsilon^2},\frac{B}{\epsilon},\epsilon\big) + \sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)\big)$ quantum gates and queries to*

Oracle 7, *Oracle 8*, and *Oracle 9*.

Algorithm 4.5: Efficiently SDP feasibility testing: Quantum input model.

1 Initialize the weight matrix $W^{(1)} = I_n$, and $T = \frac{16 \ln n}{\epsilon^2}$;
2 **for** $t = 1, 2, \ldots, T$ **do**
3 Prepare $\tilde{O}(\mathcal{S}_{\mathrm{Tr}}(B, \epsilon))$ samples of the Gibbs state $\rho^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}[W^{(t)}]}$ by Definition 4.5.2;
4 Using these $\tilde{O}(\mathcal{S}_{\mathrm{Tr}}(B, \epsilon))$ copies of $\rho^{(t)}$, search for a $j^{(t)} \in [m]$ such that $\mathrm{Tr}(A_{j^{(t)}} \rho^{(t)}) > a_{j^{(t)}} + \epsilon$ by Lemma 4.5.1 with $\delta = \frac{\epsilon^2}{400 \ln n}$. Take $M^{(t)} = \frac{1}{2}(I_n - A_{j^{(t)}})$ if such $j^{(t)}$ is found; otherwise, claim that $\mathcal{S}_\epsilon \neq \varnothing$ (the SDP is feasible);
5 Define the new weight matrix: $W^{(t+1)} = \exp\left[-\frac{\epsilon}{2} \sum_{\tau=1}^{t} M^{(\tau)}\right]$;
6 Claim that $\mathcal{S}_0 = \varnothing$ and terminate the algorithm.

Proof. The correctness of Algorithm 4.5 is automatically established by Theorem 4.2.1; it suffices to analyze the gate cost of Algorithm 4.5.

In Line 3 of Algorithm 4.5 we apply Definition 4.5.2 to compute the Gibbs state $\rho^{(t)}$. In round t, because $M_j = \frac{1}{2}[I_n - (A_j^+ - A_j^-)] = \frac{1}{2} I_n + \frac{1}{2} A_j^- - \frac{1}{2} A_j^+ \ \forall j \in [m]$, we take $K_t^+ = \frac{\epsilon}{2} \sum_{\tau=1}^{t} \frac{1}{2} A_{j^{(\tau)}}^+$ and $K_t^- = \frac{\epsilon}{2} \sum_{\tau=1}^{t} \frac{1}{2} A_{j^{(\tau)}}^-$. Because $t \leq \frac{16 \ln n}{\epsilon^2}$, K_t^+, K_t^- have rank at most $t \cdot r = O(\log n \cdot r / \epsilon^2)$, and $\mathrm{Tr}[K_t^+]$, $\mathrm{Tr}[K_t^-]$ are at most $\frac{\epsilon t}{4} \cdot B = O(\log n \cdot B / \epsilon)$, Definition 4.5.2 guarantees that

$$\mathcal{T}_{\mathrm{Gibbs}}\left(\frac{r \log n}{\epsilon^2}, \frac{16 \ln n}{\epsilon^2}, \frac{B \log n}{\epsilon}, \epsilon\right) = \tilde{O}\left(\mathcal{T}_{\mathrm{Gibbs}}\left(\frac{r}{\epsilon^2}, \frac{1}{\epsilon^2}, \frac{B}{\epsilon}, \epsilon\right)\right) \tag{4.5.4}$$

quantum gates and queries to Oracle 7, Oracle 8, and Oracle 9 suffice to prepare the Gibbs state $\rho^{(t)}$. Because there are at most $\frac{16 \ln n}{\epsilon^2}$ iterations and in each iteration

$\rho^{(t)}$ is prepared for $\tilde{O}(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon))$ copies, the total cost for Gibbs state preparation is

$$\frac{16\ln n}{\epsilon^2}\cdot\tilde{O}\Big(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)\mathcal{T}_{\mathrm{Gibbs}}\Big(\frac{r}{\epsilon^2},\frac{1}{\epsilon^2},\frac{B}{\epsilon},\epsilon\Big)\Big). \tag{4.5.5}$$

Furthermore, by Lemma 4.5.1, Line 4 finds a $j^{(t)}\in[m]$ such that $\mathrm{Tr}(A_{j^{(t)}}\rho^{(t)})>a_{j^{(t)}}+\epsilon$ with success probability at least $1-\frac{\epsilon^2}{400\ln n}$, using $\tilde{O}(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon))$ quantum gates and queries to Oracle 7, Oracle 8, and Oracle 9. Because Algorithm 4.5 has at most $\frac{16\ln n}{\epsilon^2}$ iterations, with probability at least $1-\frac{16\ln n}{\epsilon^2}\cdot\frac{\epsilon^2}{400\ln n}=0.96$ we can assume that Lemma 4.5.1 works correctly, and the total cost of running Line 4 is

$$\frac{16\ln n}{\epsilon^2}\cdot\tilde{O}\big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)\big). \tag{4.5.6}$$

In all, by (4.5.5) and (4.5.6), the gate complexity of running Algorithm 4.5 is

$$\begin{aligned}&\frac{16\ln n}{\epsilon^2}\cdot\tilde{O}\Big(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)\mathcal{T}_{\mathrm{Gibbs}}\Big(\frac{r}{\epsilon^2},\frac{1}{\epsilon^2},\frac{B}{\epsilon},\epsilon\Big)\Big)+\frac{16\ln n}{\epsilon^2}\cdot\tilde{O}\big(\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)\big)\\&\quad=\frac{1}{\epsilon^2}\tilde{O}\Big(\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)\mathcal{T}_{\mathrm{Gibbs}}\Big(\frac{r}{\epsilon^2},\frac{1}{\epsilon^2},\frac{B}{\epsilon},\epsilon\Big)+\sqrt{m}\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)\Big).\end{aligned} \tag{4.5.7}$$

□

To be more explicit, in later sections we prove that:

- Lemma 4.5.2: $\mathcal{S}_{\mathrm{Tr}}(B,\epsilon)=\mathcal{T}_{\mathrm{Tr}}(B,\epsilon)=O(B^2\log m/\epsilon^2)$.
- Lemma 4.5.4: $\mathcal{T}_{\mathrm{Gibbs}}(r_K,\Phi,B_K,\epsilon)=O(\Phi\cdot\mathrm{poly}(\log n,r_K,B_K,\epsilon^{-1}))$.

As a consequence, we have the following complexity result for solving SDPs under the quantum input model:

Corollary 4.5.1. *Assume we are given Oracle 7, Oracle 8, and Oracle 9. Furthermore, assume* $\mathrm{Tr}[A_j^+] + \mathrm{Tr}[A_j^-] \leq B$ *for some bound* B*, and* A_j *have rank at most* r *for all* $j \in [m]$*. Then for any* $\epsilon > 0$*, feasibility of the SDP in (6.1.6), (6.1.7), and (6.1.8) can be tested by Algorithm 4.5 with success probability at least 0.96 and at most* $(\sqrt{m} + \mathrm{poly}(r)) \cdot \mathrm{poly}(\log m, \log n, B, \epsilon^{-1})$ *quantum gates and queries to Oracle 7, Oracle 8, and Oracle 9.*

Proof. By Theorem 4.5.1, the complexity of solving the SDP is

$$\begin{aligned} &\frac{1}{\epsilon^2}\tilde{O}\Big(\frac{B^2 \log m}{\epsilon^2} \cdot \frac{1}{\epsilon^2}\,\mathrm{poly}\left(\log n, \frac{r}{\epsilon}, \frac{B}{\epsilon}, \frac{1}{\epsilon}\right) + \sqrt{m} \cdot \frac{B^2 \log m}{\epsilon^2}\Big) \\ &\quad = (\sqrt{m} + \mathrm{poly}(r)) \cdot \mathrm{poly}(\log m, \log n, B, \epsilon^{-1}). \end{aligned} \tag{4.5.8}$$

□

Remark 4.5.1. *When we use Definition 4.5.2 to prepare the Gibbs state* $\rho^{(t)}$ *in Line 3 of Algorithm 4.5, we have* $W^{(t)} = \exp\left[-\frac{\epsilon t}{4} I_n + K_t^+ - K_t^-\right]$ *by Line 5 which actually has an extra* $-\frac{\epsilon t}{4} I_n$ *term. However, for any constant* $c \in \mathbb{R}$ *and Hermitian matrix* H *we have*

$$\frac{e^{cI-H}}{\mathrm{Tr}[e^{cI-H}]} = \frac{e^c e^{-H}}{\mathrm{Tr}[e^c e^{-H}]} = \frac{e^{-H}}{\mathrm{Tr}[e^{-H}]}, \tag{4.5.9}$$

hence this $-\frac{\epsilon t}{4} I_n$ *term does not change* $\rho^{(t)}$*.*

Remark 4.5.2. *In Corollary 4.5.1, the only restriction on the decomposition* $A_j = A_j^+ - A_j^-$ *for all* $j \in [m]$ *is that* $\mathrm{Tr}[A_j^+] + \mathrm{Tr}[A_j^-] \leq B$*. If we assume this decomposition to be the eigen-decomposition, i.e.,* A_j^+ *represents the subspace spanned by the*

eigenvectors of A_j with positive eigenvalues, and A_j^- represents the subspace spanned by the eigenvectors of A_j with negative eigenvalues, then by the low-rank assumption and $-I \preceq A_j \preceq I$, $\mathrm{Tr}[A_j^+] + \mathrm{Tr}[A_j^-] \leq r$. In this case, Corollary 4.5.1 takes at most $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ quantum gates and queries to Oracle 7, Oracle 8, and Oracle 9.

Remark 4.5.3. *The $\sqrt{m}$ dependence is optimal compared to Theorem 4.5.2 proved later.*

Remark 4.5.4. *Using more elaborated techniques and analyses, Ref. [23] explicitly computed the degrees of the parameters in (4.5.8) and improved the complexity of Corollary 4.5.1 to $\tilde{O}(\frac{B\sqrt{m}}{\epsilon^4} + \frac{B^{3.5}}{\epsilon^{7.5}})$ (the rank r is implicitly contained in B and hence this complexity is independent of r).*

4.5.4 Trace estimation

In this subsection, we prove:

Lemma 4.5.2. *Assume we are given Oracle 7, Oracle 8, Oracle 9, and $O(B^2 \log m/\epsilon^2)$ copies of a state $\rho \in \mathbb{C}^{n\times n}$, where $\mathrm{Tr}[A_j^+] + \mathrm{Tr}[A_j^-] \leq B$ for some bound B for all $j \in [m]$. Then for any $\epsilon > 0$, Algorithm 4.6 distinguishes whether $\mathrm{Tr}(A_j\rho) > a_j + \epsilon$ or $\mathrm{Tr}(A_j\rho) \leq a_j$ with success probability at least $1 - O(1/\mathrm{poly}(m))$. In other words, $\mathcal{S}_{\mathrm{Tr}}(B, \epsilon) = \mathcal{T}_{\mathrm{Tr}}(B, \epsilon) = O(B^2 \log m/\epsilon^2)$.*

Proof. Recall that the SWAP test [62] on ρ and $\frac{A_j^+}{\mathrm{Tr}[A_j^+]}$ outputs 1 with probability $\frac{1}{2} + \frac{\mathrm{Tr}(A_j^+\rho)}{2\,\mathrm{Tr}[A_j^+]}$, and the SWAP test on ρ and $\frac{A_j^-}{\mathrm{Tr}[A_j^-]}$ outputs 1 with probability $\frac{1}{2} + \frac{\mathrm{Tr}(A_j^-\rho)}{2\,\mathrm{Tr}[A_j^-]}$.

Algorithm 4.6: Implementation of the POVM M_j.

1 Using Oracle 8, apply the SWAP test on ρ and $\frac{A_j^+}{\operatorname{Tr}[A_j^+]}$ for $\operatorname{poly}(\log m, \log n, B, \epsilon^{-1})$ times. Denote the frequency of getting 1 to be $\widetilde{p_{j,+}}$;

2 Using Oracle 8, apply the SWAP test on ρ and $\frac{A_j^-}{\operatorname{Tr}[A_j^-]}$ for $\operatorname{poly}(\log m, \log n, B, \epsilon^{-1})$ times. Denote the frequency of getting 1 to be $\widetilde{p_{j,-}}$;

3 Apply Oracle 7 to compute $\operatorname{Tr}[A_j^+]$ and $\operatorname{Tr}[A_j^-]$. Claim that $\operatorname{Tr}(A_j\rho) > a_j + \epsilon$ if $\big(2\widetilde{p_{j,+}} - 1\big)\operatorname{Tr}[A_j^+] - \big(2\widetilde{p_{j,-}} - 1\big)\operatorname{Tr}[A_j^-] > a_j + \epsilon/2$, and claim that $\operatorname{Tr}(A_j\rho) < a_j$ if $\big(2\widetilde{p_{j,+}} - 1\big)\operatorname{Tr}[A_j^+] - \big(2\widetilde{p_{j,-}} - 1\big)\operatorname{Tr}[A_j^-] \le a_j + \epsilon/2$;

Therefore, by Chernoff's bound and the fact that $\operatorname{Tr}[A_j^+], \operatorname{Tr}[A_j^-] \le B$, we have

$$\begin{aligned}\Pr\Big[\Big|\widetilde{p_{j,+}} - \Big(\frac{1}{2} + \frac{\operatorname{Tr}(A_j^+\rho)}{2\operatorname{Tr}[A_j^+]}\Big)\Big| \ge \frac{\epsilon}{8\operatorname{Tr}[A_j^+]}\Big] &\le \Pr\Big[\Big|\widetilde{p_{j,+}} - \Big(\frac{1}{2} + \frac{\operatorname{Tr}(A_j^+\rho)}{2\operatorname{Tr}[A_j^+]}\Big)\Big| \ge \frac{\epsilon}{8B}\Big] \\ &\le 2e^{-\frac{O(B^2\log m/\epsilon^2)\cdot\epsilon^2}{64B^2\cdot 2}} \qquad (4.5.10) \\ &\le O\Big(\frac{1}{\operatorname{poly}(m)}\Big) \qquad (4.5.11)\end{aligned}$$

for a large constant in the big-O in (4.5.10). Similarly,

$$\Pr\Big[\Big|\widetilde{p_{j,-}} - \Big(\frac{1}{2} + \frac{\operatorname{Tr}(A_j^-\rho)}{2\operatorname{Tr}[A_j^-]}\Big)\Big| \ge \frac{\epsilon}{8\operatorname{Tr}[A_j^-]}\Big] \le O\Big(\frac{1}{\operatorname{poly}(m)}\Big). \qquad (4.5.12)$$

In other words, with probability at least $1 - O\big(\frac{1}{\operatorname{poly}(m)}\big)$,

$$\big|\big(2\widetilde{p_{j,+}} - 1\big)\operatorname{Tr}[A^+] - \operatorname{Tr}(A_j^+\rho)\big| \le \frac{\epsilon}{4}, \big|\big(2\widetilde{p_{j,-}} - 1\big)\operatorname{Tr}[A_j^-] - \operatorname{Tr}(A_j^-\rho)\big| \le \frac{\epsilon}{4}. \quad (4.5.13)$$

Therefore, if $\operatorname{Tr}(A_j\rho) = \operatorname{Tr}(A_j^+\rho) - \operatorname{Tr}(A_j^-\rho) > a_j + \epsilon$, then with probability at least

$1 - O\big(\frac{1}{\mathrm{poly}(m)}\big)$,

$$\big(2\widetilde{p_{j,+}} - 1\big)\operatorname{Tr}[A_j^+] - \big(2\widetilde{p_{j,-}} - 1\big)\operatorname{Tr}[A_j^-] > a_j + \epsilon/2, \tag{4.5.14}$$

which is exactly the first part of Line 3. Similarly, we can use Chernoff's bound to prove that if $\operatorname{Tr}(A_j\rho) \le a_j$, then with probability at least $1 - O\big(\frac{1}{\mathrm{poly}(m)}\big)$,

$$\big(2\widetilde{p_{j,+}} - 1\big)\operatorname{Tr}[A_j^+] - \big(2\widetilde{p_{j,-}} - 1\big)\operatorname{Tr}[A_j^-] \le a_j + \epsilon/2, \tag{4.5.15}$$

which is the second part of Line 3.

Because Algorithm 4.6 only uses SWAP which only takes $O(1)$ quantum gates, in total we have $\mathcal{S}_{\mathrm{Tr}}(B,\epsilon) = \mathcal{T}_{\mathrm{Tr}}(B,\epsilon) = O(B^2 \log m/\epsilon^2)$. □

4.5.5 Gibbs state preparation

With the access to Oracle 7 and Oracle 8, the following lemma shows how to prepare two normalized quantum states $K^\pm/\operatorname{Tr}[K^\pm]$ where $K^\pm = \sum_{j\in S} c_j A_j^\pm$, $c_j > 0$ and $A_j^\pm$ refers to either A_j^+ or A_j^-.

Lemma 4.5.3. *$K^{sgn}/\operatorname{Tr}[K^{sgn}]$ can be prepared by $|S|$ samples to Oracle 7 and one sample to Oracle 8, for both $sgn = +$ and $sgn = -$.*

Proof. Consider the following protocol, where we choose all $\pm$ to be $+$ when preparing $K^+/\operatorname{Tr}[K^+]$, and choose all $\pm$ to be $-$ when preparing $K^-/\operatorname{Tr}[K^-]$:

1. For all $j \in S$, sample Oracle 7 to obtain $\operatorname{Tr}[A_j^\pm]$;

2. To prepare $K^{\pm}/\operatorname{Tr}[K^{\pm}]$, toss a coin $i \in S$ such that $\Pr[i=j] = \frac{c_j \operatorname{Tr}[A_j^{\pm}]}{\sum_{k\in S} c_k \operatorname{Tr}[A_k^{\pm}]}$, take one sample of Oracle 8 to obtain $A_j^{\pm}/\operatorname{Tr}[A_j^{\pm}]$, and output this state.

By symmetry, we only consider the preparation of $K^{\pm}/\operatorname{Tr}[K^{\pm}]$. With probability $\frac{c_j \operatorname{Tr}[A_j^{\pm}]}{\sum_{k\in S} c_k \operatorname{Tr}[A_k^{\pm}]}$, the output state is $A_j^{\pm}/\operatorname{Tr}[A_j^{\pm}]$; therefore, in average the density matrix prepared is

$$\sum_{j\in S} \frac{c_j \operatorname{Tr}[A_j^{\pm}]}{\sum_{k\in S} c_k \operatorname{Tr}[A_k^{\pm}]} \cdot \frac{A_j^{\pm}}{\operatorname{Tr}[A_j^{\pm}]} = \frac{\sum_{j\in S} c_j A_j^{\pm}}{\sum_{k\in S} c_k \operatorname{Tr}[A_k^{\pm}]} = \frac{K^{\pm}}{\operatorname{Tr}[K^{\pm}]}. \tag{4.5.16}$$

Furthermore, Step 1 takes $|S|$ samples to Oracle 7, and Step 2 takes one sample to Oracle 8; this exactly matches the sample complexity claimed in Lemma 4.5.3. □

Combining Lemma 4.5.3 and Theorem 4.7.1 leads to a lemma that generates the Gibbs state in Line 4 of Algorithm 4.2:

Lemma 4.5.4. *Suppose $K = K^+ - K^-$, where $K^{\pm} = \sum_{j\in S} c_j A_j^{\pm}$, $c_j > 0$ and $A_j^{\pm}$ refers to either A_j^+ or A_j^-. Moreover, assume that $\operatorname{Tr}(K^+) + \operatorname{Tr}(K^-) \le B_K$ for some bound B_K, and that K^+, K^- have rank at most r_K. Then it is possible to prepare the Gibbs state $\rho_G = \exp(-K)/\operatorname{Tr}(\exp(-K))$ to ϵ precision in trace distance, with $|S| \cdot \operatorname{poly}(\log n, r_K, B_K, \epsilon^{-1})$ quantum gates and queries to Oracle 7 and Oracle 8. In other words, $\mathcal{T}_{Gibbs}(r_K, \Phi, B_K, \epsilon) = O(\Phi \cdot \operatorname{poly}(\log n, r_K, B_K, \epsilon^{-1}))$.*

4.5.6 Lower bound for quantum SDP solvers with quantum inputs

In this section, we prove quantum lower bounds in the quantum input setting.

Theorem 4.5.2 (Lower bound on Theorem 4.5.1)**.** *There exists an SDP feasibility testing problem such that* $B, r, \epsilon = \Theta(1)$*, and solving the problem requires* $\Omega(\sqrt{m})$ *calls to Oracle 7 and Oracle 8.*

Proof. Consider the following two instances of the SDP feasibility testing problem:

1. For all $j \in [m]$, set $A_j^- = 0$. For a random $i^* \in [n]$, set $(A_j^+)_{i^*i^*} = 1$ for all $j \in [m]$. All other elements of matrices A_j^+ are set to zero. For a random $j^* \in [m]$, set $a_{j^*} = -1/2$. Set $a_j = 1/2$ for all $j \neq j^*$. Set $\epsilon = 1/4$.

2. For all $j \in [m]$, set $A_j^- = 0$. For a random $i^* \in [n]$, set $(A_j^+)_{i^*i^*} = 1$ for all $j \in [m]$. All other elements of matrices A_j^+ are set to zero. Set $a_j = 1/2$ for all $j \in [m]$. Set $\epsilon = 1/4$.

Note that the first problem is not feasible because there is no X such that $X \succeq 0$ and $\mathrm{Tr}[A_{j^*}X] \leq -1/2 + 1/4 < 0$; the second problem is always feasible. For both problems, we have $B = r = 1$, and the state $\frac{A_j^+}{\mathrm{Tr}[A_j^+]}$ is always $|i^*\rangle\langle i^*|$ for all $j \in [m]$. Therefore, Oracle 8 provides no information for distinguishing between the two problems, and we should only rely on Oracle 7. But this is equivalent to searching for the j^* such that $a_{j^*} = -1/2$, and by reduction to the lower bound on Grover search it takes at least $\Omega(\sqrt{m})$ queries to Oracle 7 for distinguishing between the two problems. □

Combining Theorem 4.5.1 and Theorem 4.5.2, we obtain the optimal bound on SDP feasibility testing using Oracle 7 and Oracle 8, up to poly-logarithmic factors.

4.6 Application: Efficient learnability of quantum states

We consider the following quantum state learning problem, also named "shadow tomography" in [3]: Let ρ be an unknown quantum state in an n-dimensional Hilbert space, $E_1, \ldots, E_m$ be known two-outcome POVMs, and $0 < \epsilon < 1$. Given independent copies of ρ, one wants to obtain an explicit quantum circuit for a state σ such that with probability at least $2/3$, $|\operatorname{Tr}[\sigma E_i] - \operatorname{Tr}[\rho E_i]| \leq \epsilon \ \forall\, i \in [m]$. What is the sample complexity (i.e., the number of required copies of ρ) and gate complexity (i.e., the total running time) of the best such procedure?

Aaronson provides a solution with the sample complexity (i.e., the number of copies of ρ) of $\tilde{O}\big(\log^4 m \cdot \log n/\epsilon^5\big)$ in [3]. In this section we show that, for low rank matrices and small m, we can also make the learning process *computationally efficient* while keeping a comparable sample complexity, by using our previous result on speeding up solutions to SDPs.

4.6.1 Reduction from shadow tomography to SDP feasibility

We start with a simple explanation of using the solution to SDP feasibility to address shadow tomography. Given (many copies of) any unknown quantum state ρ and two-outcome POVMs $E_1, \ldots, E_m$, in order to estimate $\operatorname{Tr}[\rho E_i]$, it suffices to

find a state σ that is the solution to the following SDP feasibility problem:

$$\mathrm{Tr}[\sigma E_i] \leq \mathrm{Tr}[\rho E_i] + \epsilon \quad \forall\, i \in [m]; \tag{4.6.1}$$

$$\mathrm{Tr}[\sigma E_i] \geq \mathrm{Tr}[\rho E_i] - \epsilon \quad \forall\, i \in [m]; \tag{4.6.2}$$

$$\mathrm{Tr}[\sigma] = 1; \tag{4.6.3}$$

$$\sigma \succeq 0. \tag{4.6.4}$$

Any feasible solution σ satisfies that $|\,\mathrm{Tr}[\sigma E_i] - \mathrm{Tr}[\rho E_i]| < \epsilon$ for all $i \in [m]$. Thus, our quantum SDP solver will generate a description of such σ. However, we do not know $\mathrm{Tr}[\rho E_i]$, and hence the constraints of the SDP feasibility problem, in advance. The *key* observation is that our SDP solver only relies on the implementation of Oracle 6, which does not need the knowledge of $\mathrm{Tr}[\rho E_i]$ for each i explicitly. It turns out that with the help of the fast quantum OR lemma, one only needs a few copies of ρ for the implementation of Oracle 6.

4.6.2 Finding the violated constraint using $\tilde{O}(\sqrt{m})$ gates

Similar to Section 4.5, we assume the existence of Oracle 7 and Oracle 8 to achieve efficient quantum circuits. Specifically, for the feasibility problem (6.1.4)-(4.6.4), we have:

Oracle 7 for traces of E_i: A unitary O_{Tr} such that for any $i \in [m]$, $O_{\mathrm{Tr}}|i\rangle|0\rangle = |i\rangle|\,\mathrm{Tr}[E_i]\rangle$.

Oracle 8 for preparing E_i: A unitary O (and its inverse $O^\dagger$) acting on $\mathbb{C}^m \otimes (\mathbb{C}^n \otimes \mathbb{C}^n)$ such that for any $i \in [m]$,

$$O|i\rangle\langle i| \otimes |0\rangle\langle 0|O^\dagger = |i\rangle\langle i| \otimes |\psi_i\rangle\langle\psi_i|, \tag{4.6.5}$$

where $|\psi_i\rangle\langle\psi_i|$ is any purification of $\frac{E_i}{\mathrm{Tr}[E_i]}$.

Furthermore, we assume that the POVM operator E_i has rank at most r, for all $i \in [m]$. Using Oracle 7 and Oracle 8, Oracle 6 (searching for violation) can be implemented by the following lemma:

Lemma 4.6.1. *Given $\epsilon, \delta \in (0,1)$. Assume we have Oracle 7, Oracle 8, and $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1}, \log \delta^{-1})$ copies of two states $\rho, \sigma \in \mathbb{C}^n$. Assume either $\exists\, i \in [m]$ such that $|\mathrm{Tr}[\sigma E_i] - \mathrm{Tr}[\rho E_i]| \geq \epsilon$, or $|\mathrm{Tr}[\sigma E_i] - \mathrm{Tr}[\rho E_i]| \leq \epsilon/2$ for all $i \in [m]$. Then there is an algorithm that in the former case, finds such an i; and in the latter case, returns "FEASIBLE". This algorithm has success probability $1-\delta$ and uses in total $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1}, \log \delta^{-1})$ quantum gates and queries to Oracle 7 and Oracle 8.*

Lemma 4.6.1 also follows from our fast quantum OR lemma (Lemma 4.3.2) by combining Lemma 4.5.2 and Lemma 4.5.1, where we replace ρ by $\rho^{\otimes C} \otimes \sigma^{\otimes C} \otimes |0\rangle\langle 0|^a$ for some $C = \mathrm{poly}(\log m, \log n, \epsilon^{-1})$; also notice that because $0 \leq E_i \leq I$ and $\mathrm{rank}(E_i) \leq r$, we have $B \leq r$. As a result, the detailed proof is omitted here.

4.6.3 Gate complexity of learning quantum states

Similar to Corollary 4.5.1, we solve shadow tomography by using Lemma 4.5.4 to generate (copies) of the Gibbs state $\rho^{(t)}$ and relying on Lemma 4.6.1 to implement Oracle 6.

Corollary 4.6.1. *Assume we are given Oracle 7 and Oracle 8. Then for any* $\epsilon > 0$*, shadow tomography can be solved by Algorithm 4.7 with success probability at least 0.96, using at most* $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ *copies of* ρ*, and at most* $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ *quantum gates and queries to Oracle 7 and Oracle 8.*

Algorithm 4.7: Efficiently learn a quantum state via measurements.

1 Initialize the weight matrix $W^{(1)} = I_n$, and $T = \frac{16 \ln n}{\epsilon^2}$;
2 **for** $t = 1, 2, \ldots, T$ **do**
3 Prepare $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ samples of the Gibbs state $\rho^{(t)} = \frac{W^{(t)}}{\mathrm{Tr}[W^{(t)}]}$ by Lemma 4.5.4, and take $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ copies of ρ;
4 Using these $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ copies of $\rho^{(t)}$ and ρ, apply Lemma 4.6.1 with $\delta = \frac{\epsilon^2}{400 \ln n}$ to search for an $i^{(t)} \in [m]$ such that $|\mathrm{Tr}[\rho E_{i^{(t)}}] - \mathrm{Tr}[\rho^{(t)} E_{i^{(t)}}]| \geq \epsilon$. **if** *such* $i^{(t)}$ *is found* **then**
5 **if** $(\mathrm{Tr}[\rho E_{i^{(t)}}] - \mathrm{Tr}[\rho^{(t)} E_{i^{(t)}}] \geq \epsilon)$ **then**
6 Take $M^{(t)} = \frac{1}{2}\big(I_n - (-E_{i^{(t)}} + \mathrm{Tr}[\rho E_{i^{(t)}}] I_n)\big)$;
7 **else** $(\mathrm{Tr}[\rho E_{i^{(t)}}] - \mathrm{Tr}[\rho^{(t)} E_{i^{(t)}}] \leq -\epsilon)$
8 Take $M^{(t)} = \frac{1}{2}\big(I_n - (E_{i^{(t)}} - \mathrm{Tr}[\rho E_{i^{(t)}}] I_n)\big)$;
9 **else** (no such $i^{(t)}$ exists)
10 Claim $\rho^{(t)}$ to be the solution, and terminate the algorithm;
11 Define the new weight matrix: $W^{(t+1)} = \exp\big[-\frac{\epsilon}{2} \sum_{\tau=1}^{t} M^{(\tau)}\big]$;

Proof. Similar to Corollary 4.5.1, the correctness of Algorithm 4.7 is automatically established by Theorem 4.2.1; it suffices to analyze the gate cost of Algorithm 4.7.

In Line 3 of Algorithm 4.7 we apply Lemma 4.5.4 to compute the Gibbs state

$\rho^{(t)}$. In round t, because either

$$M^{(t)} = \frac{1}{2}\big(I_n - (-E_{i^{(t)}} + \mathrm{Tr}[\rho E_{i^{(t)}}]I_n)\big) = \frac{1 - \mathrm{Tr}[\rho E_{i^{(t)}}]}{2} I_n + \frac{1}{2} E_{i^{(t)}} \tag{4.6.6}$$

when Line 6 executes, or

$$M^{(t)} = \frac{1}{2}\big(I_n - (E_{i^{(t)}} - \mathrm{Tr}[\rho E_{i^{(t)}}]I_n)\big) = \frac{1 + \mathrm{Tr}[\rho E_{i^{(t)}}]}{2} I_n - \frac{1}{2} E_{i^{(t)}} \tag{4.6.7}$$

when Line 8 executes, we can take $K_t^+ = \frac{\epsilon}{2}\sum_{\tau=1}^{t} \frac{1}{2} E_{i^{(\tau)}}^+$ and $K_t^- = \frac{\epsilon}{2}\sum_{\tau=1}^{t} \frac{1}{2} E_{i^{(\tau)}}^-$, where $E_{i^{(\tau)}}^+ = E_{i^{(\tau)}}$, $E_{i^{(\tau)}}^- = 0$ when (4.6.6) holds for round τ, and $E_{i^{(\tau)}}^+ = 0$, $E_{i^{(\tau)}}^- = E_{i^{(\tau)}}$ when (4.6.7) holds for round τ. Because $t \le \frac{16 \ln n}{\epsilon^2}$, K_t^+, K_t^- have rank at most $t \cdot r = O(\log n \cdot r/\epsilon^2)$ and $\mathrm{Tr}[K_t^+]$, $\mathrm{Tr}[K_t^-]$ are at most $\frac{\epsilon t}{4} \cdot r = O(\log n \cdot r/\epsilon)$, Lemma 4.5.4 guarantees that

$$\frac{16 \ln n}{\epsilon^2} \cdot \mathrm{poly}\left(\log n, \frac{r \log n}{\epsilon^2}, \frac{r \log n}{\epsilon}, \epsilon^{-1}\right) = \mathrm{poly}(\log n, r, \epsilon^{-1}) \tag{4.6.8}$$

quantum gates and queries to Oracle 7 and Oracle 8 suffice to prepare the Gibbs state $\rho^{(t)}$. Because there are at most $\frac{16 \ln n}{\epsilon^2} = \mathrm{poly}(\log n, \epsilon^{-1})$ iterations and in each iteration $\rho^{(t)}$ is prepared for $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ copies, in total the gate cost for Gibbs state preparation is $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$.

Furthermore, by Lemma 4.6.1, Algorithm 4.7 finds an $i^{(t)} \in [m]$ such that $|\mathrm{Tr}[\rho E_{i^{(t)}}] - \mathrm{Tr}[\rho^{(t)} E_{i^{(t)}}]| \ge \epsilon$ with success probability at least $1 - \frac{\epsilon^2}{400 \ln n}$, using $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ copies of ρ, and $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ quantum gates and queries to Oracle 7 and Oracle 8. Because Algorithm 4.7 has at most $\frac{16 \ln n}{\epsilon^2}$

iterations, with success probability at least $1 - \frac{16 \ln n}{\epsilon^2} \cdot \frac{\epsilon^2}{400 \ln n} = 0.96$ we can assume that the quantum search in Lemma 4.6.1 works correctly, and the total gate cost of calling Algorithm 4.7 is $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$.

In conclusion, $\mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ is an upper bound on the number of copies of ρ, and $\sqrt{m} \cdot \mathrm{poly}(\log m, \log n, r, \epsilon^{-1})$ is an upper bound on the total number of quantum gates and queries to Oracle 7 and Oracle 8. □

Remark 4.6.1. *Using the same idea as Theorem 4.5.2, we can prove that there exists a shadow tomography problem such that $r, \epsilon = \Theta(1)$, and solving the problem requires $\Omega(\sqrt{m})$ calls to Oracle 7 and Oracle 8. Therefore Corollary 4.6.1 is also optimal up to poly-logarithmic factors.*

4.7 Gibbs sampling of low-rank Hamiltonians

In this section, we demonstrate how to sample from the Gibbs state of low-rank Hamiltonians given a quantum oracle generating desired states. We repeatedly use the following result of [195] (with a straightforward generalization in [173]):

Lemma 4.7.1 ([173, 195])**.** *Suppose we are given a quantum oracle that prepares copies of two unknown (normalized) l-qubit quantum states ρ^+ and ρ^-, and we wish to evolve under the Hamiltonian $H = a_+\rho^+ - a_-\rho^-$ for some nonnegative numbers $a_+, a_- \geq 0$. Then we can approximately implement the unitary $\exp(iHt)$ up to diamond-norm error δ, using $O(a^2t^2/\delta)$ copies of ρ^+ and ρ^- and $O(la^2t^2/\delta)$ other 1- or 2-qubit gates, where $a = a_+ + a_-$.*

By using phase estimation on the operator $\exp(iHt)$ with $t = O(1/a)$, we have

Lemma 4.7.2. *Under the same assumptions as Lemma 4.7.1, we can perform eigenvalue estimation of H: given an eigenstate of H, we can estimate its eigenvalue up to precision ϵ, with probability $1-\xi$, using $O(a^2\epsilon^{-2}\xi^{-2})$ copies of ρ^+ and ρ^- and $O(la^2\epsilon^{-2}\xi^{-2})$ other 1- or 2-qubit gates, where $a = a_+ + a_-$. This procedure disturbs the input state by at most a trace distance error of $O(\sqrt{\xi})$.*

4.7.1 Computing the partition function

As a warm-up, we start with the following lemma:

Lemma 4.7.3. *Suppose $K = K^+ - K^-$, where K^+ and K^- are $n \times n$ PSD matrices, and there is a quantum oracle that prepares copies of the states $\rho^+ = K^+/\operatorname{Tr}(K^+)$, $\rho^- = K^-/\operatorname{Tr}(K^-)$, and an oracle for the numbers $\operatorname{Tr}(K^+)$, $\operatorname{Tr}(K^-)$. Moreover, assume that $\operatorname{Tr}(K^+) + \operatorname{Tr}(K^-) \le B$ for some bound B,[6] and that K^+, K^- have rank at most r_K. Then it is possible to estimate the partition function $Z = \operatorname{Tr}(\exp(-K))$ to multiplicative error ϵ with success probability at least $1-\xi$, with $\operatorname{poly}(\log n, r_K, B, \epsilon^{-1}, \xi^{-1})$ quantum gates.*

Proof Sketch. We assume that we can implement the unitary evolution $\exp(iKt)$ as well as the phase estimation protocol, perfectly to infinite precision. This assumption is not true, but it helps to simplify the exposition; the assumption can be removed by Appendix G of [55] where a careful error analysis of this scheme is presented. This idealization is made here for the sake of flashing out the core ideas behind the proposed protocol.

[6] The B here is denoted as B_K in Definition 4.5.2 and Lemma 4.5.4; for simplicity we make this abbreviation throughout Section 4.7.

Under these assumptions, let us first consider the estimation of

$$Z_{\text{supp}} \equiv \sum_{|\lambda_i| \geq \delta} e^{-\lambda_i}, \tag{4.7.1}$$

where $0 < \delta < 1$ is a small threshold and λ_i's are eigenvalues of K. Since $\delta > 0$ is a small parameter, Z_{supp} is the partition function when considering the approximated support of K.

The main idea in the estimation of Z_{supp} is to perform phase estimation of the unitary operator $e^{2\pi i K}$ on ρ^+ and ρ^-, after which we obtain

$$\rho^{\pm} = \frac{K^{\pm}}{\text{Tr}(K^{\pm})} \to \bar{\rho}^{\pm} = \frac{1}{\text{Tr}(K^{\pm})} \sum_{\lambda} \Pi_\lambda K^{\pm} \Pi_\lambda \otimes |\lambda\rangle \langle\lambda| , \tag{4.7.2}$$

where Π_λ is the projection onto the λ-eigenspace of K, and λ is any eigenvalue of K. Let us define

$$K_\lambda^+ := \Pi_\lambda K^+ \Pi_\lambda, \quad K_\lambda^- := \Pi_\lambda K^- \Pi_\lambda. \tag{4.7.3}$$

Then,

$$K_\lambda^+ - K_\lambda^- = \Pi_\lambda K \Pi_\lambda = \lambda \Pi_\lambda, \tag{4.7.4}$$

and therefore K_λ^+ and K_λ^- differ by a multiple of the identity in their support space (the λ-eigenspace of K). Hence K_λ^+ and K_λ^- are simultaneously diagonalizable, and their corresponding eigenvalues differ by exactly λ. In other words, there exists an eigenbasis of K, which we call $\{|v_i\rangle\}_i$ with corresponding eigenvalues λ_i, such that

K_λ^+ and K_λ^- are diagonal in this eigenbasis for all λ. We can therefore write

$$K_\lambda^+ = \sum_{i:\lambda_i=\lambda} \lambda_i^+ |v_i\rangle \langle v_i| , \qquad K_\lambda^- = \sum_{i:\lambda_i=\lambda} \lambda_i^- |v_i\rangle \langle v_i| , \tag{4.7.5}$$

for some nonnegative numbers λ_i^+, λ_i^- satisfying $\lambda_i^+ - \lambda_i^- = \lambda_i$. Combining Eqs. (4.7.2) and (4.7.5), we obtain that $\bar{\rho}^+$ ($\bar{\rho}^-$) – the state after performing phase estimation of the unitary operator $e^{2\pi i K}$ on ρ^+ (ρ^-) is given by

$$\bar{\rho}^\pm = \frac{1}{\mathrm{Tr}(K^\pm)} \sum_{\lambda_i} \lambda_i^\pm |v_i\rangle \langle v_i| \otimes |\lambda_i\rangle \langle \lambda_i| . \tag{4.7.6}$$

Algorithm 4.8: Estimation of Z_{supp}

---

1. Let $\mathrm{sgn} = +$ with probability $\mathrm{Tr}(K^+)/[\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-)]$, and $\mathrm{sgn} = -$ otherwise.
2. Perform phase estimation of the operator $e^{2\pi i K}$ on ρ^{sgn}; Let the output state be $\bar{\rho}^{\mathrm{sgn}} = \frac{1}{\mathrm{Tr}(K^{\mathrm{sgn}})} \sum_{\lambda_i} \lambda_i^{\mathrm{sgn}} |v_i\rangle \langle v_i| \otimes |\lambda_i\rangle \langle \lambda_i|$. Measure the second register and let the obtained eigenvalue of K be λ.
3. If $|\lambda| < \delta$ output 0; else if $\mathrm{sgn} = +$ output $\lambda^{-1} e^{-\lambda}$; else output $-\lambda^{-1} e^{-\lambda}$.

Now consider the procedure in Algorithm 4.8, and let its output be the random variable X. Then under the assumption of perfect phase estimation, we have

$$\begin{aligned} \mathbb{E}[X] &= \frac{\mathrm{Tr}(K^+)}{\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-)} \sum_{|\lambda_i| \geq \delta} \frac{\lambda_i^+}{\mathrm{Tr}(K^+)} \frac{e^{-\lambda_i}}{\lambda_i} - \frac{\mathrm{Tr}(K^-)}{\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-)} \sum_{|\lambda_i| \geq \delta} \frac{\lambda_i^-}{\mathrm{Tr}(K^-)} \frac{e^{-\lambda_i}}{\lambda_i} \\ &= \frac{1}{\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-)} \sum_{|\lambda_i| \geq \delta} e^{-\lambda_i} = \frac{Z_{\mathrm{supp}}}{\mathrm{Tr}(K^+) + \mathrm{Tr}(K^-)}, \end{aligned} \tag{4.7.7}$$

where $\lambda_i^\pm$ are the eigenvalues of $K_{\lambda_i}^\pm$, satisfying $\lambda_i^+ - \lambda_i^- = \lambda_i$. Therefore $\mathbb{E}[X]$ is

proportional to Z_{supp}, and obtaining a multiplicative estimate of $\mathbb{E}[X]$ gives us a multiplicative estimate of Z_{supp}.

The second moment of X is bounded by

$$\mathbb{E}[X^2] = \frac{1}{\text{Tr}(K^+) + \text{Tr}(K^-)} \sum_{|\lambda_i| \geq \delta} (\lambda_i^+ + \lambda_i^-) \frac{e^{-2\lambda_i}}{\lambda_i^2} \leq \max_{|\lambda_i| \geq \delta} |\lambda_i|^{-2} e^{-2\lambda_i} \leq \delta^{-2} Z_{\text{supp}}^2.$$

We see that $\mathbb{E}[X^2] \leq B^2 \delta^{-2} \mathbb{E}[X]^2$, and therefore by Chebyshev's inequality we can obtain, with constant probability, an ϵ-error multiplicative estimate of $\mathbb{E}[X]$, hence of Z_{supp}, by running the above procedure $O(B^2 \delta^{-2} \epsilon^{-2})$ times and taking the mean.

We still need to calculate Z, the full partition function including small eigenvalues of K. Let R denote the number of eigenvalues of K (including degeneracy) with absolute value at least δ, and note that $R \leq 2r_K$, where recall that r_K upper bounds the rank of K^+ and K^- . Define the following approximation of Z:

$$Z' \equiv Z_{\text{supp}} + (n - R) = \sum_{|\lambda_i| \geq \delta} e^{-\lambda_i} + \sum_{|\lambda_i| < \delta} e^0. \tag{4.7.8}$$

Using $e^\delta \leq 1 + 2\delta$ and $e^{-\delta} \geq 1 - \delta$, we get that

$$|Z - Z'| \leq 2\delta(n - R). \tag{4.7.9}$$

Therefore if we make δ small enough, say $\delta = O(\epsilon)$, Z' gives a good multiplicative estimate for Z.

To compute Z', we need a good multiplicative estimate of $n - R$. This can essentially be done by estimating the probability of a random state having eigenvalue

smaller than δ. Let the output of the following procedure be Y:

Algorithm 4.9: Estimation of $n - R$

---

1. Perform phase estimation of the operator $e^{2\pi iK}$ on the uniformly random state I/n; let the output eigenvalue be λ.
2. If $|\lambda| < \delta$ output 1; otherwise output 0.

Y is a Bernoulli random variable with mean $\mathbb{E}[Y] = (n - R)/n$ and variance $\mathrm{Var}[Y] = R(n - R)/n^2 \leq R\mathbb{E}[Y]^2$. By Chebyshev's inequality, $O(r_K\epsilon^{-2})$ repetitions of the above procedure gives us an ϵ-error multiplicative estimate of $\mathbb{E}[Y]$, and thus of $n-R$. Putting everything together, we see that $O(B^2\epsilon^{-4}+r_K\epsilon^{-2})$ uses of (perfect) phase estimation of $e^{2\pi iK}$ suffices to get a $O(\epsilon)$-error multiplicative estimate of Z, completing the proof. □

4.7.2 Sampling from the Gibbs state

Theorem 4.7.1 (Complete proof given in Appendix G of [55])**.** *Suppose $K = K^+ - K^-$, where K^+ and K^- are $n \times n$ PSD matrices, and there is a quantum oracle that prepares copies of the states $\rho^+ = K^+/\operatorname{Tr}(K^+)$, $\rho^- = K^-/\operatorname{Tr}(K^-)$, and an oracle for the numbers $\operatorname{Tr}(K^+)$, $\operatorname{Tr}(K^-)$. Moreover, assume that $\operatorname{Tr}(K^+)+\operatorname{Tr}(K^-) \leq B$ for some bound B, and that K^+, K^- have rank at most r_K. Then it is possible to prepare the Gibbs state $\rho_G = \exp(-K)/\operatorname{Tr}(\exp(-K))$ to ϵ precision in trace distance, with $\mathrm{poly}(\log n, r_K, B, \epsilon^{-1})$ quantum gates.*

Proof sketch. Similar to the proof sketch of the partition function, here as well we assume an infinite precision implementation of the unitary evolution operator

$\exp(iKt)$ as well as of the phase estimation protocol. In addition we assume that quantum principal component analysis can be implemented perfectly. The complete proof is given in Appendix G.3 of [55].

The procedure is similar to that of calculating the partition function above. We pick $\delta = O(\epsilon)$, a small threshold, and first consider a procedure to sample from

$$\rho_{\text{supp}} \equiv \sum_{|\lambda_i| \geq \delta} e^{-\lambda_i} |v_i\rangle \langle v_i| / Z_{\text{supp}}, \tag{4.7.10}$$

where λ_i's and $|v_i\rangle$'s are eigenvalues and eigenstates of K. (In the case that ρ_{supp} is undefined, i.e. that all eigenvalues of K have magnitude less than δ, it is easy to see that the uniformly mixed state I/n is already an $O(\epsilon)$-trace distance error approximation to ρ_G. This is the case when $Z_{\text{supp}} = 0$.) ρ_{supp} is the Gibbs state when considering only the (approximated) support of K. Consider the procedure in Algorithm 4.10.

Note that $\frac{\delta}{\lambda^+ + \lambda^-} \frac{(1-\epsilon)e^{-\lambda}}{Z'_{\text{supp}}} \leq 1$ since $\lambda^{\pm} \geq 0$ and by assumption $|\lambda| = |\lambda^+ - \lambda^-| \geq \delta$, and $Z'_{\text{supp}} \leq (1+\epsilon) Z_{\text{supp}} \leq (1+\epsilon) \sum_{|\lambda_i| \geq \delta} e^{-\lambda_i}$ by (4.7.1). Moreover assuming that K has at least one eigenvalue with magnitude at least δ, the success probability in Line 5 of Algorithm 4.10 is at least

$$\frac{\delta}{\text{Tr}(K^+) + \text{Tr}(K^-)} \sum_{\lambda_i : |\lambda_i| \geq \delta} \frac{(1-\epsilon)e^{-\lambda_i}}{Z'_{\text{supp}}} \geq \frac{\delta(1-\epsilon)}{B(1+\epsilon)}, \tag{4.7.11}$$

and therefore we can output ρ_{supp} efficiently by repeating Algorithm 4.10 until success, which takes $O(B/\delta)$ trials in expectation.

Algorithm 4.10: Estimation of ρ_{supp}

1. Let $\text{sgn} = +$ with probability $\text{Tr}(K^+)/[\text{Tr}(K^+) + \text{Tr}(K^-)]$, and $\text{sgn} = -$ otherwise.
2. Perform phase estimation of the unitary operator $e^{2\pi i K}$ on ρ^{sgn}; let the output state be $\bar{\rho}^{\text{sgn}} = \frac{1}{\text{Tr}(K^{\text{sgn}})} \sum_{\lambda_i} \lambda_i^{\text{sgn}} |v_i\rangle \langle v_i| \otimes |\lambda_i\rangle \langle \lambda_i|$.
3. Project $\bar{\rho}^{\text{sgn}}$ onto $\bar{\varrho}^{\text{sgn}} = \frac{1}{\text{Tr}(K^{\text{sgn}})} \sum_{\lambda_i : |\lambda_i| \geq \delta} \lambda_i^{\text{sgn}} |v_i\rangle \langle v_i| \otimes |\lambda_i\rangle \langle \lambda_i|$.
4. The average state at this stage is

$$\bar{\varrho} = \frac{1}{\text{Tr}(K^+) + \text{Tr}(K^-)} \sum_{\lambda_i : |\lambda_i| \geq \delta} (\lambda_i^+ + \lambda_i^-) |v_i\rangle \langle v_i| \otimes |\lambda_i\rangle \langle \lambda_i| .$$

   Perform phase estimation of the operator $e^{2\pi i K}$ on $\bar{\varrho}$; let the measured eigenvalue be $\mu = \lambda^+ + \lambda^-$, and the resulting state be ϱ_μ.
5. Accept the state ϱ_μ with probability $\frac{\delta}{\mu} \frac{(1-\epsilon)e^{-\lambda_i}}{Z'_{\text{supp}}}$, for Z'_{supp} a ϵ-multiplicative error approximation of Z_{supp} by Algorithm 4.8.

Accounting for the randomness in Step 1, at the end of Step 3, we obtain the mixed state $\bar{\varrho}$. However, for Gibbs sampling, we should have factors of the form $e^{-\lambda_i} |v_i\rangle \langle v_i|$ instead of $(\lambda_i^+ + \lambda_i^-) |v_i\rangle \langle v_i|$ that appear in $\bar{\varrho}$. Therefore, at this stage of the protocol, to accept $|v_i\rangle \langle v_i|$ with probability proportional to $e^{-\lambda_i}/(\lambda_i^+ + \lambda_i^-)$, but for that we need to measure $\lambda_i^+ + \lambda_i^-$. This is done in steps 4 and 5 of the above procedure, which is equivalent to applying $\sum_{\lambda_i : |\lambda_i| \geq \delta} \frac{\delta}{\lambda_i^+ + \lambda_i^-} \frac{e^{-\lambda_i}}{Z_{\text{supp}}} |v_i\rangle \langle v_i| \otimes |\lambda_i\rangle \langle \lambda_i|$ to $\bar{\varrho}$. Upon keeping only the first register we obtain

$$\frac{\delta}{\text{Tr}(K^+) + \text{Tr}(K^-)} \sum_{|\lambda_i| \geq \delta} \frac{e^{-\lambda_i}}{Z_{\text{supp}}} |v_i\rangle \langle v_i| \propto \frac{1}{Z_{\text{supp}}} \sum_{|\lambda_i| \geq \delta} e^{-\lambda_i} |v_i\rangle \langle v_i| \equiv \rho_{\text{supp}}, \quad (4.7.12)$$

where ρ_{supp} is the Gibbs state when considering only the (approximated) support of K.

We still need to calculate ρ_G, the full Gibbs state including small eigenvalues of K. Recall that R denotes the number of eigenvalues (including degeneracy) of K with absolute value at least δ, and note that $R \leq 2r_K$, where r_K upper bounds the rank of K^+ and K^- . Define the following approximation of ρ_G:

$$\rho'_G \equiv \frac{Z_{\text{supp}}}{Z'}\rho_{\text{supp}} + \frac{n-R}{Z'}\rho_{\text{ker}} = \frac{1}{Z'}\Big(\sum_{|\lambda_i|\geq\delta} e^{-\lambda_i}\,|v_i\rangle\langle v_i| + \sum_{|\lambda_i|<\delta} |v_i\rangle\langle v_i|\Big), \quad (4.7.13)$$

where $\rho_{\text{ker}} = \frac{1}{n-R}\sum_{|\lambda_i|<\delta}|v_i\rangle\langle v_i|$ is the uniformly random state on the orthogonal complement of the (approximate) support of K. Then

$$\|\rho_G - \rho'_G\|_{\text{Tr}} = \Big|\frac{1}{Z} - \frac{1}{Z'}\Big|\sum_{|\lambda_i|\geq\delta} e^{-\lambda_i} + \sum_{|\lambda_i|<\delta}\Big|\frac{e^{-\lambda_i}}{Z} - \frac{1}{Z'}\Big| \quad (4.7.14)$$

$$\leq \Big|\frac{1}{Z} - \frac{1}{Z'}\Big|\sum_{|\lambda_i|\geq\delta} e^{-\lambda_i} + \left[\sum_{|\lambda_i|<\delta}\Big|\frac{1}{Z} - \frac{1}{Z'}\Big| + \frac{2\delta}{Z}e^{-\lambda_i}\right] \quad (4.7.15)$$

$$\leq \Big|\frac{1}{Z} - \frac{1}{Z'}\Big|Z' + 2\delta \leq 4\delta. \quad (4.7.16)$$

Therefore if we make δ small enough, ρ'_G gives a good estimate (in trace distance) for ρ_G.

To estimate ρ_{ker}, we consider the output of Algorithm 4.11 below.

Algorithm 4.11: Estimation of ρ_{ker}

1. Perform phase estimation of the operator $e^{2\pi iK}$ on the uniformly random state I/n; let the output eigenvalue be λ and the resulting state be Π_λ.
2. If $|\lambda| \geq \delta$ abort; otherwise, accept the state.

Finally, ρ_G is generated by running the Algorithm 4.10 with probability $\frac{Z_{\text{supp}}}{Z} =$

$\Omega(\frac{\delta}{B})$ (by (4.7.11)) until we accept ρ_{supp}, and running the Algorithm 4.11 with probability

$$\frac{n-R}{Z} \geq \frac{n-2r_K}{n} = \Omega(1), \tag{4.7.17}$$

until we accept ρ_{ker}. The detailed analysis is given in Appendix G.3 of Ref. [55].

In the previous subsection we proved that, upon setting $\delta = O(\epsilon)$ we can obtain Z_{supp}, Z and $n-R$ up to an $O(\epsilon)$ multiplicative error with $\text{poly}(\log n, r_K, B, \epsilon^{-1})$ quantum gates. Therefore, Using Lemma 7 of [24] we obtain $\frac{Z_{\text{supp}}}{Z}$ and $\frac{n-R}{Z}$ to $O(\epsilon)$ multiplicative error. This, in turns, implies the with the above procedure we prepare the Gibbs state ρ_G up to error $O(\epsilon)$ in trace distance, with $\text{poly}(\log n, r_K, B, \epsilon^{-1})$ quantum gates. □

4.8 Conclusions and discussion

In this chapter, we present two new quantum algorithms for solving semidefinite programs (SDPs) providing quantum speed-ups. We consider SDP instances with m constraint matrices, each of dimension n, rank at most r, and sparsity s. The first algorithm assumes an input model where one is given access to an oracle to the entries of the matrices at unit cost. We show that it has run time $\tilde{O}(s^2(\sqrt{m}\epsilon^{-10} + \sqrt{n}\epsilon^{-12}))$, with ϵ the error of the solution. This gives an optimal dependence in terms of m, n and quadratic improvement over previous quantum algorithms (when $m \approx n$). The second algorithm assumes a fully quantum input model in which the input matrices are given as quantum states. We show that its

run time is $\tilde{O}(\sqrt{m} + \text{poly}(r)) \cdot \text{poly}(\log m, \log n, B, \epsilon^{-1})$, with B an upper bound on the trace-norm of all input matrices. In particular the complexity depends only poly-logarithmically in n and polynomially in r.

We apply the second SDP solver to learn a good description of a quantum state with respect to a set of measurements: Given m measurements and a supply of copies of an unknown state ρ with rank at most r, we show we can find in time $\sqrt{m} \cdot \text{poly}(\log m, \log n, r, \epsilon^{-1})$ a description of the state as a quantum circuit preparing a density matrix which has the same expectation values as ρ on the m measurements, up to error ϵ. The density matrix obtained is an approximation to the maximum entropy state consistent with the measurement data considered in Jaynes's principle from statistical mechanics.

Subsequent work. Van Apeldoorn and Gilyén [23] have improved the complexity of trace-estimation and Gibbs sampling. After a personal communication with Ronald de Wolf introducing our fast version of the quantum OR lemma, the authors of Ref. [23] observed independently that the application of the quantum OR lemma [138] can be applied to decouple the dependence of m and n. As a result, Ref. [23] improved the complexity of Corollary 4.1.1 to $\tilde{O}(s(\frac{\sqrt{m}}{\epsilon^4} + \frac{\sqrt{n}}{\epsilon^5}))$ in the quantum operator model, a stronger input model than the plain one proposed by Ref. [23]. Using novel techniques, it also has improved the complexity of Corollary 4.1.2 to $\tilde{O}(\frac{B\sqrt{m}}{\epsilon^4} + \frac{B^{3.5}}{\epsilon^{7.5}})$ in the quantum input model. Note there is no explicit dependence on the rank r, which is an important advance (though it can be argued that rank r is implicitly included in the parameter B).

Open questions. This work leaves several natural open questions for future work. For example:

- Are there more examples of interesting SDPs where our form of input is meaningful? We have shown the example of learning quantum states. Intuitively, we are looking for SDP instances where the constraints are much "simpler" than the solution space. Is there any such example in the context of big data and/or machine learning?

- Our work has identified one setting where Gibbs sampling has a poly-log dependence on the dimension? Is there any other setting for the same purpose?

- For any reasonable quantum input setting, what is the effect of potential noises on quantum inputs in practice?

- Can we improve further on other parameters (e.g., the dependence on m and $1/\epsilon$)? In particular, is it possible to improve the error dependence to $\operatorname{poly}\log(1/\epsilon)$? This probably implies that we have to consider a quantum version of the interior point method.

- Are there other classes of measurements for which the quantum learning problem can solved in a computationally efficient way beyond the low-rank measurements we consider in this work? We note that most measurements of interest are not low rank (e.g. local measurements) and therefore the practical applicability of the present result is limited.

Chapter 5: Linear and Kernel-based Classification[1]

The main focus of previous chapters has been quantum speedups of optimization problems. In this chapter, we study classification problems to demonstrate quantum advantages for solving machine learning tasks.

5.1 Introduction

Motivations. Classification is a fundamental problem of supervised learning, which takes a set of data points with known classes as inputs and aims to training a model for predicting the classes of future data points. It is also ubiquitous due to its broad connections to computer vision, natural language processing, statistics, etc.

A fundamental case of classification is *linear classification*, where we are given n data points $X_1, \ldots, X_n$ in $\mathbb{R}^d$ and a label vector $y \in \{1, -1\}^n$. The goal is to find a separating hyperplane, i.e., a unit vector w in $\mathbb{R}^d$, such that

$$y_i \cdot X_i^\top w \geq 0 \quad \forall\, i \in [n]. \tag{5.1.1}$$

By taking $X_i \leftarrow (-1)^{y_i} X_i$, it reduces to a *maximin* problem $\max_w \min_i X_i^\top w \geq 0$.

[1] This chapter is based on the paper [191] under the permission of all the authors.

The approximation version of linear classification is to find a unit vector $\bar{w} \in \mathbb{R}^d$ s.t.

$$X_i^\top \bar{w} \geq \max_{w \in \mathbb{R}_d} \min_{i' \in [n]} X_{i'}^\top w - \epsilon \quad \forall i \in [n], \tag{5.1.2}$$

i.e., $\bar{w}$ approximately solves the maximin problem. More generally, we can regard a (nonlinear) classifier as a *kernel-based* classifier by replacing X_i by $\Psi(X_i)$ (Ψ being a kernel function). We will focus on algorithms finding approximate classifiers (in the sense of (5.1.2)) with *provable guarantees*.

The Perceptron Algorithm for linear classification is one of the oldest algorithms studied in machine learning [206, 218], which runs in time $O(nd/\epsilon^2)$ for finding an $\bar{w} \in \mathbb{R}^d$ satisfying (5.1.2). The state-of-the-art classical result along this line [85] solves linear classification in time $\tilde{O}((n+d)/\epsilon^2)$. A careful reader might notice that the input to linear classification is n d-dimensional vectors with total size $O(nd)$. Hence, the result of [85] is *sublinear* in its input size. To make it possible, [85] assumes the following entry-wise input model:

Input model: given any $i \in [n]$ and $j \in [d]$, the j-th entry of X_i can be recovered in $O(1)$ time.

The output of [85] is an *efficient* classical representation of $\bar{w}$ in the sense that every entry of $\bar{w}$ can be recovered with $\tilde{O}(1)$ cost. It is no surprise that $\bar{w}$ per se gives such a representation. However, there could be more *succinct and efficient representations* of $\bar{w}$, which could be reasonable alternatives of $\bar{w}$ for sublinear algorithms that run in time less the dimension of $\bar{w}$ (as we will see in the quantum case). The complexity of [85] is also optimal (up to poly-logarithmic factors) in the

above input/output model as shown by the same paper.

Recent developments in quantum computation, especially in the emerging topic of "quantum machine learning" (see the surveys [32, 51, 243]), suggest that quantum algorithms might offer significant speed-ups for optimization and machine learning problems. In particular, a quantum counterpart of the Perceptron algorithm has been proposed in [161] with improved time complexity from $O(nd/\epsilon^2)$ to $\tilde{O}(\sqrt{n}d/\epsilon^2)$ (details in related works). Motivated both by the significance of classification and the promise of quantum algorithms, we investigate the *optimal* quantum algorithm for classification. Specifically, we aim to design a quantum counterpart of [85].

It is natural to require that quantum algorithms make use of the classical input/output model as much as possible to make the comparison fair. In particular, it is favorable to avoid the use of too powerful input data structure which might render any finding of quantum speedup inconclusive, especially in light of a recent development of quantum-inspired classical machine learning algorithms (e.g., [256]). Our choice of input/output models for quantum algorithms is hence almost the same as the classical one, except we allow *coherent* queries to the entries of X_i as in (1.3.3):

Quantum input model: given any $i \in [n]$ and $j \in [d]$, the j-th entry of X_i can be recovered in $O(1)$ time *coherently*.

Coherent queries allow the quantum algorithm to query many locations in superposition, which is a *standard* assumption that accounts for many quantum speed-ups (e.g., Grover's algorithm [128]). See Section 1.3 for details.

On the other side, our output is exactly the same as classical algorithms, which guarantees no overhead when using our quantum algorithms as subroutines for any applications.

Contributions. Inspired by [85], our main contribution is a *tight* characterization (up to poly-log factors) of quantum algorithms for various classification problems in the aforementioned input/output model.

Theorem 5.1.1 (Main theorem)**.** *Given $\epsilon = \Theta(1)$, we have quantum algorithms that return an efficient representation of $\bar{w} \in \mathbb{B}_d$ for the following problems*[2]*, respectively, with complexity $\tilde{O}(\sqrt{n} + \sqrt{d})$ and high success probability:*

- *Linear classification (Theorem 5.2.3):*

$$\min_{i\in[n]} X_i^\top \bar{w} \geq \max_{w\in\mathbb{B}_d} \min_{i\in[n]} X_i^\top w - \epsilon. \tag{5.1.3}$$

- *Kernel-based classification:*

$$\min_{i\in[n]} \langle \Psi(X_i), \bar{w} \rangle \geq \max_{w\in\mathbb{B}_d} \min_{i\in[n]} \langle \Psi(X_i), w \rangle - \epsilon, \tag{5.1.4}$$

  where $k(a,b) := \langle \Psi(a), \Psi(b) \rangle$ can be the polynomial kernel $k_q(a,b) = (a^\top b)^q$ (Corollary 5.3.1) or the Gaussian kernel $k_{Gauss}(a,b) = \exp(-\|a-b\|^2)$ (Corollary 5.3.2).

[2] Here $\mathbb{B}_d$ is the ℓ_2-norm unit ball in $\mathbb{R}^d$.

- *Minimum enclosing ball (Theorem 5.3.2):*

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \leq \min_{w\in\mathbb{R}^d} \max_{i\in[n]} \|w - X_i\|^2 + \epsilon. \tag{5.1.5}$$

- *ℓ_2-margin SVM (Corollary 5.3.3):*

$$\min_{i\in[n]} (X_i^\top \bar{w})^2 \geq \max_{w\in\mathbb{R}^d} \min_{i\in[n]} 2X_i^\top w - \|w\|^2 - \epsilon. \tag{5.1.6}$$

On the other hand, we show that it requires $\Omega(\sqrt{n} + \sqrt{d})$ queries to the quantum input model to prepare such $\bar{w}$ for these classification problems (Theorem 5.4.1, Theorem 5.4.2).

Our matching upper and lower bounds $\sqrt{n}+\sqrt{d}$ give a *quadratic* improvement in both n and d comparing to the classical state-of-the-art results in [85].

Technically, our result is also inspired by the recent development of quantum semidefinite program (SDP) solvers (e.g., see Chapter 4) which provide quantum speed-ups for approximating zero-sum games for the purpose of solving SDPs. Note that such a connection was leveraged classically in another direction in a follow-up work of [85] for solving SDPs [114]. However, our algorithm is even simpler because we only use simple quantum state preparation instead of complicated quantum operations in quantum SDP solvers; this is because quantum state preparation is a direct counterpart of the ℓ_2 sampling used in [85] (see Section 5.2.1 for details). In a nutshell, our result is a demonstration of quantum speed-ups for sampling-based classical algorithms. Moreover, our algorithms are hybrid classical-quantum algo-

rithms where the quantum part is isolated pieces of state preparation connected by classical processing. All of the above suggest the possibility of implementing these algorithms on near-term quantum machines [230].

In general, we deem our result as a proposal of one end-to-end quantum application in machine learning, with both provable guarantees and the perspective of implementation (at least in prototype) on near-term quantum machines.

Related works. We make the following comparisons with existing literatures in quantum machine learning.

- The most relevant result is the quantum perceptron models in [161]. The classical perceptron method [206, 218] is a pivotal linear classification algorithm. In each iteration, it checks whether (5.1.1) holds; if not, then it searches for a violated constraint i_0 (i.e., $y_{i_0} X_{i_0}^\top \bar{w} < 0$) and update $\bar{w} \leftarrow \bar{w} + X_{i_0}$ (up to normalization). This classical perceptron method has complexity $\tilde{O}(nd/\epsilon^2)$; the quantum counterpart in [161] improved the complexity to $\tilde{O}(\sqrt{n}d/\epsilon^2)$ by applying Grover search [128] to find a violated constraint. In contrast, we quantize the sublinear algorithm for linear classification in [85] with techniques inspired by quantum SDP solvers [55]. As a result, we establish a better quantum complexity $\tilde{O}(\sqrt{n} + \sqrt{d})$.

  In addition, [161] relies on an unusual input model where a data point in $\mathbb{R}^d$ is represented by concatenating the the binary representations of the d floating point numbers; if we were only given standard inputs with entry-wise queries to the coordinates of data points, we need a cost of $\Omega(d)$ to transform the data into their input form, giving the total complexity $\tilde{O}(\sqrt{n}d)$.

The same group of authors also gave a quantum algorithm for nearest-neighbor classification with complexity $\tilde{O}(\sqrt{n})$ [271]. This complexity also depends on the sparsity of the input data; in the worst case where every data point has $\Theta(d)$ nonzero entries, the complexity becomes $\tilde{O}(\sqrt{n}d^2)$.

- There have been rich developments on quantum algorithms for linear algebraic problems. One prominent example is the quantum algorithm for solving linear systems [81, 137]; in particular, they run in time $\mathrm{poly}(\log d)$ for any sparse d-dimensional linear systems. These linear system solvers are subsequently applied to machine learning applications such as cluster assignment [194], support vector machine (SVM) [232], etc.

  However, these quantum algorithms have two drawbacks. First, they require the input matrix to be *sparse* with efficient access to nonzero elements, i.e., every row/column of the matrix has at most $\mathrm{poly}(\log d)$ nonzero elements and their indexes can be queried in $\mathrm{poly}(\log d)$ time. Second, the outputs of these algorithms are quantum states instead of classical vectors, and it takes $\Omega(d)$ copies of the quantum state to reveal one entry of the output in the worst case. More caveats are listed in [2].

  In contrast, our quantum algorithms do not have the sparsity constraint and work for arbitrary input data, and the outputs of our quantum algorithms are succinct but efficient classical representations of vectors in $\mathbb{R}^d$, which can be directly used for classical applications.

- There are two lines of quantum machine learning algorithms with different input

requirements. One of them is based on quantum principal component analysis [195] and requires purely quantum inputs.

Another line is the recent development of quantum-inspired classical poly-logarithmic time algorithms for various machine learning tasks such as recommendation systems [256], principal component analysis [255], solving linear systems [78, 122], SDPs [77], and so on. These algorithms follow a Monte-Carlo approach for low-rank matrix approximation [113] and assume the ability to take samples according to the spectral norms of all rows. In other words, these results enforce additional requirements on their input: the input matrix should not only be low-rank but also be preprocessed as the sampling data structure.

- There are also a few heuristic quantum machine learning approaches for classification [108, 141, 165] without theoretical guarantees. We, however, look forward to further experiments based on their proposals.

Notations. Throughout this chapter, we denote $X \in \mathbb{R}^{n\times d}$ to be the matrix whose i^{th} row is $X_i^\top$ for all $i \in [n]$. Without loss of generality, we assume $X_1, \ldots, X_n \in \mathbb{B}_d$, i.e., all the n data points are normalized to have ℓ_2-norm at most 1. As introduced in Section 1.3, we assume a quantum oracle O_X (a unitary on $\mathbb{C}^n \otimes \mathbb{C}^d \otimes \mathbb{C}^{d_{\text{acc}}}$) s.t.

$$O_X(|i\rangle \otimes |j\rangle \otimes |z\rangle) = |i\rangle \otimes |j\rangle \otimes |z \oplus X_{ij}\rangle \tag{5.1.7}$$

for any $i \in [n]$, $j \in [d]$ and $z \in \mathbb{C}^{d_{\text{acc}}}$ such that X_{ij} can be represented in $\mathbb{C}^{d_{\text{acc}}}$.

5.2 Linear classification

5.2.1 Techniques

At a high level, our quantum algorithm leverages ideas from both classical and quantum algorithm design. We use a primal-dual approach under the multiplicative weight framework [110], in particular its improved version in [85] by sampling the update of weight vectors. An important observation of ours is that such classical algorithms can be accelerated significantly in quantum computation, which relies on a seminal technique in quantum algorithm design: amplitude amplification and estimation [57, 128].

Multiplicative weight under a primal-dual approach. Note that linear classification is essentially a minimax problem (zero-sum game); by strong duality,

$$\sigma = \max_{w \in \mathbb{R}_d} \min_{p \in \Delta_n} p^\top X w = \min_{p \in \Delta_n} \max_{w \in \mathbb{R}_d} p^\top X w. \tag{5.2.1}$$

To find its equilibrium point, we adopt an online primal-dual approach with T rounds; at round $t \in [T]$, the primal computes $p_t \in \Delta_n$ and the dual computes $w_t \in \mathbb{R}_d$, both based on p_τ and w_τ for all $\tau \in [t-1]$. After T rounds, the average solution $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$ approximately solves the zero-sum game with high probability, i.e., $\min_{p \in \Delta_n} p^\top X \bar{w} \geq \sigma - \epsilon$.

For the primal problem, we pick p_t by the *multiplicative weight* (MW) method. Given a sequence of vectors $r_1, \ldots, r_T \in \mathbb{R}^n$, MW sets $w_1 := \mathbf{1}_n$ and for all $t \in [T]$,

$p_t := w_t/\|w_t\|_1$ and $w_{t+1}(i) := w_t(i) f_{\mathrm{w}}(-\eta r_t(i))$ for all $i \in [n]$, where f_{w} is a weight function and η is the parameter representing the step size. MW promises an upper bound on $\sum_{t=1}^{T} p_t^\top r_t$, whose precise form depends on the choice of the weight function f_{w}. The most common update is the *exponential weight update*: $f_{\mathrm{w}}(x) = e^{-x}$ [110], but in this chapter we use a quadratic weight update suggested by [85], where $w_{t+1}(i) := w_t(i)(1 - \eta r_t(i) + \eta^2 r_t(i)^2)$. In our primal problem, we set $r_t = X w_t$ for all $t \in [T]$ to find p_t.

For the dual problem, we pick w_t by the *online gradient descent* method [279]. Given a set of vectors $q_1, \ldots, q_T \in \mathbb{R}^d$ such that $\|q_i\|_2 \leq 1$. Let $w_0 := \mathbf{0}_d$, and $y_{t+1} := w_t + \frac{1}{\sqrt{T}} q_t$, $w_{t+1} := \frac{y_{t+1}}{\max\{1, \|y_{t+1}\|\}}$. Then

$$\max_{w \in \mathbb{B}_d} \sum_{t=1}^{T} q_t^\top w - \sum_{t=1}^{T} q_t^\top w_t \leq 2\sqrt{T}. \tag{5.2.2}$$

This can be regarded as a *regret* bound, i.e., $\sum_{t=1}^{T} q_t^\top w_t$ has at most a regret of $2\sqrt{T}$ compared to the best possible choice of w. In our dual problem, we set q_t as a sample of rows of X following the distribution p_t.

This primal-dual approach gives a correct algorithm with only $T = \tilde{O}(1/\epsilon^2)$ iterations. However, the primal step runs in $\Theta(nd)$ time to compute $X w_t$. To obtain an algorithm that is *sublinear* in the size of X, a key observation by [85] is to replace the precise computation of $X w_t$ by an unbiased random variable. This is achieved via ℓ_2 sampling of w: we pick $j_t \in [d]$ by $j_t = j$ with probability $w_t(j)^2/\|w_t\|^2$, and for all $i \in [n]$ we take $\tilde{v}_t(i) = X_i(j_t)\|w_t\|^2/w_t(j_t)$. The expectation of the random

variable $\tilde{v}_t(i)$ satisfies

$$\mathbb{E}[\tilde{v}_t(i)] = \sum_{j=1}^{d} \frac{w_t(j)^2}{\|w_t\|^2} \frac{X_i(j)\|w_t\|^2}{w_t(j)} = X_i w_t. \tag{5.2.3}$$

In a nutshell, the update of weight vectors in each iteration need not to be precisely computed because an ℓ_2 sample from w suffices to promise the provable guarantee of the framework. This trick improves the running time of MW to $O(n)$ and online gradient descent to $O(d)$; since there are $\tilde{O}(1/\epsilon^2)$ iterations, the total complexity is $\tilde{O}(\frac{n+d}{\epsilon^2})$ as claimed in [85].

Amplitude amplification and estimation. Consider a search problem where we are given a function $f_\omega\colon [n] \to \{-1, 1\}$ such that $f_\omega(i) = 1$ iff $i \neq \omega$. To search for ω, classically we need $\Omega(n)$ queries to f_ω as checking all n positions is the only method.

Quantumly, given a unitary U_ω such that $U_\omega|i\rangle = |i\rangle$ for all $i \neq \omega$ and $U_\omega|\omega\rangle = -|\omega\rangle$, Grover's algorithm [128] finds ω with complexity $\tilde{O}(\sqrt{n})$. Denote $|s\rangle = \frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle$ (the uniform superposition), $|s'\rangle = \frac{1}{\sqrt{n-1}} \sum_{i\in[n]/\{\omega\}} |i\rangle$, and $U_s = 2|s\rangle\langle s| - I$, the unitary U_ω reflects a state with respect to $|s'\rangle$ and the unitary U_s reflects a state with respect to $|s\rangle$. If we start with $|s\rangle$ and denote $\theta = 2\arcsin(1/\sqrt{n})$ (the angle between $U_\omega|s\rangle$ and $|s\rangle$), then the angle between $U_\omega|s\rangle$ and $U_s U_\omega|s\rangle$ is *amplified* to 2θ, and in general the angle between $U_\omega|s\rangle$ and $(U_s U_\omega)^k|s\rangle$ is $2k\theta$. To find ω, it suffices to take $k = \Theta(\sqrt{n})$ in this quantum algorithm.

This trick of alternatively applying two unitaries is called *amplitude amplifica-*

tion; in general, this provides a quadratic speedup for search. For the quantitative version of estimating θ (not only finding ω), quadratic quantum speedup also holds via an improved version of amplitude amplification called *amplitude estimation* [57].

Our **main technical contribution** is the implementations of amplitude amplification and estimation in the primal-dual approach for solving minimax problems. On the one hand, we achieve quadratic quantum speedup for multiplicative weight update, i.e., we improve the complexity from $\tilde{O}(n)$ to $\tilde{O}(\sqrt{n})$. This is because the ℓ_2 sampling of w is identical to measuring the quantum state $|w\rangle$ in the computational basis, which is prepared by amplitude amplification.

On the other hand, we also achieve quadratic quantum speedup for online gradient descent (improving $\tilde{O}(d)$ to $\tilde{O}(\sqrt{d})$). This is because the main cost of online gradient descent comes from estimating the norms $\|y_t\|$, which can be regarded as an amplitude estimation problem.

Comparison between classical and quantum results. Although our quantum algorithms enjoy quadratic speedups in n and d, their executions incur a larger dependence in ϵ: we have worst case $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^8}\big)$ compared to the classical $\tilde{O}\big(\frac{n}{\epsilon^2}+\frac{d}{\epsilon^2}\big)$ in [85]. The main reason of having a larger ϵ-dependence in quantum is because we cannot prepare the weight states in MW via those in previous iterations (i.e., the quantum state $|w_t\rangle$ cannot be prepared by $|w_{t-1}\rangle$), and we have to start over every time; this is an intrinsic difficulty due to quantum state preparation.

Therefore, there is a trade-off between [85] and our results for arbitrary ϵ: we provide faster training of the classifiers if we allow a constant error, while the

classical algorithms in [85] might work better if we require high-accuracy classifiers.

5.2.2 Quantum speedup for multiplicative weights

First, we give a quantum algorithm for linear classification with complexity $\tilde{O}(\sqrt{n})$:

Theorem 5.2.1. *With success probability at least 2/3, Algorithm 5.1 returns a succinct classical representation of a vector $\bar{w} \in \mathbb{R}^d$ such that*

$$X_i \bar{w} \geq \max_{w \in \mathbb{B}_d} \min_{i' \in [n]} X_{i'} w - \epsilon \quad \forall\, i \in [n], \tag{5.2.4}$$

using $\tilde{O}\left(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon^2}\right)$ quantum gates.

Algorithm 5.1: Quantum linear classification algorithm.

Input: $\epsilon > 0$, a quantum oracle O_X for $X \in \mathbb{R}^{n \times d}$.
Output: $\bar{w}$ that satisfies (5.2.4).

1 Let $T = 23^2 \epsilon^{-2} \log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}} \sum_{i \in [n]} |i\rangle$;
2 **for** $t = 1$ **to** T **do**
3 Define[3] $w_t := \frac{y_t}{\max\{1, \|y_t\|\}}$;
4 Measure $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
5 Define $y_{t+1} := y_t + \frac{1}{\sqrt{2T}} X_{i_t}$;
6 Choose $j_t \in [d]$ by $j_t = j$ with probability $\frac{w_t(j)^2}{\|w_t\|^2}$;
7 Denote $\tilde{v}_t(i) = X_i(j_t) \frac{\|w_t\|^2}{w_t(j_t)}$, $v_t(i) = \min\{1/\eta, \max\{-1/\eta, \tilde{v}_t(i)\}\}$, and $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$ for all $i \in [n]$. Implement a quantum oracle O_t such that for all $i \in [n]$, $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ by Algorithm 5.3 in Section 5.2.2.2;
8 Prepare $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2} \sum_{i \in [n]} u_{t+1}(i)|i\rangle$ by applying Algorithm 5.2 to O_t;
9 Return $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$;

[3] By defining w_t here, we do not write down the whole vector but we construct any query to its entries in $O(1)$ time. For example, the i^{th} coordinate of w_t is $w_t(i) = \frac{y_t(i)}{\max\{1, \|y_t\|\}}$, constructed by one query to $y_t(i)$. The y_{t+1} in Line 5 is defined in the same sense.

Note that Algorithm 5.1 is inspired by the classical sublinear algorithm [85] by using online gradient descent in Line 5 and ℓ^2 sampling in Line 6 and Line 7. However, to achieve the $\tilde{O}(\sqrt{n})$ quantum complexity we use two quantum building blocks: a state preparation procedure in Line 7, and an oracle implementation procedure in Line 8; their details are covered in Section 5.2.2.2 and Section 5.2.2.1, respectively. The full proof of Theorem 5.2.1 is given in Section 5.2.2.3.

5.2.2.1 Quantum state preparation with oracles

We use the following result for quantum state preparation (see, e.g., [129]):

Proposition 5.2.1. *Assume that $a \in \mathbb{C}^n$, and we are given a unitary oracle O_a such that $O|i\rangle|0\rangle = |i\rangle|a_i\rangle$ for all $i \in [n]$. Then Algorithm 5.2 takes $O(\sqrt{n})$ calls to O_a for preparing the quantum state $\frac{1}{\|a\|_2}\sum_{i\in[n]} a_i|i\rangle$ with success probability $1 - O(1/n)$.*

Note that the coefficient in (5.2.7) satisfies $\frac{\|a\|_2}{\sqrt{n}a_{\max}} \geq \frac{1}{\sqrt{n}}$; therefore, applying amplitude amplification for $O(\sqrt{n})$ times indeed promises that we obtain $|1\rangle$ on the second system with success probability $1 - O(1/n)$, i.e., the state $\frac{1}{\|a\|_2}\sum_{i\in[n]} a_i|i\rangle$ is prepared in the first system.

Remark 5.2.1. *Algorithm 5.2 is incomparable to state preparation via quantum random access memory (QRAM). QRAM relies on the weak assumption that we start from zero, and every added datum is processed in poly-logarithmic time. In total, this takes at least linear time in the size of the data (see, for instance, [167]). For the task of Proposition 5.2.1, QRAM takes at least $\Omega(n)$ cost.*

In this paper, we use the standard model where the input is formulated as an

Algorithm 5.2: Prepare a pure state given an oracle to its coefficients.

1 Apply Dürr-Høyer's algorithm [100] to find $a_{\max} := \max_{i\in[n]} |a_i|$ in $O(\sqrt{n})$ time;

2 Prepare the uniform superposition $\frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle$;

3 Perform the following unitary transformations:

$$\frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle \overset{O_a}{\longmapsto} \frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle|a_i\rangle \tag{5.2.5}$$

$$\mapsto \frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle|a_i\rangle \Big(\frac{a_i}{a_{\max}}|0\rangle + \sqrt{1 - \frac{|a_i|^2}{a_{\max}^2}}|1\rangle \Big)$$

$$\overset{O_a^{-1}}{\longmapsto} \frac{1}{\sqrt{n}} \sum_{i\in[n]} |i\rangle|0\rangle \Big(\frac{a_i}{a_{\max}}|0\rangle + \sqrt{1 - \frac{|a_i|^2}{a_{\max}^2}}|1\rangle \Big); \tag{5.2.6}$$

4 Delete the second system in Eq. (5.2.6), and rewrite the state as

$$\frac{\|a\|_2}{\sqrt{n} a_{\max}} \cdot \Big(\frac{1}{\|a\|_2} \sum_{i\in[n]} a_i |i\rangle \Big) |1\rangle + |a^\perp\rangle|0\rangle, \tag{5.2.7}$$

where $|a^\perp\rangle := \frac{1}{\sqrt{n}} \sum_{i\in[n]} \sqrt{1 - \frac{|a_i|^2}{a_{\max}^2}} |i\rangle$ is a garbage state;

5 Apply amplitude amplification [57] for the state in (5.2.7) conditioned on the second system being 1. Return the output;

oracle, also widely assumed and used in existing quantum algorithm literatures (e.g., [55, 81, 128, 137]). Under the standard model, Algorithm 5.2 prepares states with only $O(\sqrt{n})$ cost.

Nevertheless, it is an interesting question to ask whether there is a $\mathrm{poly}(\log(nd))$*-time quantum algorithm for linear classification given the existence of a pre-loaded QRAM of X. This would require the ability to take summations of the vectors $\frac{1}{\sqrt{2T}} X_{i_t}$ in Line 5 of Algorithm 5.1 in* $\mathrm{poly}(\log(nd))$*-time as well as the ability to update the weight state u_{t+1} in Line 8 in* $\mathrm{poly}(\log(nd))$*-time, both using QRAM. These two tasks are plausible as suggested by classical poly-log time sample-based al-*

gorithms for matrix arithmetics under multiplicative weight frameworks [77], which can potentially be combined with the analysis of QRAM data structures in [167]; we leave this possibility as an open question.

5.2.2.2 Implementing the quantum oracle for weight vectors

The quantum oracle O_t in Line 7 of Algorithm 5.1 is implemented by Algorithm 5.3. For convenience, we denote $\mathrm{clip}(v, 1/\eta) := \min\{1/\eta, \max\{-1/\eta, v\}\}$ for all $v \in \mathbb{R}$.

Algorithm 5.3: Quantum oracle for updating the weight state.

Input: $w_1, \ldots, w_t \in \mathbb{R}^d$, $j_1, \ldots, j_t \in [d]$.

Output: An oracle O_t such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$.

1 Define three classical oracles: $O_{s,j}(0) = j_s$, $O_{s,w}(j_s) = \frac{\|w_s\|^2}{w_s(j_s)}$, and $O_{\mathrm{clip}}(a, b, c) = c \cdot \left(1 - \eta\,\mathrm{clip}(ab, 1/\eta) + \eta^2\,\mathrm{clip}(ab, 1/\eta)^2\right)$;

2 **for** $s = 1$ **to** t **do**

3 Perform the following maps:

$$|i\rangle|0\rangle|0\rangle|u_s(i)\rangle \xmapsto{O_{s,j}} |i\rangle|j_s\rangle|0\rangle|0\rangle|u_s(i)\rangle \tag{5.2.8}$$

$$\xmapsto{O_X} |i\rangle|j_s\rangle|X_i(j_s)\rangle|0\rangle|u_s(i)\rangle \tag{5.2.9}$$

$$\xmapsto{O_{s,w}} |i\rangle|j_s\rangle|X_i(j_s)\rangle\left|\frac{\|w_s\|^2}{w_s(j_s)}\right\rangle|u_s(i)\rangle \tag{5.2.10}$$

$$\xmapsto{O_{\mathrm{clip}}} |i\rangle|j_s\rangle|X_i(j_s)\rangle\left|\frac{\|w_s\|^2}{w_s(j_s)}\right\rangle|u_{s+1}(i)\rangle \tag{5.2.11}$$

$$\xmapsto{O_{s,w}^{-1}} |i\rangle|j_s\rangle|X_i(j_s)\rangle|0\rangle|u_{s+1}(i)\rangle \tag{5.2.12}$$

$$\xmapsto{O_X^{-1}} |i\rangle|j_s\rangle|0\rangle|0\rangle|u_{s+1}(i)\rangle \tag{5.2.13}$$

$$\xmapsto{O_{s,j}^{-1}} |i\rangle|0\rangle|0\rangle|0\rangle|u_{s+1}(i)\rangle. \tag{5.2.14}$$

Because we have stored w_s and j_s, we could construct classical oracles $O_{s,j}(0) = j_s$, $O_{s,w}(j_s) = \frac{\|w_s\|^2}{w_s(j_s)}$ with $O(1)$ complexity. In the algorithm, we first call $O_{s,j}$ to

compute j_s and store it into the second register in (5.2.8). In (5.2.9), we call the quantum oracle O_X for the value $X_i(j_s)$, which is stored into the third register. In (5.2.10), we call $O_{s,w}$ to compute $\frac{\|w_s\|^2}{w_s(j_s)}$ and store it into the fourth register. In (5.2.11), because we have $X_i(j_s)$ and $\frac{\|w_s\|^2}{w_s(j_s)}$ at hand, we could use $\tilde{O}(1)$ arithmetic computations to compute $\tilde{v}_s(i) = X_i(j_s)\|w_s\|^2/w_t(j_s)$ and

$$u_{s+1}(i) = u_s(i)\big(1 - \eta\,\mathrm{clip}(\tilde{v}_s(i), 1/\eta) + \eta^2\,\mathrm{clip}(\tilde{v}_s(i), 1/\eta)^2\big). \tag{5.2.15}$$

We then store $u_{s+1}(i)$ into the fifth register. In (5.2.12), (5.2.13), and (5.2.14), we uncompute the steps in (5.2.10), (5.2.9), and (5.2.8), respectively (we need these steps in Algorithm 5.3 to keep its unitarity).

In total, between (5.2.8)-(5.2.14) we use 2 queries to O_X and $\tilde{O}(1)$ additional arithmetic computations. Because s goes from 1 to t, in total we use $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations.

5.2.2.3 Proof of Theorem 5.2.1

To prove Theorem 5.2.1, we use the following five lemmas proved in [85] for analyzing the online gradient gradient descent and ℓ^2 sampling outcomes:

Lemma 5.2.1 (Lemma A.2 of [85])**.** *The updates of w in Line 3 and y in Line 5 satisfy*

$$\max_{w\in\mathbb{B}_n} \sum_{t\in[T]} X_{i_t} w \le \sum_{t\in[T]} X_{i_t} w_t + 2\sqrt{2T}. \tag{5.2.16}$$

Lemma 5.2.2 (Lemma 2.3 of [85]). *For any $t \in [T]$, denote p_t to be the unit vector in $\mathbb{R}^n$ such that $(p_t)_i = |\langle i|p_t\rangle|^2$ for all $i \in [n]$. Then the update for p_{t+1} in Line 8 satisfies*

$$\sum_{t\in[T]} p_t^\top v_t \leq \min_{i\in[n]} \sum_{t\in[T]} v_t(i) + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \frac{\log n}{\eta}, \tag{5.2.17}$$

where v_t^2 is defined as $(v_t^2)_i := (v_t)_i^2$ for all $i \in [n]$.

Lemma 5.2.3 (Lemma 2.4 of [85]). *With probability at least $1 - O(1/n)$,*

$$\max_{i\in[n]} \sum_{t\in[T]} \left[v_t(i) - X_i w_t\right] \leq 4\eta T. \tag{5.2.18}$$

Lemma 5.2.4 (Lemma 2.5 of [85]). *With probability at least $1 - O(1/n)$,*

$$\left| \sum_{t\in[T]} X_{i_t} w_t - \sum_{t\in[T]} p_t^\top v_t \right| \leq 10\eta T. \tag{5.2.19}$$

Lemma 5.2.5 (Lemma 2.6 of [85]). *With probability at least $3/4$,*

$$\sum_{t\in[T]} p_t^\top v_t^2 \leq 8T. \tag{5.2.20}$$

Proof. We first prove the correctness of Algorithm 5.1. By Lemma 5.2.1, we have

$$\sum_{t\in[T]} X_{i_t} w_t \geq \max_{w\in\mathbb{B}_n} \sum_{t\in[T]} X_{i_t} w - 2\sqrt{2T} \geq T\sigma - 2\sqrt{2T}. \tag{5.2.21}$$

On the other hand, Lemma 5.2.3 implies that for any $i \in [n]$,

$$\sum_{t\in[T]} X_i w_t \geq \sum_{t\in[T]} v_t(i) - 4\eta T. \tag{5.2.22}$$

Together with Lemma 5.2.2, we have

$$\sum_{t\in[T]} p_t^\top v_t \leq \min_{i\in[n]} \sum_{t\in[T]} X_i w_t + \eta \sum_{t\in[T]} p_t^\top v_t^2 + \frac{\log n}{\eta} + 4\eta T. \tag{5.2.23}$$

Plugging Lemma 5.2.4, Lemma 5.2.5, and (5.2.21) into (5.2.23), with probability at least $\frac{3}{4} - 2 \cdot O(\frac{1}{n}) \geq \frac{2}{3}$,

$$\min_{i\in[n]} \sum_{t\in[T]} X_i w_t \geq -\frac{\log n}{\eta} - 8\eta T - 4\eta T + T\sigma - 2\sqrt{2T} - 10\eta T \tag{5.2.24}$$

$$\geq T\sigma - 22\eta T - \frac{\log n}{\eta}. \tag{5.2.25}$$

Since $T = 23^2\epsilon^{-2}\log n$ and $\eta = \sqrt{\frac{\log n}{T}}$, we have

$$\min_{i\in[n]} X_i \bar{w} = \frac{1}{T} \min_{i\in[n]} \sum_{t=1}^{T} X_i w_t \geq \sigma - 23\sqrt{\frac{\log n}{T}} \geq \sigma - \epsilon \tag{5.2.26}$$

with probability at least $2/3$, which is exactly (5.2.4).

Now we analyze the gate complexity of Algorithm 5.1. To run Line 3 and Line 5, we need d time and space to compute and store w_t and y_{t+1}; for all $t \in [T]$, this takes total complexity $O(dT) = \tilde{O}(\frac{d}{\epsilon^2})$. It takes another $O(dT) = \tilde{O}(\frac{d}{\epsilon^2})$ cost to compute

j_t for all $t \in [T]$ in Line 6.

The quantum part of Algorithm 5.1 mainly happens at Line 7 and Line 8, where we prepare the quantum state $|p_{t+1}\rangle$ instead of computing the coefficients $u_{t+1}(i)$ one by one for all $i \in [n]$. To be more specific, we construct an oracle O_t such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$. This is achieved iteratively, i.e., at iteration s we map $|i\rangle|u_s(i)\rangle$ to $|i\rangle|u_{s+1}(i)\rangle$. The full details are given in Algorithm 5.3 in Section 5.2.2.2; in total, one query to O_t is implemented by $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations.

Finally, we prepare the state $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2} \cdot \sum_{i\in[n]} u_{t+1}(i)|i\rangle$ in Line 8 using $O(\sqrt{n})$ calls to O_t, which are equivalent to $O(\sqrt{n}t)$ calls to O_X by Line 7 and $\tilde{O}(\sqrt{n}t)$ additional arithmetic computations. Therefore, the total complexity of Line 8 for all $t \in [T]$ is

$$\sum_{t=1}^{T} \tilde{O}(\sqrt{n}t) = \tilde{O}(\sqrt{n}T^2) = \tilde{O}\Big(\frac{\sqrt{n}}{\epsilon^4}\Big). \tag{5.2.27}$$

In all, the total complexity of Algorithm 5.1 is $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon^2}\big)$, giving our statement.

Finally, the output $\bar{w}$ has a succinct classical representation with space complexity $O(\log n/\epsilon^2)$. To achieve this, we save $2T = O(\log n/\epsilon^2)$ values in Algorithm 5.1: $i_1, \ldots, i_T$ and $\|y_1\|, \ldots, \|y_T\|$; it then only takes $O(\log n/\epsilon^2)$ cost to recover any coordinate of $\bar{w}$ by Line 3 and Line 5. □

Remark 5.2.2. *Theorem 5.2.1 could also be applied to the PAC model. For the case where there exists a hyperplane classifying all data points correctly with margin σ, and assume that the margin is not small in the sense that $\frac{1}{\sigma^2} < d$, PAC learning*

theory implies that the number of examples needed for training a classifier of error δ is $O(1/\sigma^2\delta)$. As a result, we have a quantum algorithm that computes a $\sigma/2$-approximation to the best classifier with cost

$$\tilde{O}\Big(\frac{\sqrt{1/\sigma^2\delta}}{\sigma^4}+\frac{d}{\sigma^2}\Big)=\tilde{O}\Big(\frac{1}{\sigma^5\sqrt{\delta}}+\frac{d}{\sigma^2}\Big). \tag{5.2.28}$$

This is better than the classical complexity $O(\frac{1}{\sigma^4\delta}+\frac{d}{\sigma^2})$ in [85] as long as $\delta\leq\sigma^2$, which is plausible under the assumption that the margin σ is large.

5.2.3 Quantum speedup for online gradient descent

Norm estimation by amplitude estimation. We further improve the dependence in d to $\tilde{O}(\sqrt{d})$. To achieve this, we cannot update w_t and y_t in Line 3 and Line 5 by each coordinate because storing w_t or y_t would already take cost at least d. We solve this issue by not updating w_t and y_t explicitly and instead only computing $\|y_t\|$ for all $i\in[T]$. This norm estimation is achieved by the following lemma:

Lemma 5.2.6. *Assume that $F\colon[d]\to[0,1]$ with a quantum oracle $O_F|i\rangle|0\rangle=|i\rangle|F(i)\rangle$ for all $i\in[d]$. Denote $m=\frac{1}{d}\sum_{i=1}^{d}F(i)$. Then for any $\delta>0$, there is a quantum algorithm that uses $O(\sqrt{d}/\delta)$ queries to O_F and returns an $\tilde{m}$ such that $|\tilde{m}-m|\leq\delta m$ with probability at least 2/3.*

Our proof of Lemma 5.2.6 is based on amplitude estimation:

Theorem 5.2.2 (Theorem 15 of [57])**.** *For any $0<\epsilon<1$ and Boolean function $f\colon[d]\to\{0,1\}$ with quantum oracle $O_f|i\rangle|0\rangle=|i\rangle|f(i)\rangle$ for all $i\in[d]$, there is a*

quantum algorithm that outputs an estimate $\hat{t}$ *to* $t = |f^{-1}(1)|$ *such that*

$$|\hat{t} - t| \leq \epsilon t \tag{5.2.29}$$

with probability at least $8/\pi^2$*, using* $O(\frac{1}{\epsilon}\sqrt{\frac{d}{t}})$ *evaluations of* O_f*. If* $t = 0$*, the algorithm outputs* $\hat{t} = 0$ *with certainty and* O_f *is evaluated* $O(\sqrt{d})$ *times.*

Proof. Assume that $F(i)$ has l bits for precision for all $i \in [d]$ (in our paper, we take $l = O(1)$, say $l = 64$ for double float precision), and for all $k \in [l]$ denote $F_k(i)$ as the k^{th} bit of $F(i)$; denote $n_k = \sum_{i\in[d]} F_k(i)$.

We apply Theorem 7.4.5 to all the l bits of n_k using $O(\sqrt{d}/\delta)$ queries (taking $\epsilon = \delta/2$), which gives an approximation $\hat{n}_k$ of n_k such that with probability at least $8/\pi^2$ we have $|n_k - \hat{n}_k| \leq \delta n_k/2$ if $n_k \geq 1$, and $\hat{n}_k = 0$ if $n_k = 0$. Running this procedure for $\Theta(\log l)$ times and take the median of all returned $\hat{n}_k$, and do this for all $k \in [l]$, Chernoff's bound promises that with probability $2/3$ we have

$$|n_k - \hat{n}_k| \leq \delta n_k \quad \forall\, k \in [l]. \tag{5.2.30}$$

As a result, if we take $\tilde{m} = \frac{1}{d}\sum_{k\in[l]} \frac{\hat{n}_k}{2^k}$, and observe that $m = \frac{1}{d}\sum_{k\in[l]} \frac{n_k}{2^k}$, with probability at least $2/3$ we have

$$|\tilde{m} - m| \leq \frac{1}{d}\sum_{k\in[l]} \left|\frac{\hat{n}_k}{2^k} - \frac{n_k}{2^k}\right| \leq \frac{1}{d}\sum_{k\in[l]} \frac{\delta n_k}{2^k} = \delta m. \tag{5.2.31}$$

The total quantum query complexity is $O(l \log l \cdot \sqrt{d}/\delta) = O(\sqrt{d}/\delta)$. □

Quantum algorithm with $\tilde{O}(\sqrt{d})$ cost. Instead of updating y_t explicitly in Line 5 of Algorithm 5.1, we save the i_t for all $t \in [T]$ in Line 4, which only takes $\tilde{O}(1/\epsilon^2)$ cost in total but we can directly generate y_t given $i_1, \ldots, i_t$. Furthermore, notice that the probabilities in the ℓ^2 sampling in Line 6 do not change because $w_t(j)^2/\|w_t\|^2 = y_t(j)^2/\|y_t\|^2$; it suffices to replace $\tilde{v}_t(i) = X_i(j_t)\|w_t\|^2/w_t(j_t)$ by $\tilde{v}_t(i) = X_i(j_t)\|y_t\|^2/(y_t(j_t)\max\{1, \|y_t\|\})$ in Line 7. These observations result in Algorithm 5.4 with the following result:

Theorem 5.2.3. *With success probability at least $2/3$, there is a quantum algorithm that returns a succinct classical representation of a vector $\bar{w} \in \mathbb{R}^d$ such that*

$$X_i \bar{w} \geq \max_{w \in \mathbb{B}_d} \min_{i' \in [n]} X_{i'} w - \epsilon \quad \forall\, i \in [n], \tag{5.2.32}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$ quantum gates.

Proof. For clarification, for all $i \in [n]$ we denote

$$\tilde{v}_{t,\mathrm{approx}}(i) = \frac{X_i(j_t)\widetilde{\|y_t\|}^2}{y_t(j_t)\max\{1, \widetilde{\|y_t\|}\}}, \quad \tilde{v}_{t,\mathrm{true}}(i) = \frac{X_i(j_t)\|y_t\|^2}{y_t(j_t)\max\{1, \|y_t\|\}}. \tag{5.2.33}$$

In other words, the $\tilde{v}_t$ in Line 7 of Algorithm 5.4 is $\tilde{v}_{t,\mathrm{approx}}$, an approximation of $\tilde{v}_{t,\mathrm{true}}$. We prove:

$$|\tilde{v}_{t,\mathrm{approx}}(i) - \tilde{v}_{t,\mathrm{true}}(i)| \leq \eta \quad \forall\, i \in [n]. \tag{5.2.34}$$

[4]The meaning of the definition here is the same as Footnote 3 in Algorithm 5.1.

Algorithm 5.4: Quantum linear classification algorithm with $\tilde{O}(\sqrt{d})$ cost.

Input: $\epsilon > 0$, a quantum oracle O_X for $X \in \mathbb{R}^{n\times d}$.
Output: $\bar{w}$ that satisfies (5.2.4).

1 Let $T = 27^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
2 **for** $t = 1$ **to** T **do**
3 Measure $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
4 Define[4] $y_{t+1} := y_t + \frac{1}{\sqrt{2T}}X_{i_t}$;
5 Apply Lemma 5.2.6 for $2\lceil\log T\rceil$ times to estimate $\|y_t\|^2$ with precision $\delta = \eta^2$, and take the median of all the $2\lceil\log T\rceil$ outputs, denoted $\widetilde{\|y_t\|}^2$;
6 Choose $j_t \in [d]$ by $j_t = j$ with probability $y_t(j)^2/\|y_t\|^2$, which is achieved by applying Algorithm 5.2 to prepare the quantum state $|y_t\rangle$ and measure in the computational basis;
7 For all $i \in [n]$, denote $\tilde{v}_t(i) = X_i(j_t)\widetilde{\|y_t\|}^2/\big(y_t(j_t)\max\{1, \widetilde{\|y_t\|}\}\big)$, $v_t(i) = \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$, and $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Apply Algorithm 5.3 to prepare an oracle O_t such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$, using $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations;
8 Prepare $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]}u_{t+1}(i)|i\rangle$ using Algorithm 5.2 and O_t;
9 Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T}\frac{y_t}{\max\{1,\widetilde{\|y_t\|}\}}$;

Without loss generality, we can assume that $\tilde{v}_{t,\mathrm{true}}(i), \tilde{v}_{t,\mathrm{approx}}(i) \le 1/\eta$; otherwise, they are both truncated to $1/\eta$ by the clip function in Line 7 and no error occurs. For convenience, we denote $m = \|y_t\|^2$ and $\tilde{m} = \widetilde{\|y_t\|}^2$. Then

$$|\tilde{v}_{t,\mathrm{approx}}(i) - \tilde{v}_{t,\mathrm{true}}(i)| = \tilde{v}_{t,\mathrm{true}}(i)\cdot\left|\frac{\tilde{v}_{t,\mathrm{approx}}(i)}{\tilde{v}_{t,\mathrm{true}}(i)} - 1\right| \le \frac{1}{\eta}\cdot\left|\frac{\tilde{v}_{t,\mathrm{approx}}(i)}{\tilde{v}_{t,\mathrm{true}}(i)} - 1\right|. \quad (5.2.35)$$

When $\|y_t\| \ge 1$ we have $\frac{\tilde{v}_{t,\mathrm{approx}}(i)}{\tilde{v}_{t,\mathrm{true}}(i)} = \frac{\tilde{m}}{m}$; when $\|y_t\| \le 1$ we have $\frac{\tilde{v}_{t,\mathrm{approx}}(i)}{\tilde{v}_{t,\mathrm{true}}(i)} = \sqrt{\frac{\tilde{m}}{m}}$. Because in Line 5 $\widetilde{\|y_t\|}^2$ is the median of $2\lceil\log T\rceil$ executions of Lemma 5.2.6, with failure probability at most $1 - (2/3)^{2\log T} = O(1/T^2)$ we have $|\frac{\tilde{m}}{m} - 1| \le \delta$; given there are T iterations in total, the probability that Line 5 always succeeds is at least

$1 - T \cdot O(1/T^2) = 1 - o(1)$, and we have

$$\left|\frac{\tilde{m}}{m} - 1\right|, \ \left|\sqrt{\frac{\tilde{m}}{m}} - 1\right| \leq \delta. \tag{5.2.36}$$

Plugging this into (5.2.35), we have

$$|\tilde{v}_{t,\text{approx}}(i) - \tilde{v}_{t,\text{true}}(i)| \leq \frac{\delta}{\eta} = \eta, \tag{5.2.37}$$

which proves (5.2.34).

Now we prove the correctness of Algorithm 5.4. By (5.2.34) and Lemma 5.2.3, with probability at least $1 - O(1/n)$ we have

$$\max_{i\in[n]} \sum_{t\in[T]} \left[v_t(i) - X_i w_t\right] \leq 4\eta T + \eta T = 5\eta T, \tag{5.2.38}$$

where $w_t = \frac{y_t}{\max\{1, \widetilde{\|y_t\|}\}}$ for all $t \in [T]$. By (5.2.34) and Lemma 5.2.4, with probability at least $1 - O(1/n)$ we have

$$\left|\sum_{t\in[T]} X_{i_t} w_t - \sum_{t\in[T]} p_t^\top v_t\right| \leq 10\eta T + \eta T = 11\eta T; \tag{5.2.39}$$

by (5.2.34) and Lemma 5.2.5, with probability at least $3/4$ we have

$$\sum_{t\in[T]} p_t^\top v_t^2 \leq 8T + 2T = 10T. \tag{5.2.40}$$

As a result, similar to the proof of Theorem 5.2.1, we have

$$\min_{i\in[n]} \sum_{t\in[T]} X_i w_t \geq -\frac{\log n}{\eta} - 10\eta T - 5\eta T + T\sigma - 2\sqrt{2T} - 11\eta T \tag{5.2.41}$$

$$\geq T\sigma - 26\eta T - \frac{\log n}{\eta}. \tag{5.2.42}$$

Since $T = 27^2\epsilon^{-2}\log n$ and $\eta = \sqrt{\frac{\log n}{T}}$, we have

$$\min_{i\in[n]} X_i\bar{w} = \frac{1}{T}\min_{i\in[n]} \sum_{t=1}^{T} X_i w_t \geq \sigma - 27\sqrt{\frac{\log n}{T}} \geq \sigma - \epsilon \tag{5.2.43}$$

with probability at least $2/3$, which is exactly (5.2.32).

It remains to analyze the time complexity. Same as the proof of Theorem 5.2.1, the complexity in n is $\tilde{O}(\frac{\sqrt{n}}{\epsilon^4})$. It remains to show that the complexity in d is $\tilde{O}(\frac{\sqrt{n}}{\epsilon^8})$. The cost in d in Algorithm 5.1 and Algorithm 5.4 differs at Line 5 and Line 6. We first look at Line 5; because

$$y_t = \frac{1}{\sqrt{2T}} \sum_{\tau=1}^{T} X_{i_\tau}, \tag{5.2.44}$$

one query to a coefficient of y_t takes $t = \tilde{O}(1/\epsilon^2)$ queries to O_X. Next, since $X_i \in \mathbb{B}_n$ for all $i \in [n]$, we know that $X_{ij} \in [-1, 1]$ for all $i \in [n]$, $j \in [d]$; to apply Lemma 5.2.6 (F should have image domain in $[0, 1]$) we need to renormalize y_t by a factor of $t = \tilde{O}(1/\epsilon^2)$. In addition, notice that $\delta = \eta^2 = \Theta(\epsilon^2)$; as a result, the query complexity of executing Lemma 5.2.6 is $\tilde{O}(\sqrt{d}/\epsilon^2)$. Finally, there are in

total $T = \tilde{O}(1/\epsilon^2)$ iterations. Therefore, the total complexity in Line 5 is

$$\tilde{O}\Big(\frac{1}{\epsilon^2}\Big) \cdot \tilde{O}\Big(\frac{1}{\epsilon^2}\Big) \cdot \tilde{O}\Big(\frac{\sqrt{d}}{\epsilon^2}\Big) \cdot \tilde{O}\Big(\frac{1}{\epsilon^2}\Big) = \tilde{O}\Big(\frac{\sqrt{d}}{\epsilon^8}\Big). \tag{5.2.45}$$

Regarding the complexity in d in Line 6, the cost is to prepare the pure state $|y_t\rangle$ whose coefficient is proportional to y_t. To achieve this, we need $t = \tilde{O}(1/\epsilon^2)$ queries to O_X (for summing up the rows $X_{i_1}, \ldots, X_{i_t}$) such that we have an oracle O_{y_t} satisfying $O_{y_t}|j\rangle|0\rangle = |j\rangle|y_t(j)\rangle$ for all $j \in [d]$. By Algorithm 5.2, the query complexity of preparing $|y_t\rangle$ using O_{y_t} is $O(\sqrt{d})$. Because there are in total $T = \tilde{O}(1/\epsilon^2)$ iterations, the total complexity in Line 6 is

$$\tilde{O}\Big(\frac{1}{\epsilon^2}\Big) \cdot O(\sqrt{d}) \cdot \tilde{O}\Big(\frac{1}{\epsilon^2}\Big) = \tilde{O}\Big(\frac{\sqrt{d}}{\epsilon^4}\Big). \tag{5.2.46}$$

In all, the total complexity in d is $\tilde{O}(\sqrt{d}/\epsilon^8)$ as dominated by (5.2.45). Finally, $\bar{w}$ has a succinct classical representation: using $i_1, \ldots, i_T$ obtained from Line 3 and $\widetilde{\|y_1\|}^2, \ldots, \widetilde{\|y_T\|}^2$ obtained from Line 5, we could restore a coordinate of $\bar{w}$ in time $T = \tilde{O}(1/\epsilon^2)$. □

Remark 5.2.3. *For practical applications of linear classification, typically the number of data points n is larger than the dimension d, so in practice Theorem 5.2.1 might perform better than Theorem 5.2.3. Nevertheless, the $\tilde{O}(\sqrt{d})$ complexity in Theorem 5.2.3 matches our quantum lower bound (see Theorem 5.4.1).*

5.3 Applications

As introduced in Section 5.2.1, the ℓ_2 sampling of w picks $j_t \in [d]$ by $j_t = j$ with probability $w(j)^2/\|w\|^2$, and the expectation of the random variable $X_i(j_t)\|w\|^2/w(j_t)$ is $X_i w$. Here, if we consider some alternate random variables, we could give unbiased estimators of nonlinear functions of X. We first look at the general case of applying kernel functions [242] in Section 5.3.1. We then look at the special case of quadratic problems in Section 5.3.2 as they enjoy simple forms that can be applied to finding minimum enclosing balls [240] and ℓ_2-margin support vector machines [253].

5.3.1 Kernel methods

Having quantum algorithms for solving linear classification at hand, it is natural to consider linear classification under kernels. Let $\Psi\colon \mathbb{R}^d \mapsto \mathcal{H}$ be a mapping into a reproducing kernel Hilbert space (RKHS), and the problem is to find the classifier $h \in \mathcal{H}$ that solves the maximin problem

$$\sigma = \max_{h\in\mathcal{H}} \min_{i\in[n]} \langle h, \Psi(X_i)\rangle, \tag{5.3.1}$$

where the kernel is defined as $k(a,b) := \langle \Psi(a), \Psi(b)\rangle$ for all $a, b \in \mathbb{R}^d$.

Classically, [85] gave the following result for classification under efficiently-computable kernels, following the linear classification algorithm therein:

Theorem 5.3.1 (Lemma 5.3 of [85])**.** *Denote T_k as the time cost for computing $k(X_i, X_j)$ for some $i, j \in [n]$, and denote L_k as the time cost for computing a ran-*

dom variable $\tilde{k}(X_i, X_j)$ *for some* $i, j \in [n]$ *such that* $\mathbb{E}[\tilde{k}(X_i, X_j)] = k(X_i, X_j)$ *and* $\mathrm{Var}[k(X_i, X_j)] \leq 1$. *Then there is a classical algorithm that runs in time*

$$\tilde{O}\Big(\frac{L_k n + d}{\epsilon^2} + \min\Big\{\frac{T_k}{\epsilon^4}, \frac{L_k}{\epsilon^6}\Big\}\Big) \tag{5.3.2}$$

and returns a vector $\bar{h} \in \mathcal{H}$ *such that with high success probability* $\langle \bar{h}, \Psi(X_i)\rangle \geq \sigma - \epsilon$ *for all* $i \in [n]$.

Algorithm 5.5: Quantum kernel-based classification.

Input: $\epsilon > 0$, a quantum oracle O_X for $X \in \mathbb{R}^{n\times d}$.
Output: $\bar{w}$ that satisfies (5.2.4).

Let $T = 27^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$, $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
for $t = 1$ **to** T **do**
  Measure $|p_t\rangle$ in the computational basis and denote the output as $i_t \in [n]$;
  Define $y_{t+1} := y_t + \frac{1}{\sqrt{2T}}\Psi(X_{i_t})$;
  Apply Lemma 5.2.6 for $2\lceil\log T\rceil$ times to estimate $\|y_t\|^2$ with precision $\delta = \eta^2$, and take the median of all the $2\lceil\log T\rceil$ outputs, denoted $\widetilde{\|y_t\|}^2$;
  Choose $j_t \in [d]$ by $j_t = j$ with probability $y_t(j)^2/\|y_t\|^2$, which is achieved by applying Algorithm 5.2 to prepare the quantum state $|y_t\rangle$ and measure in the computational basis;
  For all $i \in [n]$, denote $\tilde{v}_t(i) = \frac{\Psi(X_i)(j_t)\widetilde{\|y_t\|}^2}{y_t(j_t)\max\{1,\widetilde{\|y_t\|}\}}$, $v_t(i) = \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$, and $u_{t+1}(i) = u_t(i)\cdot(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Apply Algorithm 5.3 to prepare an oracle O_t such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$, using $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations;
  Prepare $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]} u_{t+1}(i)|i\rangle$ using Algorithm 5.2 and O_t;
Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T}\frac{y_t}{\max\{1,\widetilde{\|y_t\|}\}}$;

Quantumly, we give Algorithm 5.5 for classification under kernels based on Algorithm 5.1. Theorem 5.2.3 and Theorem 5.3.1 imply that our quantum kernel-

based classifier has time complexity

$$\tilde{O}\Big(\frac{L_k\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8} + \min\Big\{\frac{T_k}{\epsilon^4}, \frac{L_k}{\epsilon^6}\Big\}\Big). \tag{5.3.3}$$

For polynomial kernels of degree q, i.e., $k_q(x,y) = (x^\top y)^q$, we have $L_{k_q} = q$ by taking the product of q independent ℓ_2 samples (this is an unbiased estimator of $(x^\top y)^q$ and the variance of each sample is at most 1). As a result of (5.3.3),

Corollary 5.3.1. *For the polynomial kernel of degree q, there is a quantum algorithm that solves the classification task within precision ϵ with gate complexity $\tilde{O}\big(\frac{q\sqrt{n}}{\epsilon^4} + \frac{q\sqrt{d}}{\epsilon^8}\big)$.*

Compared to the classical complexity $\tilde{O}\big(\frac{q(n+d)}{\epsilon^2} + \min\big\{\frac{d\log q}{\epsilon^4}, \frac{q}{\epsilon^6}\big\}\big)$ in Corollary 5.4 of [85], our quantum algorithm gives quadratic speedups in n and d.

For Gaussian kernels, i.e., $k_{\text{Gauss}}(x,y) = \exp(-\|x-y\|^2)$, Corollary 5.5 of [85] proved that $L_{k_{\text{Gauss}}} = 1/s^4$ if the Gaussian has standard deviation s. As a result,

Corollary 5.3.2. *For the polynomial kernel of degree q, there is a quantum algorithm that solves the classification task within precision ϵ with gate complexity $\tilde{O}\big(\frac{\sqrt{n}}{s^4\epsilon^4} + \frac{\sqrt{d}}{s^4\epsilon^8}\big)$.*

This still gives quadratic speedups in n and d compared to the classical complexity $\tilde{O}\big(\frac{n+d}{s^4\epsilon^2} + \min\big\{\frac{d}{\epsilon^4}, \frac{1}{s^4\epsilon^6}\big\}\big)$ in Corollary 5.5 of [85].

5.3.2 Quadratic machine learning problems

We consider the maximin problem of a quadratic function:

$$\max_{w\in\mathbb{R}^d}\min_{p\in\Delta_n} p^\top(b+2Xw-\mathbf{1}_n\|w\|^2)=\max_{w\in\mathbb{R}^d}\min_{i\in[n]} b_i+2X_iw-\|w\|^2, \tag{5.3.4}$$

where $b\in\mathbb{R}^n$ and $X\in\mathbb{R}^{n\times d}$. Note that the function $b_i+2X_iw-\|w\|^2$ in Eq. (5.3.4) is 2-strongly convex; as a result, the regret of the online gradient descent after T rounds can be improved to $O(\log T)$ by [250] instead of $O(\sqrt{T})$ as in Eq. (5.2.2). In addition, ℓ_2 sampling of the w in Algorithm 5.1 still works: consider the random variable $w=b_i+\frac{2X_i(j)\|w\|^2}{w(j)}-\|w\|^2$ where $j=k$ with probability $\frac{w(k)^2}{\|w\|^2}$. Then the expectation of w is

$$\mathbb{E}[X]=\sum_{j=1}^{d}\frac{w(j)^2}{\|w\|^2}\left(b_i+\frac{2X_i(j)\|w\|^2}{w(j)}-\|w\|^2\right)=b_i+2X_iw-\|w\|^2, \tag{5.3.5}$$

i.e., w is an unbiased estimator of the quadratic form in (5.3.4). As a result, given the quantum oracle O_X in (5.1.7), we could give sublinear quantum algorithms for such problems; these include two important problems: minimum enclosing balls (MEB) and ℓ_2-margin supper vector machines (SVM).

5.3.2.1 Minimum enclosing ball

In the minimum enclosing ball (MEB) problem we have $b_i=-\|X_i\|^2$ for all $i\in[n]$; Eq. (5.3.4) then becomes $\max_{w\in\mathbb{R}^d}\min_{i\in[n]}-\|X_i\|^2+2X_iw-\|w\|^2=$

$-\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w-X_i\|^2$, which is the smallest radius of the balls that contain all the data points $X_1,\ldots,X_n$.

Denote $\sigma_{\mathrm{MEB}}=\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w-X_i\|^2$, the following theorem is a direct consequence of Theorem 5.2.1 (see also Theorem 3.1 in [85]) and Theorem 5.2.3:

Theorem 5.3.2. *There is a quantum algorithm that returns a vector $\bar{w}\in\mathbb{R}^d$ such that with probability at least $2/3$,*

$$\max_{i\in[n]}\|\bar{w}-X_i\|^2\leq\sigma_{\mathrm{MEB}}+\epsilon, \tag{5.3.6}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{d}{\epsilon}\big)$ quantum gates; the quantum gate complexity can also be improved to $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^2}\big)$.

5.3.2.2 ℓ_2-margin SVM

To estimate the margin of a support vector machine (SVM) in ℓ_2-norm, we take $b_i=0$ for all $i\in[n]$; Eq. (5.3.4) then becomes solving $\sigma_{\mathrm{SVM}}:=\max_{w\in\mathbb{R}^d}\min_{i\in[n]}2X_iw-\|w\|^2$.

Notice that $\sigma_{\mathrm{SVM}}\geq 0$ because $2X_iw-\|w\|^2=0$ for all $i\in[n]$ when $w=0$. For the case $\sigma_{\mathrm{SVM}}>0$ and taking $0<\epsilon<\sigma_{\mathrm{SVM}}$, similar to Theorem 5.3.2 we have:

Corollary 5.3.3. *There is a quantum algorithm that returns a vector $\bar{w}\in\mathbb{R}^d$ such that with probability at least $2/3$,*

$$\min_{i\in[n]}2X_i\bar{w}-\|\bar{w}\|^2\geq\sigma_{\mathrm{SVM}}-\epsilon>0, \tag{5.3.7}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{d}{\epsilon}\big)$ *quantum gates; the quantum gate complexity can also be improved to* $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^7}\big)$.

Note that (5.3.7) implies that $X_i\bar{w} > 0$ for all $i \in [n]$; furthermore, by the AM-GM inequality we have $\frac{(X_i\bar{w})^2}{\|\bar{w}\|^2} + \|\bar{w}\|^2 \geq 2X_i\bar{w}$, and hence

$$\min_{i\in[n]} \left(\frac{X_i\bar{w}}{\|\bar{w}\|}\right)^2 \geq \min_{i\in[n]} 2X_i\bar{w} - \|\bar{w}\|^2 \geq \sigma_{\mathrm{SVM}} - \epsilon. \tag{5.3.8}$$

If we denote $\hat{w} = \bar{w}/\|\bar{w}\|$, then $X_i\hat{w} \geq \sqrt{\sigma_{\mathrm{SVM}} - \epsilon} > 0$ for all $i \in [n]$. Consequently, if the data X is from an SVM, we obtain a normalized direction $\hat{w}$ (in ℓ_2-norm) such that all data points have a margin of at least $\sqrt{\sigma_{\mathrm{SVM}} - \epsilon}$. Classically, this task takes time $\tilde{O}(n+d)$ for constant σ_{SVM} by [85], but our quantum algorithm only takes time $\tilde{O}(\sqrt{n}+\sqrt{d})$.

5.4 Quantum lower bounds

All quantum algorithms (upper bounds) above have matching lower bounds in n and d. Assuming $\epsilon = \Theta(1)$ and given the oracle O_X in (5.1.7), we prove quantum lower bounds on linear classification and minimum enclosing ball in Section 5.4.1 and Section 5.4.2, respectively. Both theorems are proven by constructing reductions to the quantum search lower bound [46].

5.4.1 Linear classification

Recall that the input of the linear classification problem is a matrix $X \in \mathbb{R}^{n\times d}$ such that $X_i \in \mathbb{B}_d$ for all $i \in [n]$ (X_i being the i^{th} row of X), and the goal is to approximately solve

$$\sigma := \max_{w\in\mathbb{B}_d} \min_{p\in\Delta_n} p^\top X w = \max_{w\in\mathbb{B}_d} \min_{i\in[n]} X_i w. \tag{5.4.1}$$

Given the quantum oracle O_X such that $O_X|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|X_{ij}\rangle \ \forall\, i \in [n], j \in [d]$, Theorem 5.2.3 solves this task with high success probability with cost $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$. We prove a quantum lower bound that matches this upper bound in n and d for constant ϵ:

Theorem 5.4.1. *Assume $0 < \epsilon < 0.04$. Then to return an $\bar{w} \in \mathbb{B}_d$ satisfying*

$$X_j \bar{w} \geq \max_{w\in\mathbb{B}_d} \min_{i\in[n]} X_i w - \epsilon \quad \forall\, j \in [n] \tag{5.4.2}$$

with probability at least $2/3$, we need $\Omega(\sqrt{n} + \sqrt{d})$ quantum queries to O_X.

Proof. Assume we are given the promise that X is from one of the two cases below:

1. There exists an $l \in \{2, \ldots, d\}$ such that $X_{11} = -\frac{1}{\sqrt{2}}$, $X_{1l} = \frac{1}{\sqrt{2}}$; $X_{21} = X_{2l} = \frac{1}{\sqrt{2}}$; there exists a unique $k \in \{3, \ldots, n\}$ such that $X_{k1} = 1$, $X_{kl} = 0$; $X_{ij} = \frac{1}{\sqrt{2}}$ for all $i \in \{3, \ldots, n\}/\{k\}$, $j \in \{1, l\}$, and $X_{ij} = 0$ for all $i \in [n]$, $j \notin \{1, l\}$.

2. There exists an $l \in \{2, \ldots, d\}$ such that $X_{11} = -\frac{1}{\sqrt{2}}$, $X_{1l} = \frac{1}{\sqrt{2}}$; $X_{21} = X_{2l} = \frac{1}{\sqrt{2}}$; $X_{ij} = \frac{1}{\sqrt{2}}$ for all $i \in \{3, \ldots, n\}$, $j \in \{1, l\}$, and $X_{ij} = 0$ for all $i \in [n]$,

$j \notin \{1, l\}$.

Notice that the only difference between these two cases is a row where the first entry is 1 and the l^{th} entry is 0; they have the following pictures, respectively.

$$\text{Case 1: } X = \begin{pmatrix} -\frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \end{pmatrix}; \tag{5.4.3}$$

$$\text{Case 2: } X = \begin{pmatrix} -\frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & 0 & \cdots & 0 & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \end{pmatrix}. \tag{5.4.4}$$

We denote the maximin value in (5.4.1) of these cases as σ_1 and σ_2, respectively. We have:

- $\sigma_2 = \frac{1}{\sqrt{2}}$.

On the one hand, consider $\bar{w} = \vec{e}_l \in \mathbb{B}_d$ (the vector in $\mathbb{R}^d$ with the l^{th} coordinate

being 1 and all other coordinates being 0). Then $X_i\bar{w} = \frac{1}{\sqrt{2}}$ for all $i \in [n]$, and hence $\sigma_2 \geq \min_{i\in[n]} X_i\bar{w} = \frac{1}{\sqrt{2}}$. On the other hand, for any $w = (w_1, \ldots, w_d) \in \mathbb{B}_d$,

$$\min_{i\in[n]} X_i w = \min\Big\{ -\frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l, \frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l \Big\} \leq \frac{1}{\sqrt{2}}w_l \leq \frac{1}{\sqrt{2}}, \tag{5.4.5}$$

where the first inequality comes from the fact that $\min\{a, b\} \leq \frac{a+b}{2}$ for all $X, b \in \mathbb{R}$ and the second inequality comes from the fact that $w \in \mathbb{B}_d$ and $|w_l| \leq 1$. As a result, $\sigma_2 = \max_{w\in\mathbb{B}_d} \min_{i\in[n]} X_i w \leq \frac{1}{\sqrt{2}}$. In conclusion, we have $\sigma_2 = \frac{1}{\sqrt{2}}$.

- $\sigma_1 = \frac{1}{\sqrt{4+2\sqrt{2}}}$.

  On the one hand, consider $\bar{w} = \frac{1}{\sqrt{4+2\sqrt{2}}}\vec{e}_1 + \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}}\vec{e}_l \in \mathbb{B}_d$. Then

$$\begin{aligned}
X_1\bar{w} &= -\frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{4+2\sqrt{2}}} + \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{1}{\sqrt{4+2\sqrt{2}}}; \\
X_i\bar{w} &= \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{4+2\sqrt{2}}} + \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} > \frac{1}{\sqrt{4+2\sqrt{2}}} \quad \forall\, i \in [n]/\{1, k\}; \\
X_k\bar{w} &= 1 \cdot \frac{1}{\sqrt{4+2\sqrt{2}}} + 0 \cdot \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{1}{\sqrt{4+2\sqrt{2}}}.
\end{aligned}$$

In all, $\sigma_1 \geq \min_{i\in[n]} X_i\bar{w} = \frac{1}{\sqrt{4+2\sqrt{2}}}$.

On the other hand, for any $w = (w_1, \ldots, w_d) \in \mathbb{B}_d$, we have

$$\min_{i\in[n]} X_i w = \min\Big\{ -\frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l, \frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l, w_1 \Big\}. \tag{5.4.6}$$

If $w_1 \leq \frac{1}{\sqrt{4+2\sqrt{2}}}$, then (5.4.6) implies that $\min_{i\in[n]} X_i w \leq \frac{1}{\sqrt{4+2\sqrt{2}}}$; if $w_1 \geq \frac{1}{\sqrt{4+2\sqrt{2}}}$,

then

$$w_l \le \sqrt{1-w_1^2} = \sqrt{1-\frac{1}{4+2\sqrt{2}}} = \frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}},$$

and hence by (5.4.6) we have

$$\min_{i\in[n]} X_i w \le -\frac{1}{\sqrt{2}}w_1 + \frac{1}{\sqrt{2}}w_l \le -\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{4+2\sqrt{2}}} + \frac{1}{\sqrt{2}}\cdot\frac{\sqrt{2}+1}{\sqrt{4+2\sqrt{2}}} = \frac{1}{\sqrt{4+2\sqrt{2}}}.$$

In all, we always have $\min_{i\in[n]} X_i w \le \frac{1}{\sqrt{4+2\sqrt{2}}}$. As a result, $\sigma_1 = \max_{w\in\mathbb{B}_d} \min_{i\in[n]} X_i w \le \frac{1}{\sqrt{4+2\sqrt{2}}}$. In conclusion, we have $\sigma_1 = \frac{1}{\sqrt{4+2\sqrt{2}}}$.

Now, we prove that an $\bar{w} \in \mathbb{B}_d$ satisfying (5.4.2) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2,\dots,d\}$, by the following algorithm:

1. Check if one of $\bar{w}_2,\dots,\bar{w}_d$ is larger than 0.94; if there exists an $l' \in \{2,\dots,d\}$ such that $\bar{w}_{l'} > 0.94$, return 'Case 2' and $l = l'$;
2. Otherwise, return 'Case 1' and $l = \arg\max_{i\in\{2,\dots,d\}} \bar{w}_i$.

We first prove that the classification of X (between Case 1 and Case 2) is correct. On the one hand, assume that X comes from Case 1. If we wrongly classified X as from Case 2, we would have $\bar{w}_{l'} > 0.94$ and $\bar{w}_1 < \sqrt{1-0.94^2} < 0.342$; this would imply

$$\min_{i\in[n]} X_i \bar{w} = \min\Big\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \bar{w}_1\Big\} \le \bar{w}_1 < \frac{1}{\sqrt{4+2\sqrt{2}}} - 0.04 \le,$$

which is smaller than $\sigma_1 - \epsilon$ by $0.342 < \frac{1}{\sqrt{4+2\sqrt{2}}} - 0.04$, contradicts with (5.4.2). Therefore, we must make correct classification that X comes from Case 1.

On the other hand, assume that X comes from Case 2. If we wrongly classified X as from Case 1, we would have $\bar{w}_l \le \max_{i\in\{2,\dots,d\}} \bar{w}_i \le 0.94$; this would imply

$$\min_{i\in[n]} X_i\bar{w} = \min\Big\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l\Big\} \le \frac{1}{\sqrt{2}}\bar{w}_l < \frac{1}{\sqrt{2}} - 0.04 \le \sigma_2 - \epsilon$$

by $\frac{0.94}{\sqrt{2}} < \frac{1}{\sqrt{2}} - 0.04$, contradicts with (5.4.2). Therefore, we must make correct classification that X comes from Case 2. In all, our classification is always correct.

It remains to prove that the value of l is correct. If X is from Case 1, we have

$$\sigma_1 - \epsilon \le \min_{i\in[n]} X_i\bar{w} = \min\Big\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \bar{w}_1\Big\}; \tag{5.4.7}$$

as a result, $\bar{w}_1 \ge \sigma_1 - \epsilon > 0.38 - 0.04 = 0.34$, and

$$-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l > 0.34 \implies \bar{w}_l > 0.34\sqrt{2} + \bar{w}_1 > 0.34(\sqrt{2}+1) > 0.82. \tag{5.4.8}$$

Because $2 \cdot 0.82^2 > 1$, $\bar{w}_l$ must be the largest among $\bar{w}_2, \dots, \bar{w}_d$ (otherwise $l' = \arg\max_{i\in\{2,\dots,d\}} \bar{w}_i$ and $l \ne l'$ would imply $\|\bar{w}\|^2 = \sum_{i\in[d]} |\bar{w}_i|^2 \ge \bar{w}_l^2 + \bar{w}_{l'}^2 \ge 2\bar{w}_l^2 > 1$, contradiction). Therefore, Line 2 of our algorithm correctly returns the value of l.

If X is from Case 2, we have

$$\sigma_2 - \epsilon \le \min_{i\in[n]} X_i\bar{w} = \min\Big\{-\frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l, \frac{1}{\sqrt{2}}\bar{w}_1 + \frac{1}{\sqrt{2}}\bar{w}_l\Big\} \le \frac{1}{\sqrt{2}}\bar{w}_l, \tag{5.4.9}$$

and hence $\bar{w}_l \geq \sqrt{2}(\sigma_2 - \epsilon) \geq \sqrt{2}(\frac{1}{\sqrt{2}} - 0.04) > 0.94$. Because $2 \cdot 0.94^2 > 1$, only one coordinate of $\bar{w}$ could be at least 0.94 and we must have $l = l'$. Therefore, Line 1 of our algorithm correctly returns the value of l.

In all, we have proved that an ϵ-approximate solution $\bar{w} \in \mathbb{B}_d$ for (5.4.2) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2, \ldots, d\}$. On the one hand, notice that distinguishing these two cases requires $\Omega(\sqrt{n-2}) = \Omega(\sqrt{n})$ quantum queries to O_X for searching the position of k because of the quantum lower bound for search [46]; therefore, it gives an $\Omega(\sqrt{n})$ quantum lower bound on queries to O_X for returning an $\bar{w}$ that satisfies (5.4.2). On the other hand, finding the value of l is also a search problem on the entries of X, which requires $\Omega(\sqrt{d-1}) = \Omega(\sqrt{d})$ quantum queries to O_X also due to the quantum lower bound for search [46]. These observations complete the proof of Theorem 5.4.1. □

Because the kernel-based classifier in Section 5.3.1 contains the linear classification in Section 5.2 as a special case, Theorem 5.4.1 implies an $\Omega(\sqrt{n} + \sqrt{d})$ quantum lower bound on the kernel method.

5.4.2 Minimum enclosing ball (MEB)

Similarly, the input of the MEB problem is a matrix $X \in \mathbb{R}^{n \times d}$ such that $X_i \in \mathbb{B}_d$ for all $i \in [n]$, and we are given the quantum oracle O_X such that $O_X|i\rangle|j\rangle|0\rangle = |i\rangle|j\rangle|X_{ij}\rangle \ \forall\, i \in [n], j \in [d]$. The goal of MEB is to approximately solve

$$\sigma_{\text{MEB}} = \min_{w \in \mathbb{R}^d} \max_{i \in [n]} \|w - X_i\|^2. \tag{5.4.10}$$

Theorem 5.3.2 solves this task with high success probability with cost $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^7}\big)$. In this subsection, we prove a quantum lower bound that matches this upper bound in n and d for constant ϵ:

Theorem 5.4.2. *Assume $0 < \epsilon < 0.01$. Then to return an $\bar{w} \in \mathbb{R}_d$ satisfying*

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \leq \min_{w\in\mathbb{R}^d} \max_{i\in[n]} \|w - X_i\|^2 + \epsilon \tag{5.4.11}$$

with probability at least $2/3$, we need $\Omega(\sqrt{n} + \sqrt{d})$ quantum queries to O_X.

Proof. We also assume that X is from one of the two cases in Theorem 5.4.1; see also (5.4.3) and (5.4.4). We denote the maximin value in (5.4.10) of these cases as $\sigma_{\text{MEB},1}$ and $\sigma_{\text{MEB},2}$, respectively. We have:

- $\sigma_{\text{MEB},2} = \frac{1}{2}$.

  On the one hand, consider $\bar{w} = \frac{1}{\sqrt{2}}\vec{e}_l$. Then

$$\|\bar{w} - X_1\|^2 = \Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 = \Big(\frac{1}{\sqrt{2}}\Big)^2 = \frac{1}{2};$$
$$\|\bar{w} - X_i\|^2 = \Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 = \Big(\frac{1}{\sqrt{2}}\Big)^2 = \frac{1}{2} \quad \forall\, i \in \{2,\ldots,n\}.$$

Therefore, $\|\bar{w} - X_i\|^2 = \frac{1}{2}$ for all $i \in [n]$, and hence $\sigma_{\text{MEB},2} \leq \max_{i\in[n]} \|\bar{w} - X_i\|^2 = \frac{1}{2}$.

On the other hand, for any $w = (w_1, \ldots, w_d) \in \mathbb{R}_d$, we have

$$\begin{aligned}
&\max_{i\in[n]} \|w - X_i\|^2 \\
=\ &\max\Big\{\Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2, \Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2\Big\} \\
\geq\ &\frac{1}{2}\Big[\Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2\Big] + \frac{1}{2}\Big[\Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2\Big] + \sum_{i\neq 1,l} w_i^2 \\
=\ &w_1^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 + \frac{1}{2} \qquad (5.4.12) \\
\geq\ &\frac{1}{2}, \qquad (5.4.13)
\end{aligned}$$

where the first inequality comes from the fact that $\max\{a, b\} \geq \frac{1}{2}(a + b)$ and $\sum_{i\neq 1,l} w_i^2 \geq 0$. Therefore, $\sigma_{\mathrm{MEB},2} \geq \frac{1}{2}$. In all, we must have $\sigma_{\mathrm{MEB},2} = \frac{1}{2}$.

- $\sigma_{\mathrm{MEB},1} = \frac{2+\sqrt{2}}{4}$.

  On the one hand, consider $\bar{w} = \big(\frac{1}{2} - \frac{\sqrt{2}}{4}\big)\vec{e}_1 + \frac{\sqrt{2}}{4}\vec{e}_l$. Then

$$\begin{aligned}
&\|\bar{w} - X_1\|^2 = \Big(w_1 + \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 = \Big(\frac{1}{2} + \frac{\sqrt{2}}{4}\Big)^2 + \Big(\frac{\sqrt{2}}{4}\Big)^2 = \frac{2+\sqrt{2}}{4}; \\
&\|\bar{w} - X_k\|^2 = (w_1 - 1)^2 + w_l^2 + \sum_{i\neq 1,l} w_i^2 = \Big(\frac{1}{2} + \frac{\sqrt{2}}{4}\Big)^2 + \Big(\frac{\sqrt{2}}{4}\Big)^2 = \frac{2+\sqrt{2}}{4}; \\
&\|\bar{w} - X_i\|^2 = \Big(w_1 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(w_l - \frac{1}{\sqrt{2}}\Big)^2 + \sum_{i\neq 1,l} w_i^2 = \frac{6-3\sqrt{2}}{4} < \frac{2+\sqrt{2}}{4} \quad \forall\, i \in [n]/\{1, k\}.
\end{aligned}$$

  In all, $\sigma_{\mathrm{MEB},1} \leq \max_{i\in[n]} \|\bar{w} - X_i\|^2 = \frac{2+\sqrt{2}}{4}$.

On the other hand, for any $w = (w_1, \ldots, w_d) \in \mathbb{R}_d$, we have

$$\begin{aligned}
\max_{i\in[n]} \|w - X_i\|^2 &\geq \max\left\{\left(w_1 + \frac{1}{\sqrt{2}}\right)^2 + \left(w_l - \frac{1}{\sqrt{2}}\right)^2 + \sum_{i\neq 1,l} w_i^2, (w_1-1)^2 + w_l^2 + \sum_{i\neq 1,l} w_i^2\right\} \\
&\geq \frac{1}{2}\left[\left(w_1 + \frac{1}{\sqrt{2}}\right)^2 + \left(w_l - \frac{1}{\sqrt{2}}\right)^2\right] + \frac{1}{2}\left[(w_1-1)^2 + w_l^2\right] + \sum_{i\neq 1,l} w_i^2 \\
&= \left[w_1 - \left(\frac{1}{2} - \frac{\sqrt{2}}{4}\right)\right]^2 + \left(w_l - \frac{\sqrt{2}}{4}\right)^2 + \sum_{i\neq 1,l} w_i^2 + \frac{2+\sqrt{2}}{4} \qquad (5.4.14) \\
&\geq \frac{2+\sqrt{2}}{4}. \qquad (5.4.15)
\end{aligned}$$

Therefore, $\sigma_{\mathrm{MEB},2} \geq \frac{2+\sqrt{2}}{4}$. In all, we must have $\sigma_{\mathrm{MEB},2} = \frac{2+\sqrt{2}}{4}$.

Now, we prove that an $\bar{w} \in \mathbb{R}_d$ satisfying (5.4.11) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2, \ldots, d\}$, by the following algorithm:

1. Check if one of $\bar{w}_2, \ldots, \bar{w}_d$ is larger than $\frac{3\sqrt{2}}{8}$; if there exists an $l' \in \{2, \ldots, d\}$ such that $\bar{w}_{l'} > \frac{3\sqrt{2}}{8}$, return 'Case 1' and $l = l'$;

2. Otherwise, return 'Case 2' and $l = \arg\max_{i\in\{2,\ldots,d\}} \bar{w}_i$.

We first prove that the classification of X (between Case 1 and Case 2) is correct. On the one hand, assume that X comes from Case 1. If we wrongly classified X as from Case 2, we would have $\bar{w}_l \leq \max_{i\in\{2,\ldots,d\}} \bar{w}_i \leq \frac{3\sqrt{2}}{8}$. By (5.4.12), this would imply

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \left(\bar{w}_l - \frac{1}{\sqrt{2}}\right)^2 + \frac{1}{2} \geq \frac{1}{32} + \frac{1}{2} > \sigma_{\mathrm{MEB},1} + \epsilon, \qquad (5.4.16)$$

contradicts with (5.4.11). Therefore, for this case we must make correct classification that X comes from Case 2.

On the other hand, assume that X comes from Case 2. If we wrongly classified X as from Case 1, we would have $\bar{w}_{l'} > \frac{3\sqrt{2}}{8}$. If $l = l'$, then (5.4.14) implies that

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \left(\bar{w}_l - \frac{\sqrt{2}}{4}\right)^2 + \frac{2+\sqrt{2}}{4} \geq \frac{1}{32} + \frac{2+\sqrt{2}}{4} > \sigma_{\mathrm{MEB},2} + \epsilon, \quad (5.4.17)$$

contradicts with (5.4.11). If $l \neq l'$, then (5.4.14) implies that

$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \bar{w}_{l'}^2 + \frac{2+\sqrt{2}}{4} \geq \frac{9}{32} + \frac{2+\sqrt{2}}{4} > \sigma_{\mathrm{MEB},2} + \epsilon, \quad (5.4.18)$$

also contradicts with (5.4.11). Therefore, for this case we must make correct classification that X comes from Case 1.

In all, our classification is always correct. It remains to prove that the value of l is correct. If X is from Case 1, by (5.4.12) we have

$$\frac{1}{2} + 0.01 \geq \max_{i\in[n]} \|\bar{w} - X_i\|^2 \geq \bar{w}_1^2 + \left(\bar{w}_l - \frac{1}{\sqrt{2}}\right)^2 + \sum_{i\neq 1,l} \bar{w}_i^2 + \frac{1}{2}. \quad (5.4.19)$$

As a result, $\bar{w}_i \leq 0.1 < \frac{3\sqrt{2}}{8}$ for all $i \in [n]/\{l\}$ and $\bar{w}_l \geq \frac{1}{\sqrt{2}} - 0.1 > \frac{3\sqrt{2}}{8}$. Therefore, we must have $l = l'$, i.e., Line 1 of our algorithm correctly returns the value of l.

If X is from Case 2, by (5.4.14) we have

$$\frac{2+\sqrt{2}}{4} + 0.01 \geq \max_{i\in[n]} \|\bar{w} - X_i\|^2 \tag{5.4.20}$$

$$\geq \left[\bar{w}_1 - \left(\frac{1}{2} - \frac{\sqrt{2}}{4}\right)\right]^2 + \left(\bar{w}_l - \frac{\sqrt{2}}{4}\right)^2 + \sum_{i\neq 1,l} \bar{w}_i^2 + \frac{2+\sqrt{2}}{4}. \tag{5.4.21}$$

As a result, $\bar{w}_i \leq 0.1 < 0.25$ for all $i \in [n]/\{1, l\}$, $\bar{w}_1 \leq \frac{1}{2} - \frac{\sqrt{2}}{4} + 0.1 < 0.25$, and $\bar{w}_l \geq \frac{\sqrt{2}}{4} - 0.1 > 0.25$. Therefore, we must have $\bar{w}_l = \max_{i\in\{2,\ldots,d\}} \bar{w}_i$, i.e., Line 1 of our algorithm correctly returns the value of l.

In all, we have proved that an ϵ-approximate solution $\bar{w} \in \mathbb{R}_d$ for (5.4.11) would simultaneously reveal whether X is from Case 1 or Case 2 as well as the value of $l \in \{2, \ldots, d\}$. On the one hand, notice that distinguishing these two cases requires $\Omega(\sqrt{n-2}) = \Omega(\sqrt{n})$ quantum queries to O_X for searching the position of k because of the quantum lower bound for search [46]; therefore, it gives an $\Omega(\sqrt{n})$ quantum lower bound on queries to O_X for returning an $\bar{w}$ that satisfies (5.4.11). On the other hand, finding the value of l is also a search problem on the entries of X, which requires $\Omega(\sqrt{d-1}) = \Omega(\sqrt{d})$ quantum queries to O_X also due to the quantum lower bound for search [46]. These observations complete the proof of Theorem 5.4.2. □

Because MEB and ℓ_2-margin SVM are both maximin problems of a quadratic function (see Section 5.3.2), an $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bound on the ℓ_2-margin SVM can be proved similarly.

5.5 Conclusions and discussion

We give quantum algorithms for training linear and kernel-based classifiers with complexity $\tilde{O}(\sqrt{n}+\sqrt{d})$, where n and d are the number and dimension of data points, respectively; furthermore, our quantum algorithms are optimal as we prove matching $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bounds. Our quantum algorithms take standard entry-wise inputs and give classical outputs with succinct representations. Technically, our result is a demonstration of quantum speed-ups for sampling-based classical algorithms using the technique of amplitude amplification and estimation.

Our paper raises a couple of natural open questions for future work. For instance:

- Can we improve the dependence in ϵ? Recall our quantum algorithms have worst-case complexity $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^8}\big)$ whereas the classical complexities in [85] are $\tilde{O}\big(\frac{n}{\epsilon^2}+\frac{d}{\epsilon^2}\big)$; as a result, the quantum algorithms perform better only when we tolerate a significant error. It would be interesting to check if some clever tricks could be applied to circumventing some dependence in ϵ.

- Can we solve equilibrium point problems other than classification? Recall that our results in Theorem 6.5.1 are all formulated as maximin problems where the minimum is taken over $[n]$ and the maximum is taken over $\mathbb{B}_d$ or $\mathbb{R}_d$. It would be interesting to study other type of equilibrium point problems in game theory, learning theory, etc.

- What happens if we work with more sophisticated data structures such as QRAM

or its augmented variants? Their preprocessing time will likely be at least linear. However, it might be still advantageous to do so, e.g., to reduce the amortized complexity when one needs to perform multiple classification tasks on the same data set.

Chapter 6: Quantum-inspired Classical Machine Learning Algorithms[1]

In previous chapters, we have proposed some quantum algorithms for machine learning and optimization tasks. In fact, there are also quite a few other quantum algorithms for various machine learning problems, in particular those motivated by the Harrow-Hassidim-Lloyd (HHL) algorithm for solving sparse linear systems in poly-logarithmic time [137]. These include principal component analysis [195], cluster assignment and finding [194], support vector machines [232], recommendation systems [167], etc. However, their quantum speedups are not as "robust" as, say, Shor's algorithm for factoring [245], mainly because it is unclear how to load the input into a quantum computer efficiently or conclude useful information from the quantum outputs of these quantum algorithms [2].

In 2018, Tang gave a classical analog to the quantum recommendation systems algorithm [256], previously believed to be one of the most seminal candidates for obtaining quantum speedup for machine learning. One of the biggest implications of Tang's breakthrough result is that its techniques can be generalized to "dequantize" a wide range of quantum machine learning algorithms for low-rank cases, including principal component analysis and supervised clustering [255], linear

[1]This chapter is based on the papers [76, 77] under the permission of all the authors.

system solving [78, 122], semidefinite program solving [77], support vector machines (SVM) [97], nonnegative matrix factorization [75], minimal conical hull [99], etc. The framework that Tang used is a sampling-based model of the input matrices and vectors that replicates known quantum machine learning algorithms while running on a classical computer in the regime where the inputs are low-rank matrices. In short, "dequantized" algorithms either provide strong barriers for or completely disprove the existence of exponential speedups from their corresponding quantum machine learning algorithms in low-rank settings, which is a practical assumption in many of the applications.

In this chapter, we further extend the study of quantum-inspired classical algorithms. In particular, we will introduce a quantum-inspired classical algorithm for solving low-rank SDPs from Section 6.1 to Section 6.5. In Section 6.6, we will briefly sketch a general framework of quantum-inspired classical algorithms proposed by my recent work [76].

6.1 Revisiting semidefinite programming

6.1.1 Motivations and contributions

As introduced in Chapter 4, semidefinite programming (SDP) is a central topic in the studies of mathematical optimization and theoretical computer science, with a wide range of applications including algorithm design, machine learning, operations research, etc. Specifically, we consider the following mathematical form of SDPs in

this chapter:

$$\max \quad \mathrm{Tr}[CX] \tag{6.1.1}$$

$$\text{s.t.} \quad \mathrm{Tr}[A_i X] \leq b_i \quad \forall\, i \in [m]; \tag{6.1.2}$$

$$X \succeq 0, \tag{6.1.3}$$

where m is the number of constraints, $A_1, \ldots, A_m, C$ are $n \times n$ Hermitian matrices, and $b_1, \ldots, b_m \in \mathbb{R}$; (6.1.3) restricts the variable matrix X to be *positive semidefinite* (PSD), i.e., X is an $n \times n$ Hermitian matrix with non-negative eigenvalues (more generally, $X \succeq Y$ means that $X - Y$ is a PSD matrix). An ε-approximate solution of this SDP is an X^* that satisfies (6.1.2)-(6.1.3) while $\mathrm{Tr}[CX^*] \geq \mathsf{OPT} - \varepsilon$ (OPT being the optimum of the SDP).

All of the classical SDP solvers mentioned in Chapter 4 use the standard entry-wise access to matrices $A_1, \ldots, A_m$, and C. In contrast, a common methodology in algorithm design is to assume a certain natural *preprocessed data structure* such that the problem can be solved in *sublinear* time, perhaps even in *poly-logarithmic* time, given queries to the preprocessed data structure (e.g., see the examples discussed in Section 6.1.3). Considering this, a very natural question is to ask *whether we can solve an SDP with sublinear time and queries to a reasonable classical data structure.*

Contributions. We show that when the constraint and cost matrices are low-rank, with a low-overhead data structure that supports the following sampling ac-

cess, there exists a classical algorithm whose runtime is logarithmic in the matrices dimension n.

Definition 6.1.1 (Sampling access)**.** *Let $M \in \mathbb{C}^{n\times n}$ be a matrix. We say that we have the* sampling access *to M if we can*

1. *sample a row index $i \in [n]$ of M where the probability of choosing i is*

$$\frac{\|M(i,\cdot)\|^2}{\|M\|_F^2};$$

2. *for all $i \in [n]$, sample an index $j \in [n]$ where the probability of choosing j is*

$$\frac{|M(i,j)|^2}{\|M(i,\cdot)\|^2};\ \textit{and}$$

3. *evaluate norms of $\|M\|_F$ and $\|M(i,\cdot)\|$ for $i \in [n]$,*

with time complexity $O(\mathrm{poly}(\log n))$ for each sampling and norm access.

A low-overhead data structure that allows for this sampling access is shown in Section 6.2.1. Our main result is as follows.

Theorem 6.1.1 (informal; see Theorem 6.5.1, Algorithm 6.5, Theorem 6.2.2)**.** *Let $C, A_1, \ldots, A_m \in \mathbb{C}^{n\times n}$ be an SDP instance as in (6.1.1)-(6.1.3). Suppose $\mathrm{rank}(C), \max_{i\in[n]} \mathrm{rank}(A_i) \leq r$. Given sampling access to $A_1, \ldots, A_m, C$ in Definition 6.1.1, there is an algorithm that gives sampling access as well as the (i,j)-th entry (for any index $(i,j) \in [n]\times[n]$) of an ε-approximate solution of the SDP with probability at least $2/3$; the algorithm runs in time $O(m \cdot \mathrm{poly}(\log n, r, R_p R_d/\varepsilon))$,*

where R_p, R_d are upper bounds on the ℓ_1-norm of the optimal primal and dual solutions.

Comparing our results to existing classical SDP solvers (e.g., [29, 114, 115]), our algorithm outperforms existing classical SDP solvers given sampling access to the constraint matrices (which can be realized with a low-overhead data structure). Specifically, the running time of our algorithm is $O(m \cdot \mathrm{poly}(\log n, r, R_{\mathrm{p}} R_{\mathrm{d}}/\varepsilon))$ according to Theorem 6.1.1, which achieves exponential speedup in terms of n with the data structure given in Theorem 6.2.1. It is worth noting that there are other ways to implement the sampling access. For example, Drineas et al. [98] showed that the sampling access in Definition 6.1.1 can be achieved with poly-logarithmic space if the matrix elements are streamed. Therefore, Theorem 6.1.1 also implies that there exists a one-pass poly-logarithmic space algorithm for low-rank SDP in the data-streaming model.

Comparing to quantum algorithms, our algorithm has comparable running time. It is because existing quantum SDP solvers that achieve exponential speedup in terms of n, for instance Corollary 4.5.1, have polynomial dependence on the rank r, so they also have $\mathrm{poly}(\log n, r)$ complexity. It is worth noting that quantum SDP solvers have the requirement that the input matrices are stored as quantum oracles. Furthermore, we give query access to the solution matrix which was not achieved by existing quantum SDP solvers in Chapter 4, as only sampling access to the solution matrix is given there. In this regard, it is easy to obtain the sampling access of the solution matrix from our algorithm by extending the rejection sampling techniques

of [256] as pointed out by Tang.

Our result aligns with the studies of sampling-based algorithms for solving linear algebraic problems. In particular, [113] gave low-rank approximations of a matrix in *poly-logarithmic* time with sampling access to the matrix as in Definition 6.1.1. Recently, Tang extended the idea of [113] to give a poly-logarithmic time algorithm for solving recommendation systems [256]. Subsequently, still under the same sampling assumption, Tang [255] sketched poly-logarithmic algorithms for principal component analysis and clustering assignments, and two followup papers [78, 122] gave poly-logarithmic algorithms for solving low-rank linear systems. All these sampling-based sublinear algorithms directly exploit the sampling approach in [113] (see Section 6.1.2 for details); to solve SDPs, we derive an augmented sampling toolbox which includes two novel techniques: *weighted sampling* and *symmetric approximation*.

As a corollary, our SDP solver can be applied to learning quantum states[2] efficiently. A particular task of learning quantum states is *shadow tomography* [3], where we are asked to find a description of an unknown quantum state ρ such that we can approximate $\mathrm{Tr}[\rho E_i]$ up to error ϵ for a specific collection of Hermitian matrices $E_1, \ldots, E_m$ where $0 \preceq E_i \preceq I$ and $E_i \in \mathbb{C}^{n \times n}$ for all $i \in [m]$ (such E_i is also known as a POVM measurement in quantum computing). Mathematically, shadow

[2] A quantum state ρ is a PSD matrix with trace one.

tomography can be formulated as the following SDP feasibility problem:

$$\text{Find } \sigma \text{ such that} \qquad |\operatorname{Tr}[\sigma E_i] - \operatorname{Tr}[\rho E_i]| \leq \epsilon \quad \forall\, i \in [m]; \tag{6.1.4}$$

$$\sigma \succeq 0, \quad \operatorname{Tr}[\sigma] = 1. \tag{6.1.5}$$

Under a quantum model proposed by [55] where $\rho, E_1, \ldots, E_m$ are stored as quantum states, the state-of-the-art quantum algorithm [23] solves shadow tomography with time $O\big((\sqrt{m} + \min\{\sqrt{n}/\epsilon, r^{2.5}/\epsilon^{3.5}\})r/\epsilon^4\big)$ where $r = \max_{i\in[m]} \operatorname{rank}(E_i)$; in other words, quantum algorithms achieve poly-logarithmic complexity in n for low-rank shadow tomography. We observe the same phenomenon under our sampling-based model:

Corollary 6.1.1 (informal; see Corollary 6.5.1)**.** *Given sampling access of matrices $E_1, \ldots, E_m \in \mathbb{C}^{n\times n}$ as in Definition 6.1.1 and real numbers $\operatorname{Tr}[\rho E_1], \ldots, \operatorname{Tr}[\rho E_m]$, there is an algorithm that gives a succinct description as in Remark 6.5.1 and any entry of an ϵ-approximate solution σ of the shadow tomography problem defined as (6.1.4), (6.1.5) with probability at least $2/3$; the algorithm runs in time $O(m \cdot \operatorname{poly}(\log n, r, 1/\epsilon))$.*

6.1.2 Techniques

Matrix multiplicative weight method (MMW). We study a normalized SDP feasibility testing problem defined as follows:

Definition 6.1.2 (Feasibility of SDP)**.** *Given an $\epsilon > 0$, m real numbers $a_1, \ldots, a_m \in$*

$\mathbb{R}$, *and Hermitian* $n \times n$ *matrices* $A_1, \ldots, A_m$ *where* $-I \preceq A_i \preceq I, \forall j \in [m]$, *define* $\mathcal{S}_\epsilon$ *as the set of all* X *such that*

$$\mathrm{Tr}[A_i X] \leq a_i + \epsilon \quad \forall\, i \in [m]; \tag{6.1.6}$$

$$X \succeq 0; \tag{6.1.7}$$

$$\mathrm{Tr}[X] = 1. \tag{6.1.8}$$

For ϵ*-approximate feasibility testing of the SDP, we require that:*

- *If* $\mathcal{S}_\epsilon = \varnothing$, *output "infeasible";*
- *If* $\mathcal{S}_0 \neq \varnothing$, *output an* $X \in \mathcal{S}_\epsilon$.[3]

It is a well-known fact that one can use binary search to reduce ε-approximation of the SDP in (6.1.1)-(6.1.3) to $O(\log(R_{\mathrm{p}} R_{\mathrm{d}}/\varepsilon))$ calls of the feasibility problem in Definition 6.1.2 with $\epsilon = \varepsilon/(R_{\mathrm{p}} R_{\mathrm{d}})$.[4] Therefore, throughout the paper we focus on solving feasibility testing of SDPs.

To solve the feasibility testing problem in Definition 6.1.2, we follow the *matrix multiplicative weight* (MMW) framework [28]. To be more specific, we follow the approach of regarding MMW as a zero-sum game with two players (see, e.g., [55, 131, 142, 181, 274]), where the first player wants to provide a feasible $X \in \mathcal{S}_\epsilon$,

[3] If $\mathcal{S}_\epsilon \neq \varnothing$ and $\mathcal{S}_0 = \varnothing$, either output is acceptable.

[4] For the normalized case $R_{\mathrm{p}} R_{\mathrm{d}} = 1$, we first guess a candidate value $c_1 = 0$ for the objective function, and add that as a constraint $\mathrm{Tr}[CX] \geq c_1$ to the optimization problem. If this problem is feasible, the optimum is larger than c_1 and we accordingly take $c_2 = c_1 + \frac{1}{2}$; if this problem is infeasible, the optimum is smaller than c_1 and we accordingly take $c_2 = c_1 - \frac{1}{2}$; we proceed similarly for all c_i. As a result, we could solve the optimization problem with precision ϵ using $\lceil \log_2 \frac{1}{\epsilon} \rceil$ calls to the feasibility problem in Definition 6.1.2. For renormalization, it suffices to take $\epsilon = \varepsilon/(R_{\mathrm{p}} R_{\mathrm{d}})$.

and the second player wants to find any violation $j \in [m]$ of any proposed X, i.e., $\mathrm{Tr}[A_j X] > a_j + \epsilon$. At the t^{th} round of the game, if the second player points out a violation j_t for the current proposed solution X_t, the first player proposes a new solution

$$X_{t+1} \leftarrow \exp[-(A_{j_1} + \cdots + A_{j_t})] \tag{6.1.9}$$

(up to normalization); such solution by matrix exponentiation is formally named as a *Gibbs state*. A feasible solution is actually an equilibrium point of the zero-sum game, which can also be approximated by the MMW method [28]; formal discussions are given in Section 6.2.2.

Improved sampling tools. Before describing our improved sampling tools, let us give a brief review of the techniques introduced by [113]. The basic idea of [113] comes from the fact that a low-rank matrix A can be well-approximated by sampling a small number of rows. More precisely, suppose that A is an $n \times n$ matrix with rank r, where $n \gg r$. Because n is large, it is preferable to obtain a "description" of A without using $\mathrm{poly}(n)$ resources. If we have the sampling access to A in the form of Definition 6.1.1, we can sample rows from A according to statement 1 of Definition 6.1.1. Suppose S is the $p \times n$ submatrix of A formed by sampling $p = \mathrm{poly}(r)$ rows from A with some normalization. It can be shown that $S^\dagger S \approx A^\dagger A$. Further, we apply the similar sampling techniques to sampling p columns of S with some normalization to form a $p \times p$ matrix W such that $WW^\dagger \approx SS^\dagger$. Then the

singular values and singular vectors of W, which are easy to compute because p is small, together with the row indices that form S, can be viewed as a succinct description of some matrix $V \in \mathbb{C}^{n\times r}$ satisfying $A \approx AVV^\dagger$, which gives a low-rank projection of A. In [78], this method was extended to approximating the spectral decomposition of $AA^\dagger$, i.e., calculating a small diagonal matrix D and finding a succinct description of V such that $VD^2V^\dagger \approx AA^\dagger$.

To implement the MMW framework, we need an approximate description of the matrix exponentiation $X_{t+1} := \exp\left[-\sum_{i=1}^{t} A_{j_i}\right]$ in (6.1.9). We achieve this in two steps. First, we get an approximate description of the spectral decomposition of A: $A \approx \hat{V}\hat{D}\hat{V}^\dagger$, where $\hat{V}$ is an $n \times r$ matrix and $\hat{D}$ is an $r \times r$ real diagonal matrix. Then, we approximate the matrix exponentiation e^{-A} by $\hat{V}e^{-D}\hat{V}^\dagger$.

There are two main technical difficulties that we overcome with new tools while following the above strategy. First, since A changes dynamically throughout the MMW method, we cannot assume the sampling access to A; a more reasonable assumption is to have sampling access to each individual constraint matrix A_{j_t}, but it is hard to directly sample from A by sampling from each individual A_{j_t}.[5][6] In Section 6.3.1, we sidestep this difficulty by devising the *weighted sampling* procedure which gives a succinct description of a low-rank approximation of $A = \sum_t A_{j_t}$ by sampling each individual A_{j_t}. In other words, we cannot sample according to A, but

[5]For example, assume we have $A = A_1 + A_2$ such that $A_2 = -A_1 + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a matrix with small entries. In this case, A_1 and A_2 mostly cancel out each other and leave $A = \boldsymbol{\epsilon}$. Since $\boldsymbol{\epsilon}$ can be arbitrarily small compared to A_1 and A_2, it is hard to sample from $\boldsymbol{\epsilon}$ by sampling from A_1 and A_2.

[6]Gilyén and Tang pointed out to us that one might be able to sample from A by lower-bounding the cancellation and doing a rejection sampling. We did not explore this approach in detail, but this is a possible alternative to weighted sampling.

we can still find a succinct description of a low-rank approximation of A.

Second, the original sampling procedure of [113] and the extension by [78] give $VD^2V^\dagger \approx A^\dagger A$ instead of a spectral decomposition $\hat{V}\hat{D}\hat{V}^\dagger \approx A$, even if A is Hermitian. For our purpose of matrix exponentiation, singular value decomposition is problematic because the singular values ignore the signs of the eigenvalues; specifically, we get a large error if we approximate e^{-A} by naively exponentiate the singular value decomposition: $e^{-A} \not\approx Ve^{-D}V^\dagger$. Note that this issue of missing negative signs is intrinsic to the tools in [113] because they are built upon the approximation $S^\dagger S \approx A^\dagger A$; Suppose that A has the decomposition $A = UDV^\dagger$, where D is a diagonal matrix, and U and V are isometries. Then $A^\dagger A = VD^\dagger DV^\dagger$, cancelling out any phase on D. We resolve this issue by a novel approximation procedure, *symmetric approximation*. Symmetric approximation is based on the result $A \approx AVV^\dagger$ shown by [113]. It then holds that $A \approx V(V^\dagger AV)V^\dagger$ because the symmetry of A implies that $VV^\dagger$ acts roughly as the identity on the image of A. Since $(V^\dagger AV)$ is a small matrix of size $r \times r$, we can calculate it explicitly and diagonalize it, getting approximate eigenvalues of A. Together with the description of V, we get the desired description of the spectral decomposition of A. See Section 6.3.2 for more details.[7]

[7] It might be illustrative to describe some of our failed attempts before achieving symmetric approximation. We tried to separate the exponential function into even and odd parts; unfortunately that decomposes e^{-x} into $e^{-x} = \cosh x - \sinh x$, resulting in large cancellation and unbounded error. We also tried to obtain the eigenvectors of A from V; this approach faces multiple difficulties, the most serious one being the "fake degeneracy" as shown by the following example. Suppose $A = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. A has two distinct eigenvectors. However, $A^\dagger A = VDDV^\dagger$ can be satisfied by taking $D = I$ together with any unitary V. In this case, V does not give any information about the eigenvectors.

6.1.3 Related work

Our work follows the general methodology of leveraging preprocessed data structures; more specifically, we use sampling-based data structures to fulfill the MMW framework in our SDP solver. In this subsections, we delve into related works about preprocessed data structures and MMW-based SDP solvers.

Preprocessed data structures. Preprocessed data structures are ubiquitous in algorithm design, which enable further computation tasks to be completed within sublinear or even poly-logarithmic time. For the task of nearest neighbor search, we are given a set P of n points in $\mathbb{R}^d$ and the goal is to preprocess a data structure such that given any point q, it returns a point in P that is closest to q. In the case $d = 2$, there exists a data structure using $O(n)$ space with $O(\log n)$ time for each query [193]; more general cases are discussed in the survey paper [21]. A related problem is orthogonal range search, where the goal is to preprocess a data structure such that one can efficiently report the points contained in an axis-aligned query box. When $d = 2$, Ref. [74] preprocessed a data structure with $O(n \log \log n)$ space and only $O(\log \log n)$ query time; for larger d, the query time $O(\log \log n)$ can be kept with a slight overhead on its space complexity. If preprocessed data structures are not exploited for these problems, we have to check all n points in brute-force, *inefficient* for applications in data analytics, machine learning, computer vision, etc.

This methodology is also widespread in graph problems. It concerns *fully dynamic* graphs, where we start from an empty graph on n fixed vertices and maintain

a data structure such that edge insertions and deletions only take sublinear update time for specific graph properties. For instance, the data structure in [35] maintains the maximal independent set of the graph deterministically in $O(m^{3/4})$ amortized update time (m being the dynamic number of edges). There also exist data structures with sublinear update time for minimum vertex cover size [222] and all-pairs shortest paths [7, 259]; furthermore, data structures with poly-logarithmic update time can be constructed for connectivity, minimum spanning tree, and bipartiteness [144, 145]; maximum matching [49, 247], graph coloring [48], etc.

Solving SDPs by the MMW framework. As introduced previously, many SDP solvers use cutting-plane methods or interior-point methods with complexity $\mathrm{poly}(\log(1/\epsilon))$ but larger complexities in m and n. In contrast, our SDP solver follows the MMW framework, and we briefly summarize such SDP solvers in existing literature. They mainly fall into two categories as follows.

First, MMW is adopted in solvers for *positive* SDPs, i.e., $A_1, \ldots, A_m, C \succeq 0$. In this case, the power of MMW lies in its efficiency of having only $\tilde{O}(\mathrm{poly}(1/\epsilon))$ iterations and the fact that it admits *width-independent* solvers whose complexities are independent of R_p and R_d. Ref. [204] first gave a width-independent positive LP solver that runs in $O(\log^2(mn)/\epsilon^4)$ iterations; [151] subsequently generalized this result to give the first width-independent positive SDP solver, but the number of iterations can be as large as $O(\log^{14}(mn)/\epsilon^{13})$. The state-of-the-art positive SDP solver was proposed by [16] with only $O(\log^2(mn)/\epsilon^3)$ iterations.

Second, as far as we know, the vast majority of *quantum* SDP solvers use the MMW framework. The first quantum SDP solver was proposed by [56] with worst-case running time $\tilde{O}(\sqrt{mn}s^2(R_\mathrm{p}R_\mathrm{d}/\epsilon)^{32})$, where s is the sparsity of input matrices, i.e., every row or column of $A_1, \ldots, A_m, C$ has at most s nonzero elements. Subsequently, the quantum complexity of solving SDPs was improved by [24, 55], and the state-of-the-art quantum SDP solver runs in time $\tilde{O}\big((\sqrt{m}+\sqrt{n}R_\mathrm{p}R_\mathrm{d}/\epsilon)s(R_\mathrm{p}R_\mathrm{d}/\epsilon)^4\big)$ [23]. This is optimal in the dependence of m and n because [56] proved a quantum lower bound of $\Omega(\sqrt{m}+\sqrt{n})$ for constant $R_\mathrm{p}, R_\mathrm{d}, s$, and ϵ.

Notations. We let $[n]$ denote the set $\{1, \ldots, n\}$. For a vector $v \in \mathbb{C}^n$, we use $\mathcal{D}_v$ to denote the probability distribution on $[n]$ where the probability of i being chosen is $\mathcal{D}_v(i) = |v(i)|^2/\|v\|^2$ for all $i \in [n]$. When it is clear from the context, a sample from $\mathcal{D}_v$ is often referred to as a sample from v. For a matrix $A \in \mathbb{C}^{n\times n}$, we use $\|A\|$ and $\|A\|_F$ to denote its spectral norm and Frobenius norm, respectively; we use $A(i,\cdot)$ and $A(\cdot,j)$ to denote the i^{th} row and j^{th} column of A, respectively. We use $\mathrm{rows}(A)$ to denote the n-dimensional vector formed by the norms of its row vectors, i.e., $(\mathrm{rows}(A))(i) = \|A(i,\cdot)\|$, for all $i \in [n]$.

6.2 Preliminary tools

6.2.1 Sampling-based data structure

As we develop sublinear-time algorithms for solving SDP in this paper, the whole constraint matrices cannot be loaded into memory since storing them re-

quires at least linear space and time. Instead, we assume the *sampling access* of each constraint matrix as defined in Definition 6.1.1. This sampling access relies on a natural probability distribution that arises in many machine learning applications [78, 122, 165, 167, 255, 256].

Technically, Ref. [113] used this sampling access to develop sublinear algorithms for low-rank matrix approximation. It is well-known (as pointed out by [167] and also used in [78, 122, 165, 255, 256]) that there exist low-overhead preprocessed data structures that allow for the sampling access. More precisely, the existence of the data structures for the sampling access defined in Definition 6.1.1 is stated as follows.

Theorem 6.2.1 ([167]). *Given a matrix $M \in \mathbb{C}^{n\times n}$ with s non-zero entries, there exists a data structure storing M in space $O(s\log^2 n)$, which supports the following operators:*

1. *Reading and writing $M(i,j)$ in $O(\log^2 n)$ time.*
2. *Evaluating $\|M(i,\cdot)\|$ in $O(\log^2 n)$ time.*
3. *Evaluating $\|M\|_F^2$ in $O(1)$ time.*
4. *Sampling a row index of M according to statement 1 of Definition 6.1.1 in $O(\log^2 n)$ time.*
5. *For each row, sampling an index according to statement 2 of Definition 6.1.1 in $O(\log^2 n)$ time.*

Readers may refer to [167, Theorem A.1] for the proof of Theorem 6.2.1. In the following, we give the intuition of the data structure, which is demonstrated in Figure 6.1. We show how to sample from a row vector: we use a binary tree to store the date of each row. The square of the absolute values of all entries, along with their original values are stored in the leaf nodes. Each internode contains the sum of the values of its two immediate children. It is easy to see that the root node contains the square of the norm of this row vector. To sample an index and to query an entry from this row, logarithmic steps suffice. To fulfill statement 1 of Definition 6.1.1, we treat the norms of rows as a vector $(\|M(1,\cdot)\|,\ldots,\|M(n,\cdot)\|)$ and organize the data of this vector in a binary tree.

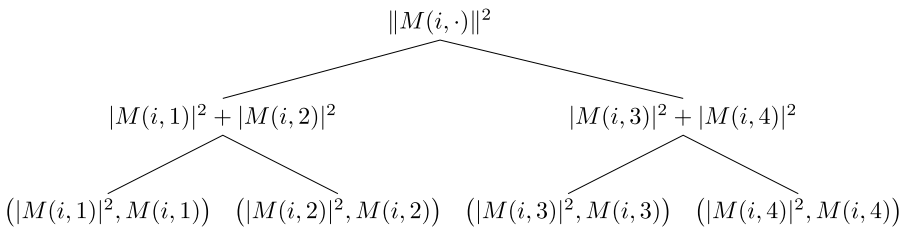

Figure 6.1: Illustration of a data structure that allows for sampling access to a row of $M \in \mathbb{C}^{4\times 4}$.

6.2.2 Feasibility testing of SDPs

We adopt the MMW framework to solve SDPs under the zero-sum approach [55, 131, 142, 181, 274]. This is formulated as the following theorem:

Theorem 6.2.2 (Master algorithm)**.** *Feasibility of the SDP in (6.1.6)-(6.1.8) can be tested by Algorithm 6.1.*

This theorem is proved in [55, Theorem 2.3]; note that the weight matrix

Algorithm 6.1: MMW for testing feasibility of SDPs (also Algorithm 4.2).

1 Set the initial Gibbs state $\rho_1 = \frac{I_n}{n}$, and number of iterations $T = \frac{16 \ln n}{\epsilon^2}$;
2 **for** $t = 1, \ldots, T$ **do**
3 Find a $j_t \in [m]$ such that $\mathrm{Tr}[A_{j_t} \rho_t] > a_{j_t} + \epsilon$. If we cannot find such j_t, claim that $\rho_t \in \mathcal{S}_\epsilon$ and terminate the algorithm;
4 Define the new weight matrix $W_{t+1} := \exp\left[-\frac{\epsilon}{2} \sum_{i=1}^{t} A_{j_i}\right]$ and Gibbs state $\rho_{t+1} := \frac{W_{t+1}}{\mathrm{Tr}[W_{t+1}]}$;
5 Claim that the SDP is infeasible and terminate the algorithm;

therein is $W_{t+1} = \exp\left[-\frac{\epsilon}{2} \sum_{\tau=1}^{t} M_\tau\right]$ where $M_\tau = \frac{1}{2}(I_n - A_{j_\tau})$, but this gives the same Gibbs state as in Line 4 since for any Hermitian matrix $W \in \mathbb{C}^{n \times n}$ and real number $c \in \mathbb{R}$,

$$\frac{e^{W+cI}}{\mathrm{Tr}[e^{W+cI}]} = \frac{e^W e^c I}{\mathrm{Tr}[e^W e^c I]} = \frac{e^W}{\mathrm{Tr}[e^W]}. \tag{6.2.1}$$

It should also be understood that this master algorithm is *not* the final algorithm; the step of trace estimation with respect to the Gibbs state (Line 3) will be fulfilled by our sampling-based approach.

6.3 Two new tools

6.3.1 Weighted sampling

The objective of this subsection is to provide a method for sampling a small submatrix of A of the form $A = A_1 + \cdots + A_\tau$ where the sampling access of each A_ℓ is given. Note that the standard FKV sampling method [113] is not capable of this task, as the sampling access of each A_ℓ does not trivially imply the sampling access of A. In the following, we propose the *weighted sampling* method. The intuition is to

assign each A_ℓ a different weight when computing the probability distribution, and then sampling a row/column index of A according to this probability distribution.

We first give the method for sampling row indices of A as in Procedure Weighted sampling of rows. The objective of this procedure is to sample a submatrix S such that $S^\dagger S \approx A^\dagger A$.

Procedure Weighted sampling of rows

Input: $A = \sum_{l=1}^{\tau} A_\ell$ where each A_ℓ has the sampling access as in Definition 6.1.1; integer p.

1 Sample p indices $i_1, \ldots, i_p$ from $[n]$ according to the probability distribution $\{P_1, \ldots, P_n\}$ where $P_i = \sum_{j=1}^{\tau} \mathcal{D}_{\mathrm{rows}(A_j)}(i) \left\|A_j\right\|_F^2 / \left(\sum_{\ell=1}^{\tau} \left\|A_\ell\right\|_F^2\right)$;

After applying Procedure Weighted sampling of rows, we obtain the row indices $i_1, \ldots, i_p$. Let $S_1, \ldots, S_\tau$ be matrices such that $S_\ell(t, \cdot) = A_\ell(i_t, \cdot)/\sqrt{pP_{i_t}}$ for all $t \in [p]$ and $\ell \in [\tau]$. Define the matrix S as

$$S = S_1 + \cdots + S_\tau. \tag{6.3.1}$$

Next, we give the method for sampling column indices of S as in Procedure Weighted sampling of columns: we need to sample a submatrix W from S such that $WW^\dagger \approx SS^\dagger$.

Now, we obtained column indices $j_1, \ldots, j_p$. Let $W_1, \ldots, W_\tau$ be matrices such that $W_\ell(\cdot, t) = S_\ell(\cdot, j_t)/\sqrt{pP'_{j_t}}$ for all $t \in [p]$ and $\ell \in [\tau]$, where $P'_j = \frac{1}{p}\sum_{t=1}^{p} Q_{j|i_t}$ for $j \in [n]$. Define the matrix W as

$$W = W_1 + \cdots + W_\tau. \tag{6.3.2}$$

Procedure Weighted sampling of columns

Input: $A = \sum_{l=1}^{\tau} A_\ell$ where each A_ℓ has the sampling access as in Definition 6.1.1; $i_1, \ldots, i_p$ obtained in Procedure Weighted sampling of rows; integer p.

Do the following p times independently to obtain samples $j_1, \ldots, j_p$. **begin**

  Sample a row index $t \in [p]$ uniformly at random;

  Sample a column index $j \in [n]$ from the probability distribution $\{Q_{1|i_t}, \ldots, Q_{n|i_t}\}$ where $Q_{j|i_t} = \sum_{k=1}^{\tau} \mathcal{D}_{A_k(i_t,\cdot)}(j) \|A_k(i_t,\cdot)\|^2 / \left(\sum_{\ell=1}^{\tau} \|A_\ell(i_t,\cdot)\|^2\right)$;

Before showing $S^\dagger S \approx A^\dagger A$ and $SS^\dagger \approx WW^\dagger$, we first prove the following general result.

Lemma 6.3.1. *Let $M_1, \ldots, M_\tau \in \mathbb{C}^{n\times n}$ be a matrices. Independently sample p rows indices $i_1, \ldots, i_p$ from $M = M_1 + \cdots + M_\tau$ according to the probability distribution $\{P_1, \ldots, P_n\}$ where*

$$P_i \geq \frac{\sum_{j=1}^{\tau} \mathcal{D}_{\mathrm{rows}(M_j)}(i) \left\|M_j\right\|_F^2}{(\tau+1)\sum_{\ell=1}^{\tau} \left\|M_\ell\right\|_F^2}. \tag{6.3.3}$$

Let $N_1, \ldots, N_\tau \in \mathbb{C}^{n\times n}$ be matrices with

$$N_\ell(i_t,\cdot) = \frac{M_\ell(i_t,\cdot)}{\sqrt{pP_{i_t}}}, \tag{6.3.4}$$

for $t \in [p]$ and $\ell \in [\tau]$. Define $N = N_1 + \cdots + N_\tau$. Then for all $\theta > 0$, it holds that

$$\Pr\left(\left\|M^\dagger M - N^\dagger N\right\|_F \geq \theta \sum_{\ell=1}^{\tau} \left\|M_\ell\right\|_F^2\right) \leq \frac{(\tau+1)^2}{\theta^2 p}. \tag{6.3.5}$$

Proof. We first show that the expected value of each entry of $N^\dagger N$ is the corre-

sponding entry of $M^\dagger M$ as follows.

$$\mathbb{E}\left(N^\dagger(i,\cdot)N(\cdot,j)\right) = \sum_{t=1}^{p}\mathbb{E}\left(N^*(t,i)N(t,j)\right) = \sum_{t=1}^{p}\sum_{k=1}^{n} P_k \frac{M^*(k,i)M(k,j)}{pP_k}$$
$$= M^\dagger(i,\cdot)M(\cdot,j). \tag{6.3.6}$$

Furthermore, we have

$$\begin{aligned}
&\mathbb{E}\left(|N^\dagger(i,\cdot)N(\cdot,j) - M^\dagger(i,\cdot)M(\cdot,j)|^2\right)\\
&\quad\le \sum_{t=1}^{p}\mathbb{E}\left((N^*(t,i)N(t,j))^2\right)\\
&\quad= \sum_{t=1}^{p}\sum_{k=1}^{n} P_k \frac{(M^*(k,i))^2(M(k,j))^2}{p^2P_k^2}\\
&\quad\le \frac{(\tau+1)\sum_{\ell=1}^{\tau}\left\|M_\ell\right\|_F^2}{p}\sum_{k=1}^{n}\frac{(M^*(k,i))^2(M(k,j))^2}{\sum_{\ell'=1}^{\tau}\mathcal{D}_{\mathrm{rows}(M_{\ell'})}(k)\left\|M_{\ell'}\right\|_F^2}\\
&\quad= \frac{(\tau+1)\sum_{\ell=1}^{\tau}\left\|M_\ell\right\|_F^2}{p}\sum_{k=1}^{n}\frac{(M^*(k,i))^2(M(k,j))^2}{\sum_{\ell'=1}^{\tau}\left\|M_{\ell'}(k,\cdot)\right\|^2}.
\end{aligned}\tag{6.3.7}$$

Now, we bound the expected distance between $N^\dagger N$ and $M^\dagger M$:

$$\begin{aligned}
\mathbb{E}\left(\left\|M^\dagger M - N^\dagger N\right\|_F^2\right) &= \sum_{i,j=1}^{n}\mathbb{E}\left(|N^\dagger(i,\cdot)N(\cdot,j) - M^\dagger(i,\cdot)M(\cdot,j)|^2\right)\\
&\le \frac{(\tau+1)\sum_{\ell=1}^{\tau}\left\|M_\ell\right\|_F^2}{p}\sum_{k=1}^{n}\frac{\sum_{i,j=1}^{n}(M^*(k,i))^2(M(k,j))^2}{\sum_{\ell'=1}^{\tau}\left\|M_{\ell'}(k,\cdot)\right\|^2}\\
&= \frac{(\tau+1)\sum_{\ell=1}^{\tau}\left\|M_\ell\right\|_F^2}{p}\sum_{k=1}^{n}\frac{\left\|M(k,\cdot)\right\|^4}{\sum_{\ell'=1}^{\tau}\left\|M_{\ell'}(k,\cdot)\right\|^2}\\
&= \frac{\tau(\tau+1)\sum_{\ell=1}^{\tau}\left\|M_\ell\right\|_F^2}{p}\left\|M\right\|_F^2\\
&\le \frac{(\tau+1)^2\left(\sum_{\ell=1}^{\tau}\left\|M_\ell\right\|_F^2\right)^2}{p}.
\end{aligned}\tag{6.3.8}$$

Consequently, the result of this lemma follows from Markov's inequality. □

The following technical claim will be used multiple times in this paper. It relates the three quantities: $\sum_{\ell=1}^{\tau}\|A_\ell\|_F^2$, $\sum_{\ell=1}^{\tau}\|S_\ell\|_F^2$, and $\sum_{\ell=1}^{\tau}\|W_\ell\|_F^2$:

Claim 6.3.1. *Let $A = A_1 + \cdots + A_m$ be a matrix with the sampling access for each A_ℓ as in Definition 6.1.1. Let S and W be defined by (6.3.1) and (6.3.2). Then, with probability at least $1 - 2\tau^2/p$ it holds that*

$$\frac{1}{\tau+1}\sum_{\ell=1}^{\tau}\|A_\ell\|_F^2 \le \sum_{\ell=1}^{\tau}\|S_\ell\|_F^2 \le \frac{2\tau+1}{\tau+1}\sum_{\ell=1}^{\tau}\|A_\ell\|_F^2, \tag{6.3.9}$$

and

$$\frac{1}{\tau+1}\sum_{\ell=1}^{\tau}\|S_\ell\|_F^2 \le \sum_{\ell=1}^{\tau}\|W_\ell\|_F^2 \le \frac{2\tau+1}{\tau+1}\sum_{\ell=1}^{\tau}\|S_\ell\|_F^2, \tag{6.3.10}$$

Proof. We first evaluate $\mathbb{E}(\|S_\ell\|_F^2)$ as follows. For all $\ell \in [\tau]$,

$$\mathbb{E}\left(\|S_\ell\|_F^2\right) = \sum_{i=1}^{p}\mathbb{E}\left(\|S_\ell(i,\cdot)\|^2\right) = \sum_{i=1}^{p}\sum_{j=1}^{n} P_j \frac{\|A_\ell(j,\cdot)\|^2}{pP_j} \tag{6.3.11}$$

$$= \sum_{j=1}^{n}\|A_\ell(j,\cdot)\|^2 = \|A_\ell\|_F^2. \tag{6.3.12}$$

Then we have

$$\|S_\ell(i,\cdot)\|^2 = \sum_{j=1}^{n}\frac{|A_\ell(i,j)|^2}{pP_i} \le \sum_{j=1}^{n}\frac{2|A_\ell(i,j)|^2\sum_{\ell=1}^{\tau}\|A_\ell\|_F^2}{p\sum_{j=1}^{\tau}\|A_j(i,\cdot)\|^2} \tag{6.3.13}$$

$$= \frac{2\|A_\ell(i,\cdot)\|^2\sum_{\ell=1}^{\tau}\|A_\ell\|_F^2}{p\sum_{j=1}^{\tau}\|A_j(i,\cdot)\|^2} \le \frac{2\sum_{\ell=1}^{\tau}\|A_\ell\|_F^2}{p}. \tag{6.3.14}$$

Note that the quantity $\left\|S_\ell\right\|_F^2$ can be viewed as a sum of p independent random variables $\|S_\ell(1,\cdot)\|^2, \ldots, \|S_\ell(p,\cdot)\|^2$. As a result,

$$\begin{aligned}
\mathrm{Var}(\left\|S_\ell\right\|_F^2) &= p\mathrm{Var}(\|S_\ell(i,\cdot)\|^2) && (6.3.15)\\
&\leq p\mathbb{E}\left(\|S_\ell(i,\cdot)\|^4\right) && (6.3.16)\\
&\leq p\sum_{i=1}^{n} P_i \left(\frac{2\sum_{j=1}^{\tau}\left\|A_j\right\|_F^2}{p}\right)^2 = \frac{2\left(\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2\right)^2}{p}. && (6.3.17)
\end{aligned}$$

According to Chebyshev's inequality, we have

$$\Pr\left(\left|\left\|S_\ell\right\|_F^2 - \left\|A_\ell\right\|_F^2\right| \geq \frac{\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2}{\tau}\right) \leq \frac{2\left(\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2\right)^2}{p} = \frac{2\tau^2}{p}. \tag{6.3.18}$$

Therefore, with probability at least $1-\frac{2\tau^2}{p}$, it holds that

$$\begin{aligned}
-\frac{1}{\tau+1}\sum_{j\neq\ell}\left\|A_j\right\|_F^2 + \frac{\tau}{\tau+1}\left\|A_\ell\right\|_F^2 &\leq \left\|S_\ell\right\|_F^2 && (6.3.19)\\
&\leq \frac{1}{\tau+1}\sum_{j\neq\ell}\left\|A_j\right\|_F^2 + \frac{\tau+2}{\tau+1}\left\|A_\ell\right\|_F^2, && (6.3.20)
\end{aligned}$$

which implies that

$$\frac{1}{\tau+1}\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2 \leq \sum_{\ell=1}^{n}\left\|S_\ell\right\|_F^2 \leq \frac{2\tau+1}{\tau+1}\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2. \tag{6.3.21}$$

(6.3.10) can be proven in a similar way. □

Now, the main result of the weighted sampling method, namely $A^\dagger A \approx S^\dagger S$ and $WW^\dagger \approx SS^\dagger$, is a consequence of Lemma 6.3.1:

Corollary 6.3.1. *Let $A = A_1 + \cdots + A_m$ be a matrix with the sampling access for each A_ℓ as in Definition 6.1.1. Let S and W be defined by (6.3.1) and (6.3.2). Letting $\theta = (\tau+1)\sqrt{\frac{100}{p}}$, then, with probability at least $9/10$, the following holds:*

$$\left\|A^\dagger A - S^\dagger S\right\|_F \le \theta \sum_{\ell=1}^{\tau} \left\|A_\ell\right\|_F^2, \text{ and} \tag{6.3.22}$$

$$\left\|SS^\dagger - WW^\dagger\right\|_F \le \theta \sum_{\ell=1}^{\tau} \left\|S_\ell\right\|_F^2 \le 2\theta \sum_{\ell=1}^{\tau} \left\|A_\ell\right\|_F^2. \tag{6.3.23}$$

Proof. First note that (6.3.22) follows from Lemma 6.3.1. For the second statement, we need the probability P'_j to satisfy (6.3.3) in Lemma 6.3.1. In fact,

$$P'_j = \sum_{t=1}^{p} \frac{Q_{j|i_t}}{p} = \frac{1}{p}\sum_{t=1}^{p} \frac{\sum_{k=1}^{\tau} \mathcal{D}_{A_k(i_t,\cdot)}(j)\|A_k(i_t,\cdot)\|^2}{\sum_{\ell=1}^{\tau}\|A_\ell(i_t,\cdot)\|^2} \tag{6.3.24}$$

$$= \frac{1}{p}\sum_{t=1}^{p} \frac{\sum_{k=1}^{\tau} |A_k(i_t,j)|^2}{\sum_{\ell=1}^{\tau}\|A_\ell(i_t,\cdot)\|^2} \tag{6.3.25}$$

$$= \frac{1}{p}\sum_{t=1}^{p} \frac{pP_{i_t}\sum_{k=1}^{\tau} |S_k(i_t,j)|^2}{\sum_{\ell=1}^{\tau}\|A_\ell(i_t,\cdot)\|^2} \tag{6.3.26}$$

$$= \sum_{t=1}^{p} \frac{\sum_{j=1}^{\tau}\|A_j(i_t,\cdot)\|^2}{\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2} \frac{\sum_{k=1}^{\tau} |S_k(i_t,j)|^2}{\sum_{\ell=1}^{\tau}\|A_\ell(i_t,\cdot)\|^2} \tag{6.3.27}$$

$$= \sum_{t=1}^{p} \frac{\sum_{k=1}^{\tau} |S_k(i_t,j)|^2}{\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2} \tag{6.3.28}$$

$$= \frac{\sum_{k=1}^{\tau}\|S_k(\cdot,j)\|^2}{\sum_{\ell=1}^{\tau}\left\|A_\ell\right\|_F^2} \tag{6.3.29}$$

$$\ge \frac{\sum_{k=1}^{\tau}\|S_k(\cdot,j)\|^2}{(\tau+1)\sum_{\ell=1}^{\tau}\left\|S_\ell\right\|_F^2}, \tag{6.3.30}$$

where the last inequality follows from Claim 6.3.1. Note that the probability satisfies (6.3.3); as a result of Lemma 6.3.1, (6.3.23) holds. □

With the weighted sampling method, we obtained a small submatrix W from A. Now, we use the singular values and singular vectors of W to approximate the ones of A. This is shown in Algorithm 6.2.

Algorithm 6.2: Approximation of singular vectors.

Input: $A = A_1 + \cdots + A_\tau$ with the sampling access as in Definition 6.1.1 for each A_ℓ and $\mathrm{rank}(A_\ell) \leq r$; error parameter ϵ.

1 Set $p = 2 \cdot 10^{20} \frac{\tau^{12} r^{19}}{\epsilon^6}$, $\gamma = \frac{\epsilon^2}{3 \times 10^6 \tau^2 r^6}$;

2 Use Procedure Weighted sampling of rows to obtain row indices $i_1, \ldots, i_p$;

3 Let $S_1, \ldots, S_\tau$ be matrices such that $S_\ell(t, \cdot) = A_\ell(i_t, \cdot)/\sqrt{pP_{i_t}}$ for all $t \in [p]$ and $\ell \in [\tau]$, where P_i is defined in Line 1 in Procedure Weighted sampling of rows. Let $S = S_1 + \cdots + S_\tau$;

4 Use Procedure Weighted sampling of columns to obtain column indices $j_1, \ldots, j_p$;

5 Let $W_1, \ldots, W_\tau$ be matrices such that $W_\ell(\cdot, t) = S_\ell(\cdot, j_t)/\sqrt{pP'_{j_t}}$ for all $t \in [p]$ and $\ell \in [\tau]$, where $P'_j = \frac{1}{p}\sum_{t=1}^{p} Q_{j|i_t}$ for $j \in [n]$ and $Q_{j|i}$ is defined in Line 3 in Procedure Weighted sampling of columns. Let $W = W_1 + \cdots + W_\tau$;

6 Compute the top $\hat{r}$ singular values $\sigma_1, \ldots, \sigma_{\hat{r}}$ and their corresponding left singular vectors $u_1, \ldots, u_{\hat{r}}$;

7 Discard the singular values and their corresponding singular vectors satisfying $\sigma_j^2 < \gamma \sum_{\ell=1}^{m} \|W_\ell\|_F^2$. Let the remaining number of singular values be $\tilde{r}$;

8 Output $\sigma_1, \ldots, \sigma_{\tilde{r}}$ and $u_1, \ldots, u_{\tilde{r}}$;

An important result of Algorithm 6.2 is that the vectors $u_1, \ldots, u_{\tilde{r}}$ are approximately orthonormal, as stated in the following lemma:

Lemma 6.3.2. *Let $A = A_1 + \cdots + A_\tau$ be a matrix with the sampling access to each A_ℓ as in Definition 6.1.1. Assume $\|A_\ell\| \leq 1$ and $\mathrm{rank}(A_\ell) \leq r$ for all $\ell \in [\tau]$. Take A and error parameter ϵ as the input of Algorithm 6.2 and obtain the $\sigma_1, \ldots, \sigma_{\tilde{r}}$ and $u_1, \ldots, u_{\tilde{r}}$. Let $V \in \mathbb{C}^{n \times \tilde{r}}$ be the matrix such that $V(\cdot, j) = \frac{S^\dagger}{\sigma_j} u_j$ for $j \in \{1, \ldots, \tilde{r}\}$. Then, with probability at least $9/10$, the following statements hold:*

1. *There exists an isometry $U \in \mathbb{C}^{n \times \tilde{r}}$ whose column vectors span the column*

space of V *satisfying* $\left\|U - V\right\|_F \leq \frac{\epsilon}{300r^2(\tau+1)}$.

2. $|\|V\| - 1| \leq \frac{\epsilon}{300r^2(\tau+1)}$.

3. *Let* Π_V *be the projector on the column space of* V*, then it holds that* $\left\|VV^\dagger - \Pi_V\right\|_F \leq \frac{\epsilon}{300r^2(\tau+1)}$.

4. $\left\|V^\dagger V - I\right\|_F \leq \frac{\epsilon}{300r^2(\tau+1)}$.

The complete proof of this lemma is given in [77, Appendix C]. Note that following the proof, one can get a tight bound which is $\frac{\sqrt{2}\epsilon}{\tau^3 r^2} + O(\epsilon^2)$. However, for the convenience of the analysis in the rest of the paper, we choose a looser bound $\frac{\epsilon}{300r^2(\tau+1)}$ as in Lemma 6.3.2.

Algorithm 6.2 is similar to the main algorithm in [113] except for the different sampling method used here. In terms of the low-rank approximation, a similar result holds as follows.

Lemma 6.3.3. *Let* $A = A_1 + \cdots + A_\tau \in \mathbb{C}^{n\times n}$ *be a Hermitian matrix where* $A_\ell \in \mathbb{C}^{n\times n}$ *is Hermitian,* $\|A_\ell\| \leq 1$*, and* $\mathrm{rank}(A_\ell) \leq r$ *for all* $\ell \in [\tau]$*. The sampling access each* A_ℓ *is given as in Definition 6.1.1. Take* A *and error parameter* ϵ *as the input of Algorithm 6.2 to obtain the* $\sigma_1, \ldots, \sigma_{\tilde{r}}$ *and* $u_1, \ldots, u_{\tilde{r}}$*. Let* $V \in \mathbb{C}^{n\times\tilde{r}}$ *be the matrix such that* $V(\cdot, j) = \frac{S^\dagger}{\sigma_j} u_j$ *for* $j \in \{1, \ldots, \tilde{r}\}$*. Then, with probability at least* $9/10$*, it holds that* $\left\|AVV^\dagger - A\right\|_F \leq \frac{\epsilon}{300r^2}$.

The proof of this lemma mostly follows the proof of the FKV algorithm but with the weighted sampling method; see [77, Appendix B].

To our purpose, the main consequence of Algorithm 6.2 is summarized in the following theorem.

Theorem 6.3.1. *Let $A = A_1 + \cdots + A_\tau \in \mathbb{C}^{n\times n}$ be a Hermitian matrix where $A_\ell \in \mathbb{C}^{n\times n}$ is Hermitian, $\|A_\ell\| \le 1$, and $\operatorname{rank}(A_\ell) \le r$ for all $\ell \in [\tau]$. The sampling access each A_ℓ is given as in Definition 6.1.1. Take A and error parameter ϵ as the input of Algorithm 6.2 to obtain the $\sigma_1, \ldots, \sigma_{\tilde{r}}$ and $u_1, \ldots, u_{\tilde{r}}$. Let $V \in \mathbb{C}^{n\times\tilde{r}}$ be the matrix such that $V(\cdot, j) = \frac{S^\dagger}{\sigma_j} u_j$ for $j \in \{1, \ldots, \tilde{r}\}$. Then with probability at lease 9/10, it holds that $\left\|VV^\dagger AVV^\dagger - A\right\|_F \le \frac{\epsilon}{300r^2}(1 + \frac{\epsilon}{300r^2(\tau+1)}) + \frac{\epsilon}{300r^2}$.*

Proof. By Lemma 6.3.3, we have

$$\left\|AVV^\dagger - A\right\|_F \le \frac{\epsilon}{300r^2}. \tag{6.3.31}$$

By taking adjoint, we have

$$\left\|VV^\dagger A - A\right\|_F \le \frac{\epsilon}{300r^2}. \tag{6.3.32}$$

Then,

$$\begin{aligned}\left\|VV^\dagger AVV^\dagger - A\right\|_F &\le \left\|VV^\dagger AVV^\dagger - AVV^\dagger\right\|_F + \left\|AVV^\dagger - A\right\|_F &(6.3.33)\\ &\le \frac{\epsilon}{300r^2}\left(1 + \frac{\epsilon}{300r^2(\tau+1)}\right) + \frac{\epsilon}{300r^2}, &(6.3.34)\end{aligned}$$

where the last inequality follows from Lemma 6.3.2. Then the result follows. □

6.3.2 Symmetric approximation of low-rank Hermitian matrices

In this section, we show that the spectral decomposition of the sum of low-rank Hermitian matrices can be approximated in time logarithmic in the dimension with the given data structure. We call this technique *symmetric approximation*.

Briefly speaking, suppose we are given the approximated left singular vectors V of A from Algorithm 6.2 such that $\|VV^\dagger AVV^\dagger - A\|$ is bounded as in Theorem 6.3.1, then we can approximately do spectral decomposition of A as follows. First, we approximate the matrix $V^\dagger AV$ by sampling. Then, since $V^\dagger AV$ is a matrix with low dimension, we can do spectral decomposition of the matrix efficiently as $UDU^\dagger$. Finally, we show that (VU) is close to an isometry. Therefore, $(VU)D(VU)^\dagger$ is an approximation to the spectral decomposition of A.

Algorithm 6.3: Approximation of the spectral decomposition of A.

Input: $A = A_1 + \cdots + A_\tau$ with the query and sampling access as in Definition 6.1.1 for each A_ℓ; error parameter ϵ.

1 Compute the matrix $\tilde{B}$ according to Lemma 6.3.4.;
2 Compute the spectral decomposition $UDU^\dagger$ of matrix $\tilde{B}$.;
3 Output an isometry U and a diagonal matrix D such that $UDU^\dagger$ is the spectral decomposition of $\tilde{B}$. U and $\tilde{B}$ satisfy Lemma 6.3.5.

The algorithm for approximating the spectral decomposition of A is Algorithm 6.3. We first introduce a useful Claim from [122, Lemma 11].

Claim 6.3.2 (Trace inner product estimation)**.** *Let $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times n}$ be two Hermitian matrices. Given sampling and query access to A and query access to B. Then one can estimate* $\mathrm{Tr}[AB]$ *with the additive error ϵ_s with probability at least*

$1-\delta$ *by using*

$$O\Big(\frac{\|A\|_F\|B\|_F}{\epsilon_s^2}\big(Q(A)+Q(B)+S(A)+N(A)\big)\log\frac{1}{\delta}\Big)$$

time and queries, where $Q(B)$ *is the cost of query access to* B*, and* $Q(A), S(A), N(A)$ *are the cost of query access, sampling access and norm access to* A*.*

By using Claim 6.3.2, we approximate $V^\dagger AV$ as follows.

Lemma 6.3.4. *Let* $V \in \mathbb{C}^{n\times r}$ *and* $A = \sum_\ell^\tau A_\ell \in \mathbb{C}^{n\times n}$ *be a Hermitian matrix. Given query access and sampling access to* A_ℓ *for* $\ell \in [\tau]$*, and query access to* V*. Then, one can output a Hermitian matrix* $\tilde{B} \in \mathbb{C}^{r\times r}$ *such that* $\|V^\dagger AV - \tilde{B}\|_F \le \epsilon_s$ *with probability* $1-\delta$ *by using* $O((p+\log n)\frac{2^{16}r^9\tau^3}{\epsilon^2}\log\frac{1}{\delta})$ *samples and time.*

Proof. Let $B_t = V^\dagger A_t V$ for $t \in [\tau]$ and $B = \sum_{t=1}^\tau B_t$. $B_t(i,j) = V^\dagger(i,\cdot)A_tV(\cdot,j)$. Then, by Claim 6.3.2, one can estimate $V^\dagger(i,\cdot)A_tV(\cdot,j)$ with error at most $\epsilon_s/r\sqrt{\tau}$ with probability $1-\frac{2\delta}{\tau(r^2+r)}$ by using $O(\|A_t\|_F\|V(i,\cdot)\|\|V(j,\cdot)\|\frac{r^2\tau}{\epsilon_s^2}\log\frac{(r^2+r)\tau}{2\delta})$ queries. We denote the estimation to $B_t(i,j)$ as $\tilde{B}(i,j)$.

Since A_t is a Hermitian matrix, we only need to compute $(r^2+r)/2$ elements. Hence,

$$\Pr\Big[|B_t(i,j)-\tilde{B}_t(i,j)| \le \epsilon_s/r\sqrt{\tau} \text{ for all } i,j\in[r]\Big] \ge 1-\delta/\tau. \tag{6.3.35}$$

Then, let us consider $B_1,\dots,B_t$,

$$\Pr\Big[|B_t(i,j)-\tilde{B}_t(i,j)| \le \epsilon_s/r\sqrt{\tau} \text{ for all } i,j\in[r],\, t\in[\tau]\Big] \ge 1-\delta. \tag{6.3.36}$$

Now, we are guaranteed that for all $t \in [\tau]$, $-\epsilon\vec{1}\vec{1}^\dagger \leq B_t - \tilde{B}_t \leq \epsilon\vec{1}\vec{1}^\dagger$ with probability at least $1-\delta$. Let $\tilde{B} = \sum_t \tilde{B}_t$. With probability $1-\delta$,

$$\|B - \tilde{B}\|_F \leq \sqrt{r^2\tau(\epsilon_s^2/r^2\tau)} = \epsilon_s. \tag{6.3.37}$$

□

Then, we prove that the matrix multiplication of an isometry and a matrix satisfying Lemma 6.3.2 is still close to an isometry.

Lemma 6.3.5. *Let $U \in \mathbb{C}^{r\times r}$ be a unitary matrix and $V \in \mathbb{C}^{n\times r}$ be a matrix which satisfies Lemma 6.3.2 with error parameter $\frac{\epsilon}{300r^2(\tau+1)}$. Then the following properties hold for the matrix VU.*

1. *There exists an isometry $W \in \mathbb{C}^{n\times r}$ such that W spans the column space of VU and $\|VU - W\|_F \leq \frac{\epsilon}{300r^2(\tau+1)}$.*
2. *$|\|VU\| - 1| \leq \frac{\epsilon}{300r^2(\tau+1)}$.*
3. *$\|(VU)^\dagger(VU) - I_r\|_F \leq \frac{\epsilon}{300r^2(\tau+1)}$.*
4. *Let Π_{VU} be the projector of the column space of UV. Then $\|(VU)(VU)^\dagger - \Pi_{VU}\|_F \leq \frac{3\epsilon}{300r^2(\tau+1)}$.*

Proof. By Lemma 6.3.2, there exists an isometry $W' \in \mathbb{C}^{n\times r}$ such that W' spans the column space of V and $\|V - W'\|_F \leq \frac{\epsilon}{300r^2(\tau+1)}$. Let $W = W'U$,

$$\|VU - W\|_F = \|VU - W'U\|_F \leq \|V - W'\|_F\|U\| \leq \frac{\epsilon}{300r^2(\tau+1)}. \tag{6.3.38}$$

Note that W is also an isometry.

For the second property, by (6.3.38), we can get the following inequality

$$|\|VU\| - 1| = |\|VU\| - \|W\|| \leq \|VU - W\| \leq \frac{\epsilon}{300r^2(\tau+1)}. \tag{6.3.39}$$

For the third inequality,

$$\|(VU)^\dagger(VU) - I\|_F = \|U^\dagger V^\dagger VU - U^\dagger U\|_F \leq \|U^\dagger\|\|V^\dagger V - I\|_F\|U^\dagger\| \tag{6.3.40}$$

$$\leq \frac{\epsilon}{300r^2(\tau+1)}. \tag{6.3.41}$$

The last inequality holds because of Lemma 6.3.2. Finally,

$$\|VUU^\dagger V^\dagger - \Pi_{VU}\|_F = \|VUU^\dagger V^\dagger - WW^\dagger\|_F \tag{6.3.42}$$

$$= \|VUU^\dagger V^\dagger - VUW' + VUW' - W'UU^\dagger W'^\dagger\|_F \tag{6.3.43}$$

$$\leq \|VU\|\|U^\dagger V^\dagger - W'^\dagger\|_F + \|VU - W'\|_F\|W'^\dagger\| \tag{6.3.44}$$

$$\leq \left(1 + \frac{\epsilon}{300r^2(\tau+1)}\right)\frac{\epsilon}{300r^2(\tau+1)} + \frac{\epsilon}{300r^2(\tau+1)} \tag{6.3.45}$$

$$\leq \frac{3\epsilon}{300r^2(\tau+1)}. \tag{6.3.46}$$

□

We conclude by the following main theorem of this section:

Theorem 6.3.2. *Let $A_1, \ldots, A_\tau \in \mathbb{C}^{n\times n}$ be Hermitian matrices with rank at most r, and $A = \sum_{\ell=1}^{\tau} A_\ell$. Suppose given V which satisfies $\|AVV^\dagger - A\| \leq \frac{\epsilon}{300r^2}$ and statements 1 to 4 in Lemma 6.3.2. Then, there exists an algorithm which outputs*

a Hermitian matrix $\tilde{B} \in \mathbb{C}^{r\times r}$ *with probability at least* $1-\delta$ *with time and query complexity* $O((p+\log n)\frac{2^{16}r^9\tau^3}{\epsilon^2}\log\frac{1}{\delta})$ *such that the following properties holds.*

1. $\|V\tilde{B}V^\dagger - A\| \leq (1+\frac{\epsilon}{300r^2(\tau+1)})^2\frac{\epsilon}{400r^2} + (2+\frac{\epsilon}{300r^2(\tau+1)})\frac{\epsilon}{300r^2}$.
2. *Let* $UDU^\dagger$ *be the spectral decomposition of* $\tilde{B}$ *and, then statements 1 to 4 in Lemma 6.3.5 hold for* UV.

Proof. By Lemma 6.3.4, we can compute $\tilde{B}$ in time $O((p+\log n)\frac{2^{16}r^9\tau^3}{\epsilon^2}\log\frac{1}{\delta})$.

For the first statement, we have

$$\begin{aligned}
&\|V\tilde{B}V^\dagger - VV^\dagger AVV^\dagger + VV^\dagger AVV^\dagger - A\| \\
&\quad\leq \|V\tilde{B}V^\dagger - VV^\dagger AVV^\dagger\| + \|VV^\dagger AVV^\dagger - A\| && (6.3.47)\\
&\quad\leq \left(1+\frac{\epsilon}{300r^2(\tau+1)}\right)^2\frac{\epsilon}{400r^2} + \left(2+\frac{\epsilon}{300r^2(\tau+1)}\right)\frac{\epsilon}{300r^2}. && (6.3.48)
\end{aligned}$$

The first term of the last inequality comes from Lemma 6.3.4 with $\epsilon_s = \frac{\epsilon}{400r^2}$. The second statement directly follows from Lemma 6.3.5. □

6.4 Gibbs states

In this section, we combine our techniques from Section 6.3.1 and Section 6.3.2 to give a sampling-based estimator of the traces of a Gibbs state times a constraint A_ℓ. This is formulated as Algorithm 6.4.

We show that the output of Algorithm 6.4 ϵ-approximates $\mathrm{Tr}[A_\ell\rho]$ for $\rho = e^{-\frac{\epsilon}{2}A}/\mathrm{Tr}[e^{-\frac{\epsilon}{2}A}]$ in the following two subsections.

Algorithm 6.4: Approximation of the trace.

Input: Given query and sampling access to a constraint A_ℓ, query access to U, and the matrix D where $UDU^\dagger$ is an spectral decomposition of $\tilde{B}$ such that $(VU)D(VU)^\dagger$ is an approximated spectral decompistion of $A = \sum_i A_i$ as in Theorem 6.3.2.

1 Compute $\mathrm{Tr}[e^{-\frac{\epsilon}{2}D}]$;

2 Approximate $\mathrm{Tr}[A_\ell(VU)(e^{-\frac{\epsilon}{2}D}/\mathrm{Tr}[e^{-\frac{\epsilon}{2}D}])(VU)^\dagger]$ by ζ according to Claim 6.3.2;

3 Output ζ.

6.4.1 Estimating matrices inner product

Lemma 6.4.1. *Let $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times n}$, and $B' \in \mathbb{C}^{n\times n}$ be Hermitian matrices. Suppose $\|B - B'\| \leq \frac{\epsilon}{300r^2(\tau+1)}$. Then*

$$|\mathrm{Tr}[AB] - \mathrm{Tr}[AB']| \leq \frac{\epsilon}{300r^2(\tau+1)} \mathrm{Tr}\,|A|. \tag{6.4.1}$$

Proof. Let $A = \sum_i \sigma_i v_i v_i^\dagger$. We have

$$|\mathrm{Tr}[AB] - \mathrm{Tr}[AB']| = \sum_i \sigma_i v_i^\dagger (B - B') v_i \tag{6.4.2}$$

$$\leq \sum_i |\sigma_i| \|B - B'\| \leq \frac{\epsilon}{300r^2(\tau+1)} \mathrm{Tr}\,|A|. \tag{6.4.3}$$

□

Lemma 6.4.2. *Let A and B be Hermitian matrices, and $\|A-B\| \leq \epsilon$. Let $\rho_A(\frac{\epsilon}{2}) = \frac{e^{-\frac{\epsilon}{2}A}}{\mathrm{Tr}[e^{-A}]}$ and $\rho_B(\frac{\epsilon}{2}) = \frac{e^{-\frac{\epsilon}{2}B}}{\mathrm{Tr}[e^{-B}]}$. Then $F(\rho_A(\frac{\epsilon}{2}), \rho_B(\frac{\epsilon}{2})) \geq e^{-\frac{\epsilon}{2}\epsilon}$, where $F(\rho_A, \rho_B) := \mathrm{Tr}\left[\sqrt{\sqrt{\rho_A}\rho_B\sqrt{\rho_A}}\right]$ is the fidelity between ρ_A and ρ_B.*

This lemma has been proven in [229, Appendix C]; its complete proof is also

given in [77, Appendix A].

Lemma 6.4.2 implies that the trace distance between ρ_A and ρ_B is

$$\frac{1}{2}\operatorname{Tr}|\rho_A - \rho_B| \leq \sqrt{1 - e^{-2\frac{\epsilon}{2}\epsilon}}, \tag{6.4.4}$$

and the spectral distance is

$$\|\rho_A(\epsilon/2) - \rho_B(\epsilon/2)\| \leq 2\sqrt{1 - e^{-2\frac{\epsilon}{2}\epsilon}}. \tag{6.4.5}$$

6.4.2 Approximating the Gibbs state

Let $\tilde{A} = VV^\dagger AVV^\dagger$ and U, D, and $\tilde{B}$ be outputs of Algorithm 6.3. In this section, we suppose $\|\tilde{A} - A\| \leq (1 + \frac{\epsilon}{300r^2(\tau+1)})^2\frac{\epsilon}{400r^2} + (2 + \frac{\epsilon}{300r^2(\tau+1)})\frac{\epsilon}{300r^2}$ as in Theorem 6.3.1.

Theorem 6.4.1. *Let* $\rho = \frac{e^{-\frac{\epsilon}{2}A}}{\operatorname{Tr} e^{-\frac{\epsilon}{2}A}}$ *and* $\hat{\rho} = \frac{\tilde{V}e^{-\frac{\epsilon}{2}D}\tilde{V}^\dagger}{\operatorname{Tr} e^{-\frac{\epsilon}{2}D}}$. *Suppose* $\|A - \tilde{A}\|_F \leq (1 + \frac{\epsilon}{300r^2(\tau+1)})^2\frac{\epsilon}{400r^2} + (2 + \frac{\epsilon}{300r^2(\tau+1)})\frac{\epsilon}{300r^2}$. *Let* A_ℓ *be a Hermitian matrix with the promise that* $\|A_\ell\| \leq 1$ *and* $\operatorname{rank}(A_\ell) \leq r$. *Then Algorithm 6.4 outputs* ζ *such that*

$$|\operatorname{Tr}[A_\ell\rho] - \zeta| \leq \epsilon \tag{6.4.6}$$

with probability $1 - \delta$ *in time* $O(\frac{4}{\epsilon^2}(\log^2 n + \tau pr)\log\frac{1}{\delta})$.

Proof. As we have proven in Lemma 6.3.5, there exists an isometry $\tilde{U}$ such that $\|\tilde{U} - \tilde{V}\| \leq \frac{\epsilon}{300r^2(\tau+1)}$ and $\tilde{U}$ spans the column space of $\tilde{V}$. We define two additional

Gibbs states $\rho' = \frac{\tilde{U}e^{-\frac{\epsilon}{2}D}\tilde{U}^\dagger}{\operatorname{Tr} e^{-\frac{\epsilon}{2}D}}$ and $\tilde{\rho} = \frac{e^{-\frac{\epsilon}{2}\tilde{A}}}{\operatorname{Tr} e^{-\frac{\epsilon}{2}\tilde{A}}}$.

$$
\begin{aligned}
&|\operatorname{Tr}[A_\ell \rho] - \operatorname{Tr}[A_\ell \hat{\rho}]| \\
&= |\operatorname{Tr}[A_\ell \rho] - \operatorname{Tr}[A_\ell \tilde{\rho}] + \operatorname{Tr}[A_\ell \tilde{\rho}] - \operatorname{Tr}[A_\ell \rho'] + \operatorname{Tr}[A_\ell \rho'] - \operatorname{Tr}[A_\ell \hat{\rho}] + \operatorname{Tr}[A_\ell \hat{\rho}] - \zeta| \\
&\leq |\operatorname{Tr}[A_\ell \rho] - \operatorname{Tr}[A_\ell \tilde{\rho}]| + |\operatorname{Tr}[A_\ell \tilde{\rho}] - \operatorname{Tr}[A_\ell \rho']| \\
&\quad + |\operatorname{Tr}[A_\ell \rho'] - \operatorname{Tr}[A_\ell \hat{\rho}]| + |\operatorname{Tr}[A_\ell \hat{\rho}] - \zeta|. \qquad (6.4.7)
\end{aligned}
$$

We give bounds on each term as follows. First,

$$
\begin{aligned}
|\operatorname{Tr}[A_\ell \rho] - \operatorname{Tr}[A_\ell \tilde{\rho}]| &\leq \operatorname{Tr}|A| \|\rho - \tilde{\rho}\| \qquad (6.4.8) \\
&\leq 2 \operatorname{Tr}|A| \sqrt{1 - e^{-2\frac{\epsilon}{2}((1+\frac{\epsilon}{300r^2(\tau+1)})^2 \frac{\epsilon}{400r^2} + (2+\frac{\epsilon}{300r^2(\tau+1)})\frac{\epsilon}{300r^2})}}. \qquad (6.4.9)
\end{aligned}
$$

For $|\operatorname{Tr}[A_\ell \tilde{\rho}] - \operatorname{Tr}[A_\ell \rho']|$, we first compute an upper bound on $\|\tilde{V}D\tilde{V} - \tilde{U}D\tilde{U}\|$.

$$
\|\tilde{V}D\tilde{V}^\dagger - \tilde{U}D\tilde{U}^\dagger\| \leq \|\tilde{U} - \tilde{V}\| \|D\| (\|\tilde{V}\| + \|\tilde{U}\|) \leq 3\frac{\epsilon}{300r^2(\tau+1)}\|D\|. \qquad (6.4.10)
$$

Then, by applying Lemma 6.4.2 and Lemma 6.4.1 again, we get

$$
|\operatorname{Tr}[A_\ell \tilde{\rho}] - \operatorname{Tr}[A_\ell \rho']| \leq \operatorname{Tr}|A| \|\tilde{\rho} - \rho'\| \leq \operatorname{Tr}|A| \left(2\sqrt{1 - e^{-6\frac{\epsilon}{2}\frac{\epsilon}{300r^2(\tau+1)}\|D\|}}\right). \qquad (6.4.11)
$$

For the second last term $|\operatorname{Tr}[A_\ell \rho'] - \operatorname{Tr}[A_\ell \hat{\rho}]|$, it is not hard to show that

$$
\|\tilde{U}e^{-\frac{\epsilon}{2}D}\tilde{U}^\dagger - \tilde{V}e^{-\frac{\epsilon}{2}D}\tilde{V}^\dagger\| \leq 2\|\tilde{U} - \tilde{V}\| \operatorname{Tr}[e^{-\frac{\epsilon}{2}D}]. \qquad (6.4.12)
$$

Then,

$$|\operatorname{Tr}[A_\ell \rho'] - \operatorname{Tr}[A_\ell \hat{\rho}]| \leq \operatorname{Tr}|A_\ell| \|\rho' - \hat{\rho}\| \leq (2\|\tilde{U} - \tilde{V}\|) \operatorname{Tr}|A_\ell| \tag{6.4.13}$$

$$\leq 2\frac{\epsilon}{300r^2(\tau+1)} \operatorname{Tr}|A_\ell|. \tag{6.4.14}$$

The last term follows from Claim 6.3.2 by setting the precision to be $\epsilon/5$. Hence

$$|\operatorname{Tr}[A_\ell \hat{\rho}] - \zeta| \leq \epsilon/5. \tag{6.4.15}$$

By adding (6.4.9), (6.4.11), and (6.4.14) together,

$$|\operatorname{Tr}[A_\ell \rho] - \zeta| \leq \epsilon. \tag{6.4.16}$$

$\operatorname{Tr}[A_\ell \hat{\rho}]$ can be approximated with precision $\epsilon/5$ with probability $1-\delta$ in time

$$O\left(\frac{4}{\epsilon^2}(Q(A_\ell) + Q(VU) + S(A_\ell) + N(A_\ell)) \log\frac{1}{\delta}\right) = O\left(\frac{1}{\epsilon^2}(\log^2 n + \tau p r) \log\frac{1}{\delta}\right),$$

where p is the number of rows sampled in Algorithm 6.2 and the maximum rank of the Gibbs state is τr. The last equality is true since one can compute $(VU)(i,j)$ by computing $V(i,j)$ as $(S^\dagger(i,\cdot)u_j/\sigma_j$ and then compute the inner product $V(i,\cdot)U(\cdot,j)$, which takes $O(p\tau r)$ time. □

6.5 Main results: sampling-based SDP and shadow tomography solvers

We finally prove our main results on solving SDPs via sampling.

Theorem 6.5.1. *Given Hermitian matrices $\{A_1, \ldots, A_m\}$ with the promise that each of $A_1, \ldots, A_m$ has rank at most r, spectral norm at most 1, and the sampling access of each A_i is given as in Definition 6.1.1. Also given $a_1, \ldots, a_m \in \mathbb{R}$. Then for any $\epsilon > 0$, Algorithm 6.5 gives a succinct description and any entry (see Remark 6.5.1) of the solution of the SDP feasibility problem*

$$\mathrm{Tr}[A_i X] \leq a_i + \epsilon \quad \forall\, i \in [m]; \tag{6.5.1}$$

$$X \succeq 0; \tag{6.5.2}$$

$$\mathrm{Tr}[X] = 1 \tag{6.5.3}$$

with probability at least 2/3 in $O(\frac{mr^{57}\ln^{37} n}{\epsilon^{92}})$ time.

Algorithm 6.5: Feasibility testing of SDPs by our sampling approach.

1 Set the initial Gibbs state $\rho_1 = \frac{I_n}{n}$, and number of iterations $T = \frac{16 \ln n}{\epsilon^2}$;
2 **for** $t = 1, \ldots, T$ **do**
3 Find a $j_t \in [m]$ such that $\mathrm{Tr}[A_{j_t}\rho_t] > a_{j_t} + \epsilon$ using Algorithm 6.2, Algorithm 6.3, and Algorithm 6.4. If we cannot find such j_t, claim that $\rho_t \in \mathcal{S}_\epsilon$ and terminate the algorithm;
4 Define the new weight matrix $W_{t+1} := \exp\left[-\frac{\epsilon}{2}\sum_{i=1}^{t} A_{j_i}\right]$ and Gibbs state $\rho_{t+1} := \frac{W_{t+1}}{\mathrm{Tr}[W_{t+1}]}$;
5 Claim that the SDP is infeasible and terminate the algorithm;

The algorithm follows the master algorithm in Theorem 6.2.2. The main challenge is to estimate $\mathrm{Tr}[A_{j_t}\rho_t]$ where ρ_t is the Gibbs state at iteration t; this is achieved by Theorem 6.4.1 in Section 6.4.

Proof. We prove Theorem 6.5.1 by showing the correctness and the time complexity of Algorithm 6.5.

Correctness: The correctness of Algorithm 6.5 directly follows from Theorem 6.4.1. Specifically, we have shown that one can estimate the quantity $\mathrm{Tr}[A_{j_t}\rho_t]$ with precision ϵ with high probability by applying Algorithm 6.2, Algorithm 6.3, and Algorithm 6.4.

Time complexity: First, we show that given the data structure in Theorem 6.2.1, Algorithm 6.2 can be computed in time $O(p^3 + p\tau \log n)$. The Procedure Weighted sampling of rows and Procedure Weighted sampling of columns both can be done in time $O(p\tau \log n)$. For Procedure Weighted sampling of rows, let $A = A_1 + \cdots + A_\tau$, the probability that the i^{th} row is sampled in Procedure Weighted sampling of rows is

$$P_i = \frac{\sum_{\ell=1}^{\tau} \|A_\ell(i,\cdot)\|^2}{\sum_{k=1}^{\tau} \|A_k\|_F^2}. \tag{6.5.4}$$

With the data structure, the accumulated probability $P_1 + \cdots + P_t$ can be computed in time $O(\tau \log(n-t))$ for any $t \leq n$ since $\sum_{i=1}^{t} \|A_\ell(i,\cdot)\|^2$ and $\|A_\ell\|_F^2$ can be accessed in time $O(\log(n-t))$ and $O(\log n)$ given the data structure. Then we can use binary search to implement Procedure Weighted sampling of rows in time $O(\tau \log n)$. Specifically, we generate a random number $p \in [0,1]$, and then do the binary search in the data structure to find the index i such that $p \in [\sum_{j=1}^{i-1} P_j, \sum_{j=1}^{i} P_j]$. Similarly, we can implement Procedure Weighted sampling of columns in time $O(\tau \log n)$. Hence, the time complexity to construct the matrix W and compute its SVD is $O(p\tau \log n + p^3)$. Algorithm 6.2 succeeds with probability $9/10$.

Then, Algorithm 6.3 and Algorithm 6.4 take $O((p+\log n)\frac{2^{16}r^9\tau^3}{\epsilon^2}\log\frac{1}{\delta})$ and $O(\frac{4}{\epsilon^2}(\log^2 n+\tau pr)\log\frac{1}{\delta})$ and succeed with probability at least $1-2\delta$. By setting δ as a small enough constant (say $\delta=1/6$), Algorithm 6.5 succeeds with probability at least $2/3$ in time $O(\tau mp^3)=O(\frac{mr^{57}\ln^{37}n}{\epsilon^{92}})$. □

Remark 6.5.1. *Theorem 6.5.1 solves the SDP feasibility problem, i.e., to decide $\mathcal{S}_0=\varnothing$ or $\mathcal{S}_\epsilon\neq\varnothing$. For the SDP optimization problem in (6.1.1)-(6.1.3), the optimal value can be approximated by binary search (see Footnote 4); however, writing down the approximate solution would take n^2 space, ruining the poly-logarithmic complexity in n. Nevertheless,*

- *we have its* succinct representation

$$\frac{\exp\left[\frac{\epsilon}{2}\sum_{i=1}^t A_{j_i}\right]}{\mathrm{Tr}\left[\exp\left[\frac{\epsilon}{2}\sum_{i=1}^t A_{j_i}\right]\right]},\ \textit{and}$$

- *we can query any entry of the solution matrix.*

The succinct representation is given by Algorithm 6.5, where $t\le T$ and $j_\tau\in[m]$ for all $\tau\in[t]$. Storing all j_τ takes $t\log_2 m=O(\log m\log n/\epsilon^2)$ bits. A query to the solution is accessed by computing the element of $(VU)(e^{-\frac{\epsilon}{2}D}/\mathrm{Tr}[e^{-\frac{\epsilon}{2}D}])(VU)^\dagger$, which is an ϵ-approximation to the solution by Theorem 6.4.1 (this suffices because the SDP feasibility problem of deciding $\mathcal{S}_0=\varnothing$ or $\mathcal{S}_\epsilon\neq\varnothing$ is ϵ-approximate).

Shadow tomography. As a corollary of Theorem 6.5.1, we have:

Corollary 6.5.1. *Given Hermitian matrices $\{E_1,\dots,E_m\}$ with the promise that each of $E_1,\dots,E_m$ has rank at most r, $0\preceq E_i\preceq I$ and the sampling access to E_i*

is given as in Definition 6.1.1 for all $i \in [m]$. Also given $p_1, \ldots, p_m \in \mathbb{R}$. Then for any $\epsilon > 0$, the shadow tomography problem

$$\text{Find } \sigma \text{ such that} \qquad |\operatorname{Tr}[\sigma E_i] - p_i| \leq \epsilon \quad \forall\, i \in [m]; \tag{6.5.5}$$

$$\sigma \succeq 0, \quad \operatorname{Tr}[\sigma] = 1 \tag{6.5.6}$$

can be solved with probability $1 - \delta$ with cost $O(m \cdot \operatorname{poly}(\log n, 1/\epsilon, \log(1/\delta), r))$.

Here, $p_i = \operatorname{Tr}[\rho E_i]$ in (6.1.4) for all $i \in [m]$. Notice that the assumption of knowing $p_1, \ldots, p_m$ makes our problem slightly different from the shadow tomography problem in [3, 23, 55] where we are only given copies of the quantum state ρ without the knowledge of $\operatorname{Tr}[\rho E_1], \ldots, \operatorname{Tr}[\rho E_m]$. However, quantum state is a concept without a counterpart in classical computing, hence we follow the conventional assumption in SDPs that these real numbers are given.

Proof. We denote $A_i = E_i$ for all $i \in [m]$ and $A_i = -E_{i-m}$ for all $i \in \{m+1, \ldots, 2m\}$; also denote $a_i = p_i$ for all $i \in [m]$ and $a_i = -p_{i-m}$ for all $i \in \{m+1, \ldots, 2m\}$. As a result, $\operatorname{Tr}[\sigma E_i] - p_i \leq \epsilon$ is equivalent to $\operatorname{Tr}[\sigma A_i] \leq a_i + \epsilon$ for all $i \in [m]$, and $\operatorname{Tr}[\sigma E_i] - p_i \geq -\epsilon$ is equivalent to $\operatorname{Tr}[\sigma A_{i+m}] \leq a_{i+m} + \epsilon$ for all $i \in [m]$; therefore, the shadow tomography problem in (6.5.5) and (6.5.6) is equivalent to the following SDP feasibility problem:

$$\text{Find } \sigma \text{ such that} \qquad \operatorname{Tr}[A_i \sigma] \leq a_i + \epsilon \quad \forall\, i \in [2m]; \tag{6.5.7}$$

$$\sigma \succeq 0, \quad \operatorname{Tr}[\sigma] = 1. \tag{6.5.8}$$

Consequently, Corollary 6.5.1 reduces to the SDP in (6.5.1) to (6.5.3) with $2m$ constraints; the result hence follows from Theorem 6.5.1. □

Remark 6.5.2. *Similar to Remark 6.5.1, σ can be stored as a succinct representation. This is because*

$$\sigma = \frac{\exp\left[\frac{\epsilon}{2}\sum_{\tau=1}^{t}(-1)^{i_\tau}A_{j_\tau}\right]}{\mathrm{Tr}\left[\exp\left[\frac{\epsilon}{2}\sum_{\tau=1}^{t}(-1)^{i_\tau}A_{j_\tau}\right]\right]} \tag{6.5.9}$$

by the proof of Corollary 6.5.1, where $t \leq T$ and $i_\tau \in \{0,1\}$, $j_\tau \in [m]$ for all $\tau \in [t]$. Storing all i_τ, j_τ takes $t(\log_2 m + 1) = O(\log m \log n/\epsilon^2)$ bits.

6.6 Generalization: a framework for quantum-inspired classical algorithms

A central goal of the research into "quantum-inspired" classical machine learning is to guide quantum machine learning research in the future. However, previous research in this topic focuses on particular problems and only describes the particular tools that are necessary in each case. In this section, we will sketch an *easy-to-understand* framework of quantum-inspired classical algorithms recently proposed by [76], exploring the capabilities and limitations of these techniques.

Similar to the quantum-inspired classical SDP solver from Section 6.1 to Section 6.5, our framework assumes the sampling access in Definition 6.1.1 as well as query access to the entry of the input vectors and matrices. Our core primitive is *singular value transformation* [123]. Roughly speaking, given a Hermitian matrix A

with sampling and query access, along with a Lipschitz function f, we can achieve the sampling access of $f(A)$ where f is applied to the singular values up to additive Frobenius norm error. Moreover, we can gain sampling and query access to the decomposition of $f(A)$ into rank-1 matrices. This primitive has previously been noted to generalize a large portion of quantum machine learning research [123]; we bring this observation into the quantum-inspired landscape.

Theorem 6.6.1. *Let $A = UDV^\dagger$ be the singular value decomposition of $A = A^{(1)} + \cdots + A^{(\tau)}$ and let f be L-Lipschitz continuous. We can implement the sampling and query access of $Uf(D)V^\dagger$ up to ℓ_2-norm error ϵ in time $\widetilde{\mathcal{O}}\left(\left(\frac{\tau L^2(\sum_\ell \|A^{(\ell)}\|_F^2)}{\epsilon^2}\right)^{18}\right)$.*

With Theorem 6.6.1, we can recover existing quantum-inspired machine learning algorithms in [76]:

- Recommendation systems: Given a matrix $A \in \mathbb{R}^{m\times n}$ with the sampling access in Definition 6.1.1, a row index $i \in [m]$, and a singular value threshold σ, the goal is to sample from the i^{th} row of a low-rank approximation of A which singular values $\geq \sigma$ with additive error $\epsilon\|A\|_F$. We apply the main theorem to a constant-Lipschitz continuous function f such that $f(x) = x$ on singular values in $[\frac{7}{6}\sigma, 1]$ and $f(x) = 0$ in $[0, \frac{5}{6}\sigma]$, which gives us the sampling and query access to an approximated singular-value transformation of $f(A)$. Finally, we obtain a sample from the i^{th} row by the sampling techniques we have developed in [76, Section 3]. The running time is $\widetilde{\mathcal{O}}\left(\frac{\|A\|_F^{24}}{\epsilon^{12}\sigma^{24}}\right)$.

- Principal component analysis: Given a matrix $X \in \mathbb{R}^{m\times n}$ with the sampling access in Definition 6.1.1 such that $\text{rank}(X) = r$ and X^TX has nonzero eigenval-

ues $\{\lambda_i\}_{i=1}^r$ and eigenvectors $\{v_i\}_{i=1}^r$ (without loss of generality $\lambda_1 \geq \cdots \geq \lambda_r$), the goal is to output λ_i up to additive error $\epsilon \operatorname{Tr}(X^T X)$ and $|v_i\rangle$ with probability $\lambda_i / \operatorname{Tr}(X^T X)$. This is in general impossible because distinguishing between λ_i and λ_{i+1} such that $\lambda_i - \lambda_{i+1} = O(1/\operatorname{poly}(n))$ necessarily takes $\operatorname{poly}(n)$ samples. However, if we know $K := \operatorname{Tr}(X^T X)/\lambda_k \geq k$ and $\eta := \min_{i \in [k]} |\lambda_i - \lambda_{i+1}| / \operatorname{Tr}(X^T X)$, then we can apply our main theorem to the function $f(x) = x^2$ to get an approximated singular-value decomposition of $X^T X$ and apply sampling access as a coupon collector problem; by doing that, we get all $\{\lambda_i\}_{i=1}^r$ and the sampling access of $\{v_i\}_{i=1}^r$ in time $\widetilde{\mathcal{O}}\left(\frac{K}{(\epsilon\eta)^{18}}\right)$.

- Supervised clustering: Given a dataset of points $q_1, \ldots, q_m \in \mathbb{R}^n$ in $\mathbb{R}^n$, the goal is to estimate the distance between their centroid and a new point $p \in \mathbb{R}^n$, i.e., $\|p - \frac{1}{m}(q_1 + \cdots + q_m)\|^2$. We show how to use the sampling and query access to estimate inner products: given the sampling access of (M^T, w) where

$$M := \left[\frac{p}{\|p\|}, \frac{-q_1}{\|q_1\|}, \ldots, \frac{-q_m}{\|q_m\|}\right] \quad \text{and} \quad w := \left[\|p\|, \frac{\|q_1\|}{m}, \ldots, \frac{\|q_m\|}{m}\right]^T, \tag{6.6.1}$$

  we approximate $\|p - \frac{1}{m}(q_1 + \cdots + q_m)\|^2$ to additive ϵ error in time $\mathcal{O}\left(\|M\|_F^2 \|w\| \frac{1}{\epsilon^2}\right)$.

- Matrix inversion: Given a matrix $A \in \mathbb{R}^{n \times n}$ with the sampling access in Definition 6.1.1 and condition number κ, the goal is to obtain the sampling access of A^+ where A^+ is the pseudo-inverse of A. We apply our main theorem to an $\mathcal{O}(\kappa)$-Lipschitz function that is $1/x$ for $x \in [1/\kappa, 1]$ and 0 when $x \in [0, (1-\xi)/\kappa]$ for a $0 < \xi < 1$, and we get the sampling access of A^+ with ϵ-error in spectral

norm in time $\widetilde{\mathcal{O}}\left(\left(\frac{\|A\|_F^2\kappa^4}{\epsilon^2\xi^2}\right)^{18}\right)$.

- Support vector machines: Given input data points $x_1, \ldots, x_m \in \mathbb{R}^n$ and their corresponding labels $y_1 \ldots, y_m = \pm 1$, let $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ be the specification of hyperplanes separating these points. The goal is to minimize the squared norm of the residuals:

$$\min_{w,b} \quad \frac{\|w\|^2}{2} + \frac{\gamma}{2}\|e\|^2 \tag{6.6.2}$$

$$\text{s.t.} \quad y_i(w^T x_i + b) = 1 - e(i), \quad \forall i \in [m], \tag{6.6.3}$$

where $e \in \mathbb{R}^m$ is a slack vector such that $e(j) \geq 0$ for $j \in [m]$. The dual of this problem is to maximize over the Karush-Kuhn-Tucker multipliers of a Lagrange function, taking partial derivatives of which yields a linear system:

$$\begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & X^T X + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}, \tag{6.6.4}$$

Therefore, solving this SVM can be regarded as solving a matrix inversion problem. Assuming the sampling access of X and the minimum nonzero singular value of $X^T X$ is at least $m\epsilon_\kappa$, SVM can be solved with error ϵ in time $\widetilde{\mathcal{O}}\left(\frac{\|X\|_F^{88}}{\epsilon^{26}\epsilon_\kappa^{72}}\right)$.

- Hamiltonian simulation: Given a Hermitian matrix $H \in \mathbb{R}^{n\times n}$ with the sampling access of H such that $\|H\| \leq 1$, a unit vector $b \in \mathbb{R}^n$ with the sampling access in Definition 6.1.1, and a time $t > 0$, the goal is to obtain the sampling access of v where $\|v - e^{itH}b\|_F \leq \epsilon$. We apply our main theorem to the function $f(x) = e^{itx}$

(which is 2π-Lipschitz) and obtain the sampling access of e^{itH}; we furthermore apply the matrix-vector product $e^{itH}b$ by a generalization of our main theorem. The final time complexity is $\widetilde{\mathcal{O}}\left(\frac{t^{36}\|H\|_F^{36}}{\epsilon^{36}}\right)$.

- Discriminant analysis: Given M input data points $\{x_i \in \mathbb{R}^N : 1 \leq i \leq M\}$, each belonging to one of k classes. Let μ_c denote the centroid (mean) of class $c \in [k]$, and $\bar{x}$ denote the centroid of all data points. Let

$$S_B = \sum_{c=1}^{k} (\mu_c - \bar{x})(\mu_c - \bar{x})^T, \qquad S_W = \sum_{c=1}^{k} \sum_{x \in c} (\mu_c - x)(\mu_c - x)^T \tag{6.6.5}$$

be the between-class scatter matrix and the weight matrix of the dataset, respectively. The goal is to find the largest p eigenvalues and eigenvectors of $S_W^{-1} S_B$. Given the sampling access of $X_{[k]}$, we apply our main theorem to an ϵ-approximation of the function $\frac{1}{\sqrt{x}}$ that is $\mathcal{O}(1/\epsilon)$-Lipschitz, with threshold θ; the overall complexity is $\mathcal{O}(\text{poly}(\|X\|_F, \epsilon^{-1}, \theta^{-1}))$.

For all these applications, please refer to [76] for their complete proofs.

6.7 Conclusions and discussion

We present a poly-logarithmic time classical algorithm for solving SDPs with low-rank constraints; specifically, given an SDP with m constraint matrices, each of dimension n and rank r, our algorithm can compute any entry and efficient descriptions of the spectral decomposition of the solution matrix. The algorithm runs in time $O(m \cdot \text{poly}(\log n, r, 1/\varepsilon))$ given access to a sampling-based low-overhead

data structure for the constraint matrices, where ε is the precision of the solution. Furthermore, our techniques can be improved to give a general framework of quantum-inspired classical algorithms, including applications such as recommendation systems, principal component analysis, supervised clustering, matrix inversion, support vector machines, Hamiltonian simulation, and discriminant analysis.

This chapter raises a few natural open questions for future work. For example:

- Can we give faster sampling-based algorithms for solving LPs? Note that a recent breakthrough by [86] solves LPs with complexity[8] $\tilde{O}(n^\omega)$, significantly faster than the state-of-the-art SDP solver [184] with complexity $\tilde{O}(m(m^2 + n^\omega + mn^2))$.

- Can we prove lower bounds on sampling-based methods? In particular, a lower bound in terms of the rank r can help us understand the limit of our current approach. It is also of interest to prove a lower bound in $1/\epsilon$ to better understand the trade-off between $1/\epsilon$ and n, r.

- What is the empirical performance of our sampling-based method? It is worthing mentioning that [30] conducted various numerical experiments on quantum-inspired classical algorithms and suggested that their performance in practice might work better than their theoretical guarantee. We look forward to more numerical evidence for sampling-based methods.

[8] Without loss of generality, we can assume $m \leq n$ for LPs by deleting overcomplete constraints. The result $\tilde{O}(n^\omega)$ only holds for the current matrix multiplication exponent $\omega \approx 2.373$; when $\omega = 2$, the complexity becomes $\tilde{O}(n^{13/6})$.

Chapter 7: Distribution Property Testing[1]

Having studied quantum algorithms for various machine learning and optimization problems, in the last chapter of this thesis we consider quantum algorithms for statistics. Specifically, we focus on testing properties of probability distributions: on the one hand, we study potential speedup of sample complexities if using quantum computers, and on the other hand we generalize to the problems of testing quantum states.

7.1 Introduction

Property testing is a rapidly developing field in theoretical computer science (e.g. see the survey [237]). It aims to determine properties of an object with the least number of independent samples of the object. Property testing is a theoretically appealing topic with intimate connections to statistics, learning theory, and algorithm design. One important topic in property testing is to estimate statistical properties of unknown distributions (e.g., [125, 265]), which are fundamental questions in statistics and information theory, given that much of science relies on samples furnished by nature.

[1]This chapter is based on the papers [121, 192] under the permission of all the authors.

The merit of distributional property testing mainly comes from the fact that the testing of many properties admits *sublinear* algorithms. For instance, given the ability to take samples from a discrete distribution p on $[n] := \{1, \ldots, n\}$, it requires $\Theta(n/\epsilon^2)$ samples to "learn" p, i.e., to construct a distribution q on $[n]$ such that $\|p - q\|_1 \leq \epsilon$ with success probability at least $2/3$ ($\|\cdot\|_1$ being ℓ^1-distance). However, testing whether $p = q$ or $\|p - q\|_1 > \epsilon$ requires only $\Theta(\max\{\frac{n^{2/3}}{\epsilon^{4/3}}, \frac{n^{1/2}}{\epsilon^2}\})$ samples from p and q [73], which is sublinear in n and significantly smaller than the complexity of learning the entire distributions. See Section 7.1.4 for more examples and discussions.

In this chapter, we study the impact of quantum computation on distributional property testing problems. We are motivated by the emerging topic of "quantum property testing" (see the survey of [210]) which focuses on investigating the quantum advantage in testing classical statistical properties. Quantum speed-ups have already been established for a few specific problems such as testing closeness between distributions [59, 208], testing identity to known distributions [70], estimating entropies [192], etc. In this chapter we propose a generic approach for quantum distributional property testing, and illustrate its power on a few examples. This is our attempt to make progress on the question:

> *Can quantum computers test properties of distributions systematically and more efficiently?*

7.1.1 Problem statements

Throughout the chapter, we denote probability distributions on $[n]$ by p and q; their ℓ^α-distance is defined as $\|p-q\|_\alpha := (\sum_{i=1}^n |p_i - q_i|^\alpha)^{\frac{1}{\alpha}}$. Similarly, we denote $n \times n$ density operators[2] (i.e., quantum distributions) by ρ and σ; their ℓ^α-distance is defined via the corresponding Schatten norm.

Input models. To formulate the problems we address, we define classical and quantum access models for distributions on $[n]$. We begin with the very natural model of sampling.

Definition 7.1.1 (Sampling)**.** *A classical distribution $(p_i)_{i=1}^n$ is accessible via* classical sampling *if we can request independent samples from the distribution, i.e., get a random $i \in [n]$ with probability p_i. A quantum distribution $\rho \in \mathbb{C}^{n\times n}$ is accessible via* quantum sampling *if we can request independent copies of the state ρ.*

Now we define a coherent analogue of the above sampling model. To our knowledge this type of query-access was only studied by a few earlier works [134, 208] and only in the special case of classical distributions. The motivation for this input model is the following: we can think about a density operator as the outcome of some physical process modeled by some black-box. Suppose that the black-box can generate samples on demand. Unlike in the classical (randomized) setting, in a quantum scenario in principle it is always possible to reverse every computational /

[2]For readers less familiar with quantum computing, a density operator (=quantum distribution) $\rho \in \mathbb{C}^{n\times n}$ is a positive semidefinite matrix with $\mathrm{Tr}[\rho] = 1$. Please refer to the textbook [217] for more information.

physical process – including this black-box. If reversion is not feasible, then we get the plain sampling model; however if it is possible to reverse the (quantum) black-box then we get the purified query access model that we describe. For example, if a quantum computer produces the samples via, say, a Monte Carlo method, then the process is easily reversible. However, if the samples come from some source "outside the lab", then reversing the process might not be possible. Therefore, both input models (purified quantum query access and sampling access) are well-motivated. The surprising fact is that this subtle difference in the input models gives rise to significantly different complexities, as we show later for several problems.

Definition 7.1.2 (Purified quantum query-access)**.** *A density operator $\rho \in \mathbb{C}^{n\times n}$, has* purified quantum query-access *if we have access to a unitary oracle U_ρ (and its inverse) acting as*[3]

$$U_\rho \left|0\right\rangle_A \left|0\right\rangle_B = \left|\psi_\rho\right\rangle_{AB} = \sum_{i=1}^{n} \sqrt{p_i} \left|\phi_i\right\rangle_A \left|\psi_i\right\rangle_B, \text{ where } \langle\phi_i|\phi_j\rangle = \langle\psi_i|\psi_j\rangle = \delta_{ij}$$

such that $\mathrm{Tr}_A\left(|\psi_\rho\rangle\langle\psi_\rho|\right) = \rho$*. If* $|\psi_i\rangle = |i\rangle$*, then* $\rho = \sum_i^n p_i |i\rangle\langle i|$ *is a diagonal density operator which can be identified with the classical distribution p, so we can simply write U_p instead of U_ρ. With a slight abuse of notation sometimes we will concisely write $|\rho\rangle$ instead of $|\psi_\rho\rangle$.*

We also define an even stronger input model that is considered in a series of earlier works, see, e.g., [59, 63, 70, 192].

[3] $|\psi\rangle \in \mathbb{C}^n$ denotes a "ket" vector and $\langle\psi| = (|\psi\rangle)^\dagger$ stands for its conjugate transpose, called "bra" in Dirac notation; $|i\rangle = \vec{e}_i$ is the i^{th} basis vector. An ℓ^2-normalized $|\psi\rangle$ is called a pure state, and corresponds to density operator $|\psi\rangle\langle\psi|$. For $A = \mathbb{C}^k, B = \mathbb{C}^n$ and $|\phi\rangle \in A \otimes B$ we denote by $\mathrm{tr}\{|\phi\rangle\langle\phi|\}_A \in B \otimes B^* = \mathbb{C}^{n\times n}$ the partial trace over A.

Definition 7.1.3 (Classical distribution with discrete query-access)**.** *A classical distribution* $(p_i)_{i=1}^n$*, has* discrete query-access *if we have classical / quantum query-access to a function* $f\colon S \to [n]$ *such that for all* $i \in [n]$*,* $p_i = |\{s \in [S] : f(s) = i\}|/|S|$*. (Typically the interesting regime is when* $|S| \gg n$*.) In the quantum case a query oracle is a unitary operator* O *acting on* $\mathbb{C}^{|S|} \otimes \mathbb{C}^n$ *as*

$$\mathrm{O}\colon\ |s, 0\rangle \leftrightarrow |s, f(s)\rangle \ \text{ for all } s \in S.$$

Note that if one first creates a uniform superposition over S and then makes a query, then the above oracle turns into a purified query oracle to a classical distribution as in Definition 7.1.2. Therefore all lower bounds that are proven in this model also apply to the purified query-access oracles. In fact all algorithms that the authors are aware of do this conversion, so they effectively work in the purified query-access model. Moreover, we conjecture that the two input models are equivalent when $|S| \gg n$. For this reason we only work with the purified query-access model in this work.

Another strengthening of the purified query-access model for classical distributions is when we assume access to a unitary (and its inverse) acting as $|0\rangle \mapsto \sum_{i=1}^n \sqrt{p_i}\,|i\rangle$. A very similar input model was thoroughly studied by [14].

Definition 7.1.4 (Classical distribution with pure-state preparation access)**.** *A classical distribution* $(p_i)_{i=1}^n$*, is accessible via* pure state preparation oracle *if we have*

access to a unitary oracle U_{pure} (and its inverse) acting as

$$U_{\text{pure}}\colon\ |0\rangle \mapsto \sum_{i=1}^{n} \sqrt{p_i}\,|i\rangle\,. \tag{7.1.1}$$

This is strictly stronger[4] than the purified query-access model. In order to simulate purified queries we can first do a pure state query and then copy $|i\rangle$ to a second fresh ancillary register using, e.g., some CNOT gates. Finally, for completeness we mention that one could also consider a model similar to the above where one can only request samples of pure states of the form $\sum_{i=1}^{n} \sqrt{p_i}\,|i\rangle$, as studied for example in [31, 33].

We will focus on the first two input models and will only use the latter strengthenings of the purified query-access model for invoking and proving lower bounds.

Property testing problems. We study three distributional properties: ℓ^α-closeness testing, independence testing, and entropy estimation. In the classical literature these are well-studied properties, and the corresponding testers motivate general algorithms for testing properties of discrete distributions [8, 96].

For brevity we only give the definitions for classical distributions; similar definitions apply to quantum density matrices if we replace vector norms by the corresponding Schatten norms.

[4]This can be seen in various ways. We give an argument in the spirit of distributional property testing. Closeness of two unknown distributions p, q can be tolerantly tested in the squared Hellinger distance $H(p,q)^2 = \frac{1}{2}\left\|\sqrt{p} - \sqrt{q}\right\|_2^2$ to precision ϵ in query complexity $\mathcal{O}\left(1/\sqrt{\epsilon}\right)$ in the model of Definition 7.1.4 using amplitude estimation. On the other hand the classical sample complexity of testing equality to ϵ precision in this metric is $\widetilde{\Theta}(\min(n^{2/3}/\epsilon^{4/3}, n^{3/4}/\epsilon))$, as shown in [96]. The results of Chailloux [65] imply that this query complexity improves at most cubically in the model of Definition 7.1.2, showing that the input model of Definition 7.1.4 is strictly stronger.

Definition 7.1.5 (ℓ^α-closeness testing)**.** *Given $\epsilon > 0$ and two probability distributions p, q on $[n]$,* ℓ^α-closeness testing *is to decide whether $p=q$ or $\|p-q\|_\alpha \geq \epsilon$ with success probability at least $\frac{2}{3}$. Tolerant testing: decide whether $\|p-q\|_\alpha \leq 0.99\epsilon$ or $\|p-q\|_\alpha \geq \epsilon$ with success probability at least $\frac{2}{3}$.*

Definition 7.1.6 (Independence testing)**.** *Given $\epsilon > 0$ and a probability distribution p on $[n]\times[m]$ with $n \geq m$,* independence testing *is to decide, with success probability at least $\frac{2}{3}$, whether p is a product distribution or p is ϵ-far in ℓ^1-norm from any product distribution on $[n] \times [m]$.*

Definition 7.1.7 (Entropy estimation)**.** *Given $\epsilon > 0$ and a density operator $\rho \in \mathbb{C}^{n\times n}$,* entropy estimation *is to estimate the Shannon / von Neumann entropy $H(\rho) = -\operatorname{tr}\{\rho \log(\rho)\}$ within additive ϵ-precision with success probability at least $\frac{2}{3}$.*

7.1.2 New results

We give a systematic study of distributional property testing for classical / quantum distributions, and obtain the following results for the purified quantum query model of Definition 7.1.2:

- Shannon entropy estimation of classical / quantum distributions costs $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\right)$ and $\widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)$ queries respectively, as we prove in Theorem 7.3.1 and Theorem 7.3.2.

- Tolerant ℓ^2-closeness testing of classical / quantum distributions costs $\tilde{\Theta}\left(\frac{1}{\epsilon}\right)$ and $\mathcal{O}\left(\min\left(\frac{\sqrt{n}}{\epsilon}, \frac{1}{\epsilon^2}\right)\right)$ queries respectively, as we prove in Theorem 7.3.3 and Theorem 7.3.4.

- ℓ^1-closeness testing of classical / quantum distributions costs $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon}\right)$ and $\mathcal{O}\left(\frac{n}{\epsilon}\right)$ queries respectively, as we prove in Corollary 7.3.1.

- Independence testing of classical / quantum distributions costs $\widetilde{\mathcal{O}}\left(\frac{\sqrt{nm}}{\epsilon}\right)$ and $\mathcal{O}\left(\frac{nm}{\epsilon}\right)$ queries respectively, as we prove in Corollary 7.3.2.

- For all $\alpha \geq 0$, there is quantum speedup on α-Rényi entropy estimation, as we prove in Theorem 7.4.1.

For context, we compare our results with previous classical and quantum results in Table 7.1 and Table 7.2. (Note that all of our results are gate efficient, because they are based on singular value transformation and amplitude estimation, both of which have gate-efficient implementations.)

problem / model	ℓ^1-closeness testing	(tolerant) ℓ^2-closeness testing	Shannon / von Neumann entropy
Classical distribution sampling	$\Theta\left(\max\left\{\frac{n^{2/3}}{\epsilon^{4/3}}, \frac{n^{1/2}}{\epsilon^2}\right\}\right)$ [73]	$\Theta\left(\frac{1}{\epsilon^2}\right)$ [73]	$\Theta\left(\frac{n}{\epsilon \log n} + \frac{\log^2 n}{\epsilon^2}\right)$ [153, 275]
Classical distribution with purified query-access	$\boldsymbol{\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon}\right)}$	$\boldsymbol{\widetilde{\Theta}\left(\frac{1}{\epsilon}\right)}$	$\boldsymbol{\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\right)}$; $\tilde{\Omega}(\sqrt{n})$ [63]
Quantum state with purified query-access	$\boldsymbol{\mathcal{O}\left(\frac{n}{\epsilon}\right)}$	$\boldsymbol{\mathcal{O}\left(\min\left(\frac{\sqrt{n}}{\epsilon}, \frac{1}{\epsilon^2}\right)\right)}$	$\boldsymbol{\widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)}$
Quantum state sampling	$\Theta\left(\frac{n}{\epsilon^2}\right)$ [38]	$\Theta\left(\frac{1}{\epsilon^2}\right)$ [38]	$\mathcal{O}\left(\frac{n^2}{\epsilon^2}\right)$, $\Omega\left(\frac{n^2}{\epsilon}\right)$ [10]

Table 7.1: Summary of sample and query complexity results of distributional property testing. Our new bounds are printed in **bold**. For classical distributions with quantum query-access we prove (almost) matching upper and lower bounds for ℓ^2-testing, and improve the previous best complexity $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^{2.5}\right)$ for ℓ^1-testing by [208] and $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^2\right)$ for Shannon entropy estimation by [192]. Note that Ref. [65] imply that in this model quantum speed-ups are at most cubic. The results for Rényi entropy estimation are summarized in Table 7.3 separately.

As we show our quantum algorithms for classical distributional property testing problems with purified access can be naturally lifted to the case of quantum distributions, incurring an overhead of $\approx \sqrt{n}$, which is manifested in the complexities of Table 7.1.

	Sample complexity	(Purified) Query complexity
Classical	$\Theta\left(\frac{n}{\log n}\right)$ [262]	$\widetilde{\Theta}\left(\sqrt{n}\right)$ [63, 192]
Quantum	$\Theta\left(n^2\right)$ [10]	$\boldsymbol{\widetilde{\mathcal{O}}\left(n\right)}$

Table 7.2: Complexities of Shannon / von Neumann entropy estimation with constant precision. It seems that the n-dependence is roughly quadratically higher for quantum distributions, while coherent quantum access gives a quadratic advantage for both classical and quantum distributions. This suggests that our entropy estimation algorithm has essentially optimal n-dependence for density operators with purified access, however we do not have a matching lower bound yet.

7.1.3 New techniques

The motivating idea behind our approach is that if we can prepare a purification of a quantum distribution / density operator ρ, then we can construct a unitary U, which has this density operator in the top-left corner, using only two queries to U_ρ. This observation is due to [203]. We call such a unitary a *block-encoding* of ρ:

$$U = \begin{bmatrix} \rho & . \\ . & . \end{bmatrix} \quad \Longleftrightarrow \quad \rho = \left(\langle 0|^{\otimes a} \otimes I\right) U \left(|0\rangle^{\otimes a} \otimes I\right). \tag{7.1.2}$$

One can think of a block-encoding as a post-selective implementation of the linear map ρ: given an input state $|\psi\rangle$, applying the unitary U to the state $|0\rangle^{\otimes a}|\psi\rangle$, measuring the first a-qubit register and post-selecting on the $|0\rangle^{\otimes a}$ outcome, we get a state $\propto \rho|\psi\rangle$ in the second register. Block-encodings are easy to work with, for example given a block-encoding of ρ and σ we can easily construct a block-encoding of $(\rho - \sigma)/2$, see for example in the work of [68].

Example application to ℓ^3-testing. The problem is to decide whether $\rho = \sigma$ or $\|\rho - \sigma\|_3 \geq \epsilon$, with query complexity $\mathcal{O}\left(\epsilon^{-\frac{3}{2}}\right)$. The first idea is that if we can

prepare a purification of ρ and σ, then we can also prepare a purification of $(\rho+\sigma)/2$ by setting a qubit to the state $(|0\rangle + |1\rangle)/\sqrt{2}$, and then controlled on the $|0\rangle$ or $|1\rangle$ value of the qubit run the process that samples from ρ or σ, respectively. The second idea is to combine the block-encodings of ρ and σ to apply the map $\frac{\rho-\sigma}{2}$ to the purification of $(\rho+\sigma)/2$, to get

$$\left|\frac{\rho+\sigma}{2}\right\rangle \mapsto \left(\frac{\rho-\sigma}{2}\otimes I\right)\left|\frac{\rho+\sigma}{2}\right\rangle|0\rangle + \ldots|1\rangle\,. \tag{7.1.3}$$

Finally, apply amplitude estimation with setting $M = \Theta(\epsilon^{-\frac{3}{2}})$. This works since if $\|\rho-\sigma\|_3 \geq \epsilon$, then the $|0\rangle$ ancilla state has probability $\mathrm{tr}\{(\rho-\sigma)^2(\rho+\sigma)\}/8 \geq \mathrm{tr}\{|\rho-\sigma|^3\}/8 \geq \epsilon^3/8$.

Working with singular values. The above is a promising approach because it directly makes the density operator in question operationally accessible. However, it turns out that using this simple block-encodings is often suboptimal for distribution testing, because a query in some sense gives access to the square-root of ρ, whereas this unitary has ρ itself in the top-left corner. Since the problems often heavily depend on smaller eigenvalues of ρ, the square root of ρ is more desirable since it has quadratically larger singular/eigenvalues.

One of our main technical contributions is to use a new type of block-encoding, which is a unitary matrix having a certain block proportional to a matrix A such that $A^\dagger A = \rho$, i.e., we use a "square-root" of the (quantum) distribution ρ (in the case of classical distributions rho is a diagonal matrix with the probabilities as

diagonal entries). This new technique allows us to develop a unified approach for distributional property testing, which we consider one of our major contributions. It is this new perspective that enables us to derive several results in a relatively short paper. Once we establish this methodology the results are relatively easy to derive in a systematic way, both for classical and quantum distributions.

Therefore, we show how to efficiently construct a unitary matrix whose top-left corner contains a matrix with singular values $\sqrt{p_1}, \ldots, \sqrt{p_n}$, given purified access to a classical distribution p. To be more precise, we define a slight generalization of block-encodings called *projected unitary encodings*, which represent a matrix A in the form of $\Pi U \widetilde{\Pi}$, where $\Pi, \widetilde{\Pi}$ are orthogonal projectors and U is a unitary matrix. One can think about U in a projected unitary encoding as a post-selective implementation of the map $A\colon \operatorname{img}\left(\widetilde{\Pi}\right) \to \operatorname{img}(\Pi)$. Take for example $U := (U_p \otimes I)$, $\Pi := \left(\sum_{i=1}^{n} I \otimes |i\rangle\langle i| \otimes |i\rangle\langle i|\right)$, and $\widetilde{\Pi} := (|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I)$. As we show in Section 7.2.4 these operators form a projected unitary encoding of

$$A = \Pi U \widetilde{\Pi} = \sum_{i=1}^{n} \sqrt{p_i}\, |\phi_i\rangle\langle 0| \otimes |i\rangle\langle 0| \otimes |i\rangle\langle i| \,. \tag{7.1.4}$$

We can use a similar trick for a general density operator ρ too. However, there is a major difficulty which arises from the fact that we do not a prioiri know the diagonalizing basis of ρ. Therefore we use slightly different operators. Let W be a unitary,[6] mapping $|0\rangle\,|0\rangle \mapsto \sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}}$. Let $U' := \left(I \otimes U_\rho^\dagger\right)\left(W^\dagger \otimes I\right)$, $\Pi' := (I \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0|)$ and $\widetilde{\Pi}$ as above. As we show in Section 7.2.4 these operators

[6] This unitary is easy to implement, e.g., by using a few Hadamard and CNOT gates.

form a projected unitary encoding of

$$A' = \Pi' U' \widetilde{\Pi} = \sum_{i=1}^{n} \sqrt{\frac{p_i}{n}} \, |\phi_i'\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes |0\rangle\langle \psi_i| \,, \tag{7.1.5}$$

where $\sum_{j=1}^{n} \frac{|\phi_j'\rangle|\phi_j\rangle}{\sqrt{n}} = \sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}}$ is the Schmidt decomposition of the maximally entangled state under the basis $(|\phi_1\rangle, \ldots, |\phi_n\rangle)$.

As we can see, the case of general density operators is less efficient, it only gives operational access to the "square root" of ρ/n. We note that for (approximately) transforming a block-encoding of $A/\sqrt{n}$ to a block-encoding of $A/\mathcal{O}(1)$ it is necessary and sufficient to use the block-encoding of $A/\sqrt{n}$ about $\sqrt{n}$ times [123, Theorems 3 and 17]. This is essentially the reason for the $\approx \sqrt{n}$ overhead in our quantum algorithms for quantum distributions in Table 7.1. If the $1/\sqrt{n}$ factor could be directly improved, that would speed up our von Neumann entropy estimation algorithm Theorem 7.3.2, which seems unlikely, cf. Table 7.2. This suggests that it is impossible to obtain a more efficient block-encoding in the general case.

General recipe. We summarize our algorithms as follows.

1. Construct the quantum circuit / unitary matrix representing the distribution.

2. Transform the singular values of the matrix according to a desired function.

3. Apply the resulting map to the purification of the distribution.

4. Estimate the amplitude of the flagged output state and conclude.

The above general scheme describes our approach to the problems we discuss in this

paper. Sometimes it is useful to divide the probabilities / singular values into bins, and fine-tune the algorithm by using the approximate knowledge of the size of the singular values. This divide-and-conquer strategy is at the core of our improved tolerant ℓ^2-closeness tester of Theorem 7.3.3.

7.1.4 Related works on distributional property testing

Classical algorithms. Many distributional property testing problems fall into the category of *closeness testing*, where we are given the ability to take independent samples from two unknown distributions p and q with cardinality n, and the goal is to determine whether they are the same versus significantly different. For ℓ^1-*closeness testing*, which is about testing whether $p = q$ or $\|p - q\|_1 \geq \epsilon$, [42] first gave a sublinear algorithm using $\tilde{O}(n^{2/3}/\epsilon^{8/3})$ samples to p and q. The follow-up work by [73] determined the optimal sample complexity as $\Theta(\max\{\frac{n^{2/3}}{\epsilon^{4/3}}, \frac{n^{1/2}}{\epsilon^2}\})$; the same paper also gave a tight bound $\Theta(\frac{1}{\epsilon^2})$ for ℓ^2-*closeness testing*.

Besides closeness testing, a similar problem is *identity* testing where one of the distributions, say q, is known and we are given independent samples from the other distribution p. For ℓ^1 *identity testing*, it is known that the sample complexity can be smaller than that of ℓ^1-closeness testing, which was proved by [41] to be $\tilde{O}(\sqrt{n}/\epsilon^4)$ and then [227] gave the tight bound $\Theta(\sqrt{n}/\epsilon^2)$. More recently, Ref. [96] proposed a modular reduction-based approach for distributional property testing problems, which recovered all closeness and identity testing results above. Furthermore, they also studied *independence testing* (see also the previous studies by [9, 41, 189]), i.e.,

whether a distribution on $[n] \times [m]$ ($n \geq m$) is a product distribution or at least ϵ-far in ℓ^1-distance from any product distribution, and determined the optimal bound $\Theta(\max\{\frac{n^{2/3}m^{1/3}}{\epsilon^{4/3}}, \frac{(nm)^{1/2}}{\epsilon^2}\})$.

Apart from the relationship between distributions, properties of a single distribution also have been extensively studied. One of the most important properties is *Shannon entropy* [244] because it measures for example compressibility. The sample complexity of estimating $H(p)$ within additive error ϵ has been intensively studied [40, 225, 226]; in particular, [262, 263] gave an explicit algorithm for entropy estimation using $\Theta(\frac{n}{\epsilon \log n})$ samples when $\epsilon = \Omega(n^{-0.03})$ and $\epsilon = O(1)$; for the general case [153] and [275] gave the optimal estimator with $\Theta\left(\frac{n}{\epsilon \log n} + \frac{(\log n)^2}{\epsilon^2}\right)$ samples.

Quantum algorithms. The first paper on distributional property testing by quantum algorithms was by [59], which considered classical distributions with discrete quantum query-access (see Definition 7.1.3); it gives a quantum query complexity upper bound $O(\sqrt{n}/\epsilon^6)$ for ℓ^1-closeness testing and $O(n^{1/3}/\epsilon^{4/3})$ for identity testing to the uniform distribution on $[n]$. Subsequently, [70] gave an algorithm for identity testing (to an arbitrary known distribution) with $\widetilde{\mathcal{O}}\left(n^{1/3}/\epsilon^5\right)$ queries, and [208] improved the ϵ-dependence of ℓ^1-closeness testing to $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^{2.5}\right)$. More recently, [192] studied entropy estimation under this model, and gave a quantum algorithm for Shannon entropy estimation with $\widetilde{\mathcal{O}}\left(\sqrt{n}/\epsilon^2\right)$ queries and also sublinear quantum algorithms for estimating Rényi entropies ([234]).

Another type of quantum property testing results ([10, 38, 132, 219–221]) concern *density matrices*, where the ℓ^1-distance becomes the trace distance and the

Shannon entropy becomes the von Neumann entropy. To be more specific, for n-dimensional density matrices, the number of samples needed for ℓ^1 and ℓ^2-closeness testing are $\Theta(n/\epsilon^2)$ and $\Theta(1/\epsilon^2)$ ([38]), respectively. In addition [10] gave upper and lower bounds $\mathcal{O}\left(n^2/\epsilon^2\right), \Omega\left(n^2/\epsilon\right)$ for estimating the von Neumann entropy of an n-dimensional density matrix with accuracy ϵ.

7.2 Technical tools

7.2.1 Amplitude estimation

Classically, given i.i.d. samples of a Bernoulli random variable X with $\mathbb{E}[X] = p$, it takes $\Theta(1/\epsilon^2)$ samples to estimate p within ϵ with high success probability. Quantumly, if we are given a unitary U such that

$$U|0\rangle|0\rangle = \sqrt{p}|0\rangle|\phi\rangle + |0^\perp\rangle, \quad \text{where } \||\phi\rangle\| = 1 \text{ and } (\langle 0| \otimes I)|0^\perp\rangle = 0, \tag{7.2.1}$$

then if measure the output state, we get 0 in the first register with probability p. Given access to U we can estimate the value of p quadratically more efficiently than what is possible by sampling:

Theorem 7.2.1 ([57, Theorem 12])**.** *Given U satisfying (7.2.1), the amplitude estimation algorithm outputs $\tilde{p}$ such that $\tilde{p} \in [0, 1]$ and*

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2} \tag{7.2.2}$$

with success probability at least $8/\pi^2$*, using* M *calls to* U *and* $U^\dagger$*.*

In particular, if we take $M = \left\lceil 2\pi \left(\frac{2\sqrt{p}}{\epsilon} + \frac{1}{\sqrt{\epsilon}} \right) \right\rceil = \Theta\left(\frac{\sqrt{p}}{\epsilon} + \frac{1}{\sqrt{\epsilon}} \right)$ in (7.2.2),

$$|\tilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{2\pi}\epsilon + \frac{\pi^2}{4\pi^2}\epsilon^2 \leq \frac{\epsilon}{2} + \frac{\epsilon}{4} \leq \epsilon. \tag{7.2.3}$$

Therefore, using only $\Theta(1/\epsilon)$ implementations of U and $U^\dagger$, we could get an ϵ-additive approximation of p with success probability at least $8/\pi^2$, which is a quadratic speed-up compared to the classical sample complexity $\Theta(1/\epsilon^2)$. The success probability can be boosted to $1 - \nu$ by executing the algorithm for $\Theta(\log 1/\nu)$ times and taking the median of the estimates.

7.2.2 Quantum singular value transformation

Singular value decomposition (SVD) is one of the most important tools in linear algebra, generalizing eigen-decomposition of Hermitian matrices. Recently, [123] proposed *quantum singular value transformation* which turns out to be very useful for property testing. Mathematically, it is defined as follows:

Definition 7.2.1 (Singular value transformation)**.** *Let* $f : \mathbb{R} \to \mathbb{C}$ *be an even or odd function. Let* $A \in \mathbb{C}^{\tilde{d} \times d}$ *have the following singular value decomposition*

$$A = \sum_{i=1}^{d_{\min}} \varsigma_i \left| \tilde{\psi}_i \right\rangle \left\langle \psi_i \right|,$$

where $d_{\min} := \min(d, \tilde{d})$*. For the function* f *we define the* singular value transform

of A as

$$f^{(SV)}(A) := \begin{cases} \sum_{i=1}^{d_{\min}} f(\varsigma_i) \left|\tilde{\psi}_i\right\rangle\!\left\langle\psi_i\right| & \text{if } f \text{ is odd, and} \\ \sum_{i=1}^{d} f(\varsigma_i) \left|\psi_i\right\rangle\!\left\langle\psi_i\right| & \text{if } f \text{ is even, where for } i \in [d] \setminus [d_{\min}] \text{ we define } \varsigma_i := 0. \end{cases}$$

Quantum singular value transformation by real polynomials can be efficiently implemented on a quantum computer as follows:

Theorem 7.2.2 ([123, Corollary 18])**.** *Let $\mathcal{H}_U$ be a finite-dimensional Hilbert space and let $U, \Pi, \widetilde{\Pi} \in \mathrm{End}(\mathcal{H}_U)$ be linear operators on $\mathcal{H}_U$ such that U is a unitary, and $\Pi, \widetilde{\Pi}$ are orthogonal projectors. Suppose that $P = \sum_{k=0}^{n} a_k x^k \in \mathbb{R}[x]$ is a degree-n polynomial such that*

- *$a_k \neq 0$ only if $k \equiv n \mod 2$, and*
- *for all $x \in [-1, 1]$: $|P(x)| \leq 1$.*

Then there exist $\Phi \in \mathbb{R}^n$, such that

$$P^{(SV)}\left(\widetilde{\Pi} U \Pi\right) = \begin{cases} \left(\langle +| \otimes \widetilde{\Pi}\right)\left(|0\rangle\!\langle 0| \otimes U_\Phi + |1\rangle\!\langle 1| \otimes U_{-\Phi}\right)\left(|+\rangle \otimes \Pi\right) & \text{if } n \text{ is odd, and} \\ \left(\langle +| \otimes \Pi\right)\left(|0\rangle\!\langle 0| \otimes U_\Phi + |1\rangle\!\langle 1| \otimes U_{-\Phi}\right)\left(|+\rangle \otimes \Pi\right) & \text{if } n \text{ is even,} \end{cases}$$

where $U_\Phi = e^{i\phi_1(2\widetilde{\Pi}-I)} U \prod_{j=1}^{(n-1)/2} \left(e^{i\phi_{2j}(2\Pi-I)} U^\dagger e^{i\phi_{2j+1}(2\widetilde{\Pi}-I)} U\right)$.[7]

Thus for an even or odd polynomial P of degree n, we can apply singular value transformation of the matrix $\widetilde{\Pi} U \Pi$ with n uses of U, $U^\dagger$ and the same number of controlled reflections $I - 2\Pi, I - 2\widetilde{\Pi}$.

[7] This is the mathematical form for odd n; even n is defined similarly.

7.2.3 Polynomial approximations for singular value transformation

To apply singular value transformation corresponding to our problems, we need low-degree polynomial approximations to the following functions:

Lemma 7.2.1. (Polynomial approximations) *Let $\beta \in (0,1]$, $\eta \in (0, \frac{1}{2}]$ and $t \geq 1$. There exists polynomials $\tilde{P}, \tilde{Q}, \tilde{S}$ such that*

- $\forall x \in [\frac{1}{t}, 1]\colon |\tilde{P}(x) - \frac{1}{2tx}| \leq \eta$, *and* $\forall x \in [-1,1]\colon -1 \leq \tilde{P}(x) = \tilde{P}(-x) \leq 1$,
- $\forall x \in [-\frac{1-\beta}{t}, \frac{1-\beta}{t}]\colon |\tilde{Q}(x) - tx| \leq \eta tx$, *and* $\forall x \in [-1,1]\colon \tilde{Q}(x) = -\tilde{Q}(-x) \leq 1$,
- $\forall x \in [\beta, 1]\colon |\tilde{S}(x) - \frac{\ln(1/x)}{2\ln(2/\beta)}| \leq \eta$, *and* $\forall x \in [-1,1]\colon -1 \leq \tilde{S}(x) = \tilde{S}(-x) \leq 1$,

moreover $\deg(\tilde{P}) = \mathcal{O}\left(t \log\left(\frac{1}{\eta}\right)\right)$, $\deg(\tilde{Q}) = \mathcal{O}\left(\frac{t}{\beta} \log\left(\frac{1}{\eta}\right)\right)$, *and* $\deg(\tilde{S}) = \mathcal{O}\left(\frac{1}{\beta} \log\left(\frac{1}{\eta}\right)\right)$.

To prove this lemma, we use the following result based on local Taylor series:

Lemma 7.2.2 ([123, Corollary 66])**.** *Let $x_0 \in [-1,1]$, $r \in (0,2]$, $\nu \in (0, r]$ and let $f\colon [-x_0 - r - \nu, x_0 + r + \nu] \to \mathbb{C}$ and be such that $f(x_0 + x) = \sum_{\ell=0}^{\infty} a_\ell x^\ell$ for all $x \in [-r-\nu, r+\nu]$. Suppose $B > 0$ is such that $\sum_{\ell=0}^{\infty} (r+\nu)^\ell |a_\ell| \leq B$. Let $\epsilon \in \left(0, \frac{1}{2B}\right]$, then there is an efficiently computable polynomial $P \in \mathbb{C}[x]$ of degree $\mathcal{O}\left(\frac{1}{\nu} \log\left(\frac{B}{\epsilon}\right)\right)$ such that*[8]

$$\|f(x) - P(x)\|_{[x_0 - r, x_0 + r]} \leq \epsilon$$

$$\|P(x)\|_{[-1,1]} \leq \epsilon + \|f(x)\|_{[x_0 - r - \nu/2, x_0 + r + \nu/2]} \leq \epsilon + B$$

$$\|P(x)\|_{[-1,1] \setminus [x_0 - r - \nu/2, x_0 + r + \nu/2]} \leq \epsilon.$$

[8] For a function $g\colon \mathbb{R} \to \mathbb{C}$, and an interval $[a,b] \subseteq \mathbb{R}$, we define $\|g\|_{[a,b]} := \max_{x \in [a,b]} |g(x)|$.

Proof. For the construction of the $\tilde{P}$ and $\tilde{Q}$ polynomials see Corollary 67 and Theorem 30 of [123], respectively. It remains to construct the polynomial $\tilde{S}$ above.

Denote $f(x) = \frac{\ln(1/x)}{2\ln(2/\beta)}$; by taking $\epsilon = \eta/2$, $x_0 = 1$, $r = 1-\beta$, $\nu = \frac{\beta}{2}$, and $B = \frac{1}{2}$ in Corollary 7.2.2, we have a polynomial $S \in \mathbb{C}[x]$ of degree $\mathcal{O}\left(\frac{1}{\nu}\log\left(\frac{B}{\epsilon}\right)\right) = \mathcal{O}\left(\frac{1}{\beta}\log\left(\frac{1}{\eta}\right)\right)$ such that

$$\|f(x) - S(x)\|_{[\beta,2-\beta]} \leq \eta/2 \tag{7.2.4}$$

$$\|S(x)\|_{[-1,1]} \leq B + \eta/2 \leq (1+\eta)/2 \tag{7.2.5}$$

$$\|S(x)\|_{[-1,\frac{\beta}{2}]} \leq \eta/2. \tag{7.2.6}$$

Note that $B = \frac{1}{2}$ is valid because the Taylor series of $f(x)$ at $x = 1$ is $\frac{1}{2\ln(2/\beta)}\sum_{l=1}^{\infty}\frac{(-1)^l x^l}{l}$, and as a result we could take

$$\begin{aligned} B = \frac{1}{2\ln(2/\beta)}\sum_{l=1}^{\infty}\frac{(1-\beta/2)^l}{l} &= -\frac{1}{2\ln(2/\beta)}\sum_{l=1}^{\infty}\frac{(-1)^{l-1}}{l}(-1+\beta/2)^l \\ &= -\frac{1}{2\ln(2/\beta)}\ln\frac{\beta}{2} = \frac{1}{2}. \end{aligned} \tag{7.2.7}$$

However, S is not an even polynomial in general; we instead take $\tilde{S}(x) = S(x) + S(-x)$ for all $x \in [-1,1]$. Then by (7.2.4) and (7.2.6) we have

$$\left\|f(x) - \tilde{S}(x)\right\|_{[\beta,1]} \leq \left\|f(x) - \tilde{S}(x)\right\|_{[\beta,1]} + \left\|\tilde{S}(-x)\right\|_{[\beta,1]} \leq \frac{\eta}{2} + \frac{\eta}{2} = \eta. \tag{7.2.8}$$

Furthermore, $\tilde{S}$ is an even polynomial such that $\deg(\tilde{S}) = \mathcal{O}\left(\frac{1}{\beta}\log\left(\frac{1}{\eta}\right)\right)$; hence

(7.2.5) and (7.2.6) imply

$$\left\|\tilde{S}(x)\right\|_{[-1,1]} = \left\|\tilde{S}(x)\right\|_{[0,1]} \leq \|S(x)\|_{[0,1]} + \|S(x)\|_{[-1,0]} \leq \frac{1+\eta}{2} + \frac{\eta}{2} \leq 1 \qquad (7.2.9)$$

given $\eta \leq 1/2$. (Finally we can take the real part of $\tilde{S}(x)$ if it has some complex coefficients.) □

7.2.4 Projected unitary encodings for singular value transformation

First we handle the case of classical distributions. Let U_p be a purified quantum oracle of a classical distribution p as in Definition 7.1.2, and let $U := (U_p \otimes I)$, also let $\Pi := (\sum_{i=1}^{n} I \otimes |i\rangle\langle i| \otimes |i\rangle\langle i|)$, $\widetilde{\Pi} := (|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I)$, then

$$\begin{aligned}
\Pi U \widetilde{\Pi} = \Pi \left(U_p \otimes I\right) \widetilde{\Pi} &= \Big(\sum_{i=1}^{n} I \otimes |i\rangle\langle i| \otimes |i\rangle\langle i| \Big) (U_p \otimes I) \big(|0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I \big) \\
&= \sum_{i=1}^{n} \Big((I \otimes |i\rangle\langle i|) U_p (|0\rangle\langle 0| \otimes |0\rangle\langle 0|) \Big) \otimes |i\rangle\langle i| \\
\ldots &= \sum_{i=1}^{n} \Big((I \otimes |i\rangle\langle i|) \sum_{j=1}^{n} \sqrt{p_j} \, |\phi_j\rangle \, |j\rangle \, \langle 0| \, \langle 0| \Big) \otimes |i\rangle\langle i| \\
&= \sum_{i=1}^{n} \sqrt{p_i} \, |\phi_i\rangle\langle 0| \otimes |i\rangle\langle 0| \otimes |i\rangle\langle i| \, . \qquad (7.2.10)
\end{aligned}$$

Now we turn to quantum distributions where we do not know the diagonalizing basis of the density operator ρ. Let U_ρ be a purified quantum oracle of a quantum distribution ρ as in Definition 7.1.2, and W a unitary, mapping $|0\rangle \, |0\rangle \mapsto \sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}}$.

Let $U' := \left(I \otimes U_\rho^\dagger\right)\left(W^\dagger \otimes I\right)$, $\Pi' := (I \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0|)$ and $\widetilde{\Pi}$ as above, then

$$\begin{aligned}
\Pi' U' \widetilde{\Pi} =& \Pi' \left(I \otimes U_\rho^\dagger\right)\left(W^\dagger \otimes I\right)\widetilde{\Pi} \\
=& \left(I \otimes \left(|0\rangle\langle 0| \otimes |0\rangle\langle 0|\, U_\rho^\dagger\right)\right)\left(\left(\sum_{j=1}^{n} \frac{|j\rangle\,|j\rangle}{\sqrt{n}}\right)\langle 0|\,\langle 0| \otimes I\right) \\
=& \left(I \otimes \sum_{i=1}^{n} \sqrt{p_i}\,|0\rangle\,|0\rangle\langle \phi_i|\,\langle \psi_i|\right)\left(\left(\sum_{j=1}^{n} \frac{\left|\phi_j'\right\rangle\,|\phi_j\rangle}{\sqrt{n}}\right)\langle 0|\,\langle 0| \otimes I\right) \\
=& \sum_{i=1}^{n} \sqrt{\frac{p_i}{n}}\,\left|\phi_i'\right\rangle\,|0\rangle\,|0\rangle\,\langle 0|\,\langle 0|\,\langle \psi_i|\,, \qquad (7.2.11)
\end{aligned}$$

where $\sum_{j=1}^{n} \frac{\left|\phi_j'\right\rangle|\phi_j\rangle}{\sqrt{n}} = \sum_{j=1}^{n} \frac{|j\rangle|j\rangle}{\sqrt{n}}$ is the Schmidt decomposition of the maximally entangled state under the basis $(|\phi_1\rangle, \ldots, |\phi_n\rangle)$.

7.3 Results

7.3.1 Shannon entropy estimation

Classical distributions with purified quantum query-access. Recall that we introduced purified quantum query-access in Definition 7.1.2. In particular, for a classical distribution p on $[n]$, we are given a unitary U_p acting on $\mathbb{C}^{n\times n}$ such that

$$U_p|0\rangle_A|0\rangle_B = |\psi_p\rangle = \sum_{i=1}^{n} \sqrt{p_i}|\phi_i\rangle_A|i\rangle_B. \qquad (7.3.1)$$

We use U_p and $U_p^\dagger$ to estimate the Shannon entropy $H(p)$:

Theorem 7.3.1. *For any $0 < \epsilon < 1$, we can estimate $H(p)$ with accuracy ϵ with success probability at least $2/3$ using $\mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{1.5}} \log^{1.5}\left(\frac{n}{\epsilon}\right) \log\left(\frac{\log n}{\epsilon}\right)\right)$ calls to U_p and $U_p^\dagger$.*

Proof. The general idea is to first construct a unitary matrix that has a specific matrix block with singular values $\sqrt{p_1}, \ldots, \sqrt{p_n}$. We use the construction of Eq. (7.1.4) and apply singular value transformation (Theorem 7.2.2) by a polynomial $\tilde{S}$ constructed in Lemma 7.2.1, setting $\eta = \frac{\epsilon}{24 \ln(2/\beta)}$ and $\beta = \sqrt{\Delta}$ for $\Delta = \frac{\epsilon}{12n \ln(n/\epsilon)}$. Notice that this Δ satisfies

$$\Delta\left(\ln\left(\frac{1}{\Delta}\right) + 4\ln\left(\frac{2}{\beta}\right)\right) = \frac{\epsilon}{12n\ln(n/\epsilon)} \cdot \ln \frac{16(4n\ln\frac{n}{\epsilon})^3}{\epsilon^3} \tag{7.3.2}$$

$$\leq \frac{\epsilon}{12n\ln(n/\epsilon)} \cdot \ln \frac{n^6}{\epsilon^6} = \frac{\epsilon}{2n}, \tag{7.3.3}$$

provided that $\frac{n}{\epsilon} \geq 152$. Note that the polynomial $\tilde{S}$ satisfies both conditions in Theorem 7.2.2. Applying the singular value transformed version of the operator (7.1.4) to the state $|\psi_p\rangle$ gives

$$|\widetilde{\Psi_p}\rangle = \sum_{i=1}^{n} \sqrt{p_i}\tilde{S}(\sqrt{p_i})|\phi_i\rangle_A|i\rangle_B|0\rangle + \ldots|1\rangle. \tag{7.3.4}$$

Preparing $|\widetilde{\Psi_p}\rangle$ costs $\deg \tilde{S} = \mathcal{O}\left(\frac{1}{\beta}\log\left(\frac{1}{\eta}\right)\right) = \mathcal{O}\left(\sqrt{\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)}\log\left(\frac{\log n}{\epsilon}\right)\right)$ uses of U_p and $U_p^\dagger$ and the same number of controlled reflections through $\Pi, \widetilde{\Pi}$. Furthermore, Lemma 7.2.1 implies that for all i such that $p_i \geq \Delta$,

$$\left|\frac{p_i\ln(1/p_i)}{4\ln(2/\beta)} - p_i\tilde{S}(\sqrt{p_i})\right| = p_i \cdot \left|\frac{\ln\left(1/\sqrt{p_i}\right)}{2\ln(2/\beta)} - \tilde{S}(\sqrt{p_i})\right| \leq \eta p_i. \tag{7.3.5}$$

For all i such that $p_i < \Delta$, we have

$$\left|\frac{p_i \ln(1/p_i)}{4\ln(2/\beta)} - p_i \tilde{S}(\sqrt{p_i})\right| \leq \frac{p_i \ln(1/p_i)}{4\ln(2/\beta)} + p_i \tag{7.3.6}$$

$$\leq \frac{\Delta(\ln(\frac{1}{\Delta}) + 4\ln(2/\beta))}{4\ln(2/\beta)} \leq \frac{\epsilon}{8n\ln(2/\beta)}, \tag{7.3.7}$$

where the first inequality comes from the fact that $|\tilde{S}(x)| \leq 1$ for all $x \in [-1, 1]$, the second inequality comes from the monotonicity of $x(\ln(1/x) + 4\ln(2/\beta))$ on $(0, \frac{1}{\Delta}]$, and the third inequality comes from (7.3.2). As a result of (7.3.1), (7.3.5), and (7.3.6), we have

$$\left|(\langle\psi_p| \otimes \langle 0|)|\widetilde{\Psi_p}\rangle - \frac{H(p)}{4\ln(2/\beta)}\right| = \left|p_i \tilde{S}(\sqrt{p_i}) - \sum_{i=1}^{n} \frac{p_i \log(1/p_i)}{4\ln(2/\beta)}\right| \tag{7.3.8}$$

$$\leq \sum_{i:\, p_i < \Delta} \frac{\epsilon}{8n\ln(2/\beta)} + \sum_{i:\, p_i \geq \Delta} \eta p_i \tag{7.3.9}$$

$$\leq \frac{\epsilon}{8\ln(2/\beta)} + \frac{\epsilon}{24\ln(2/\beta)} = \frac{\epsilon}{6\ln(2/\beta)}. \tag{7.3.10}$$

Therefore, $|4\ln(2/\beta)(\langle\psi_p| \otimes \langle 0|)|\widetilde{\Psi_p}\rangle - H(p)| \leq 2\epsilon/3$. By Theorem 7.2.1, we can use $\Theta(\ln(1/\beta)/\epsilon)$ applications of the unitaries (and their inverses) that implement $|\psi_p\rangle$ and $|\widetilde{\Psi_p}\rangle$ to estimate $(\langle\psi_p| \otimes \langle 0|)|\widetilde{\Psi_p}\rangle$ within additive error $\frac{\epsilon}{12\ln(2/\beta)}$. In total, this estimates $H(p)$ within additive error $\frac{\epsilon}{12\ln(2/\beta)} \cdot 4\ln(2/\beta) + \frac{2\epsilon}{3} = \epsilon$ with success probability at least $8/\pi^2$. The total complexity of the algorithm is

$$\mathcal{O}\left(\frac{\ln(1/\beta)}{\epsilon}\right) \cdot \mathcal{O}\left(\sqrt{\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)}\log\left(\frac{\log n}{\epsilon}\right)\right) = \mathcal{O}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\log^{1.5}\left(\frac{n}{\epsilon}\right)\log\left(\frac{\log n}{\epsilon}\right)\right).$$

□

Density matrices with purified quantum query-access. For a density matrix ρ, we also assume the purified quantum query-access in Definition 7.1.2, i.e., a unitary oracle U_ρ acting as $U_\rho |0\rangle_A |0\rangle_B = |\rho\rangle = \sum_{i=1}^n \sqrt{p_i} |\phi_i\rangle_A |\psi_i\rangle_B$. We use U_ρ and $U_\rho^\dagger$ to estimate the von-Neumann entropy $H(\rho) = -\operatorname{Tr}[\rho \log \rho]$:

Theorem 7.3.2. *For any $0 < \epsilon < 1$, we can estimate $H(p)$ with accuracy ϵ with success probability at least $2/3$ using $\widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)$ calls to U_ρ and $U_\rho^\dagger$.*

Proof. We use the construction of Eq. (7.1.5). The proof is essentially the same as that of Theorem 7.3.1 proceeding by constructing singular value transformation via Theorem 7.2.2, with the only difference that all probabilities are rescaled by a factor of $1/\sqrt{n}$ in (7.1.5); as a result, the number of calls to U_ρ and $U_\rho^\dagger$ is blown up to $\widetilde{\mathcal{O}}\left(\sqrt{n} \cdot \frac{\sqrt{n}}{\epsilon^{1.5}}\right) = \widetilde{\mathcal{O}}\left(\frac{n}{\epsilon^{1.5}}\right)$. □

7.3.2 Tolerant testers for ℓ^2-closeness with purified query-access

First we give an ℓ^2-closeness tester for unknown classical distributions p, q.

Theorem 7.3.3. *Given purified quantum query-access for classical distributions p, q as in Definition 7.1.2, for any $\nu, \epsilon \in (0, 1)$ the quantum query complexity of distinguishing the cases $\|p - q\|_2 \geq \epsilon$ and $\|p - q\|_2 \leq (1-\nu)\epsilon$ with success probability at least $2/3$ is*

$$\mathcal{O}\left(\frac{1}{\nu\epsilon} \log^3\left(\frac{1}{\nu\epsilon}\right) \log\log\left(\frac{1}{\nu\epsilon}\right)\right).$$

Proof. The main idea is to first bin the x elements based on the approximate value of $p(x) + q(x)$, then apply fine-tuned algorithms exploiting the knowledge of the approximate value of $p(x) + q(x)$.

Using amplitude estimation for any $k \in \mathbb{N}$ we can construct an algorithm $\mathcal{A}_k$ that for any input x with $p(x) + q(x) \geq 2^{-k}$ outputs "greater" with probability at least $2/3$, and for any x with $p(x) + q(x) \leq 2^{-k-1}$ outputs "smaller" and uses $\mathcal{O}\left(2^{\frac{k}{2}}\right)$ queries to U_p and U_q. Using $\mathcal{O}\left(\log\left(\frac{1}{\nu\epsilon}\right)\right)$ repetitions we can boost the success probability to $1 - \mathcal{O}\left(\text{poly}\left(\nu\epsilon\right)\right)$. Since our algorithm only needs to succeed with constant probability, and will use these subroutines at most $\frac{1}{\text{poly}(\nu\epsilon)}$ times, we can ignore the small failure probability. Therefore in the rest of the proof we assume without loss of generality, that $\mathcal{A}_k$ solves perfectly the above question with (query) complexity $\mathcal{O}\left(2^{\frac{k}{2}} \log\left(\frac{1}{\nu\epsilon}\right)\right)$.

Algorithm 7.1: Estimating $\log_2(p(x) + q(x))$.

Input: $x \in [n]$, $\theta \in (0, 1)$.

for $k \in K := \left\{-1, 0, 1, 2, \ldots, \left\lceil \log_2\left(\frac{1}{\theta}\right)\right\rceil\right\}$ **do**

  Run algorithm $\mathcal{A}_k$ on $|x\rangle$. **If** output is "greater" **then return** k;

return "less than θ";

For any x with $p(x) + q(x) \geq \theta$, Algorithm 7.1 outputs a k such that $p(x) + q(x) \in (2^{-k-1}, 2^{-k+1})$. However, note that this labeling is probabilistic; let us denote by $s_k(x)$ the probability that x is labeled by k. Observe that $s_k(x) = 0$ unless $k \in \left\{\left\lfloor \log_2\left(\frac{1}{p(x)+q(x)}\right)\right\rfloor, \left\lceil \log_2\left(\frac{1}{p(x)+q(x)}\right)\right\rceil\right\}$ (otherwise the return is either "greater" or "less than"). Now let us express $\|p - q\|_2^2$ in terms of this "soft-selection" function

$s(x)$.

$$\begin{aligned}
\|p-q\|_2^2 &= \sum_x |p(x)-q(x)|^2 \\
&= \sum_x \sum_{k\in K} s_k(x)\,|p(x)-q(x)|^2 + \eta \qquad (\eta \in [0,2\theta)) \\
&= \sum_{k\in K} 2^{9-k} \sum_x s_k(x) \frac{p(x)+q(x)}{2} \frac{2^{-k-2}}{p(x)+q(x)} \left(\frac{p(x)-q(x)}{2^{-k+3}}\right)^2 + \eta, \quad (7.3.11)
\end{aligned}$$

where the bound on η follows from the observation that

$$\eta \le \sum_{x:\, p(x)+q(x)<\theta} |p(x)-q(x)|^2 \le \sum_{x:\, p(x)+q(x)<\theta} (p(x)+q(x))^2 \qquad (7.3.12)$$

$$< \theta \sum_{x:\, p(x)+q(x)<\theta} p(x)+q(x) < 2\theta. \qquad (7.3.13)$$

If for all $k \in K$ we have a $2^{k-9}\frac{\theta}{|K|}$-precise estimate of

$$\sum_x s_k(x) \frac{p(x)+q(x)}{2} \frac{2^{-k-2}}{p(x)+q(x)} \left(\frac{p(x)-q(x)}{2^{-k+3}}\right)^2, \qquad (7.3.14)$$

then we get a 3θ-precise estimate of $\|p-q\|_2^2$. In particular setting $\theta := \nu\epsilon^2/6$, this solves the tolerant testing problem, since if $\|p-q\| \ge \epsilon$ then $\|p-q\|^2 \ge \epsilon^2$, on the other hand if $\|p-q\| \le (1-\nu)\epsilon$ then $\|p-q\|^2 \le (1-\nu)^2\epsilon^2 \le (1-\nu)\epsilon^2 = \epsilon^2 - \nu\epsilon^2$.

Now we describe how to construct a quantum algorithm that sets the first output qubit to $|0\rangle$ with probability (7.3.14). Start with preparing a purification of the distribution of $\frac{p(x)+q(x)}{2}$, then set the label of x to k with probability $s_k(x)$ using Algorithm 7.1 terminating it after using $\mathcal{A}_k$. Then separately apply the maps

$\sqrt{\frac{2^{-k-2}}{p(x)+q(x)}}$ and $\frac{p(x)-q(x)}{2^{-k-3}}$ to the state.

Note that we do not need to apply the above transformations exactly, it is enough if apply them with precision say $2^{k-11}\frac{\theta}{|K|}$. We analyze the complexity of (approximately) implementing the above sketched algorithm. To implement the map $\sqrt{\frac{2^{-k-2}}{p(x)+q(x)}}$, we use the unitary of Eq. (7.1.4), and transform the singular values by the polynomial $\tilde{P}$ from Lemma 7.2.1 using Theorem 7.2.2. In order to implement the map $\frac{p(x)-q(x)}{2^{-k-2}}$, we again use the unitary of Eq. (7.1.4), but now separately for p and q. We amplify both the singular values $\sqrt{p(x)}$ and $\sqrt{q(x)}$ by a factor $\sqrt{2^{k-2}}$ using the polynomial $\tilde{Q}$ from Lemma 7.2.1 in Theorem 7.2.2. Then we create a block-encoding[9] of both and $2^{k-2}p(x)$ and $2^{k-2}q(x)$ and then combine them to get a block-encoding of $\frac{p(x)-q(x)}{2^{-k-3}}$. In both cases the query complexity of $\mathcal{O}\left(\theta/|K|\right)$-precisely implementing the transformations is $\mathcal{O}\left(2^{k/2}\log\left(|K|/\theta\right)\right) = \mathcal{O}\left(2^{k/2}\log\left(1/\theta\right)\right)$. Since computing the label k also costs $\mathcal{O}\left(2^{k/2}\log\left(1/(\nu\epsilon)\right)\right)$, this is the overall complexity so far. Finally we estimate the probability of the first qubit being set to $|0\rangle$ with setting $M = \mathcal{O}\left(|K|2^{-k/2}/(\nu\epsilon)\right)$ in Theorem 7.2.1, and boost the success probability to $1-\mathcal{O}\left(1/|K|\right)$ with $\mathcal{O}\left(\log(|K|)\right)$ repetitions. Thus for any $k \in K$ the overall complexity of estimating Eq. (7.3.14) with sufficient precision has (query) complexity $\mathcal{O}\left(\frac{|K|}{\nu\epsilon}\log\left(\frac{1}{\nu\epsilon}\right)\log(|K|)\right) = \mathcal{O}\left(\frac{1}{\nu\epsilon}\log^2\left(\frac{1}{\nu\epsilon}\right)\log\log\left(\frac{1}{\nu\epsilon}\right)\right)$. Therefore estimating $\|p-q\|_2^2$ to precision $\nu\epsilon^2/6$ with high probability has (query) complexity

$$\mathcal{O}\left(\frac{1}{\nu\epsilon}\log^3\left(\frac{1}{\nu\epsilon}\right)\log\log\left(\frac{1}{\nu\epsilon}\right)\right). \qquad \square$$

[9] If we have a projected unitary encoding of $\Pi U\widetilde{\Pi} = A = \sum_i \varsigma_i |\psi_i\rangle\langle 0,i|$ with $\widetilde{\Pi} = |0\rangle\langle 0|\otimes I$, we can immediately turn it into a block-encoding of $A^\dagger A = \sum_i \varsigma_i^2 |i\rangle\langle i|$ by e.g. applying Theorem 7.2.2 with the polynomial x^2.

It is easy to see an $\Omega\left(\frac{1}{\epsilon}\right)$ lower bound on the above problem even in the strongest quantum pure state input model Definition 7.1.4. Indeed, consider the case $n = 2, q = (\frac{1}{2}, \frac{1}{2})$ (the uniform distribution on $\{1, 2\}$) and we want to test whether $p = q$ or $\|p - q\|_2 \geq \epsilon$. This is equivalent to test whether $p_1 = \frac{1}{2}$ or $|p_1 - \frac{1}{2}| \geq \frac{\epsilon}{\sqrt{2}}$; due to the optimality of amplitude estimation in Theorem 7.2.1, this task requires $\Omega(\frac{1}{\epsilon})$ quantum queries to the unitary U preparing the state $\sqrt{p_1}\,|1\rangle + \sqrt{p_2}\,|2\rangle$.

Now we prove the result below on (tolerant) ℓ^2-closeness testing for quantum distributions:

Theorem 7.3.4. *Given $\epsilon, \nu \in (0, 1)$ and two density operators $\rho, \sigma \in \mathbb{C}^{n\times n}$ with purified quantum query-access to U_ρ and U_σ as in Definition 7.1.2, it takes $\mathcal{O}\left(\min\left(\frac{\sqrt{n}}{\epsilon}, \frac{1}{\epsilon^2}\right)\frac{1}{\nu}\right)$ queries to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$ to decide whether $\|\rho-\sigma\|_2 \geq \epsilon$ or $\|\rho-\sigma\|_2 \leq (1-\nu)\epsilon$, with success probability at least $2/3$.*

Proof. We can combine the block-encodings of ρ and σ to apply the map $\frac{\rho-\sigma}{2}$ to the maximally entangled state $\sum_{j=1}^n \frac{|j\rangle|j\rangle}{\sqrt{n}}$, which gives

$$\sum_{j=1}^n \frac{|j\rangle\,|j\rangle}{\sqrt{n}} \to \left(\frac{\rho-\sigma}{2}\otimes I\right)\sum_{j=1}^n \frac{|j\rangle\,|j\rangle}{\sqrt{n}}\,|0\rangle + \ldots\,|1\rangle\,. \tag{7.3.15}$$

The probability of measuring the $|0\rangle$ ancilla state is

$$\sum_{i,j=1}^n \frac{\langle i|\,\langle i|}{\sqrt{n}}\left(\frac{(\rho-\sigma)^2}{4}\otimes I\right)\frac{|j\rangle\,|j\rangle}{\sqrt{n}} = \frac{1}{4n}\sum_{i=1}^n \langle i|\,(\rho-\sigma)^2\,|i\rangle = \frac{1}{4n}\operatorname{Tr}[(\rho-\sigma)^2]. \tag{7.3.16}$$

Thus it suffices to apply amplitude estimation with $M = \Theta\left(\frac{\sqrt{n}}{\nu\epsilon}\right)$ calls to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$.

On the other hand, we can estimate $\|\rho-\sigma\|_2^2$ by observing that $\|\rho-\sigma\|_2^2 =$

$\mathrm{tr}\{(\rho-\sigma)^2\} = \mathrm{tr}\{\rho^2\} - 2\,\mathrm{tr}\{\rho\sigma\} + \mathrm{tr}\{\sigma^2\}$. Since the success probability of the SWAP test ([62]) on input states ρ, σ is $\frac{1}{2}(1 + \mathrm{tr}\{\rho\sigma\})$, we can individually estimate the latter quantities with precision $\mathcal{O}(\nu\epsilon^2)$ using amplitude estimation (Theorem 7.2.1) with $\mathcal{O}\left(\frac{1}{\nu\epsilon^2}\right)$ queries to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$. As a result, we could decide whether $\|\rho-\sigma\|_2 \geq \epsilon$ or $\|\rho-\sigma\|_2 \leq (1-\nu)\epsilon$ using $\mathcal{O}\left(\frac{1}{\nu\epsilon^2}\right)$ queries.

The result of Theorem 7.3.4 hence follows by taking the minimum of the two complexities. □

7.3.3 ℓ^1-closeness testing with purified query-access

Corollary 7.3.1. *Given $\epsilon > 0$ and two distributions p, q on the domain $[n]$ with purified quantum query-access via U_p and U_q as in Definition 7.1.2, it takes $\widetilde{\mathcal{O}}\left(\frac{\sqrt{n}}{\epsilon}\right)$ queries to $U_p, U_p^\dagger, U_q, U_q^\dagger$ to decide whether $p=q$ or $\|p-q\|_1 \geq \epsilon$ with success probability at least $2/3$. Similarly for density operators $\rho, \sigma \in \mathbb{C}^{n\times n}$ with purified quantum query-access via U_ρ and U_σ, it takes $\mathcal{O}\left(\frac{n}{\epsilon}\right)$ queries to $U_\rho, U_\rho^\dagger, U_\sigma, U_\sigma^\dagger$ to decide whether $\rho=\sigma$ or $\|\rho-\sigma\|_1 \geq \epsilon$ with success probability at least $2/3$.*

Proof. By the Cauchy-Schwartz inequality we have $\|p-q\|_2 \geq \frac{1}{\sqrt{n}}\|p-q\|_1$, therefore Theorem 7.3.3 implies our claim by taking $\epsilon \leftarrow \epsilon/\sqrt{n}$ therein. Similarly, Theorem 7.3.4 implies our claim for quantum distributions ρ and σ. □

7.3.4 Independence testing with purified query-access

Corollary 7.3.2. *Given $\epsilon > 0$ and a classical distribution p on $[n] \times [m]$ with the purified quantum query-access via U_p as in Definition 7.1.2, it takes $\widetilde{\mathcal{O}}\left(\frac{\sqrt{nm}}{\epsilon}\right)$*

queries to $U_p, U_p^\dagger$ to decide whether p is a product distribution on $[n] \times [m]$ or p is ϵ-far in ℓ^1-norm from any product distribution on $[n] \times [m]$ with success probability at least $2/3$.

Proof. We define p_A to be the margin of p on the first marginal space, i.e., $p_A(i) = \sum_{j=1}^m p(i,j)$ for all $i \in [n]$. We similarly define p_B to be the margin of p on the second marginal space, i.e., $p_B(j) = \sum_{i=1}^n p(i,j)$ for all $j \in [m]$. Assume the quantum oracle U_p from Definition 7.1.2 acts as

$$U_p|0\rangle_A|0\rangle_B|0\rangle_C = \sum_{i=1}^n \sum_{j=1}^m \sqrt{p(i,j)}|i\rangle_A|j\rangle_B|\psi_{i,j}\rangle_C; \tag{7.3.17}$$

if we denote $|\phi_i\rangle = \sum_{j=1}^m \frac{\sqrt{p(i,j)}}{\sqrt{p_A(i)}}|j\rangle|\psi_{i,j}\rangle$ for all $i \in [n]$ and $|\varphi_j\rangle = \sum_{i=1}^n \frac{\sqrt{p(i,j)}}{\sqrt{p_B(j)}}|i\rangle|\psi_{i,j}\rangle$ for all $j \in [m]$, then we have

$$U_p|0\rangle_A|0\rangle_B|0\rangle_C = \sum_{i=1}^n \sqrt{p_A(i)}|i\rangle_A|\phi_i\rangle_{B,C} = \sum_{j=1}^m \sqrt{p_B(j)}|j\rangle_B|\varphi_j\rangle_{A,C}. \tag{7.3.18}$$

As a result,

$$(U_p \otimes U_p)(|0\rangle^{\otimes 6}) = \sum_{i=1}^n \sum_{j=1}^m \sqrt{p_A(i)}\sqrt{p_B(j)}|i\rangle|j\rangle|\phi_i\rangle|\varphi_j\rangle; \tag{7.3.19}$$

in other words, one purified quantum query to the distribution $p_A \times p_B$ can be implemented by two queries to U_p.

If p is a product distribution on $[n] \times [m]$, then $p = p_A \times p_B$; if p is ϵ-far in ℓ^1-norm from any product distribution on $[n] \times [m]$, then $\|p - p_A \times p_B\|_1 \geq \epsilon$.

Therefore, the problem of independence testing reduces to ℓ^1-closeness testing for distributions on $[n] \times [m]$, and hence Corollary 7.3.2 follows from Corollary 7.3.1. □

Similarly, Corollary 7.3.1 implies that the quantum query complexity of testing independence of quantum distributions is $\mathcal{O}\left(\frac{nm}{\epsilon}\right)$.

7.4 Rényi entropy estimation

In this section, we focus on the specific question of Rényi entropy estimation. The methodology will be significantly different from that of the previous sections; we give a sketch of the techniques, and full details and proofs can be found in [192]. For our convenience, we focus on classical distributions with discrete query-access (Definition 7.1.3), since the results for purified quantum query-access (Definition 7.1.2) naturally follows with an overhead of $\sqrt{n}$ as in Section 7.1.3.

7.4.1 Overview

One important generalization of Shannon entropy is the *Rényi entropy* of order $\alpha > 0$, denoted $H_\alpha(p)$, which is defined by

$$H_\alpha(p) := \begin{cases} \frac{1}{1-\alpha} \log \sum_{x \in X} p_x^\alpha, & \text{when } \alpha \neq 1. \\ \lim_{\alpha \to 1} H_\alpha(p), & \text{when } \alpha = 1. \end{cases} \tag{7.4.1}$$

The Rényi entropy of order 1 is simply the Shannon entropy, i.e., $H_1(p) = H(p)$. General Rényi entropy can be used as a bound on Shannon entropy, making it

useful in many applications (e.g., [27, 89]). Rényi entropy is also of interest in its own right. One prominent example is the Rényi entropy of order 2, $H_2(p)$ (also known as the *collision entropy*), which measures the quality of random number generators (e.g., [266]) and key derivation in cryptographic applications (e.g., [47, 148]). Motivated by these and other applications, the estimation of Rényi entropy has also been actively studied [11, 153, 154]. In particular, Acharya et al. [11] have shown almost tight bounds on the classical query complexity of computing Rényi entropy. Specifically, for any non-integer $\alpha > 1$, the classical query complexity of α-Rényi entropy is $\Omega(n^{1-o(1)})$ and $O(n)$. Surprisingly, for any *integer* $\alpha > 1$, the classical query complexity is $\Theta(n^{1-1/\alpha})$, i.e., *sublinear* in n. When $0 \leq \alpha < 1$, the classical query complexity is $\Omega(n^{1/\alpha-o(1)})$ and $O(n^{1/\alpha})$, which is always superlinear.

The extreme case ($\alpha \to \infty$) is known as the *min-entropy*, denoted $H_\infty(p)$, which is defined by

$$H_\infty(p) := \lim_{\alpha\to\infty} H_\alpha(p) = -\log \max_{i\in[n]} p_i. \tag{7.4.2}$$

Min-entropy plays an important role in the randomness extraction (e.g., [260]) and characterizes the maximum number of uniform bits that can be extracted from a given distribution. Classically, the query complexity of min-entropy estimation is $\Theta(n/\log n)$, which follows directly from [262].

Another extreme case ($\alpha = 0$), also known as the *Hartley entropy* [140], is the logarithm of the support size of distributions, where the *support* of any distribution

p is defined by

$$\mathrm{Supp}(p) := |\{x : x \in X,\, p_x > 0\}|. \tag{7.4.3}$$

It is a natural and fundamental quantity of distributions with various applications (e.g., [105, 109, 133, 147, 177, 228, 258]). However, estimating the support size is impossible in general because elements with negligible but nonzero probability, which are very unlikely to be sampled, could still contribute to $\mathrm{Supp}(p)$. Two related quantities (*support coverage* and *support size*) have hence been considered as alternatives of 0-Rényi entropy with roughly $\Theta(n/\log n)$ complexity.

Besides the entropic measures of a discrete distribution, we also briefly discuss an entropic measure between two distributions, namely the *Kullback-Leibler (KL) divergence*. Given two discrete distributions p and q with cardinality n, the KL divergence is defined as

$$D_{\mathrm{KL}}(p\|q) = \sum_{i\in[n]} p_i \log \frac{p_i}{q_i}. \tag{7.4.4}$$

KL divergence is a key measure with many applications in information theory [90, 178], data compression [64], and learning theory [174]. Classically, under the assumption that $\frac{p_i}{q_i} \le f(n)\ \forall\, i \in [n]$ for some $f(n)$, $D_{\mathrm{KL}}(p\|q)$ can be approximated within constant additive error with high success probability if $\Theta(\frac{n}{\log n})$ samples are taken from p and $\Theta(\frac{nf(n)}{\log n})$ samples are taken from q.

Main question. In this section, we study the impact of quantum computation on estimation of general Rényi entropies. Specifically, we aim to characterize quantum speed-ups for estimating *Rényi entropies* of classical distributions with discrete query-access (Definition 7.1.3), i.e., for a distribution $p = (p_i)_{i=1}^n$ on $[n]$ with a function $O_p \colon [S] \to [n]$ for some $S \in \mathbb{N}$ such that

$$p_i = |\{s \in [S] : O_p(s) = i\}|/S, \tag{7.4.5}$$

we assume a unitary operator $\hat{O}_p$ acting on $\mathbb{C}^S \otimes \mathbb{C}^{n+1}$ such that

$$\hat{O}_p|s\rangle|0\rangle = |s\rangle|O_p(s)\rangle \quad \forall\, s \in [S]. \tag{7.4.6}$$

This oracle model can also be readily obtained in some algorithmic settings, e.g., when distributions are generated by some classical or quantum sampling procedure. Thus, statistical property testing results in this oracle model can be potentially leveraged in algorithm design.

Our results. Our main contribution is a systematic study of both upper and lower bounds for the *quantum query complexity* of estimation of Rényi entropies (including Shannon entropy as a special case). Specifically, we obtain the following quantum speedups for different ranges of α.

Theorem 7.4.1. *There are quantum algorithms that approximate $H_\alpha(p)$ of distribution p on $[n]$ within an additive error $0 < \epsilon \leq O(1)$ with success probability at*

least 2/3 using[10]

- $\tilde{O}\left(\frac{\sqrt{n}}{\epsilon^{1.5}}\right)$ *quantum queries when* $\alpha = 0$*, i.e., Hartley entropy.*[11]
- $\tilde{O}\left(\frac{n^{1/\alpha-1/2}}{\epsilon^2}\right)$ *quantum queries*[12] *when* $0 < \alpha < 1$*.*
- $\tilde{O}\left(\frac{\sqrt{n}}{\epsilon^2}\right)$ *quantum queries when* $\alpha = 1$ *(Shannon entropy).*
- $\tilde{O}\left(\frac{n^{\nu(1-1/\alpha)}}{\epsilon^2}\right)$ *quantum queries when* $\alpha > 1, \alpha \in \mathbb{N}$ *for some* $\nu < \frac{3}{4}$*.*
- $\tilde{O}\left(\frac{n^{1-1/2\alpha}}{\epsilon^2}\right)$ *quantum queries when* $\alpha > 1, \alpha \notin \mathbb{N}$*.*
- $\tilde{O}\left(Q\left(\left\lceil\frac{16\log n}{\epsilon^2}\right\rceil\text{-}\mathit{distinctness}\right)\right)$ *quantum queries when* $\alpha = \infty$*, where we denote* $Q\left(\left\lceil\frac{16\log n}{\epsilon^2}\right\rceil\text{-}\mathit{distinctness}\right)$ *as the quantum query complexity of the* $\left\lceil\frac{16\log n}{\epsilon^2}\right\rceil$*-distinctness problem.*

Our quantum testers demonstrate advantages over classical ones for all $0 < \alpha < \infty$; in particular, our quantum tester has a quadratic speedup in the case of Shannon entropy. When $\alpha = \infty$, our quantum upper bound depends on the quantum query complexity of the $\lceil\log n\rceil$-distinctness problem, which is open to the best of our knowledge[13] and might demonstrate a quantum advantage.

As a corollary, we also obtain quadratic quantum speedup for estimating KL divergence:

[10] It should be understood that the success probability 2/3 can be boosted to close to 1 without much overhead.

[11] 0-Rényi entropy estimation is intractable without any assumption, both classically and quantumly. Here, the results are based on the assumption that nonzero probabilities are at least $1/n$.

[12] $\tilde{O}$ hides factors that are polynomial in $\log n$ and $\log 1/\epsilon$.

[13] Existing quantum algorithms for the k-distinctness problem (e.g., [17] has query complexity $O(k^2 n^{k/k+1})$ and [44] has query complexity $O(2^{k^2} n^{\nu})$ for some $\nu < 3/4$) do not behave well for super-constant ks.

Corollary 7.4.1. *Assuming p and q satisfies $\frac{p_i}{q_i} \leq f(n)\ \forall\, i \in [n]$ for some function $f : \mathbb{N} \to \mathbb{R}^+$, $D_{KL}(p\|q)$, there is a quantum algorithm that approximates $D_{KL}(p\|q)$ within an additive error $\epsilon > 0$ with success probability at least $\frac{2}{3}$ using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^2}\big)$ quantum queries to p and $\tilde{O}\big(\frac{\sqrt{n}f(n)}{\epsilon^2}\big)$ quantum queries to q.*

On the other hand, we obtain corresponding quantum lower bounds on entropy estimation using the polynomial method [6, 43], which are then combined with a couple of lower bounds shown in [63]. It is worth mentioning that lower bounds in [63] are established when assuming $\epsilon = O(1)$, whereas our lower bounds have precise error dependence.

We summarize both bounds in Table 7.3 and visualize them in Figure 7.1.

Theorem 7.4.2. *Any quantum algorithm that approximates $H_\alpha(p)$ of distribution p on $[n]$ within additive error ϵ with success probability at least 2/3 must use*

- $\Omega(\sqrt{n} + n^{\frac{1}{3}}/\epsilon^{\frac{1}{6}})$ *quantum queries when* $\alpha = 0$, *assuming* $1/n \leq \epsilon \leq 1/12$.
- $\tilde{\Omega}(n^{\frac{1}{7\alpha}-o(1)}/\epsilon^{\frac{2}{7}})$ *quantum queries when* $0 < \alpha < \frac{3}{7}$.
- $\Omega(n^{\frac{1}{3}}/\epsilon^{\frac{1}{6}})$ *quantum queries when* $\frac{3}{7} \leq \alpha \leq 3, \alpha \neq 1$, *assuming* $1/n \leq \epsilon \leq 1/2$.
- $\Omega(\sqrt{n} + n^{\frac{1}{3}}/\epsilon^{\frac{1}{6}})$ *quantum queries when* $\alpha = 1$, *assuming* $1/n \leq \epsilon \leq 1/2$.
- $\Omega(n^{\frac{1}{2}-\frac{1}{2\alpha}}/\epsilon)$ *quantum queries when* $3 \leq \alpha < \infty$.
- $\Omega(\sqrt{n}/\epsilon)$ *quantum queries when* $\alpha = \infty$.

α	classical bounds	quantum bounds (our result)
$\alpha = 0$	$\Theta(\frac{n}{\log n})$ [223, 276]	$\tilde{O}(\sqrt{n})$ (this paper), $\tilde{\Omega}(\sqrt{n})$ [63]
$0 < \alpha < 1$	$O(\frac{n^{\frac{1}{\alpha}}}{\log n})$, $\Omega(n^{\frac{1}{\alpha}-o(1)})$ [11]	$\tilde{O}(n^{\frac{1}{\alpha}-\frac{1}{2}})$, $\Omega(\max\{n^{\frac{1}{7\alpha}-o(1)}, n^{\frac{1}{3}}\})$
$\alpha = 1$	$\Theta(\frac{n}{\log n})$ [153, 262, 275]	$\tilde{O}(\sqrt{n})$ (this paper), $\tilde{\Omega}(\sqrt{n})$ [63]
$\alpha > 1, \alpha \notin \mathbb{N}$	$O(\frac{n}{\log n})$, $\Omega(n^{1-o(1)})$ [11]	$\tilde{O}(n^{1-\frac{1}{2\alpha}})$, $\Omega(\max\{n^{\frac{1}{3}}, n^{\frac{1}{2}-\frac{1}{2\alpha}}\})$
$\alpha = 2$	$\Theta(\sqrt{n})$ [11]	$\tilde{\Theta}(n^{\frac{1}{3}})$
$\alpha > 2, \alpha \in \mathbb{N}$	$\Theta(n^{1-1/\alpha})$ [11]	$\tilde{O}(n^{\nu(1-1/\alpha)})$, $\Omega(n^{\frac{1}{2}-\frac{1}{2\alpha}})$, $\nu < 3/4$
$\alpha = \infty$	$\Theta(\frac{n}{\log n})$ [262]	$\tilde{O}(Q(\lceil \log n \rceil\text{-distinctness}))$, $\Omega(\sqrt{n})$

Table 7.3: Classical and quantum query complexities of estimating α-Rényi entropy $H_\alpha(p)$, assuming $\epsilon = \Theta(1)$. © 2019 IEEE.

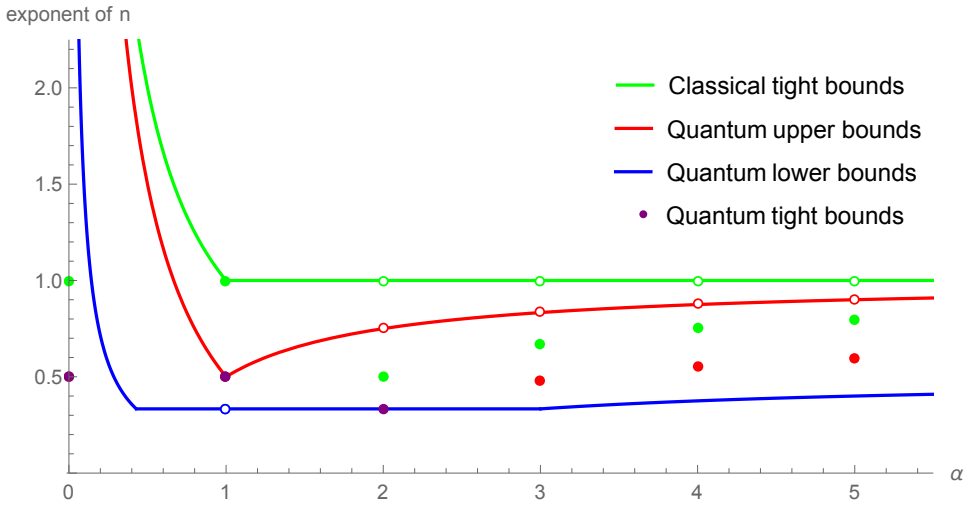

Figure 7.1: Visualization of classical and quantum query complexity of $H_\alpha(p)$. The x-axis represents α and the y-axis represents the exponent of n. Red curves and points represent quantum upper bounds. Green curves and points represent classical tight bounds. Blue curve represents quantum lower bounds. Purple points represent quantum tight bounds. © 2019 IEEE.

Techniques. At a high level, our upper bound is inspired by BHH [59], where we formulate a framework (in Section 7.4.2) that generalizes the technique in BHH and makes it applicable in our case. Let $F(p) = \sum_x p_x f(p_x)$ for some function $f(\cdot)$ and distribution p. Similar to BHH, we design a master algorithm that samples x from p and then use the *quantum counting primitive* [57] to obtain an estimate $\tilde{p}_x$ of p_x and outputs $f(\tilde{p}_x)$. It is easy to see that the expectation of the output of the master algorithm is roughly[14] $F(p)$. By choosing appropriate $f(\cdot)$s, one can recover $H(p)$ or $H_\alpha(p)$ as well as the ones used in BHH. It suffices then to obtain a good estimate of the output expectation of the master algorithm, which was achieved by multiple independent runs of the master algorithm in BHH.

The performance of the above framework (and its analysis) critically depends on how close the expectation of the algorithm is to $F(p)$ and how concentrated the output distribution is around its expectation, which in turn heavily depends on the specific $f(\cdot)$ in use. Our first contribution is a fine-tuned error analysis for specific $f(\cdot)$s, such as in the case of Shannon entropy (i.e., $f(p_x) = -\log(p_x)$) whose values could be significant for boundary cases of p_x. Instead of only considering the case when $\tilde{p}_x$ is a good estimate of p_x as in BHH, we need to analyze the entire distribution of $\tilde{p}_x$ using quantum counting. We also leverage a generic quantum speedup for estimating the expectation of the output of any quantum procedure with additive errors [208], which significantly improves our error dependence as compared to BHH. These improvements already give a quadratic quantum speedup for Shannon (Section 3 of [192]) and 0-Rényi (Section 8 of [192]) entropy estimation.

[14] The precise expectation is $\sum_x p_x \mathbb{E}[f(\tilde{p}_x)]$. Intuitively, $\tilde{p}_x$ should be a good estimate of p_x.

As an application, it also gives a quadratic speedup for estimating the KL-divergence between two distributions (Section 4 of [192]).

For general α-Rényi entropy $H_\alpha(p)$, we choose $f(p_x) = p_x^{\alpha-1}$ and let $P_\alpha(p) = F(p)$ so that $H_\alpha(p) \propto \log P_\alpha(p)$. Instead of estimating $F(p)$ with *additive errors* in the case of Shannon entropy, we switch to working with *multiplicative errors* which is harder since the aforementioned quantum algorithm [208] is much weaker in this setting. Indeed, by following the same technique, we can only obtain quantum speedups for α-Rényi entropy when $1/2 < \alpha < 2$.

For general $\alpha > 0$, our first observation is that if one knew the output expectation $\mathbb{E}[X]$ is within $[a, b]$ such that $b/a = \Theta(1)$, then one can slightly modify the technique in [208] (as shown in Theorem 7.4.4) and obtain a quadratic quantum speedup similar to the additive error setting. This approach, however, seems circular since it is unclear how to obtain such a, b in advance. Our second observation is that for any close enough α_1, α_2, $P_{\alpha_1}(p)$ can be used to bound $P_{\alpha_2}(p)$. Precisely, when $\alpha_1/\alpha_2 = 1 \pm 1/\log(n)$, we have $P_{\alpha_1}(p) = \Theta(P_{\alpha_2}(p)^{\alpha_1/\alpha_2})$. As a result, when estimating $P_\alpha(p)$, we can first estimate $P_{\alpha'}$ to provide a bound on P_α, where α', α differ by a $1 \pm 1/\log(n)$ factor and α' moves toward 1. We apply this strategy recursively on estimating $P_{\alpha'}$ until α' is very close to 1 from above when initial $\alpha > 1$ or from below when initial $\alpha < 1$, where a quantum speedup is already known. At a high level, we recursively estimate a sequence (of size $O(\log n)$) of such αs that eventually converges to 1, where in each iteration we establish some quantum speedup which leads to an overall quantum speedup. We remark that our approach is in spirit similar to the cooling schedules in simulated annealing (e.g. [251]). (See

Section 5 of [192].)

For *integer* $\alpha \geq 2$, we observe a connection between $P_\alpha(p)$ and the α-distinctness problem which leads to a more significant quantum speedup. Precisely, let $O_p : [S] \to [n]$ be the oracle in (7.4.6), we observe that $P_\alpha(p)$ is proportional to the α-*frequency moment* of $O_p(1), \ldots, O_p(S)$ which can be solved quantumly [209] based on any quantum algorithm for the α-distinctness problem (e.g., [44]). However, there is a catch that a direct application of [209] will lead to a dependence on S rather than n. We remedy this situation by tweaking the algorithm and its analysis in [209] to remove the dependence on S for our specific setting. (See Section 6 of [192].)

The integer α algorithm fails to extend to the *min-entropy* case (i.e., $\alpha = +\infty$) because the hidden constant in $O(\cdot)$ has a poor dependence on α. Instead, we develop another reduction to the $\lceil \log n \rceil$-distinctness problem by exploiting the so-called "Poissonized sampling" technique [153, 180, 262]. At a high level, we construct Poisson distributions that are parameterized by p_is and leverage the "threshold" behavior of Poisson distributions. Roughly, if $\max_i p_i$ passes some threshold, with high probability, these parameterized Poisson distributions will lead to a collision of size $\lceil \log n \rceil$ that will be caught by the $\lceil \log n \rceil$-distinctness algorithm. Otherwise, we run again with a lower threshold until the threshold becomes trivial. (See Section 7 of [192].)

Some of our lower bounds come from reductions to existing ones in quantum query complexity, such as the quantum-classical separation of symmetric boolean functions [4], the collision problem [6, 179], and the Hamming weight problem [212],

for different ranges of α. We also obtain lower bounds with a better error dependence by the polynomial method, which is inspired by the celebrated quantum lower bound for the collision problem [6, 179]. (See Section 9 of [192].)

Notations. Throughout this section, we consider a discrete distribution $\{p_i\}_{i=1}^n$ on $[n]$, and $P_\alpha(p) := \sum_{i=1}^n p_i^\alpha$ represents the α-power sum of p. In the analyses of our algorithms, 'log' is natural logarithm; '$\approx$' omits lower order terms.

7.4.2 Master algorithm

Let $p = (p_i)_{i=1}^n$ be a discrete distribution on $[n]$ encoded by the quantum oracle $\hat{O}_p$ defined in (7.4.6). Inspired by [59] (BHH) and [208], we develop the following master algorithm to estimate a property F with the form $F(p) := \sum_{i\in[n]} p_i f(p_i)$ for a function $f\colon (0,1] \to \mathbb{R}$.

Algorithm 7.2: Estimate $F(p) = \sum_i p_i f(p_i)$ of a distribution p on $[n]$.

1 Set $l, M \in \mathbb{N}$;
2 **Regard the following subroutine as $\mathcal{A}$:**
3 Draw a sample $i \in [n]$ according to p;
4 Use **EstAmp** or **EstAmp′** with M queries to get an estimate $\tilde{p}_i$ of p_i;
5 Output $X = f(\tilde{p}_i)$;
6 Use $\mathcal{A}$ for l executions in Theorem 7.4.3 or Theorem 7.4.4 and output $\tilde{F}(p)$ to estimate $F(p)$;

Comparing to BHH, we introduce a few new technical ingredients, which significantly improve the performance of Algorithm 7.2 especially for specific $f(\cdot)$s in our case, e.g., $f(p_x) = -\log(p_x)$ (Shannon entropy) and $f(p_x) = p_x^{\alpha-1}$ (Rényi entropy).

The **first** one is a generic quantum speedup of Monte Carlo methods [208], in particular, a quantum algorithm that approximates the output expectation of a

subroutine with additive errors that has a quadratic better sample complexity than the one implied by Chebyshev's inequality.

Theorem 7.4.3 (Additive error; Theorem 5 of [208]). *Let $\mathcal{A}$ be a quantum algorithm with output X such that $\mathrm{Var}[X] \leq \sigma^2$. Then for ϵ where $0 < \epsilon < 4\sigma$, by using $O((\sigma/\epsilon)\log^{3/2}(\sigma/\epsilon)\log\log(\sigma/\epsilon))$ executions of $\mathcal{A}$ and $\mathcal{A}^{-1}$, Algorithm 3 in [208] outputs an estimate $\tilde{\mathbb{E}}[X]$ of $\mathbb{E}[X]$ such that*

$$\Pr\left[\left|\tilde{\mathbb{E}}[X] - \mathbb{E}[X]\right| \geq \epsilon\right] \leq 1/5. \tag{7.4.7}$$

It is worthwhile mentioning that classically one needs to use $\Omega(\sigma^2/\epsilon^2)$ executions of $\mathcal{A}$ [91] to estimate $\mathbb{E}[X]$. Theorem 7.4.3 demonstrates a quadratic improvement on the error dependence. In the case of approximating $H_\alpha(p)$, we need to work with multiplicative errors while existing results (e.g. [208]) have a worse error dependence which is insufficient for our purposes. Instead, inspired by [208], we prove the following theorem (our **second** ingredient) that takes auxiliary information about the range of $\mathbb{E}[X]$ into consideration, which might be of independent interest.

Theorem 7.4.4 (Multiplicative error). *Let $\mathcal{A}$ be a quantum algorithm with output X such that $\mathrm{Var}[X] \leq \sigma^2\mathbb{E}[X]^2$ for a known σ. Assume that $\mathbb{E}[X] \in [a, b]$. Then for ϵ where $0 < \epsilon < 24\sigma$, by using $\mathcal{A}$ and $\mathcal{A}^{-1}$ for $O((\sigma b/\epsilon a)\log^{3/2}(\sigma b/\epsilon a)\log\log(\sigma b/\epsilon a))$ executions, Algorithm 7.3 outputs an estimate $\tilde{\mathbb{E}}[X]$ of $\mathbb{E}[X]$ such that*

$$\Pr\left[\left|\tilde{\mathbb{E}}[X] - \mathbb{E}[X]\right| \geq \epsilon\mathbb{E}[X]\right] \leq 1/10. \tag{7.4.8}$$

To prove Theorem 7.4.4, the main technique that we use is Lemma 4 in [208], which approximates a random variable with an additive error as long as its second-moment is bounded:

Lemma 7.4.1 (Lemma 4 in [208]). *Assume $\mathcal{A}$ is a quantum algorithm that outputs a random variable X. Then for ϵ where $0 < \epsilon < 1/2$ (multiplicative error), by using $O((1/\epsilon)\log^{3/2}(1/\epsilon)\log\log(1/\epsilon))$ executions of $\mathcal{A}$ and $\mathcal{A}^{-1}$, Algorithm 2 in [208] outputs an estimate $\tilde{\mathbb{E}}[X]$ of $\mathbb{E}[X]$ such that*[15]

$$\Pr\left[\left|\tilde{\mathbb{E}}[X] - \mathbb{E}[X]\right| \geq \epsilon(\sqrt{\mathbb{E}[X^2]} + 1)^2\right] \leq 1/50. \tag{7.4.9}$$

Based on Lemma 7.4.1 and inspired by Algorithm 3 and Theorem 5 in [208], we propose Algorithm 7.3.

Algorithm 7.3: Estimate $\mathbb{E}[X]$ within multiplicative error ϵ.

1. Run the algorithm that gives a, b such that $\mathbb{E}[X] \in [a, b]$;
2. Set $\mathcal{A}' = \mathcal{A}/\sigma b$;
3. Run $\mathcal{A}'$ once and denote $\widetilde{m}$ to be the output. Set $\mathcal{B} = \mathcal{A}' - \widetilde{m}$;
4. Let $\mathcal{B}_-$ be the algorithm that calls $\mathcal{B}$ once; if $\mathcal{B}$ outputs $x \geq 0$ then $\mathcal{B}_-$ outputs 0, and if $\mathcal{B}$ outputs $x < 0$ then $\mathcal{B}_-$ outputs x. Similarly, let $\mathcal{B}_+$ be the algorithm such that if $\mathcal{B}$ outputs $x < 0$ then $\mathcal{B}_+$ outputs 0, and if $\mathcal{B}$ outputs $x \geq 0$ then $\mathcal{B}_+$ outputs x;
5. Apply Lemma 7.4.1 to $-\mathcal{B}_-/6$ and $\mathcal{B}_+/6$ with error $\frac{\epsilon a}{48\sigma b}$ and failure probability 1/50, and obtain estimates $\widetilde{\mu}_-$ and $\widetilde{\mu}_+$, respectively;
6. Output $\tilde{\mathbb{E}}[X] = \sigma b(\widetilde{m} - 6\widetilde{\mu}_- + 6\widetilde{\mu}_+)$;

We now give the proof of Theorem 7.4.4.

[15] The original error probability in (7.4.9) is 1/5, but it can be improved to 1/50 by rescaling the parameters in Lemma 4 in [208] up to a constant.

Proof. Because $\mathrm{Var}[X] \le \sigma^2 \mathbb{E}[X]^2 \le \sigma^2 b^2$, by Chebyshev's inequality we have

$$\Pr\left[\left|\widetilde{m} - \mathbb{E}[X/\sigma b]\right| \ge 4\right] \le 1/16. \tag{7.4.10}$$

Therefore, with probability at least 15/16 we have $|\widetilde{m} - \mathbb{E}[X/\sigma b]| \le 4$. Denote $X_B = \frac{X}{\sigma b} - \widetilde{m}$, which is the random variable output by $\mathcal{B}$; $X_{B,+} := \max\{X_B, 0\}$ is then the output of $\mathcal{B}_+$ and $X_{B,-} := \min\{X_B, 0\}$ is the output of $\mathcal{B}_-$. Assuming $|\widetilde{m} - \mathbb{E}[X/\sigma b]| \le 4$, we have

$$\mathbb{E}[X_B^2] = \mathbb{E}\Big[\Big(\big(\frac{X}{\sigma b} - \mathbb{E}[\frac{X}{\sigma b}]\big) + \big(\mathbb{E}[\frac{X}{\sigma b}] - \widetilde{m}\big)\Big)^2\Big] \tag{7.4.11}$$

$$\le 2\mathbb{E}\Big[\Big(\frac{X}{\sigma b} - \mathbb{E}[\frac{X}{\sigma b}]\Big)^2\Big] + 2\mathbb{E}\Big[\Big(\mathbb{E}[\frac{X}{\sigma b}] - \widetilde{m}\Big)^2\Big] \tag{7.4.12}$$

$$\le 2(1^2 + 4^2) = 34. \tag{7.4.13}$$

Therefore, $\mathbb{E}\big[(X_B/6)^2\big] \le 34/36 < 1$, hence $\mathbb{E}\big[(X_{B,+}/6)^2\big] < 1$ and $\mathbb{E}\big[(-X_{B,-}/6)^2\big] < 1$. By Lemma 7.4.1, we have

$$\left|\widetilde{\mu}_- - \mathbb{E}[-X_{B,-}/6]\right| \le \frac{\epsilon a}{12\sigma b} \qquad \text{and} \qquad \left|\widetilde{\mu}_+ - \mathbb{E}[X_{B,+}/6]\right| \le \frac{\epsilon a}{12\sigma b} \tag{7.4.14}$$

both with failure probability at most 1/50. Because

$$\mathbb{E}[X] = \sigma b\big(\widetilde{m} + \mathbb{E}[X_B]\big) = \sigma b\big(\widetilde{m} + \mathbb{E}[X_{B,+}] - \mathbb{E}[-X_{B,-}]\big), \tag{7.4.15}$$

with probability at least $15/16 \cdot (1-1/50)^2 > 9/10$, we have

$$\left|\tilde{\mathbb{E}}[X] - \mathbb{E}[X]\right| \leq \sigma b \cdot \left(6\left|\widetilde{\mu}_- - \mathbb{E}[-X_{B,-}/6]\right| + 6\left|\widetilde{\mu}_+ - \mathbb{E}[X_{B,+}/6]\right|\right) \tag{7.4.16}$$

$$\leq \sigma b \cdot 2 \cdot 6 \cdot \frac{\epsilon a}{12\sigma b} = \epsilon a \leq \epsilon \mathbb{E}(X). \tag{7.4.17}$$

□

The **third** ingredient is a fine-tuned error analysis due to the specific $f(\cdot)$s. Similar to BHH, we rely on quantum counting (named **EstAmp**) [57] to estimate the pre-image size of a Boolean function, which provides another source of quantum speedup. In particular, we approximate any probability p_x in the query model ((7.4.6)) by $\tilde{p}_x$ by estimating the size of the pre-image of a Boolean function $\chi\colon [S] \to \{0,1\}$ with $\chi(s) = 1$ if $O(s) = i$ and $\chi(s) = 0$ otherwise. However, for cases in BHH, it suffices to only consider the probability when p_x and $\tilde{p}_x$ are close, while in our case, we need to analyze the whole output distribution of quantum counting. Specifically, letting $t = \left|\chi^{-1}(1)\right|$ and $a = t/S = \sin^2(\omega\pi)$ for some ω, we have

Theorem 7.4.5 ([57]). *For any $k, M \in \mathbb{N}$, there is a quantum algorithm (named **EstAmp**) with M quantum queries to χ that outputs $\tilde{a} = \sin^2\left(\frac{l\pi}{M}\right)$ for some $l \in \{0, \ldots, M-1\}$ such that*

$$\Pr\left[\tilde{a} = \sin^2\left(\frac{l\pi}{M}\right)\right] = \frac{\sin^2(M\Delta\pi)}{M^2\sin^2(\Delta\pi)} \leq \frac{1}{(2M\Delta)^2}, \tag{7.4.18}$$

where $\Delta = |\omega - \frac{l}{M}|$. This promises $|\tilde{a} - a| \leq 2\pi k\frac{\sqrt{a(1-a)}}{M} + k^2\frac{\pi^2}{M^2}$ with probability at least $\frac{8}{\pi^2}$ for $k = 1$ and with probability greater than $1 - \frac{1}{2(k-1)}$ for $k \geq 2$. If $a = 0$

then $\tilde{a} = 0$ *with certainty.*

Moreover, we also need to slightly modify **EstAmp** to avoid outputting $\tilde{p}_x = 0$ in estimating Shannon entropy. This is because $f(\tilde{p}_x) = \log(\tilde{p}_x)$ is not well-defined at $\tilde{p}_x = 0$. Let **EstAmp′** be the modified algorithm. It is required that **EstAmp′** outputs $\sin^2(\frac{\pi}{2M})$ when **EstAmp** outputs 0 and outputs **EstAmp**'s output otherwise.

By leveraging Theorem 7.4.3, Theorem 7.4.4, Theorem 7.4.5, and carefully setting parameters in Algorithm 7.2, we have the following corollaries that describe the complexity of estimating any $F(p)$.

Corollary 7.4.2 (additive error)**.** *Given* $\epsilon > 0$*. If* $l = \Theta\big(\big(\frac{\sigma}{\epsilon}\big)\log^{3/2}\big(\frac{\sigma}{\epsilon}\big)\log\log\big(\frac{\sigma}{\epsilon}\big)\big)$ *where* $\mathrm{Var}[X] \leq \sigma^2$ *and* M *is large enough such that* $\big|\mathbb{E}[X] - F(p)\big| \leq \epsilon$*, then Algorithm 7.2 approximates* $F(p)$ *with an additive error* ϵ *and success probability* $2/3$ *using* $O\big(M \cdot l\big)$ *quantum queries to* p*.*

Corollary 7.4.3 (multiplicative error)**.** *Assume a procedure using* $C_{a,b}$ *quantum queries that returns an estimated range* $[a, b]$*, and that* $\mathbb{E}[X] \in [a, b]$ *with probability at least 0.9. Let* $l = \Theta\big(\big(\frac{\sigma b}{\epsilon a}\big)\log^{3/2}\big(\frac{\sigma b}{\epsilon a}\big)\log\log\big(\frac{\sigma b}{\epsilon a}\big)\big)$ *where* $\mathrm{Var}[X]/(\mathbb{E}[X])^2 \leq \sigma^2$ *and* $\epsilon > 0$*. For large enough* M *such that* $\big|\mathbb{E}[X] - F(p)\big| \leq \epsilon$*, Algorithm 7.2 estimates* $F(p)$ *with a multiplicative error* ϵ *and success probability* $2/3$ *with* $O\big(M \cdot l + C_{a,b}\big)$ *queries.*

We can use Corollary 7.4.2 and Corollary 7.4.3 to prove our entropy estimation results in Theorem 7.4.1. Complete details and proofs are given in [192].

7.5 Conclusions and discussion

In this chapter, we show that quantum computers can test properties of distributions with significant speed-ups. We also introduce a novel access model for quantum distributions, enabling the coherent preparation of quantum samples, and propose a general framework that can naturally handle both classical and quantum distributions in a unified manner. Our framework generalizes and improves previous quantum algorithms for testing closeness between unknown distributions, testing independence between two distributions, and estimating the Shannon / von Neumann / Rényi entropy of distributions. For classical distributions our algorithms significantly improve the precision dependence of some earlier results. We also show that in our framework procedures for classical distributions can be directly lifted to the more general case of quantum distributions, and thus obtain the first speed-ups for testing properties of density operators that can be accessed coherently rather than only via sampling.

There are a couple of natural open questions for future work, including:

- For which other distributional property testing problems can we get faster and simpler quantum algorithms using the presented methodology?

- Can we prove quantum lower bounds that match our upper bounds? For instance, can we prove an $\Omega\left(\frac{n}{\epsilon}\right)$ lower bound on estimating the von Neumann entropy in the purified quantum query-access model for density operators?

- Is there a lower bound technique which naturally fits our purified quantum

query input model?

- Can we prove the conjecture that the purified and discrete query input models are equivalent for classical distributions, with respect to the query complexity of (distributional) property testing problems? For some recent progress in this direction see [45].

Chapter 8: Conclusions and Future Work

This thesis presented quantum algorithms for three different fields:

- Optimization: With implicit oracle access, we studied quantum speedup of general convex optimization (Chapter 2) and volume estimation of convex bodies (Chapter 3). With explicit matrix inputs, we studied quantum SDP solvers (Chapter 4).
- Machine learning: We proposed the optimal quantum algorithm for classification (Chapter 5). We also studied quantum-inspired classical machine learning algorithms (Chapter 6).
- Statistics: We focused on quantum speedup of testing properties of distributions (Chapter 7).

I believe that our quantum algorithms can motivate further interdisciplinary research between quantum computing and optimization, machine learning, and statistics. For instance, it might be worthwhile to explore the following future directions:

Nonconvex optimization. Recently, research on nonconvex optimization has been dramatically developed because the loss functions in many machine learning

models (including neural networks) are typically not convex. However, finding the global optima of a nonconvex function is NP-hard in general. Instead, many theoretical works focus on finding local optima of nonconvex functions, since there are landscape results suggesting that local optima are nearly as good as the global optima for many learning problems (see e.g. [50, 116–119, 136]). Specifically, Refs. [155, 156] presented a breakthrough result along this line, which finds an ϵ-approximate local minimum in $\tilde{O}(1/\epsilon^2)$ iterations using only the gradient oracle $\nabla f(x)$.

Quantumly, it would be natural to explore quantum algorithms for nonconvex optimization. On the one hand, escaping saddle points is essentially tunneling through poor landscapes, and quantum tunneling can potentially be a mechanism for solving nonconvex optimization with quantum speedup. On the other hand, it would be of interest to investigate real scenarios where gradients are difficult to compute and only evaluations of the function are available, and see how quantum computers provide speedups in such cases.

Sampling from convex bodies. Sampling from convex bodies is closely related to convex optimization with wide applications in machine learning. As seen in (Chapter 3), sampling from n-dimensional log-concave distributions can be achieved in poly(n) time given query access to the function [103, 112] using the hit-and-run walk. However, in many cases other dynamics can converge significantly faster; common proposals are Metropolis sampling [101], Langevin dynamics [92], Hamiltonian Monte Carlo [185], etc.

It is a natural question to understand the convergence of quantum dynamics in

general. Such analysis might rely on better understanding of open quantum systems, for instance our paper [82] showed how to efficiently simulate sparse Markovian open systems.

Learning distributions. Following Chapter 7, it is natural to explore quantum algorithms for learning distributions. On the one hand, it is worthwhile to investigate quantum algorithms for learning distributions from given types, a common question asked in learning theory and statistics. This problem has been studied classically for log-concave [71], unimodal [71], Poisson binomial [72], and Gaussian distributions [72], and it is a natural question to ask whether quantum algorithms have advantages for them. On the other hand, it may also be of interest to understand the quantum complexity of learning the structure of an Ising model. Classical algorithms for this topic have been well-understood (see e.g. [94, 124, 176]), but given that quantum Ising models are extensively used and studied in quantum mechanics, it would be interesting to understand the cost of learning their structures.

Bibliography

[1] Scott Aaronson, *The learnability of quantum states*, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **463** (2007), no. 2088, 3089–3114, arXiv:quant-ph/0608142.

[2] Scott Aaronson, *Read the fine print*, Nature Physics **11** (2015), no. 4, 291.

[3] Scott Aaronson, *Shadow tomography of quantum states*, Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 325–338, ACM, 2018, arXiv:1711.01053.

[4] Scott Aaronson and Andris Ambainis, *The need for structure in quantum speedups*, Theory of Computing **10** (2014), no. 6, 133–166, arXiv:0911.0996.

[5] Scott Aaronson, Xinyi Chen, Elad Hazan, Satyen Kale, and Ashwin Nayak, *Online learning of quantum states*, Advances in Neural Information Processing Systems, pp. 8962–8972, 2018, arXiv:1802.09025.

[6] Scott Aaronson and Yaoyun Shi, *Quantum lower bounds for the collision and the element distinctness problems*, Journal of the ACM **51** (2004), no. 4, 595–605.

[7] Ittai Abraham, Shiri Chechik, and Sebastian Krinninger, *Fully dynamic all-pairs shortest paths with worst-case update-time revisited*, Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 440–452, SIAM, 2017, arXiv:1607.05132.

[8] Jayadev Acharya, Hirakendu Das, Alon Orlitsky, and Ananda Theertha Suresh, *A unified maximum likelihood approach for estimating symmetric properties of discrete distributions*, Proceedings of the 34th International Conference on Machine Learning, pp. 11–21, 2017, arXiv:1611.02960.

[9] Jayadev Acharya, Constantinos Daskalakis, and Gautam Kamath, *Optimal testing for properties of distributions*, Advances in Neural Information Processing Systems 28, pp. 3591–3599, 2015, arXiv:1507.05952.

[10] Jayadev Acharya, Ibrahim Issa, Nirmal V. Shende, and Aaron B. Wagner, *Measuring quantum entropy*, 2019 IEEE International Symposium on Information Theory, pp. 3012–3016, IEEE, 2019, arXiv:1711.00814.

[11] Jayadev Acharya, Alon Orlitsky, Ananda Theertha Suresh, and Himanshu Tyagi, *Estimating Rényi entropy of discrete distributions*, IEEE Transactions on Information Theory **63** (2017), no. 1, 38–56, arXiv:1408.1000.

[12] Dimitris Achlioptas and Frank McSherry, *Fast computation of low-rank matrix approximations*, Journal of the ACM **54** (2007), no. 2, 9–es.

[13] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani, *Quantum walks on graphs*, Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, pp. 50–59, 2001, arXiv:quant-ph/0012090.

[14] Dorit Aharonov and Amnon Ta-Shma, *Adiabatic quantum state generation and statistical zero knowledge*, Proceedings of the 35th Annual ACM Symposium on Theory of Computing, pp. 20–29, 2003, arXiv:quant-ph/0301023.

[15] Gorjan Alagic and Alexander Russell, *Decoherence in quantum walks on the hypercube*, Physical Review A **72** (2005), 062304, arXiv:quant-ph/0501169.

[16] Zeyuan Allen-Zhu, Yin Tat Lee, and Lorenzo Orecchia, *Using optimization to obtain a width-independent, parallel, simpler, and faster positive sdp solver*, Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1824–1831, Society for Industrial and Applied Mathematics, 2016, arXiv:1507.02259.

[17] Andris Ambainis, *Quantum walk algorithm for element distinctness*, SIAM Journal on Computing **37** (2007), no. 1, 210–239, arXiv:quant-ph/0311001.

[18] Andris Ambainis, *Variable time amplitude amplification and quantum algorithms for linear algebra problems*, Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science, vol. 14, pp. 636–647, 2012, arXiv:1010.4458.

[19] Andris Ambainis, Eric Bach, Ashwin Nayak, Ashvin Vishwanath, and John Watrous, *One-dimensional quantum walks*, Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, pp. 37–49, 2001.

[20] Andris Ambainis and Ashley Montanaro, *Quantum algorithms for search with wildcards and combinatorial group testing*, Quantum Information and Computation **14** (2014), no. 5&6, 439–453, arXiv:1210.1148.

[21] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn, *Approximate nearest neighbor search in high dimensions*, 2018, arXiv:1806.09823.

[22] Kurt M. Anstreicher, *The volumetric barrier for semidefinite programming*, Mathematics of Operations Research **25** (2000), no. 3, 365–380.

[23] Joran van Apeldoorn and András Gilyén, *Improvements in quantum SDP-solving with applications*, Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 99:1–99:15, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1804.05058.

[24] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf, *Quantum SDP-solvers: Better upper and lower bounds*, Proceedings of the 58th Annual Symposium on Foundations of Computer Science, IEEE, 2017, arXiv:1705.01843.

[25] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf, *Convex optimization using quantum oracles*, Quantum **4** (2020), 220, arXiv:1809.00643.

[26] David Applegate and Ravi Kannan, *Sampling and integration of near log-concave functions*, Proceedings of the 23nd Annual ACM Symposium on Theory of Computing, pp. 156–163, 1991.

[27] Erdal Arikan, *An inequality on guessing and its application to sequential decoding*, IEEE Transactions on Information Theory **42** (1996), no. 1, 99–105.

[28] Sanjeev Arora, Elad Hazan, and Satyen Kale, *The multiplicative weights update method: a meta-algorithm and applications*, Theory of Computing **8** (2012), no. 1, 121–164.

[29] Sanjeev Arora and Satyen Kale, *A combinatorial, primal-dual approach to semidefinite programs*, Proceedings of the 39th Annual ACM Symposium on Theory of Computing, pp. 227–236, ACM, 2007.

[30] Juan Miguel Arrazola, Alain Delgado, Bhaskar Roy Bardhan, and Seth Lloyd, *Quantum-inspired algorithms in practice*, 2019, arXiv:1905.10415.

[31] Srinivasan Arunachalam, Sourav Chakraborty, Troy Lee, Manaswi Paraashar, and Ronald de Wolf, *Two new results about quantum exact learning*, Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 16:1–16:15, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1810.00481.

[32] Srinivasan Arunachalam and Ronald de Wolf, *Guest column: a survey of quantum learning theory*, ACM SIGACT News **48** (2017), no. 2, 41–67, arXiv:1701.06806.

[33] Srinivasan Arunachalam and Ronald de Wolf, *Optimal quantum sample complexity of learning algorithms*, The Journal of Machine Learning Research **19** (2018), no. 1, 2879–2878, arXiv:1607.00932.

[34] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G.S.L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandr, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis, *Quantum supremacy using a programmable superconducting processor*, Nature **574** (2019), no. 7779, 505–510, arXiv:1910.11333.

[35] Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon, *Fully dynamic maximal independent set with sublinear update time*, Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 815–826, ACM, 2018, arXiv:1802.09709.

[36] Alp Atici and Rocco A. Servedio, *Improved bounds on quantum learning algorithms*, Quantum Information Processing **4** (2005), no. 5, 355–386, arXiv:quant-ph/0411140.

[37] Alp Atıcı and Rocco A. Servedio, *Quantum algorithms for learning and testing juntas*, Quantum Information Processing **6** (2007), no. 5, 323–348, arXiv:0707.3479.

[38] Costin Bădescu, Ryan O'Donnell, and John Wright, *Quantum state certification*, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, pp. 503–514, 2019, arXiv:1708.06002.

[39] Imre Bárány and Zoltán Füredi, *Computing the volume is difficult*, Discrete & Computational Geometry **2** (1987), no. 4, 319–326.

[40] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld, *The complexity of approximating the entropy*, SIAM Journal on Computing **35** (2005), no. 1, 132–150.

[41] Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White, *Testing random variables for independence and identity*, Proceedings 42nd IEEE Symposium on Foundations of Computer Science, pp. 442–451, IEEE, 2001.

[42] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White, *Testing closeness of discrete distributions*, Journal of the ACM **60** (2013), no. 1, 1–25, arXiv:1009.5397.

[43] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf, *Quantum lower bounds by polynomials*, Journal of the ACM **48** (2001), no. 4, 778–797, arXiv:quant-ph/9802049.

[44] Aleksandrs Belovs, *Learning-graph-based quantum algorithm for k-distinctness*, Proceedings of the 53rd Annual Symposium on Foundations of Computer Science, pp. 207–216, IEEE, 2012, arXiv:1205.1534.

[45] Aleksandrs Belovs, *Quantum algorithms for classical probability distributions*, Proceedings of the 27th Annual European Symposium on Algorithms, Leibniz International Proceedings in Informatics (LIPIcs), vol. 144, pp. 16:1–16:11, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1904.02192.

[46] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani, *Strengths and weaknesses of quantum computing*, SIAM Journal on Computing **26** (1997), no. 5, 1510–1523, arXiv:quant-ph/9701001.

[47] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli M. Maurer, *Generalized privacy amplification*, IEEE Transactions on Information Theory **41** (1995), no. 6, 1915–1923.

[48] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai, *Dynamic algorithms for graph coloring*, Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1–20, Society for Industrial and Applied Mathematics, 2018, arXiv:1711.04355.

[49] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai, *Fully dynamic approximate maximum matching and minimum vertex cover in $o(\log^3 n)$ worst case update time*, Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 470–489, SIAM, 2017, arXiv:1704.02844.

[50] Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro, *Global optimality of local search for low rank matrix recovery*, Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 3880–3888, 2016, arXiv:1605.07221.

[51] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, *Quantum machine learning*, Nature **549** (2017), no. 7671, 195, arXiv:1611.09347.

[52] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth, *Learnability and the Vapnik-Chervonenkis dimension*, Journal of the ACM **36** (1989), no. 4, 929–965.

[53] Sergio Boixo and Rolando D. Somma, *Quantum algorithms for simulated annealing*, 2015, arXiv:1512.03806.

[54] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, New York, NY, USA, 2004.

[55] Fernando G.S.L. Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M. Svore, and Xiaodi Wu, *Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning*, Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 27:1–27:14, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1710.02581.

[56] Fernando G.S.L. Brandão and Krysta Svore, *Quantum speed-ups for semidefinite programming*, Proceedings of the 58th Annual Symposium on Foundations of Computer Science, pp. 415–426, 2017, arXiv:1609.05537.

[57] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp, *Quantum amplitude amplification and estimation*, Contemporary Mathematics **305** (2002), 53–74, arXiv:quant-ph/0005055.

[58] Gilles Brassard, Peter Høyer, and Alain Tapp, *Quantum counting*, Proceedings of the 25th International Colloquium on Automata, Languages, and Programming, pp. 820–831, 1998, arXiv:quant-ph/9805082.

[59] Sergey Bravyi, Aram W. Harrow, and Avinatan Hassidim, *Quantum algorithms for testing properties of distributions*, IEEE Transactions on Information Theory **57** (2011), no. 6, 3971–3981, arXiv:0907.3920.

[60] Nader H. Bshouty and Jeffrey C. Jackson, *Learning DNF over the uniform distribution using a quantum example oracle*, SIAM Journal on Computing **28** (1998), no. 3, 1136–1153.

[61] Sébastien Bubeck, *Convex optimization: Algorithms and complexity*, Foundations and Trends® in Machine Learning **8** (2015), no. 3-4, 231–357.

[62] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf, *Quantum fingerprinting*, Physical Review Letters **87** (2001), no. 16, 167902, arXiv:quant-ph/0102001.

[63] Mark Bun, Robin Kothari, and Justin Thaler, *The polynomial method strikes back: Tight quantum query bounds via dual polynomials*, Proceedings of the 50th Annual ACM Symposium on Theory of Computing, 2018, arXiv:1710.09079.

[64] Olivier Catoni, *Statistical learning theory and stochastic optimization: Ecole d'eté de probabilités de saint-flour xxxi-2001*, Springer, 2004.

[65] André Chailloux, *A note on the quantum query complexity of permutation symmetric functions*, Proceedings of the 10th Innovations in Theoretical Computer Science Conference, Leibniz International Proceedings in Informatics (LIPIcs), vol. 124, pp. 19:1–19:7, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, arXiv:1810.01790.

[66] Shouvanik Chakrabarti, Andrew M. Childs, Shih-Han Hung, Tongyang Li, Chunhao Wang, and Xiaodi Wu, *Quantum algorithm for estimating volumes of convex bodies*, 2019, arXiv:1908.03903.

[67] Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu, *Quantum algorithms and lower bounds for convex optimization*, Quantum **4** (2020), 221, arXiv:1809.01731.

[68] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery, *The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation*, Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, Leibniz International Proceedings in Informatics (LIPIcs), vol. 132, pp. 33:1–33:14, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, arXiv:1804.01973.

[69] Shantanav Chakraborty, Kyle Luh, and Jérémie Roland, *On analog quantum algorithms for the mixing of Markov chains*, 2019, arXiv:1904.11895.

[70] Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Ronald de Wolf, *New results on quantum property testing*, Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, Leibniz International Proceedings in Informatics (LIPIcs), vol. 8, pp. 145–156, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, arXiv:1005.0523.

[71] Siu-on Chan, Ilias Diakonikolas, Rocco A. Servedio, and Xiaorui Sun, *Learning mixtures of structured distributions over discrete domains*, Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1380–1394, 2013, arXiv:1210.0864.

[72] Siu-On Chan, Ilias Diakonikolas, Rocco A. Servedio, and Xiaorui Sun, *Efficient density estimation via piecewise polynomial approximation*, Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 604–613, ACM, 2014, arXiv:1305.3207.

[73] Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant, *Optimal algorithms for testing closeness of discrete distributions*, Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1193–1203, SIAM, 2014, arXiv:1308.3946.

[74] Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu, *Orthogonal range searching on the RAM, revisited*, Proceedings of the 27th Annual Symposium on Computational Geometry, pp. 1–10, ACM, 2011, arXiv:1103.5510.

[75] Zhihuai Chen, Yinan Li, Xiaoming Sun, Pei Yuan, and Jialin Zhang, *A quantum-inspired classical algorithm for separable non-negative matrix factorization*, Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 4511–4517, AAAI Press, 2019, arXiv:1907.05568.

[76] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang, *Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning*, To appear in the proceedings of the 52nd Annual ACM Symposium on Theory of Computing, ACM, 2020, arXiv:1910.06151.

[77] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang, *Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches*, 2019, arXiv:1901.03254.

[78] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang, *Quantum-inspired sublinear classical algorithms for solving low-rank linear systems*, 2018, arXiv:1811.04852.

[79] Andrew M. Childs, *Lecture notes on quantum algorithms*, Lecture notes at University of Maryland, 2017, https://www.cs.umd.edu/%7Eamchilds/qa/qa.pdf.

[80] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman, *Exponential algorithmic speedup by quantum walk*, Proceedings of the 35th ACM Symposium on Theory of Computing, pp. 59–68, 2003, arXiv:quant-ph/0209131.

[81] Andrew M. Childs, Robin Kothari, and Rolando D. Somma, *Quantum algorithm for systems of linear equations with exponentially improved dependence on precision*, SIAM Journal on Computing **46** (2017), no. 6, 1920–1950, arXiv:1511.02306.

[82] Andrew M. Childs and Tongyang Li, *Efficient simulation of sparse Markovian quantum dynamics*, Quantum Information & Computation **17** (2017), no. 11-12, 901–947, arXiv:1611.05543.

[83] Andrew M. Childs and Wim. Van Dam, *Quantum algorithms for algebraic problems*, Reviews of Modern Physics **82** (2010), no. 1, 1, arXiv:0812.0380.

[84] Anirban Narayan Chowdhury and Rolando D. Somma, *Quantum algorithms for Gibbs sampling and hitting-time estimation*, Quantum Information & Computation **17** (2017), no. 1-2, 41–64, arXiv:1603.02940.

[85] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff, *Sublinear optimization for machine learning*, Journal of the ACM **59** (2012), no. 5, 23, arXiv:1010.4408.

[86] Michael B. Cohen, Yin Tat Lee, and Zhao Song, *Solving linear programs in the current matrix multiplication time*, Proceedings of the 51st Annual ACM Symposium on Theory of Computing, pp. 938–942, ACM, 2019, arXiv:1810.07896.

[87] Ben Cousins and Santosh Vempala, *A cubic algorithm for computing Gaussian volume*, Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1215–1228, 2014, arXiv:1306.5829.

[88] Ben Cousins and Santosh Vempala, *Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm*, Proceedings of the 47th Annual ACM Symposium on Theory of Computing, pp. 539–548, 2015, arXiv:1409.6011.

[89] Imre Csiszár, *Generalized cutoff rates and Rényi's information measures*, IEEE Transactions on Information Theory **41** (1995), no. 1, 26–34.

[90] Imre Csiszar and János Körner, *Information theory: coding theorems for discrete memoryless systems*, Cambridge University Press, 2011.

[91] Paul Dagum, Richard Karp, Michael Luby, and Sheldon Ross, *An optimal algorithm for Monte Carlo estimation*, SIAM Journal on Computing **29** (2000), no. 5, 1484–1496.

[92] Arnak S. Dalalyan, *Theoretical guarantees for approximate sampling from smooth and log-concave densities*, Journal of the Royal Statistical Society: Series B (Statistical Methodology) **79** (2017), no. 3, 651–676, arXiv:1412.7392.

[93] George B. Dantzig and Mukund N. Thapa, *Linear programming 2: Theory and extensions*, Springer, 2006.

[94] Constantinos Daskalakis, Nishanth Dikkala, and Gautam Kamath, *Testing Ising models*, Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1989–2007, Society for Industrial and Applied Mathematics, 2018, arXiv:1612.03147.

[95] Christopher M. Dawson and Michael A. Nielsen, *The Solovay-Kitaev algorithm*, Quantum Information & Computation **6** (2006), no. 1, 81–95, arXiv:quant-ph/0505030.

[96] Ilias Diakonikolas and Daniel M. Kane, *A new approach for testing properties of discrete distributions*, Proceedings of the 57th Annual Symposium on Foundations of Computer Science, pp. 685–694, IEEE, 2016, arXiv:1601.05557.

[97] Chen Ding, Tian-Yi Bao, and He-Liang Huang, *Quantum-inspired support vector machine*, 2019, arXiv:1906.08902.

[98] P. Drineas, R. Kannan, and M. Mahoney, *Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication*, SIAM Journal on Computing **36** (2006), no. 1, 132–157.

[99] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao, *A quantum-inspired algorithm for general minimum conical hull problems*, 2019, arXiv:1907.06814.

[100] Christoph Dürr and Peter Høyer, *A quantum algorithm for finding the minimum*, 1996, arXiv:quant-ph/9607014.

[101] Raaz Dwivedi, Yuansi Chen, Martin J. Wainwright, and Bin Yu, *Log-concave sampling: Metropolis-Hastings algorithms are fast*, Journal of Machine Learning Research **20** (2019), no. 183, 1–42, arXiv:1801.02309.

[102] Martin Dyer and Alan Frieze, *Computing the volume of convex bodies: a case where randomness provably helps*, Probabilistic Combinatorics and its Applications **44** (1991), 123–170.

[103] Martin Dyer, Alan Frieze, and Ravi Kannan, *A random polynomial-time algorithm for approximating the volume of convex bodies*, Journal of the ACM **38** (1991), no. 1, 1–17.

[104] Martin E. Dyer and Alan M. Frieze, *On the complexity of computing the volume of a polyhedron*, SIAM Journal on Computing **17** (1988), no. 5, 967–974.

[105] Bradley Efron and Ronald Thisted, *Estimating the number of unseen species: How many words did Shakespeare know?*, Biometrika **63** (1976), no. 3, 435–447.

[106] György Elekes, *A geometric inequality and the complexity of computing volume*, Discrete & Computational Geometry **1** (1986), no. 4, 289–292.

[107] Edward Farhi and Sam Gutmann, *Quantum computation and decision trees*, Physical Review A **58** (1998), no. 2, 915–928, arXiv:quant-ph/9706062.

[108] Edward Farhi and Hartmut Neven, *Classification with quantum neural networks on near term processors*, 2018, arXiv:1802.06002.

[109] Dinei Florencio and Cormac Herley, *A large-scale study of web password habits*, Proceedings of the 16th International Conference on World Wide Web, pp. 657–666, ACM, 2007.

[110] Yoav Freund and Robert E. Schapire, *Adaptive game playing using multiplicative weights*, Games and Economic Behavior **29** (1999), no. 1-2, 79–103.

[111] Alan Frieze and Ravi Kannan, *Log-Sobolev inequalities and sampling from log-concave distributions*, The Annals of Applied Probability **9** (1999), no. 1, 14–26.

[112] Alan Frieze, Ravi Kannan, and Nick Polson, *Sampling from log-concave distributions*, Annals of Applied Probability **4** (1994), no. 3, 812–837.

[113] Alan Frieze, Ravi Kannan, and Santosh Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, Journal of the ACM **51** (2004), no. 6, 1025–1041.

[114] Dan Garber and Elad Hazan, *Approximating semidefinite programs in sublinear time*, Advances in Neural Information Processing Systems, pp. 1080–1088, 2011.

[115] Dan Garber and Elad Hazan, *Almost optimal sublinear time algorithm for semidefinite programming*, 2012, arXiv:1208.5211.

[116] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan, *Escaping from saddle points – online stochastic gradient for tensor decomposition*, Proceedings of The 28th Conference on Learning Theory, Proceedings of Machine Learning Research, vol. 40, pp. 797–842, 2015, arXiv:1503.02101.

[117] Rong Ge, Jason D. Lee, and Tengyu Ma, *Matrix completion has no spurious local minimum*, Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 2981–2989, 2016, arXiv:1605.07272.

[118] Rong Ge, Jason D. Lee, and Tengyu Ma, *Learning one-hidden-layer neural networks with landscape design*, International Conference on Learning Representations, 2018, arXiv:1711.00501.

[119] Rong Ge and Tengyu Ma, *On the optimization landscape of tensor decompositions*, Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 3656–3666, Curran Associates Inc., 2017, arXiv:1706.05598.

[120] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe, *Optimizing quantum optimization algorithms via faster quantum gradient computation*, Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1425–1444, Society for Industrial and Applied Mathematics, 2019, arXiv:1711.00465.

[121] András Gilyén and Tongyang Li, *Distributional property testing in a quantum world*, Proceedings of the 11th Innovations in Theoretical Computer Science Conference, Leibniz International Proceedings in Informatics (LIPIcs), vol. 151, pp. 25:1–25:19, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020, arXiv:1902.00814.

[122] András Gilyén, Seth Lloyd, and Ewin Tang, *Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension*, 2018, arXiv:1811.04909.

[123] 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 Annual ACM SIGACT Symposium on Theory of Computing, pp. 193–204, 2019, arXiv:1806.01838.

[124] Surbhi Goel, Daniel M. Kane, and Adam R. Klivans, *Learning Ising models with independent failures*, Proceedings of the 32nd Conference on Learning Theory, Proceedings of Machine Learning Research, vol. 99, pp. 1449–1469, 2019, arXiv:1902.04728.

[125] Oded Goldreich, *Introduction to property testing*, Cambridge University Press, 2017.

[126] Martin Grötschel, László Lovász, and Alexander Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica **1** (1981), no. 2, 169–197.

[127] Martin Grötschel, László Lovász, and Alexander Schrijver, *Geometric algorithms and combinatorial optimization*, vol. 2, Springer Science & Business Media, 2012.

[128] Lov K. Grover, *A fast quantum mechanical algorithm for database search*, Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 212–219, ACM, 1996, arXiv:quant-ph/9605043.

[129] Lov K. Grover, *Synthesis of quantum superpositions by quantum computation*, Physical Review Letters **85** (2000), no. 6, 1334.

[130] Lov K. Grover, *A different kind of quantum search*, 2005, arXiv:quant-ph/0503205.

[131] Gus Gutoski and Xiaodi Wu, *Parallel approximation of min-max problems with applications to classical and quantum zero-sum games*, Proceedings of the 27th Annual IEEE Symposium on Computational Complexity, pp. 21–31, IEEE, 2012.

[132] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu, *Sample-optimal tomography of quantum states*, Proceedings of the 48th Annual ACM Symposium on Theory of Computing, pp. 913–925, ACM, 2016, arXiv:1508.01797.

[133] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Lynne Stokes, *Sampling-based estimation of the number of distinct values of an attribute*, Proceedings of 21th International Conference on Very Large Data Bases, vol. 95, pp. 311–322, 1995.

[134] Yassine Hamoudi and Frédéric Magniez, *Quantum Chebyshev's inequality and applications*, Proceedings of the 46th International Colloquium on Automata,

Languages, and Programming, Leibniz International Proceedings in Informatics, vol. 132, pp. 69:1–69:16, 2019, arXiv:1807.06456.

[135] Steve Hanneke, *The optimal sample complexity of PAC learning*, Journal of Machine Learning Research **17** (2016), no. 38, 1–15, arXiv:1507.00473.

[136] Moritz Hardt, Tengyu Ma, and Benjamin Recht, *Gradient descent learns linear dynamical systems*, Journal of Machine Learning Research **19** (2018), no. 29, 1–44, arXiv:1609.05191.

[137] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, *Quantum algorithm for linear systems of equations*, Physical Review Letters **103** (2009), no. 15, 150502, arXiv:0811.3171.

[138] Aram W. Harrow, Cedric Yen-Yu Lin, and Ashley Montanaro, *Sequential measurements, disturbance and property testing*, Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1598–1611, SIAM, 2017, arXiv:1607.03236.

[139] Aram W. Harrow and Annie Y. Wei, *Adaptive quantum simulated annealing for Bayesian inference and estimating partition functions*, Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 193–212, SIAM, 2020, arXiv:1907.09965.

[140] Ralph V. L. Hartley, *Transmission of information*, Bell Labs Technical Journal **7** (1928), no. 3, 535–563.

[141] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta, *Supervised learning with quantum-enhanced feature spaces*, Nature **567** (2019), no. 7747, 209–212, arXiv:1804.11326.

[142] Elad Hazan, *Efficient algorithms for online convex optimization and their applications*, Ph.D. thesis, Princeton University, 2006.

[143] Elad Hazan, *Introduction to online convex optimization*, Foundations and Trends® in Optimization **2** (2016), no. 3-4, 157–325.

[144] Monika R. Henzinger, Valerie King, and Valerie King, *Randomized fully dynamic graph algorithms with polylogarithmic time per operation*, Journal of the ACM **46** (1999), no. 4, 502–516.

[145] Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup, *Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity*, Journal of the ACM **48** (2001), no. 4, 723–760.

[146] Lars Hörmander, *The analysis of linear partial differential operators: Distribution theory and Fourier analysis*, Springer-Verlag, 1990.

[147] Jennifer B. Hughes, Jessica J. Hellmann, Taylor H. Ricketts, and Brendan J. M. Bohannan, *Counting the uncountable: statistical approaches to estimating microbial diversity*, Applied and Environmental Microbiology **67** (2001), no. 10, 4399–4406.

[148] Russell Impagliazzo and David Zuckerman, *How to recycle random bits*, 30th Annual Symposium on Foundations of Computer Science, pp. 248–253, IEEE, 1989.

[149] Prateek Jain and Purushottam Kar, *Non-convex optimization for machine learning*, Foundations and Trends® in Machine Learning **10** (2017), no. 3-4, 142–336.

[150] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous, *QIP=PSPACE*, Journal of the ACM **58** (2011), no. 6, 30, arXiv:0907.4737.

[151] Rahul Jain and Penghui Yao, *A parallel approximation algorithm for positive semidefinite programming*, Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science, pp. 463–471, IEEE, 2011, arXiv:1104.2502.

[152] Edwin T. Jaynes, *Information theory and statistical mechanics*, Physical Review **106** (1957), no. 4, 620.

[153] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman, *Minimax estimation of functionals of discrete distributions*, IEEE Transactions on Information Theory **61** (2015), no. 5, 2835–2885, arXiv:1406.6956.

[154] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman, *Maximum likelihood estimation of functionals of discrete distributions*, IEEE Transactions on Information Theory **63** (2017), no. 10, 6774–6798, arXiv:1406.6959.

[155] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan, *How to escape saddle points efficiently*, Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1724–1732, 2017, arXiv:1703.00887.

[156] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M. Kakade, and Michael I. Jordan, *Stochastic gradient descent escapes saddle points efficiently*, 2019, arXiv:1902.04811.

[157] Stephen P. Jordan, *Fast quantum algorithm for numerical gradient estimation*, Physical Review Letters **95** (2005), no. 5, 050501, arXiv:quant-ph/0405146.

[158] Adam T. Kalai and Santosh Vempala, *Simulated annealing for convex optimization*, Mathematics of Operations Research **31** (2006), no. 2, 253–266.

[159] Satyen Kale, *Efficient algorithms using the multiplicative weights update method*, Ph.D. thesis, Princeton University, 2007.

[160] Ravi Kannan, László Lovász, and Miklós Simonovits, *Random walks and an $O^*(n^5)$ volume algorithm for convex bodies*, Random Structures & Algorithms **11** (1997), no. 1, 1–50.

[161] Ashish Kapoor, Nathan Wiebe, and Krysta Svore, *Quantum perceptron models*, Advances in Neural Information Processing Systems 29, pp. 3999–4007, 2016, arXiv:1602.04799.

[162] Narendra Karmarkar, *A new polynomial-time algorithm for linear programming*, Proceedings of the 16th Annual ACM Symposium on Theory of Computing, pp. 302–311, 1984.

[163] Phillip Kaye, Raymond Laflamme, and Michele Mosca, *An introduction to quantum computing*, Oxford University Press, 2007.

[164] James E. Kelley, Jr., *The cutting-plane method for solving convex programs*, Journal of the Society for Industrial and Applied Mathematics **8** (1960), no. 4, 703–712.

[165] Iordanis Kerenidis and Alessandro Luongo, *Quantum classification of the MNIST dataset via slow feature analysis*, 2018, arXiv:1805.08837.

[166] Iordanis Kerenidis and Anupam Prakash, *Quantum gradient descent for linear systems and least squares*, 2017, arXiv:1704.04992.

[167] Iordanis Kerenidis and Anupam Prakash, *Quantum recommendation systems*, Proceedings of the 8th Innovations in Theoretical Computer Science Conference, pp. 49:1–49:21, 2017, arXiv:1603.08675.

[168] Iordanis Kerenidis and Anupam Prakash, *A quantum interior point method for LPs and SDPs*, 2018, arXiv:1808.09266.

[169] Iordanis Kerenidis, Anupam Prakash, and Dániel Szilágyi, *Quantum algorithms for second-order cone programming and support vector machines*, 2019, arXiv:1908.06720.

[170] Leonid G. Khachiyan, *Polynomial algorithms in linear programming*, USSR Computational Mathematics and Mathematical Physics **20** (1980), no. 1, 53–72.

[171] Leonid G. Khachiyan, *On the complexity of computing the volume of a polytope*, Izvestia Akad. Nauk SSSR, Engineering Cybernetics **3** (1988), 216–217.

[172] Leonid G. Khachiyan, *The problem of computing the volume of polytopes is NP-hard*, Uspekhi Mat. Nauk **44** (1989), no. 3, 199–200.

[173] Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J. Yoder, *Hamiltonian simulation with optimal sample complexity*, npj Quantum Information **3** (2017), no. 1, 13, arXiv:1608.00281.

[174] Diederik P. Kingma and Max Welling, *Auto-encoding variational bayes*, 2013, arXiv:1312.6114.

[175] Alexei Yu. Kitaev, Alexander Shen, and Mikhail N. Vyalyi, *Classical and quantum computation*, no. 47, American Mathematical Society, 2002.

[176] Adam Klivans and Raghu Meka, *Learning graphical models using multiplicative weights*, Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science, pp. 343–354, IEEE, 2017, arXiv:1706.06274.

[177] Ian Kroes, Paul W. Lepp, and David A. Relman, *Bacterial diversity within the human subgingival crevice*, Proceedings of the National Academy of Sciences **96** (1999), no. 25, 14547–14552.

[178] Solomon Kullback, *Information theory and statistics*, Courier Corporation, 1997.

[179] Samuel Kutin, *Quantum lower bound for the collision problem with small range*, Theory of Computing **1** (2005), no. 1, 29–36.

[180] Lucien Le Cam, *Asymptotic methods in statistical decision theory*, Springer Science & Business Media, 2012.

[181] James R. Lee, Prasad Raghavendra, and David Steurer, *Lower bounds on the size of semidefinite programming relaxations*, Proceedings of the 47th Annual ACM Symposium on Theory of Computing, pp. 567–576, ACM, 2015, arXiv:1411.6317.

[182] Yin Tat Lee, *personal communication*, 2018.

[183] Yin Tat Lee, Aaron Sidford, and Santosh S. Vempala, *Efficient convex optimization with membership oracles*, Proceedings of the 31st Conference on Learning Theory, Proceedings of Machine Learning Research, vol. 75, pp. 1292–1294, 2018, arXiv:1706.07357.

[184] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong, *A faster cutting plane method and its implications for combinatorial and convex optimization*, Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, pp. 1049–1065, IEEE, 2015, arXiv:1508.04874.

[185] Yin Tat Lee, Zhao Song, and Santosh S. Vempala, *Algorithmic theory of ODEs and sampling from well-conditioned logconcave densities*, 2018, arXiv:1812.06243.

[186] Yin Tat Lee and Santosh S. Vempala, *Eldan's stochastic localization and the KLS hyperplane conjecture: An improved lower bound for expansion*, Proceedings of the 58th Annual Symposium on Foundations of Computer Science, pp. 998–1007, 2017, arXiv:1612.01507.

[187] Yin Tat Lee and Santosh S. Vempala, *Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation*, Proceedings of the 50th Annual Symposium on Theory of Computing, pp. 1115–1121, 2018, arXiv:1710.06261.

[188] Yin Tat Lee and Santosh S. Vempala, *The Kannan-Lovász-Simonovits conjecture*, 2018, arXiv:1807.03465.

[189] Reut Levi, Dana Ron, and Ronitt Rubinfeld, *Testing properties of collections of distributions*, Theory of Computing **9** (2013), no. 1, 295–347.

[190] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer, *Markov chains and mixing times*, American Mathematical Society, 2017.

[191] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu, *Sublinear quantum algorithms for training linear and kernel-based classifiers*, International Conference on Machine Learning, pp. 3815–3824, 2019, arXiv:1904.02276.

[192] Tongyang Li and Xiaodi Wu, *Quantum query complexity of entropy estimation*, IEEE Transactions on Information Theory **65** (2019), no. 5, 2899–2921, arXiv:1710.06025, © 2019 IEEE.

[193] Richard J. Lipton and Robert E. Tarjan, *Applications of a planar separator theorem*, Proceedings of the 18th Annual Symposium on Foundations of Computer Science, pp. 162–170, IEEE, 1977.

[194] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost, *Quantum algorithms for supervised and unsupervised machine learning*, 2013, arXiv:1307.0411.

[195] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost, *Quantum principal component analysis*, Nature Physics **10** (2014), no. 9, 631, arXiv:1307.0401.

[196] László Lovász, *Hit-and-run mixes fast*, Mathematical Programming **86** (1999), no. 3, 443–461.

[197] László Lovász and Miklós Simonovits, *The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume*, Proceedings of the 31st Annual Symposium on Foundations of Computer Science, pp. 346–354, 1990.

[198] László Lovász and Miklós Simonovits, *Random walks in a convex body and an improved volume algorithm*, Random Structures & Algorithms **4** (1993), no. 4, 359–412.

[199] László Lovász and Santosh Vempala, *Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 57–68, 2006.

[200] László Lovász and Santosh Vempala, *Hit-and-run from a corner*, SIAM Journal on Computing **35** (2006), no. 4, 985–1005.

[201] László Lovász and Santosh Vempala, *Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm*, Journal of Computer and System Sciences **72** (2006), no. 2, 392–417.

[202] László Lovász and Santosh Vempala, *The geometry of logconcave functions and sampling algorithms*, Random Structures Algorithms **30** (2007), no. 3, 307–358.

[203] Guang Hao Low and Isaac L. Chuang, *Hamiltonian simulation by qubitization*, Quantum **3** (2019), 163, arXiv:1610.06546.

[204] Michael Luby and Noam Nisan, *A parallel approximation algorithm for positive linear programming*, Proceedings of the 25th Annual ACM Symposium on Theory of Computing, pp. 448–457, ACM, 1993.

[205] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha, *Search via quantum walk*, SIAM Journal on Computing **40** (2011), no. 1, 142–164, arXiv:quant-ph/0608026.

[206] Marvin Minsky and Seymour A. Papert, *Perceptrons: An introduction to computational geometry*, MIT Press, 1988.

[207] John E. Mitchell, *Polynomial interior point cutting plane methods*, Optimization Methods and Software **18** (2003), no. 5, 507–534.

[208] Ashley Montanaro, *Quantum speedup of Monte Carlo methods*, Proceedings of the Royal Society A **471** (2015), no. 2181, 20150301, arXiv:1504.06987.

[209] Ashley Montanaro, *The quantum complexity of approximating the frequency moments*, Quantum Information & Computation **16** (2016), no. 13&14, 1169–1190, arXiv:1505.00113.

[210] Ashley Montanaro and Ronald de Wolf, *A survey of quantum property testing*, Theory of Computing (2016), 1–81, arXiv:1310.2035.

[211] Daniel Nagaj, Pawel Wocjan, and Yong Zhang, *Fast amplification of QMA*, Quantum Information & Computation **9** (2009), no. 11, 1053–1068, arXiv:0904.1549.

[212] 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, pp. 384–393, 1999, arXiv:quant-ph/9804066.

[213] Arkadi S. Nemirovski, *Information-based complexity of convex programming*, lecture notes, 1995.

[214] Arkadi S. Nemirovski and David B. Yudin, *Problem complexity and method efficiency in optimization*, Wiley, 1983.

[215] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, Applied Optimization, vol. 87, Springer, 2013.

[216] Yurii Nesterov and Arkadi Nemirovsky, *Conic formulation of a convex programming problem and duality*, Optimization Methods and Software **1** (1992), no. 2, 95–115.

[217] Michael A. Nielsen and Isaac L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.

[218] Albert B. Novikoff, *On convergence proofs for perceptrons*, Proceedings of the Symposium on the Mathematical Theory of Automata, vol. 12, pp. 615–622, 1963.

[219] Ryan O'Donnell and John Wright, *Quantum spectrum testing*, Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, pp. 529–538, ACM, 2015, arXiv:1501.05028.

[220] Ryan O'Donnell and John Wright, *Efficient quantum tomography*, Proceedings of the 48th Annual ACM Symposium on Theory of Computing, pp. 899–912, ACM, 2016, arXiv:1508.01907.

[221] Ryan O'Donnell and John Wright, *Efficient quantum tomography II*, Proceedings of the Forty-ninth Annual ACM SIGACT Symposium on Theory of Computing, pp. 962–974, ACM, 2017, arXiv:1612.00034.

[222] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld, *A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size*, Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1123–1131, Society for Industrial and Applied Mathematics, 2012, arXiv:1110.1079.

[223] Alon Orlitsky, Ananda Theertha Suresh, and Yihong Wu, *Optimal prediction of the number of unseen species*, Proceedings of the National Academy of Sciences **113** (2016), no. 47, 13283–13288.

[224] Davide Orsucci, Hans J. Briegel, and Vedran Dunjko, *Faster quantum mixing for slowly evolving sequences of Markov chains*, Quantum **2** (2018), 105, arXiv:1503.01334.

[225] Liam Paninski, *Estimation of entropy and mutual information*, Neural Computation **15** (2003), no. 6, 1191–1253.

[226] Liam Paninski, *Estimating entropy on m bins given fewer than m samples*, IEEE Transactions on Information Theory **50** (2004), no. 9, 2200–2203.

[227] Liam Paninski, *A coincidence-based test for uniformity given very sparsely sampled discrete data*, IEEE Transactions on Information Theory **54** (2008), no. 10, 4750–4755.

[228] Bruce J. Paster, Susan K. Boches, Jamie L. Galvin, Rebecca E. Ericson, Carol N. Lau, Valerie A. Levanos, Ashish Sahasrabudhe, and Floyd E. Dewhirst, *Bacterial diversity in human subgingival plaque*, Journal of Bacteriology **183** (2001), no. 12, 3770–3783.

[229] David Poulin and Pawel Wocjan, *Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer*, Physical Review Letters **103** (2009), no. 22, 220502, arXiv:0905.2199.

[230] John Preskill, *Quantum computing in the NISQ era and beyond*, Quantum **2** (2018), 79, arXiv:1801.00862.

[231] Luis Rademacher and Santosh Vempala, *Dispersion of mass and the complexity of randomized geometric algorithms*, Advances in Mathematics **219** (2008), no. 3, 1037–1069, arXiv:cs/0608054.

[232] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd, *Quantum support vector machine for big data classification*, Physical Review Letters **113** (2014), no. 13, 130503, arXiv:1307.0471.

[233] Patrick Rebentrost, Maria Schuld, Leonard Wossnig, Francesco Petruccione, and Seth Lloyd, *Quantum gradient descent and Newton's method for constrained polynomial optimization*, 2016, arXiv:1612.01789.

[234] Alfréd Rényi, *On measures of entropy and information*, Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 547–561, 1961.

[235] Peter C. Richter, *Almost uniform sampling via quantum walks*, New Journal of Physics **9** (2007), no. 3, 72, arXiv:quant-ph/0606202.

[236] Peter C. Richter, *Quantum speedup of classical mixing processes*, Physical Review A **76** (2007), no. 4, 042306, arXiv:quant-ph/0609204.

[237] Dana Ron, *Algorithmic and analysis techniques in property testing*, Foundations and Trends in Theoretical Computer Science **5** (2010), no. 2, 73–205.

[238] Mark Rudelson, *Random vectors in the isotropic position*, Journal of Functional Analysis **164** (1999), no. 1, 60–72, arXiv:math/9608208.

[239] Walter Rudin, *Principles of mathematical analysis*, vol. 3, McGraw-Hill, 1964.

[240] Ankan Saha, S.V.N. Vishwanathan, and Xinhua Zhang, *New approximation algorithms for minimum enclosing convex shapes*, Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1146–1160, Society for Industrial and Applied Mathematics, 2011, arXiv:0909.1062.

[241] Jun John Sakurai and Jim Napolitano, *Modern quantum mechanics*, Pearson Harlow, 2014.

[242] Bernhard Schölkopf and Alexander J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT Press, 2002.

[243] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione, *An introduction to quantum machine learning*, Contemporary Physics **56** (2015), no. 2, 172–185, arXiv:1409.3097.

[244] Claude E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.

[245] Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Review **41** (1999), no. 2, 303–332, arXiv:quant-ph/9508027.

[246] Robert L. Smith, *Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions*, Operations Research **32** (1984), no. 6, 1296–1308.

[247] Shay Solomon, *Fully dynamic maximal matching in constant update time*, Proceedings of the 57th Annual Symposium on Foundations of Computer Science, pp. 325–334, IEEE, 2016, arXiv:1604.08491.

[248] Rolando D. Somma, Sergio Boixo, and Howard Barnum, *Quantum simulated annealing*, 2007, arXiv:0712.1008.

[249] Rolando D. Somma, Sergio Boixo, Howard Barnum, and Emanuel Knill, *Quantum simulations of classical annealing processes*, Physical Review Letters **101** (2008), no. 13, 130504, arXiv:0804.1571.

[250] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, *Optimization for machine learning*, MIT Press, 2012.

[251] Daniel Štefankovič, Santosh Vempala, and Eric Vigoda, *Adaptive simulated annealing: a near-optimal connection between sampling and counting*, Journal of the ACM **56** (2009), no. 3, 18, arXiv:cs/0612058.

[252] Ruoyu Sun, *Optimization for deep learning: theory and algorithms*, 2019, arXiv:1912.08957.

[253] Johan A.K. Suykens and Joos Vandewalle, *Least squares support vector machine classifiers*, Neural Processing Letters **9** (1999), no. 3, 293–300.

[254] Mario Szegedy, *Quantum speed-up of Markov chain based algorithms*, Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 32–41, 2004.

[255] Ewin Tang, *Quantum-inspired classical algorithms for principal component analysis and supervised clustering*, 2018, arXiv:1811.00414.

[256] Ewin Tang, *A quantum-inspired classical algorithm for recommendation systems*, Proceedings of the 51st Annual ACM Symposium on Theory of Computing, ACM, 2019, arXiv:1807.04271.

[257] Kristan Temme, Tobias J. Osborne, Karl G. Vollbrecht, David Poulin, and Frank Verstraete, *Quantum Metropolis sampling*, Nature **471** (2011), no. 7336, 87–90, arXiv:0911.3635.

[258] Ronald Thisted and Bradley Efron, *Did Shakespeare write a newly-discovered poem?*, Biometrika **74** (1987), no. 3, 445–455.

[259] Mikkel Thorup, *Worst-case update times for fully-dynamic all-pairs shortest paths*, Proceedings of the 37th Annual ACM Symposium on Theory of Ccomputing, pp. 112–119, ACM, 2005.

[260] Salil P. Vadhan, *Pseudorandomness*, Foundations and Trends® in Theoretical Computer Science **7** (2012), no. 1–3, 1–336.

[261] Pravin M. Vaidya, *A new algorithm for minimizing convex functions over convex sets*, Proceedings of the 30th Annual Symposium on Foundations of Computer Science, pp. 338–343, 1989.

[262] Gregory Valiant and Paul Valiant, *Estimating the unseen: an $n/log(n)$-sample estimator for entropy and support size, shown optimal via new CLTs*, Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 685–694, ACM, 2011.

[263] Gregory Valiant and Paul Valiant, *The power of linear estimators*, 52nd Annual Symposium on Foundations of Computer Science, pp. 403–412, IEEE, 2011.

[264] Leslie G. Valiant, *A theory of the learnable*, Communications of the ACM **27** (1984), no. 11, 1134–1142.

[265] Paul Valiant, *Testing symmetric properties of distributions*, SIAM Journal on Computing **40** (2011), no. 6, 1927–1968.

[266] Paul C. van Oorschot and Michael J. Wiener, *Parallel collision search with cryptanalytic applications*, Journal of Cryptology **12** (1999), no. 1, 1–28.

[267] Lieven Vandenberghe and Stephen Boyd, *Semidefinite programming*, SIAM Review **38** (1996), no. 1, 49–95.

[268] Vladimir N. Vapnik and Alexey Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability & Its Applications **16** (1971), no. 2, 264–280.

[269] Santosh Vempala, *Geometric random walks: a survey*, Combinatorial and Computational Geometry (Jacob E. Goodman, János Pach, and Emo Welzl, eds.), Mathematical Sciences Research Institute Publications, vol. 52, MSRI, 2005, pp. 573–612.

[270] Nathan Wiebe, Daniel Braun, and Seth Lloyd, *Quantum algorithm for data fitting*, Physical Review Letters **109** (2012), no. 5, 050505, arXiv:1204.5242.

[271] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore, *Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning*, Quantum Information & Computation **15** (2015), no. 3-4, 316–356, arXiv:1401.2142.

[272] Pawel Wocjan and Anura Abeyesinghe, *Speedup via quantum sampling*, Physical Review A **78** (2008), no. 4, 042336, arXiv:0804.4259.

[273] Pawel Wocjan, Chen-Fu Chiang, Daniel Nagaj, and Anura Abeyesinghe, *Quantum algorithm for approximating partition functions*, Physical Review A **80** (2009), no. 2, 022340, arXiv:0811.0596.

[274] Xiaodi Wu, *Parallelized solution to semidefinite programmings in quantum complexity theory*, 2010, arXiv:1009.2211.

[275] Yihong Wu and Pengkun Yang, *Minimax rates of entropy estimation on large alphabets via best polynomial approximation*, IEEE Transactions on Information Theory **62** (2016), no. 6, 3702–3720, arXiv:1407.0381.

[276] Yihong Wu and Pengkun Yang, *Chebyshev polynomials, moment matching, and optimal estimation of the unseen*, The Annals of Statistics **47** (2019), no. 2, 857–883, arXiv:1504.01227.

[277] Man-Hong Yung and Alán Aspuru-Guzik, *A quantum–quantum Metropolis algorithm*, Proceedings of the National Academy of Sciences **109** (2012), no. 3, 754–759, arXiv:1011.1468.

[278] Chi Zhang, *An improved lower bound on query complexity for quantum PAC learning*, Information Processing Letters **111** (2010), no. 1, 40–45.

[279] Martin Zinkevich, *Online convex programming and generalized infinitesimal gradient ascent*, Proceedings of the 20th International Conference on Machine Learning, pp. 928–936, 2003.