Learning Models in Quantum Computation and Quantum Control

Reevu Maity

St Catherine's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2020

Acknowledgements

I would like to thank my advisors Vlatko Vedral and Seth Lloyd for their support and guidance during the entire course of my DPhil studies. I thank Bobak Toussi Kiani, Srinivasan Arunachalam, Milad Marvian, Giacomo de Palma, Zi-Wen Liu, Benjamin Yadin, Felix Tennie, Christian Schilling, Davide Girolami and Tristan Farrow for helpful discussions. I thank Seth Lloyd for hosting me at Massachusetts Institute of Technology as a visiting student for a year where a substantial part of this thesis was completed. Finally and most importantly, I thank my family for supporting and believing in me throughout my DPhil studies.

Abstract

We investigate learning models for implementing arbitrary unitary operations and perform quantum machine learning tasks. First, we discuss efficient constructions for building unitary transformations in $U(d)$, thus providing upper bounds on the minimum time required to implement them. Second, we study the hardness of learning unitary operations in $U(d)$ via gradient descent on the time parameters of an alternating operator sequence, and numerically find that, despite the non-convexity of the loss landscape, gradient descent always converges to the target unitary when the sequence contains d^2 or more parameters. Third, we introduce the first quantum boosting algorithm for converting a weak quantum learner into a strong accurate quantum learner for a Boolean concept class $\mathcal{C}$; thereby achieving a quadratic quantum improvement over classical AdaBoost in terms of $\mathsf{VC}(\mathcal{C})$.

Contents

Chapter 1

Introduction

In recent years, machine learning has been applied in a wide range of areas including the fields of quantum computation and quantum control. Machine learning algorithms have been used to classify the phases of quantum matter [CM17], predict the energies and properties of molecules and solids in chemistry [BDC+18], characterize the landscape of string theories [CHKN17], and learn about the underlying waveform from a gravitational wave observed by LIGO [SGHZ19]. For a more comprehensive discussion on the exciting results of studying physics using machine learning, we recommend the review article by Carleo et al. [CCC+19]. Some of these works provide heuristic approaches to achieve an exponential speedup over classical machine learning algorithms. Indeed, one has to be careful about the assumptions required to develop efficient quantum machine learning algorithms. A brief but concise explanation of the potential issues has been provided by Aaronson [Aar15]. This encourages a thorough understanding of the advantages and limitations of learning models in quantum computation and quantum control. An interesting starting point for investigating the applications of learning models in tackling problems in physics is statistical learning theory. The techniques developed by statistical learning theory [FHT01] provides a mathematically rigorous definition of what it means to efficiently learn a desired function from labelled examples. In other words, what is the computational complexity of the learning model? How much data is required to learn the target function with a desired accuracy? Understanding these questions will be the central subject of interest of this thesis.

It is well known that quantum mechanics can produce unusual patterns in data [HG01]. An interesting property of classical machine learning models is they

can realize statistical patterns in data and generate new data which have the same statistical properties. This suggests if a quantum device can produce statistical patterns that are computationally difficult to be produced by a classical computer, then it can also realize patterns in data that might not be efficiently possible using classical algorithms. Furthermore, learning models that use data provided by a coherent quantum process can promise to efficiently solve computational tasks which might not be possible using classical resources. Building on these observations, the goal of this thesis is to advance our understanding of learning models for studying quantum physical systems by opening three directions of research:

1. A constructive and efficient method for building any unitary transformation using sequences of elementary quantum logic operations (continuous) or applications of control fields.

2. The study of the hardness of learning unitary transformations by performing gradient descent on the time parameters of sequences of alternating operators.

3. The design of a quantum boosting algorithm that can convert a weak and inaccurate quantum learner into a strong accurate quantum learner.

We discuss the main contributions of the thesis below.

1.1 Summary of our contributions

In Chapter 2, we present a brief but concise review of the basic concepts from learning theory, quantum computation and quantum control. In Chapter 3, we provide upper bounds on the minimum time required to implement a desired unitary transformation when applying sequences of Hamiltonian transformations. In Chapter 4, we discuss a simple algorithm for learning unitary transformations. Here we numerically analyze the hardness of obtaining the optimal control parameters in an alternating operator sequence for learning arbitrary unitary operations using gradient descent optimization. Furthermore, we also numerically investigate the learnability of low depth unitaries. In Chapter 5, we introduce a quantum boosting algorithm that achieves a quadratic improvement over classical AdaBoost in terms of an important combinatorial parameter. The main results of these chapters are summarised in Section 1.1.1, Section 1.1.2 and Section 1.1.3. In Section 1.1.4, we provide a statement of authorship. Finally, in Chapter 6, we

discuss the future directions of research based on this thesis and remark about the outlook of learning models to study problems in physics.

1.1.1 Constructive methods to implement a unitary transformation

One of the central tasks of quantum information processing is to implement a desired unitary operation for performing a quantum computation. A necessary and sufficient condition to build any unitary operation is that the algebra generated by a set of Hamiltonians via commutation is complete in the Lie algebra. In the gate based quantum computation model, this translates to the fact that the elementary gates from a universal gate set can build an arbitrary unitary operation. The Solovay-Kitaev theorem [Kit97] says that a sequence of one and two qubit gates from a universal discrete gate set can efficiently approximate any unitary operation. In particular, it shows that an approximating sequence of length $O(d^2 \cdot \text{polylog}(1/\varepsilon))$ can reach any unitary in the manifold $U(d)$ with accuracy ε.

In Chapter 3, we provide upper bounds on the optimal time required to implement an arbitrary unitary operation in $U(d)$. Additionally, we also solve the problem of whether one can find the correct sequence of logic gates (continuous) or control fields which can be applied to build the desired unitary operation. Our work can be understood as a continuous version of the Solovay-Kitaev algorithm where we are interested in the continuous gate based model of quantum computation. In contrast to the Solovay-Kitaev algorithm, our approach can be applied to construct arbitrary unitaries in any dimension.

We present two methods – the non-infinitesimal and infinitesimal constructions – for building a desired unitary transformation. The infinitesimal construction employs nested commutators to build a unitary transformation in the vicinity of the identity. We avail a different strategy for the non-infinitesimal approach. In the non-infinitesimal method, we first move away from the identity unitary, perform a set of operations and then return in the vicinity of the identity. This sequence of steps allow us to move along any direction in the unitary manifold and hence approximate any unitary transformation with a desired accuracy ε. The time complexity of the non-infinitesimal construction is $O(d^2/\varepsilon)$ while it is $O(d^2 \log d/\varepsilon)$ for the infinitesimal approach. We present numerical evidence that supports the conjecture that one can construct a desired unitary operation in d-dimensions by appropriately choosing $O(d^2)$ parameters in an alternating operator sequence.

1.1.2 Learning unitary operations using gradient descent

In quantum information processing, a comprehensively studied topic is the controllability of quantum systems where mathematically rigorous conditions for controllability have been established [RHR04, KRK$^+$05, RHR05, RHH$^+$06, CR07, MHR08, HDR09]. In Section 1.1.1, we mentioned two approaches for constructing an arbitrary unitary with sequences of the optimal length.

In Chapter 4, we aim to solve the hardness of learning the desired quantum controls in an alternating operator sequence that is necessary to perform an arbitrary unitary operation in d dimensions. The task is to find the optimal set of quantum controls in an alternating operator sequence using gradient descent optimization on the parameter space. Due to the complexity of the nature of the parameter landscape, analytical solutions for the control parameters can be very difficult to obtain: this is true even for simple cases where there are only a few parameters in the landscape. While multiple theoretical results about the necessary and sufficient conditions for controllability have been established [RHR04, KRK$^+$05, RHR05, RHH$^+$06], the parameter landscape is still not well understood.

First, we consider the under-parameterized case (i.e., less than d^2 parameters) where the learning process is not expected to converge to the target unitary operator and investigate the rate of convergence of the learning process to the suboptimal unitary via gradient descent. Next, we study the over-parameterized sequences where the number of control parameters is greater than d^2. Similar to the under-parameterized case, we ask how hard it is to obtain the learning sequences. In particular, we are interested in the existence of local traps and saddle points in the parameter landscape where the gradient descent based learning process can get stuck with a high probability and thus it fails to obtain the correct sequence of controls to learn the target unitary. The existence of such local traps have been observed in studies where the control problem contains a drift Hamiltonian [RHR05]. In the absence of a drift term, we find that, surprisingly, gradient descent exponentially converges to the optimal solution in the control landscape when the alternating operator sequence is over-parameterized. When the number of control parameters is exactly d^2, gradient descent converges to the solution following a power law rate of convergence. In comparison to the results of Chapter 3, the results of Chapter 4 indicate that the number of parameters required to attain

any unitary in $U(d)$ using an alternating operator sequence is not only $O(d^2)$, it is *exactly* d^2.

Due to the complex and multifaceted nature of the control landscape, we emphasize that the above problem is very difficult to prove analytically in general. However, we have attacked the problem numerically and obtained novel and unexpected results (which would not be possible to find analytically). We believe that our numerical results will foster further research and discussions.

1.1.3 Boosting weak quantum learners

A fundamental problem in the PAC learning model is whether a weak learning algorithm can be converted into a strong learning algorithm. A weak learner is one which performs slightly better than random guessing while a strong learner performs well on almost every inputs. This problem was conclusively settled by the AdaBoost algorithm of Freund and Schapire [FS97]. The AdaBoost algorithm is one of the few classical machine learning algorithms that was simple enough to have a profound impact in both theory and practice, with applications ranging from statistics, game theory, optimization, computer vision and speech recognition [SF12]. Freund and Schapire's boosting algorithm efficiently solves the following: suppose we are given a weak learner as a black-box, then the AdaBoost algorithm can boost the performance of the black-box such that it acts as a strong learner.

In the past decade, the field of quantum machine learning (QML) has seen a lot of progress by developing algorithms for various classical and quantum learning tasks. QML has given us quantum improvements to machine learning tasks such as support vector machines [RML14], linear algebra [Pra14], clustering [ABG07, KLLP19]. Several quantum algorithms have been developed for learning quantum states in various settings [Aar07, Aar18, Roc18, Yog19] and learning Boolean-valued concept classes [BV93, BJ99, AS05, ACL$^+$19]. We remark that all of the above QML algorithms promise to perform well on an ideal quantum computer. To be precise, suppose a QML algorithm is designed to theoretically perform well on a fault tolerant quantum computer. However, the performance of the algorithm can be weak when implemented on a noisy quantum computer. We ask, can the performance of a weak quantum learner be improved such that it performs well on more than 2/3 of the inputs?

In Chapter 5, we provide a positive answer to the above question. We introduce a quantum boosting algorithm that can convert a weak QML algorithm into a strong and accurate quantum learning algorithm. Our quantum boosting algorithm performs quadratically better than AdaBoost in terms of the VC dimension of the concept class. Furthermore, we discuss a *robust AdaBoost* algorithm which can be faster than standard AdaBoost or the multiplicative weight update method.

1.1.4 Statement of authorship

This thesis builds on the following papers:

1. [LM19] *Efficient implementation of unitary transformations*, arXiv preprint arXiv:1901.03431 (2019). With S. Lloyd.
2. [KLM20] *Learning Unitaries by Gradient Descent*, arXiv preprint arXiv:2001.11897v3 (2020). With B. T. Kiani and S. Lloyd.
3. [qua] *Quantum Boosting*, arXiv preprint arXiv:2002.05056 (2020). With S. Arunachalam (Accepted for publication in ICML 2020).

I have also co-authored the following articles during my D.Phil. which are not included in this thesis :

1. [ZYH$^+$17] *Detecting metrologically useful asymmetry and entanglement by a few local measurements*, Physical Review A. Vol. 96, No. 4 (2017). With C. Zhang, B. Yadin, Z.-B. Hou and others.
2. [MRS17] *Black hole phase transitions and the chemical potential*, Physics Letters B, Vol. 765 (2017). With P. Roy and T. Sarkar.

Chapter 2

Preliminaries

In this chapter, we review some standard concepts from the theory of learning and the theory of quantum computation. We discuss here some notions that are relevant for a general presentation and defer the introduction of specific topics in the background section of a particular chapter.

The reader who is familiar with the basic concepts of machine learning theory, quantum mechanics, and quantum algorithms can proceed directly to the relevant chapters, referring back to this chapter as required to review the background concepts. A comprehensive discussion on quantum computation can be found in the books by Nielsen and Chuang [NC16] and Wolf [Wol19]. The interested reader is referred to the book by Kearns and Vazirani [KV94] for a detailed introduction to learning theory.

Throughout this thesis, we will make use of standard definitions and results from group theory and classical machine learning. The relevant topics from group theory have been discussed in Appendix A.

2.1 Notation

Let $[b]$ denote the set $\{1,\ldots,b\}$ for an integer b. We use the lower-case letter with an arrow $\vec{a}$ to represent a vector in $\mathbb{R}^n$. The i-th element of the vector $\vec{a}$ is denoted by a_i. The standard norms will be used in this thesis. For convenience of the reader, we mention the norms that are used frequently in the thesis. Let $\vec{x} = \{x_1,\ldots,x_n\}$. The ℓ_1 norm is defined as $\|\vec{x}\|_1 = \sum_{i\in[n]}|x_i|$, the ℓ_2 norm is $\|\vec{x}\|_2 = \sqrt{\sum_{i\in[n]} x_i^2}$ and the ℓ_∞ norm is $\|\vec{x}\|_\infty = \max_{i\in[n]}|x_i|$.

Given a set S, the indicator function $[\cdot] : S \to \{0,1\}$ satisfies the following. The function outputs $[x] = 0$ if $x \notin S$ and $[x] = 1$ if $x \in S$. We use the notation Pr to denote probability. Let E be an event. The probability of occurrence of an event E is given by $\Pr[E]$. Let D be a distribution over all elements of the set S such that $\sum_{x \in S} D_x = 1$. The notation $x \sim D$ implies that the element x is sampled from the distribution D over the set S.

It is often useful to understand the asymptotic behaviour of a function, i.e., how does a function behave as its argument increases or decreases. Let $x \in \mathbb{R}$, $p : \mathbb{R} \to \mathbb{R}^+$ and $q : \mathbb{R} \to \mathbb{R}^+$. The $O(\cdot)$ notation specifies that the asymptotic scaling of the function $p(x)$ is upper bounded by $q(x)$ up to a constant factor and it is denoted by $p(x) = O(q(x))$. If the bound is not tight and $p(x) < q(x)$ for every $x \in \mathbb{R}$, then the asymptotic scaling is denoted by $p(x) = o(q(x))$. In a similar vein, the notation $p(x) = \Omega(q(x))$ denotes that the asymptotic scaling of the function p is strictly lower bounded by q. As before, if the bound is not tight, then we use $p(x) = w(q(x))$. When the upper and lower bounds on the asymptotic scaling of p is the same and is given by q, then the notation $f(x) = \Theta(g(x))$ is used. Furthermore, the notations $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide poly-logarithmic factors.

2.2 Learning theory

In this section, we review the standard literature on statistical learning in order to provide the background necessary for Chapter 5. Let $\mathcal{X}$ denote the *input space* and $\mathcal{Y}$ denote the *output space*. Let $\mathcal{D}$ denote a probability distribution over the space $\mathcal{X} \times \mathcal{Y}$. In this thesis, we restrict ourselves to binary classification where the set $\mathcal{Y} = \{-1, 1\}$. We assume that the examples $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are drawn from an *unknown* probability distribution $\mathcal{D}$. Let $\mathcal{T}$ be a training set which is a sequence $\mathcal{T} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of m i.i.d. pairs sampled from the distribution $\mathcal{D}$.

There are many frameworks of learning; the simplest of which is *supervised learning* where a learning algorithm $\mathcal{A}$ is trained on labelled examples in the set $\mathcal{T}$ and returns a *hypothesis* function h. The set of functions $\mathcal{H} = \{h_i : \mathcal{X} \to \mathcal{Y}\}_i$ that can be returned by the learning algorithm $\mathcal{A}$ is termed as the *hypothesis space*. The task of the learning algorithm is to output a hypothesis h that performs well on the labelled examples in $\mathcal{X} \times \mathcal{Y}$ outside the training set. In other words, the hypothesis h predicts the label $y \in \mathcal{Y}$ of an example $x \notin \mathcal{T}$ by optimizing a given loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that measures the distance of $h(x)$ from the true label y.

The goal of finding the best hypothesis h when the labels in $\mathcal{Y}$ are discrete is known as *classification*. Similarly, when the label space $\mathcal{Y}$ is continuous, the task is called *regression*. For a Boolean label space $\mathcal{Y}$, a standard loss function is the 0-1 loss given by $\mathcal{L}(h(x), y) = [h(x) \neq y]$ (Note that $[\cdot]$ is the indicator function). When the label space $\mathcal{Y} \in \mathbb{R}$ as in regression tasks, a quadratic loss function $\mathcal{L}(h(x), y) = (h(x) - y)^2$ is the most conventional. In order to find the optimal hypothesis h (for simplicity, we consider the 0-1 loss), the learning algorithm $\mathcal{A}$ aims to optimize the true risk or *generalization error* given by

$$R(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]. \tag{2.1}$$

Since the distribution $\mathcal{D}$ is unknown to the learner, it is not possible to directly measure the true generalisation error. One widely accepted resolution for optimizing the generalization error (which works quite well under some computational complexity assumptions) is to optimize an empirical estimate called the *empirical error* or training error: $\widehat{R}(h) = (1/m) \cdot \sum_{(x,y)\in\mathcal{T}} \mathcal{L}(h(x), y) = (1/m) \cdot \sum_{(x,y)\in\mathcal{T}} [h(x) \neq y]$.

We now review some common strategies to optimize both the empirical and generalisation errors. The first strategy is a learning model known as *empirical risk minimization* (ERM) where the goal of the learner is to output a hypothesis h_0 that optimizes the empirical error $\widehat{R}(h)$

$$h_0 = \arg\min_{h\in\mathcal{H}} \widehat{R}(h) = \arg\min_{h\in\mathcal{H}} \frac{1}{m} \sum_{(x,y)\in\mathcal{T}} \mathcal{L}(h(x), y). \tag{2.2}$$

A fundamental result in statistical learning theory by Vapnik and Chervonenkis [VC15] states that for a sufficiently large size of the training set m, it is possible to show that the empirical risk minimization will output a hypothesis h_0 that also minimizes the true risk or the generalization error.

One immediate issue with this line of approach is when the hypothesis space $\mathcal{H}$ is large, it is likely that ERM outputs a hypothesis that performs well on the training set but has poor generalization error. This phenomenon is known as *overfitting* by the hypothesis h_0. A potential resolution to this issue is to restrict the size of the hypothesis space $\mathcal{H}$ and a common practice in statistical learning theory is to measure the complexity of a Boolean hypothesis space $\mathcal{H}$ by a combinatorial parameter called the VC dimension. To be precise, with high probability, the ERM analysis bounds the difference between the empirical error and the generalization error given by $|R(h_0) - \widehat{R}(h_0)| \leq O(\sqrt{\mathrm{VC}/m})$. Thus, when the size of the training

set $m = O(VC)$ and $\widehat{R}(h_0) = 0$, then with high probability, we have the true error $R(h_0) \sim O(1)$.

Another strategy to prevent overfitting is called *regularization*. In this learning approach, the trick is to select a large hypothesis space $\mathcal{H}$ (the size of the hypothesis space can be infinite for continuous functions) and introduce a *regularizer*, which is usually the norm $\| h \|$. Then the *regularized empirical error* is optimized to obtain

$$h_0 = \arg\min_{h \in \mathcal{H}} \widehat{R}(h) + \lambda \| h \|^2. \tag{2.3}$$

Here λ is a free parameter called the *regularization parameter* which balances the trade-off between the computational complexity and the fit. In general, finding the appropriate λ is a difficult problem and it can vary with different data sets.

2.2.1 PAC Learning

The quantum AdaBoost algorithm presented in Chapter 5 is based on the classical technique of Probably Approximately Correct learning. The Probably Approximately Correct (PAC) model of learning was introduced by Valiant [Val84] in 1984. A concept class $\mathcal{C}$ is a collection of concepts. Often, $\mathcal{C}$ is composed of a subclass of functions $\{\mathcal{C}_n\}_{n \geq 1}$, i.e., $\mathcal{C} = \cup_{n \geq 1} \mathcal{C}_n$, where $\mathcal{C}_n$ is a collection of Boolean functions $c : \{0,1\}^n \to \{-1,1\}$, which are often referred to as *concepts*. In the PAC learning model, a learner $\mathcal{A}$ is given $n \geq 1$ and access to *labelled examples* $(x, c(x))$ where $(x, c(x))$ is drawn according to the unknown distribution $\mathcal{D} : \{0,1\}^n \to [0,1]$ and $c \in \mathcal{C}_n$ is the *unknown* target concept (which the learner is trying to learn). The goal of $\mathcal{A}$ is to output a hypothesis $h : \{0,1\}^n \to \{-1,1\}$ that is η-close to c under $\mathcal{D}$. We say that $\mathcal{A}$ is an (η, δ)-*PAC learner* for a concept class $\mathcal{C}$ if it satisfies:

> for every $n \geq 1$, $c \in \mathcal{C}_n$ and distributions $\mathcal{D}$, $\mathcal{A}$ takes as input n, δ, η and labelled examples $(x, c(x))$ and with probability $\geq 1 - \delta$, $\mathcal{A}$ outputs a hypothesis h such that
>
> $$\Pr_{x \sim \mathcal{D}} [h(x) \neq c(x)] \leq \eta.$$

The sample complexity and time complexity of a learner is the number of labelled examples and number of bit-wise operations (i.e., time taken) that suffices to learn $\mathcal{C}$ (under the hardest concept $c \in \mathcal{C}$ and distribution $\mathcal{D}$).

We assume that all concept classes $\mathcal{C}$ are defined as $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$ where $\mathcal{C}_n \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$ (in fact from here onwards, we will assume that $\mathcal{C}_n$ is always a subset of $\{c : \{0,1\}^n \to \{-1,1\}\}$ and do not explicitly mention it).

In the quantum PAC model, a learner is a *quantum algorithm* given access to the quantum examples $\sum_x \sqrt{\mathcal{D}_x}|x, c(x)\rangle$. The quantum sample complexity is the number of quantum examples used by the quantum learner to learn $\mathcal{C}$ (on the hardest $c \in \mathcal{C}$ and distribution $\mathcal{D}$) and the *time complexity* of a quantum algorithm is the total number of gates involved (i.e., the number of gates it takes to implement various unitaries during the quantum algorithm) as well as the number of gates it takes to prepare quantum states. The remaining aspects of the quantum PAC learner is defined analogous to the classical PAC model. For more on this subject, the interested reader is referred to [AW17]. We now define what it means for an algorithm $\mathcal{A}$ to be a strong and weak learner for a concept class $\mathcal{C}$.

Definition 2.2.1 (Weak learner) *Let $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$ be a concept class. We say $\mathcal{A}$ is a* weak (quantum) learner *for $\mathcal{C}$ if it satisfies the following: there exists a polynomial p such that for all $n \geq 1$, for all $c \in \mathcal{C}_n$ and distributions $\mathcal{D} : \{0,1\}^n \to [0,1]$, algorithm $\mathcal{A}$, given n and (quantum) query access to c, with probability $\geq 2/3$, outputs a hypothesis $h : \{0,1\}^n \to \{0,1\}$ satisfying*

$$\Pr_{x\sim\mathcal{D}}[h(x) = c(x)] \geq \frac{1}{2} + \frac{1}{p(n)}. \tag{2.4}$$

Definition 2.2.2 (Strong learner) *Let $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$ be a concept class. We say $\mathcal{A}$ is a* strong (quantum) learner *for $\mathcal{C}$ if it satisfies the following: for all $n \geq 1$, $c \in \mathcal{C}_n$ and distributions $\mathcal{D} : \{0,1\}^n \to [0,1]$, algorithm $\mathcal{A}$, given n and (quantum) query access to c, with probability $\geq 2/3$, outputs a hypothesis $h : \{0,1\}^n \to \{0,1\}$ satisfying*

$$\Pr_{x\sim\mathcal{D}}[h(x) = c(x)] \geq \frac{2}{3}. \tag{2.5}$$

In this thesis, we will assume that we have classical or quantum query access to the output hypothesis h. Similarly, we say h is a *weak hypothesis* (resp. *strong hypothesis*) under $\mathcal{D}$ if h satisfies Eq. (2.4) (resp. Eq. (2.5)). We now define the Vapnik-Chervonenkis dimension (also referred to as VC dimension) [VC71].

Definition 2.2.3 *(VC dimension [VC71]) Fix a concept class $\mathcal{C}$ over $\{0,1\}^n$. A set $\mathcal{S} = \{s_1, \ldots, s_t\} \subseteq \{0,1\}^n$ is said to be* shattered *by a concept class $\mathcal{C}$ if $\{(c(s_1)\cdots c(s_t)) : c \in \mathcal{C}\} = \{-1,1\}^t$. In other words, for every labeling $\ell \in \{-1,1\}^t$, there exists a $c \in \mathcal{C}$ such that $(c(s_1)\cdots c(s_t)) = \ell$. The* VC *dimension of $\mathcal{C}$ (denoted by $\mathsf{VC}(\mathcal{C})$) is the size of the largest $\mathcal{S} \subseteq \{0,1\}^n$ that is shattered by $\mathcal{C}$.*

We again define two important misclassification errors (introduced in the previous section) which we will encounter often in Chapter 5. Suppose an algorithm $\mathcal{A}$ is given a set of labelled examples $S = \{(x_1, y_1), \ldots, (x_M, y_M)\}$ where $(x_i, y_i) \in \{0,1\}^n \times \{-1,1\}$ is drawn from a joint distribution $\mathcal{D} : \{0,1\}^n \times \{-1,1\} \to [0,1]$ and suppose $\mathcal{A}$ outputs a hypothesis $h : \{0,1\}^n \to \{-1,1\}$. The *training error* of h is defined as the error of h on the *training set* S, i.e.,

$$\text{training error of } h = \frac{1}{M} \sum_{i=1}^{M} [h(x_i) \neq y_i].$$

Ultimately, the goal of $\mathcal{A}$ should be to do well on labelled examples $(x, y) \notin S$ where (x, y) is sampled from the same distribution $\mathcal{D} : \{0,1\}^n \times \{-1,1\} \to [0,1]$ that generated the training set S. In order to quantify the *goodness* of the hypothesis h, the *true* error or the *generalization error* is defined as

$$\text{generalization error of } h = \Pr_{(x,y)\sim\mathcal{D}} [h(x) \neq y].$$

2.3 Boosting

In the early 1990s, Freund and Schapire [Sch90, Fre95, FS99] came up with a boosting algorithm called *AdaBoost* that *efficiently* solves the following problem: suppose we are given a weak learner as a black-box, can we use this black-box to obtain a strong learner? The AdaBoost algorithm by Freund and Schapire was one of the few theoretical boosting algorithms that were simple enough to be extremely useful and successful in practice, with applications ranging from game theory, statistics, optimization, biology, vision and speech recognition [SF12]. Given the success of AdaBoost in theory and practice, Freund and Schapire won the Gödel prize in 2003.

AdaBoost algorithm. We now give a sketch of the classical AdaBoost algorithm and provide more details in Section 5.4. Let $\mathcal{A}$ be a weak PAC learner for $\mathcal{C} = \cup_{n\geq 1} \mathcal{C}_n$ that runs in time $R(\mathcal{C})$ and has bias $\gamma > 0$, i.e., $\mathcal{A}$ does slightly better than random guessing (think of γ as inverse-polynomial in n). The goal of boosting is the following: for every $n \geq 1$, *unknown* distribution $\mathcal{D} : \{0,1\}^n \to [0,1]$ and *unknown* concept $c \in \mathcal{C}_n$, construct a hypothesis $H : \{0,1\}^n \to \{0,1\}$ that satisfies

$$\Pr_{x\sim\mathcal{D}} [H(x) = c(x)] \geq \frac{2}{3}, \tag{2.6}$$

where $[\cdot]$ is the indicator function which outputs 1 if $H(x) = c(x)$ and outputs 0 otherwise. AdaBoost algorithm by Freund and Schapire produces such an H by invoking $\mathcal{A}$ polynomially many times. The algorithm works as follows: it first obtains M different labelled examples $S = \{(x_i, c(x_i)) : i \in [M]\}$ where $x_i \sim \mathcal{D}$ and then AdaBoost is an iterative algorithm that runs for T steps (for some M, T which we specify later). Let D^1 be the uniform distribution on S. At the tth step, AdaBoost defines a distribution D^t depending on D^{t-1} and invokes $\mathcal{A}$ on the training set S and distribution D^t. Using the output hypothesis h_t of $\mathcal{A}$, AdaBoost computes the *weighted error*

$$\varepsilon_t = \Pr_{x \sim D^t}[h_t(x) \neq c(x)], \tag{2.7}$$

which is the probability of h_t misclassifying a randomly selected training example drawn from the distribution D^t. The algorithm then uses ε_t to compute a *weight* $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ and updates the distribution D^t to D^{t+1} as follows

$$D_x^{t+1} = \frac{D_x^t}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t} & \text{otherwise}, \end{cases} \tag{2.8}$$

where $Z_t = \sum_{x \in S} D_x^t \exp(-c(x)\alpha_t h_t(x))$.[1] After T iterations, the algorithm outputs the hypothesis H

$$H(x) = \operatorname{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x)\Big),$$

where α_t is the weight and h_t is the weak hypothesis computed in the tth iteration.[2]

It remains to answer three important questions: (1) What is T, (2) What is M, (3) Why does H satisfy Eq. (2.6)? The punchline of AdaBoost is the following: by selecting the number of iterations $T = O(\log M)$, the hypothesis H satisfies $H(x) = c(x)$ for every $x \in S$. However, note that this does not imply that H is a strong hypothesis, i.e., it is not clear if H satisfies Eq. (2.6). Freund and Schapire showed that, if the number of labelled examples M is at least $O(\mathsf{VC}(\mathcal{C}))$ (where $\mathsf{VC}(\mathcal{C})$ is a combinatorial dimension that can be associated with the concept class $\mathcal{C}$), then

[1] This distribution update rule is also referred to as the *Multiplicative Weights Update Method* (MMUW). See [AHK12, Section 3.6] on how one can cast AdaBoost into the standard MMUW framework.

[2] Note that without loss of generality, we can assume $\sum_t \alpha_t = 1$ since renormalizing α_ts will not change H.

with high probability (where the probability is taken over the randomness of the algorithm and the training set S), the final hypothesis H satisfies

$$\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq 2/3.$$

In other words, by picking M large enough, not only perfectly classifies every $x \in S$, but is also 2/3-close to c under $\mathcal{D}$. Hence H is a strong hypothesis for the target concept c under the unknown distribution $\mathcal{D}$ which had support on $\{0,1\}^n$. The overall time complexity of AdaBoost is $\widetilde{O}(n \cdot R(\mathcal{C}) \cdot \mathsf{VC}(\mathcal{C}))$: the algorithm runs for $T = \log M = \log \mathsf{VC}(\mathcal{C})$ rounds, and in each round we run a weak learner with time complexity $R(\mathcal{C})$, compute the weighted error ε_t which takes time $O(M) = O(\mathsf{VC}(\mathcal{C}))$ and then update the distributions using arithmetic operations in time $O(n)$.

2.4 Gradient descent

In Chapter 4, we use gradient descent to find a sequence of N unitaries that reproduces a target unitary on a d-dimensional space. We obtain the surprising result that the gradient descent algorithm finds the correct sequence using *exactly* $N = d^2$ unitaries in the sequence. In this section, we provide a sketch of the gradient descent algorithm for optimizing a convex function and briefly mention its convergence properties. Furthermore, we discuss Adam which is an improved version of gradient descent.

The central idea of any convex optimization algorithm is to find a stationary point. In the literature, several optimization techniques and heuristics have been introduced to find a stationary point in a convex manifold. The most popular approach – gradient descent – is to start from an arbitrary point in the manifold and move along a gradient at that point to decrease the cost function. The hope is to repeat this procedure until the procedure converges to the stationary point.

In gradient descent, there are two key components for searching a stationary point in a convex manifold $f(\vec{x})$ where $\vec{x}$ is a vector of parameters: the learning rate or step size and the gradient. The learning rate determines how far do we need to move in one direction and the gradient decides which direction should we search in the next step to minimize the cost function. To be more precise, gradient descent starts from an arbitrary point $\vec{x}_0$ where $\vec{x}_0$ is in general a vector of parameters in the convex manifold. Gradient descent is an iterative method where at each iteration $t > 0$, the algorithm moves in the direction of the gradients $\Delta\vec{x}_t$

by the step size η_t to reach the next point $\vec{x}_{t+1} = \vec{x}_t + \eta_t \Delta\vec{x}_t$. The gradients $\Delta\vec{x}_t$ are given by the negative of the gradients of the cost function, i.e., $\Delta\vec{x}_t = -\nabla f(\vec{x}_t)$. We have the following recursive relation due to gradient descent,

$$\vec{x}_{t+1} = \vec{x}_t - \eta_t \nabla f(\vec{x}_t). \tag{2.9}$$

In the majority of the applications of gradient descent, the learning rate η_t is fixed to a value η during all the iterations. We now describe the gradient descent algorithm below.

Algorithm 1 Gradient Descent

Input: Cost function $f(\vec{x})$ and the parameters in $\vec{x}$.

Initialize: $\vec{x} = \vec{x}_0$, t=0.

1: **while** $\| \nabla f(\vec{x}_t) \| \geq \varepsilon$ **do**
2: $\quad \vec{x}_{t+1} = \vec{x}_t - \eta \nabla f(\vec{x}_t)$.
3: $\quad t \leftarrow t + 1$.
4: **end while**

Output: $\vec{x}_t$.

The condition $\| \nabla f(\vec{x}_t) \| \geq \varepsilon$ is called the stopping criterion. Note that this condition does not a priori guarantee that gradient descent will reach a point that is ε-close to the stationary point. However, since we are searching for the optimal parameters in a convex manifold, one can show that the relation $f(\vec{x}_t) - \min_{\vec{x}} f(\vec{x}) \leq \varepsilon$ follows as a consequence of the convexity of the domain of interest.

We now mention the convergence guarantees of the gradient descent algorithm. Given a convex, twice differentiable function f with domain$(f) = \mathbb{R}^n$ and a constant $L > 0$. Then for a fixed step size $\eta \leq 1/L$, gradient descent satisfies

$$f(\vec{x}_t) - f(\vec{x}^*) \leq \frac{\| \vec{x}_t - \vec{x}^* \|_2^2}{2\eta t},$$

where $\vec{x}^*$ is the stationary point. The above equation implies that gradient descent has a convergence rate $O(1/t)$. In other words, we need $O(1/\varepsilon)$ iterations to achieve $f(\vec{x}_t) - f(\vec{x}^*) \leq \varepsilon$.

2.4.1 Adam

In recent years, neural networks have gained prominence as one of the main tools for machine learning. It is well known that the parameter landscape of neural networks can be highly non-convex. Additionally, the landscape can posses multiple saddle points. However, these bottlenecks have not deterred the machine learning community from applying the gradient descent algorithm or its advanced versions to find a stationary point in the neural network landscape.

Gradient descent or vanilla gradient descent does not always guarantee convergence in non-convex spaces. One major issue is to choose the appropriate value of the learning rate or step size. If the learning rate is very small, then gradient descent can converge very slowly to the stationary point. Alternatively, if the learning rate is too large, then gradient descent may not converge and the cost function can oscillate around the stationary point or even diverge. Another issue is conventional gradient descent applies the same learning rate to all parameter updates. The features of data can have different frequencies and it is not prudent to apply the same learning rate to all of them, instead apply larger learning rates to the rarer features. The most prominent challenge for gradient descent is to avoid saddle points in neural network landscapes. Saddle points typically lie in a plateau of the same error which is hard for gradient descent to escape since the gradient is almost zero in all directions.

In the following, we provide a sketch of the Adam or *Adaptive Moment Estimation* algorithm [KB15]. We do not delve into the details of the algorithm but mention only the necessary details. For further details, the interested reader can refer to the review [Rud16]. Adam overcomes the problem of assigning the correct learning rate to each parameter by computing an adaptive learning rate that can vary during the course of gradient descent. The trick employed by Adam is to store the exponentially decaying averages of both the past gradients m_t as well as the second moments v_t of the past gradients. The intuition here is to think of a heavy ball running down a hill but there is friction on the surface of the hill. As the ball reaches a local minimum, the friction on the surface enables it to stop at the local minimum. The decaying averages of the past gradients and past squared gradients are computed as follows

$$\begin{aligned} m_{\boldsymbol{x}}^t &= \beta_1 m_{\boldsymbol{x}}^{t-1} + (1-\beta_1)\nabla_{\boldsymbol{x}} f(\vec{x}), \\ v_{\boldsymbol{x}}^t &= \beta_1 v_{\boldsymbol{x}}^{t-1} + (1-\beta_1)(\nabla_{\boldsymbol{x}} f(\vec{x}))^2, \end{aligned} \tag{2.10}$$

where $\boldsymbol{x}$ is an element of the parameter vector $\vec{x}$ and $(1-\beta_1),(1-\beta_2)$ are the decay rates close to zero.

In Adam, the vectors $\vec{m}_{\boldsymbol{x}}^t$ and $\vec{v}_{\boldsymbol{x}}^t$ are initialized to zero. This means the vectors are biased to zero during the initial stages of gradient descent. In order to overcome the biases of the first and second moments, they are normalized as

$$\widehat{m}_{\boldsymbol{x}}^t = \frac{m_{\boldsymbol{x}}^t}{1-\beta_1}, \quad \widehat{v}_{\boldsymbol{x}}^t = \frac{v_{\boldsymbol{x}}^t}{1-\beta_2}. \tag{2.11}$$

The final update rule of Adam is then given by

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \frac{\widehat{m}_{\boldsymbol{x}}^t}{\widehat{v}_{\boldsymbol{x}}^t + \varepsilon}. \tag{2.12}$$

The default values of the hyper-parameters $\beta_1, \beta_2, \varepsilon$ are set equal to 0.9, 0.999 and 10^{-6} respectively.

2.5 Quantum computation

We assume that the reader is familiar with the following notation in quantum information. Throughout this thesis, we let $[n] = \{1, \ldots, n\}$ and $\mathbb{C}$ denote the complex space. A qubit, which is the quantum equivalent of a bit, lives in the Hilbert space $\mathbb{C}^2$. The space $\mathbb{C}^2$ is spanned by the standard basis states: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, i.e., the qubit can be in a superposition of the states $|0\rangle$ and $|1\rangle$. The basis states for multi-qubit systems can be obtained by performing a tensor product of the basis states of single qubit systems. For example, a basis state for a two-qubit system is $|1\rangle \otimes |0\rangle \in \mathbb{C}^4$ where $|1\rangle$ and $|0\rangle$ are the states of the first and second qubits respectively. We use the shorthand $|a_1 \ldots a_n\rangle$ for a n-qubit tensor product state $|a_1\rangle \otimes \cdots \otimes |a_n\rangle$. For every $i \in [n]$, let $\alpha_i \in \mathbb{C}$ and $a_i \in \{0,1\}$. A pure quantum state $|\psi\rangle$ composed of n qubits is expressed as $|\psi\rangle = \sum_{i\in[n]} \alpha_i |a_i\rangle$ where the $\alpha_i's$ satisfy the condition $\sum_{i\in[n]} |\alpha_i|^2 = 1$. The complex conjugate of the state $|\psi\rangle$ is a row vector denoted by $\langle\psi|$. Alternatively, a quantum state can be also represented by a positive semi-definite matrix or a *density matrix* ρ with trace 1. Similar to the state vector $|\psi\rangle$, the density matrix ρ encodes all accessible information about a quantum state. Additionally, it also characterizes quantum states $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ which are statistical mixtures of the pure states $\{|\psi_i\rangle\}_i$. We remark that mixed states which are statistical ensembles of pure states encoded by ρ are more prevalent in the environment than their pure counterparts.

A quantum operation is represented by a unitary matrix U which acts on a pure quantum state $|\psi\rangle$ as $U|\psi\rangle$ and the action of U on a density matrix ρ is given by $U\rho U^{-1}$. For example, let $x \in \{0,1\}^n$ and the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. The quantum Fourier transform over $\mathbb{Z}_2^n$ is given by $H^{\otimes n}|x\rangle = 2^{-n/2}\sum_{a\in\{0,1\}^n}(-1)^{x\cdot a}|a\rangle$. Let $c : \{0,1\}^n \to \{-1,1\}$ be a Boolean valued function. We say an oracle $\mathcal{A}$ is given *query* access to c if, $\mathcal{A}$ can *query* c, i.e., $\mathcal{A}$ can obtain $c(x)$ for x of its choice. Similar, we say $\mathcal{A}$ has *quantum query* access to c, if $\mathcal{A}$ can query c in a superposition, i.e., $\mathcal{A}$ can perform the map

$$O_c : |x, b\rangle \to |x, c(x) \cdot b\rangle,$$

for every $x \in \{0,1\}^n$ and $b \in \{-1,1\}$.

A *quantum measurement* on a quantum state ρ is a procedure to obtain classical information from ρ. A *POVM* or a positive operator valued measure is a k-outcome measurement which is characterized by a set of positive semi-definite matrices $\{M_i\}_{i\in[k]}$ which satisfy $\sum_i M_i = I$. When performing a measurement on the quantum state ρ with the POVM $\{M_i\}_{i\in[k]}$, the probability of obtaining the outcome j is given by $\mathrm{Tr}(M_j\rho)$.

2.5.1 Universal quantum gates

A finite set of gates is *universal* for quantum computation if any unitary transformation can be approximated to a desired accuracy with a finite sequence of gates from the set. Note that the number of possible quantum gates is uncountable while the number of finite sequences that can be constructed from a finite gate set is countable. This signals a contradiction in the construction of an arbitrary unitary operation. However, we are only interested in approximating a unitary transformation by a sequence of gates from a finite set instead of building them exactly.

We mention the main ideas required for universality constructions in quantum computation. First, it can be shown that any unitary operation can be decomposed exactly into a product of 2-qubit unitary matrices, i.e., unitary matrices that act non-trivially on one or two qubit systems. This result can be fine-tuned to show that single qubit and CNOT gates can be used to construct any n-qubit unitary operation. Thus single qubit gates and CNOT gates are universal for quantum computation. Second, any single qubit gate can be approximated to a desired accuracy

by the Hadamard H, phase S and T gates where $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ and $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$. Putting these results together, we find that a sequence of CNOT, Hadamard, phase and T gates can approximate any unitary operation to an arbitrary precision.

Note that the above universal gate set is not unique. In fact, there can be infinite sets of gates that can be universal. The gate set comprising of XOR and single qubit gates is also universal. An example of a 2-qubit continuous gate is the $XX(\phi)$ gate where $-\pi/2 \leq \phi \leq \pi/2$. The $XX(\phi)$ gate coupled with single qubit gates can perform universal quantum computation. The $XX(\phi)$ gate is defined by the unitary matrix:

$$\begin{pmatrix} \cos(\phi) & 0 & 0 & -i\sin(\phi) \\ 0 & \cos(\phi) & -i\sin(\phi) & 0 \\ 0 & -i\sin(\phi) & \cos(\phi) & 0 \\ -i\sin(\phi) & 0 & 0 & \cos(\phi) \end{pmatrix}.$$

2.5.2 Solovay-Kitaev theorem

As seen in the previous section, a universal gate set can perform any unitary operation to a desired accuracy. However, this does not address the question of how efficiently can a sequence from a discrete gate set approximate a unitary transformation in $U(d)$ with precision ε.

The Solovay-Kitaev theorem [Kit97] is a fundamental result in quantum computation theory. Roughly speaking, the theorem says if $\mathcal{G}$ is a universal gate set for $SU(d)$ (i.e., given any unitary $U \in SU(d)$ and precision $\varepsilon > 0$, there exists a sequence $S = g_1 \dots g_m$ of gates from $\mathcal{G}$ that can approximate U with accuracy ε), then it is possible to approximate any unitary in $SU(d)$ using remarkably short sequences of gates drawn from $\mathcal{G}$. To be precise, the Solovay-Kitaev theorem says that a sequence S of depth $O(d^2 \log^c(1/\varepsilon))$ is sufficient to ε-approximate any unitary U where $c > 0$ is a constant. In other words, a discrete gate set of CNOT, H, S and T gates can be used to construct any unitary in time $O(\text{polylog}(1/\varepsilon))$.

We provide a brief overview of the Solovay-Kitaev algorithm as discussed in [DN05]. A simplified version of the algorithm for qubits is presented here. The algorithm can be extended for constructing any d-dimensional unitary transformation using techniques discussed in [DN05]. The pseudocode for constructing an arbitrary unitary in $U(2)$ is described below.

Algorithm 2 Solovay-Kitaev algorithm

Input: An arbitrary unitary $U \in U(2)$ and circuit depth ℓ.

function SOLOVAY-KITAEV(U, ℓ)
  if $\ell = 0$ **then**
    $\widetilde{U}_0 = \text{approx}(U)$
  else
    $\widetilde{U}_{\ell-1} = \text{Solovay-Kitaev}(U, \ell-1)$
    $V_{\ell-1}, W_{\ell-1} = \text{Group\_Commutator}(U, \widetilde{U}_{\ell-1})$
    $\widetilde{V}_{\ell-1} = \text{Solovay-Kitaev}(V_{\ell-1}, \ell-1)$
    $\widetilde{W}_{\ell-1} = \text{Solovay-Kitaev}(W_{\ell-1}, \ell-1)$
    $\widetilde{U}_\ell = \widetilde{V}_{\ell-1}\widetilde{W}_{\ell-1}\widetilde{V}_{\ell-1}^\dagger\widetilde{W}_{\ell-1}^\dagger\widetilde{U}_{\ell-1}$
  end if
end function

Output: The final approximating sequence $\widetilde{U}_\ell$.

The Solovay-Kitaev theorem is a recursive algorithm that takes the target unitary U and the desired depth ℓ as input and returns a unitary $\widetilde{U}_\ell$ that approximates U with a desired accuracy ε. The unitary $\widetilde{U}_\ell$ is a sequence of gates drawn from a universal gate set. The first step which is the preprocessing step involves generating the best approximating sequence $\widetilde{U}_0$ for the target unitary U (Dawson *et al.* [DN05] discuss how to generate the initial sequence $\widetilde{U}_0$). At each iteration or layer $\ell > 0$, the algorithm invokes itself to obtain a coarser approximating unitary $\widetilde{U}_{\ell-1}$ of depth $\ell - 1$. The next step involves a group commutator decomposition subroutine to obtain $V_{\ell-1}, W_{\ell-1} \in SU(2)$ such that,

$$U\widetilde{U}_{\ell-1}^\dagger = V_{\ell-1}W_{\ell-1}V_{\ell-1}^\dagger W_{\ell-1}^\dagger. \tag{2.13}$$

The matrices $V_{\ell-1}, W_{\ell-1}$ are called the group commutators which are unitaries close to the identity where

$$\delta = \max\{\| I - V_{\ell-1} \|, \| I - W_{\ell-1} \|\}. \tag{2.14}$$

The algorithm then approximates the group commutators by invoking itself at the $(\ell-1)$-th layer. Let $\widetilde{V}_{\ell-1}, \widetilde{W}_{\ell-1}$ be the unitaries approximating $V_{\ell-1}, W_{\ell-1}$ such that,

$$\Delta = \max\{\| \widetilde{V}_{\ell-1} - V_{\ell-1} \|, \| \widetilde{W}_{\ell-1} - W_{\ell-1} \|\}. \tag{2.15}$$

At the end of the ℓ-th iteration, the algorithm returns the unitary $\widetilde{U}_\ell$ where $\widetilde{U}_\ell = \widetilde{V}_{\ell-1}\widetilde{W}_{\ell-1}\widetilde{V}_{\ell-1}^\dagger\widetilde{W}_{\ell-1}^\dagger\widetilde{U}_{\ell-1}$. The crucial idea of the Solovay-Kitaev theorem is that the ℓ-th iteration produces a better approximation $\widetilde{U}_\ell$ than the one in the previous

iteration. This is due to the following relationship between the approximate and exact group commutators given by,

$$\| \widetilde{V}_\ell \widetilde{W}_\ell \widetilde{V}_\ell^\dagger \widetilde{W}_\ell^\dagger - V_\ell W_\ell V_\ell^\dagger W_\ell^\dagger \| \leq O(\Delta^2 + \Delta\delta). \tag{2.16}$$

2.5.3 Quantum subroutines

The most famous quantum algorithm to date is Shor's algorithm discovered in 1994 [Sho94]. Shor's algorithm is a quantum algorithm that can compute the prime factors of integers in polynomial time. This is exponentially faster than the best known classical algorithm for prime factorization of integers. The second most important quantum algorithm is Grover's quantum search problem discovered in 1996 [Gro96]. Grover's algorithm tackles the search problem with a quadratic improvement over its classical counterpart. Let $N = 2^n$. Suppose we have an arbitrary string $x \in \{0,1\}^N$. The goal of the search problem is to find the position i such that $x_i = 1$ and output 'no' if there is no such i. This problem is basically a simplified version of searching an N-element unstructured database. A classical randomized algorithm requires $\Theta(N)$ queries to solve the search problem. Remarkably, Grover's algorithm can solve it with only $O(\sqrt{N})$ queries and in time $O(\sqrt{N}\log N)$. Grover's algorithm can be applied more generally to problems which require the correct solution with a high probability of success. One such problem is amplitude amplification which we describe below.

In this thesis, we will use three quantum subroutines to prove our main results in Chapter 5. The first quantum subroutine is *amplitude amplification*, a well-known quantum algorithm which performs the following task: suppose we have a (classical) algorithm $\mathcal{A}$ that outputs 1 with probability p and 0 otherwise, then classically we need to repeat $\mathcal{A}$ $\Theta(1/p)$ many times before one of the repetitions of $\mathcal{A}$ outputs the number 1. Quantumly, amplitude amplification is a procedure that invokes $\mathcal{A}$ and the inverse of $\mathcal{A}$ (denoted $\mathcal{A}^{-1}$) $O(1/\sqrt{p})$ many times before outputting 1 with high probability, hence providing a quadratic quantum speedup over classical randomized algorithms.

Theorem 2.5.1 (Amplitude Amplification [BHMT02]) *Let $p, a, a' > 0$. There is a quantum algorithm $\mathcal{A}$ that satisfies the following: given access to a unitary U such that $U|0\rangle = |\psi\rangle$ where $|\psi\rangle = \sqrt{p}|\psi_0\rangle + \sqrt{1-p}|\psi_1\rangle$ for an* unknown *$p > a$ and $|\psi_0\rangle, |\psi_1\rangle$ are orthogonal quantum states, $\mathcal{A}$ makes an expected number of $\Theta(\sqrt{a'/a})$ queries to U, U^{-1} and outputs $|\psi_0\rangle$ with probability $a' > 0$.*

The second quantum algorithm estimates the mean of numbers quadratically faster on a quantum computer than classical algorithms for mean estimation. Given a black-box for the function $F : \{1,\dots,N\} \to [0,1]$, there exists a quantum algorithm that with probability at least 2/3 computes an additive ε-approximation of $\frac{1}{N}\sum_{i=1}^{N} F(i)$ using $O(1/\varepsilon)$ evaluations of F. Observe that classically estimating the mean $\frac{1}{N}\sum_{i=1}^{N} F(i)$ up to additive precision ε would take $\Theta(1/\varepsilon^2)$ many evaluations of the function F.

Theorem 2.5.2 (Amplitude Estimation [BHMT02]) *There is a quantum algorithm $\mathcal{A}$ that satisfies the following: given access to a unitary U such that $U|0\rangle = |\psi\rangle$ where $|\psi\rangle = \sqrt{a}|\psi_0\rangle + \sqrt{1-a}|\psi_1\rangle$ and $|\psi_0\rangle$, $|\psi_1\rangle$ are orthogonal quantum states, $\mathcal{A}$ makes M queries to U and U^{-1} and with probability $\geq 2/3$, outputs $\widetilde{a}$ such that*

$$|\widetilde{a} - a| \leq 2\pi \frac{\sqrt{a(1-a)}}{M} + \frac{\pi^2}{M^2}. \tag{2.17}$$

The third quantum algorithm is a modified version of amplitude estimation where the goal is estimate the probability a with a better precision than the additive error ε. This algorithm provides a technique to achieve a multiplicative error, i.e., $|a - \widetilde{a}| \leq \varepsilon\widetilde{a}$ using $\widetilde{O}(1/\varepsilon)$ queries to U and U^{-1}.

Theorem 2.5.3 (Multiplicative amplitude estimation [Amb10]) *Let $c \in (0,1]$. There is a quantum algorithm $\mathcal{A}$ that satisfies the following: given a state $|\psi\rangle$ and access to a unitary U such that $U|0\rangle = |\psi\rangle$ where $|\psi\rangle = \sqrt{a}|\psi_0\rangle + \sqrt{1-a}|\psi_1\rangle$ and $|\psi_0\rangle$, $|\psi_1\rangle$ are orthogonal quantum states and promised that either $a = 0$ or $a \geq p$, with probability $\geq 1-\delta$, the algorithm $\mathcal{A}$ outputs an estimate $\widetilde{a}$ satisfying $|a - \widetilde{a}| \leq c \cdot \widetilde{a}$ if $a \geq p$, and $\widetilde{a} = 0$ if $a = 0$. The total number of queries made by $\mathcal{A}$ to U, U^{-1} is*

$$O\left(\frac{\log(1/\delta)}{c}\left(1 + \log\log\frac{1}{p}\right)\sqrt{\frac{1}{\max\{a,p\}}}\right). \tag{2.18}$$

2.6 Quantum control

Quantum control is the study of active manipulation of physical and chemical processes in quantum systems. One of the main interests of quantum physicists and chemists is to control the dynamics of quantum systems with a desired precision. This subject has recently received a lot of attention mainly because of the need to build quantum information processing devices [BCR10, RTB+15, RWSR17,

RRW17]. The central issue with controlling a quantum system is whether it is possible to achieve a desired precision in the control objective.

A practical task in quantum control for manipulating quantum systems is to find the optimal solutions of the control parameters in the control landscape . Several studies have established the conditions which the landscape should satisfy for the optimal solutions to exist [DR93]. A necessary and sufficient condition for the controllability of a quantum system in d dimensions is that the Lie group generated by the control Hamiltonians should be complete in $u(d)$. A mathematically rigorous analysis of the quantum control landscape can characterize the nature of the critical points of the landscape: whether the critical point is a global minimum, a local minimum or a saddle point. Furthermore, it can also establish necessary conditions for an optimization algorithm to converge to a global solution in the control landscape. Consider a target unitary transformation W in d dimensions. Let $U(\vec{t},\vec{\tau})$ be the unitary for approximating the target W where $\vec{t},\vec{\tau}$ are the control parameters. The control landscape is described by the objective function

$$L(\vec{t},\vec{\tau}) = \| W - U(\vec{t},\vec{\tau}) \|^2.$$

The optimal control problem is concerned with an unconstrained search for

$$L_{\min} = \min_{\vec{t},\vec{\tau}} L(\vec{t},\vec{\tau}). \tag{2.19}$$

A critical point of the control landscape is indicated by the control parameters at which the first order derivative of the landscape $L(\vec{t},\vec{\tau})$ with respect to all the parameters is zero, i.e., $\partial L(\vec{t},\vec{\tau})/\partial t = \partial L(\vec{t},\vec{\tau})/\partial \tau = 0$ for all ts and τs. A necessary condition for the optimality of a critical point is that the first order derivatives are zero at that point. For sufficiency, one needs to check the negative definiteness of the Hessian of the objective function $L(\vec{t},\vec{\tau})$ at the critical point. It has been shown that the presence or absence of local critical points including saddle points can influence the convergence properties of search algorithms such as gradient descent optimisation [CR07]. For more details about the field of quantum control, the interested reader is referred to the recent review [KLS19].

Chapter 3

A bound on implementing unitary transformations

One of the most important applications of a quantum computer is to simulate the time evolution of a quantum system [Fey82]. It is well known that under reasonable complexity theoretic assumptions, the state-of-the-art classical algorithms (i.e., algorithms that run on traditional computers) can solve this problem in exponential time. In contrast, it was known from the early days of quantum computing [Fey82, Llo96] that quantum computers take polynomial time to simulate the quantum dynamics.

The dynamics of quantum systems are described by unitary transformations, which are implemented by applying sequences of elementary quantum logic operations or control fields. One can build any desired unitary operation U from a sequence $\{U_1, \ldots, U_N\}$ of elementary quantum logic gates given by $U = U_N \ldots U_1$. In quantum computation, the logic gates that we apply are either discrete such as CNOT, Hadarmard and phase gate or continuous of the form $U_j = e^{-iH_j t_j}$ where H_j is a local Hamiltonian acting on the physical system for time t_j. In quantum control, the desired unitary operation U is implemented by a time dependent Hamiltonian of the form $H(t) = \sum_j g_j(t) H_j$. Here $g_j(t)$ corresponds to a time-dependent control field.

A vast amount of literature in quantum computation has been devoted to the study of quantifying the resources, i.e., gate complexity and time complexity required to simulate the dynamics of a quantum physical system. This task is known in the literature as *Hamiltonian simulation* or *quantum simulation* and for a n-qubit system, a quantum algorithm can perform the task in time $O(\text{poly}(n))$ [Fey82,

Llo96]. Recent works improve on the scaling of the precision of the quantum algorithm designed to mimic a physical process [BCC+14, BCC+15, LC17, LC19].

In this chapter, we discuss the problem of providing upper bounds on the minimum time required to implement an arbitrary unitary transformation in $U(d)$ on a d-dimensional Hilbert space when applying sequences of Hamiltonian transformations. The previous paragraph discusses the problem of simulating the dynamics governed by a local Hamiltonian. Here we shift our focus towards providing an algorithm that can construct any unitary transformation in $U(d)$ with a desired accuracy. A necessary and sufficient condition for constructing an arbitrary unitary in both quantum computation and quantum control is that the algebra generated by the Hamiltonians $\{H_1, \ldots, H_N\}$ via commutation should be complete in the Lie algebra $u(d)$. Because we are not concerned about an overall global phase, in the rest of this chapter we will restrict our study to traceless Hamiltonians $\{H_1, \ldots, H_N\}$ in the Lie algebra $su(d)$.

3.1 Overview of our results

The Trotter-Suzuki decomposition or Trotterization is a technique that addresses the problem of efficient approximation of matrix exponentiation [Tro59, Suz90, Suz91]. This technique transforms the time evolution operator into a product of simpler operations that can be implemented on a quantum computer while incurring an error, namely the Trotter error.

We consider the simplest case where we have access to a pair of d-dimensional non-commuting traceless Hamiltonians A and B. Our results can be generalized to the case of three or more traceless Hamiltonians in a straightforward manner. Following the seminal works of Suzuki [Suz90, Suz91], we construct unitaries of the form

$$U(\vec{t}, \vec{\tau}) = e^{-iB\tau_N} e^{-iAt_N} \ldots e^{-iB\tau_1} e^{-iAt_1}, \tag{3.1}$$

via the Trotter-Suzuki decomposition. The parameters $\vec{t}, \vec{\tau}$ represent the times for which the Hamiltonians A, B are applied. We make three assumptions here in order to facilitate our study in a controlled setting. The first assumption is that the operators $A, B \in \mathcal{H}_d$ are bounded such that $\| A \|_1 = \| B \|_1 = 1$. The second assumption is that the operators A and B are sampled from the Gaussian Unitary Ensemble in d-dimensions so that the algebra generated by A, B via commutation is

with probability one complete in $su(d)$ [RHR04, KRK$^+$05, RHR05, RHH$^+$06, CR07, MHR08, HDR09, BCR10, RTB$^+$15, RWSR17, RRW17]. The third assumption is that the operators $\pm A, \pm B$ can be implemented; equivalently, we can consider $\vec{t}, \vec{\tau}$ to be positive or negative.

The time parameters $\vec{t}, \vec{\tau}$ can be large in the quantum logic gate construction of unitary operators. In continuous time quantum optimal control, the dynamics is governed by a time dependent control Hamiltonian $H(t) = f_1(t)A + f_2(t)B$ where $f_1(t), f_2(t)$ are time-dependent control fields. It has been shown that the continuous time Hamiltonian dynamics can be implemented in the small time $\vec{t}, \vec{\tau}$ and large N limit of Eq. (3.1) by the well-known technique of Trotterization [Llo96, LV01, WBHS10, NC16]. Here we represent the time dependent dynamics as a sequence of infinitesimal unitary transformations.

In this chapter, we address the general problem of providing upper bounds on the optimal time required to implement an arbitrary unitary operator in d dimensions. It is evident that $d^2 - 1$ parameters are required to specify an arbitrary unitary matrix in $SU(d)$. We further ask whether one can find the correct sequence of logic gates or control fields that can be applied in order to build the desired unitary matrix. We discuss two methods, namely the non-infinitesimal and infinitesimal constructions for building a desired unitary transformation in $SU(d)$. In the infinitesimal approach, we build unitary transformations in the vicinity of the identity using nested commutation relations. The non-infinitesimal approach employs a different strategy. In this approach, we first move away from the identity unitary, perform a set of operations and then return in the vicinity of the identity. The time complexity of the non-infinitesimal construction is $O(d^2)$ and it is $O(d^2 \log d)$ for the infinitesimal approach. Our first result conjectures that under certain assumptions, one can construct a desired unitary operation in the neighbourhood of the identity in time $O(d^2)$ by appropriately choosing a sequence of parameters $\vec{t}, \vec{\tau}$ in Eq. (3.1). This can be achieved either by a direct method when we have access to the target unitary or by optimizing a cost function when the unitary is described by a training set of input and output pairs.

In quantum control, a remarkable result by Rabitz *et al.* [RHR04, HDR09] demonstrates that when the control fields $\{f_j(t)\}_j$ are neither power nor band limited, the optimal control sequence for constructing a desired unitary can be achieved by using gradient descent optimization. In practice, however, control fields are power and band limited. Our second result conjectures that even with power and

band limited controls, one can construct a desired unitary near identity in time $O(d^2)$ using gradient descent. We emphasize that the main differences between the infinitesimal and non-infinitesimal constructions is the infinitesimal technique cannot build an arbitrary unitary and the process of finding the optimal path to construct the unitary using an optimization technique such as gradient descent involves a search through multiple saddle points.

3.2 Related works

It is important to note that the majority of studies in the field of quantum computation have focused on the efficient simulation of the dynamics of local Hamiltonians. The early works on Hamiltonian simulation showed that quantum computers can simulate the dynamics of any local quantum system (the Hamiltonian of the physical system is characterized by a tensor product structure of smaller subsystems) comprising of n qubits in time poly(n) [Fey82, Llo96]. In particular, Lloyd demonstrated that by Trotterizing the local quantum dynamics, a quantum computer can simulate the dynamics of a physical system by implementing a sequence of elementary logic operations.

Aharonov and Ta-Shma [AT03] extended the result of Lloyd by showing that a more general class of Hamiltonians of n qubit systems can be efficiently simulated (in time poly(n)). In particular, they considered sparse Hamiltonians which cannot be expressed as a tensor product structure of smaller subsystems. They proved that there exists an efficient quantum algorithm for computing the nonzero entries in a particular column of such Hamiltonians. Sparse Hamiltonians are common in quantum walk simulations [Chi10] and adiabatic quantum computation [FGGS00].

Subsequent works on sparse Hamiltonian simulation based on high-order product formulas, namely the Lie-Trotter-Suzuki formulas improved the complexity of the precision ε of the quantum algorithm from $O(1/\varepsilon)$ to $O(1/\varepsilon^p)$ where $p < 1$ [BACS07, WBHS10, WBHS11]. These works further showed that higher order product formulas can upper bound the number of exponentials required in the simulating sequence by a term that depends linearly on both the norm of the sparse Hamiltonian and the evolution time t. Berry and Childs [BC12] leveraged the computing power of quantum walks to present a general method for simulating black-box sparse Hamiltonians. Their quantum simulation algorithm made

a substantial progress in the complexity of the sparsity (i.e., the maximum number of non-zero elements in a column) of the Hamiltonian matrix from polynomial to linear. The scaling of the algorithmic precision was further improved by Childs and Wiebe [CW12] using a novel technique based on implementing linear combinations of unitary operations instead of higher order product formulas. Building on the linear combination of unitaries framework, Berry *et al.* and Low and Chuang [BCC+14, BCC+15, LC17, LW18b] showed that there exist quantum algorithms that can achieve exponentially better performance in the accuracy of simulating sparse Hamiltonian dynamics. Their results were further improved by Haah *et al.* [HHKL18] by presenting a quantum algorithm for simulating a time-dependent Hamiltonian on a lattice of n qubits that satisfies an optimal lower bound on the gate complexity for simulating the time evolution operator.

We also mention a modified version of the Hamiltonian simulation problem which is termed as *sample-based* Hamiltonian simulation. Given one copy of an unknown state ρ and multiple copies of an unknown state σ, the task is to implement the time evolution of the state ρ under the dynamics governed by the state σ. Note that the state σ performs the function of a Hamiltonian. This problem was first studied by [LMR14] where the authors provide a simple protocol to implement the exponential of the state σ using multiple copies of σ. A remarkable feature of their work is that the protocol is agnostic with respect to the state σ.

There has been numerous studies on the optimal construction of an arbitrary unitary transformation using quantum logic gates [MVBS04, SBM06, PISR06]. In particular, the authors proved a lower bound on the number of two-qubit logic gates required to prepare an arbitrary n-qubit quantum state using matrix decomposition techniques. There has also been a spate of activities in the fields of trapped ions and superconducting quantum computing to optimize the gate errors in implementing two-qubit logic gates [HHJ+10, WD13, GAN14, ZLW+17, GCS17, SBT+18].

One of the most fundamental results of quantum computation is the Solovay-Kitaev theorem [Kit97, DN05, NC16] which shows that a finite set of universal quantum gates can efficiently approximate an arbitrary unitary transformation. Roughly speaking, the theorem says that if a set of single and two-qubit gates is dense in $SU(d)$, then it is possible to approximate any unitary operation in $SU(d)$ using short sequences of gates from the universal set. A detailed presentation of the algorithm is discussed in Chapter 2. A series of works by Selinger *et al.*

and Kliuchnikov *et al.* [Sel12, RS16, KMM16] leveraged techniques from number theory to obtain a polynomial improvement in the construction of single-qubit gates using a Clifford+T basis. Kliuchnikov [Kli13] further extended the number theoretic gate decomposition ideas to show that any n-qubit unitary operation can be approximated with precision ε using $O(4^n n \log(1/\varepsilon))$ Clifford and T gates. Note that Kliuchnikov's approach achieves a polynomial improvement over the Solovay-Kitaev theorem (with gate complexity $O(\text{polylog}(1/\varepsilon))$) in the complexity of the precision ε. However, it is not evident how can the efficient unitary constructions using number theory be extended to odd dimensions or qudits.

In this chapter, we study the continuous version of the Solovay-Kitaev theorem for building arbitrary unitary matrices using sequences of non-commuting Hamiltonian operations. Under certain assumptions, we introduce a constructive method to build any unitary transformation in $SU(d)$ with accuracy ε using a sequence of $O(d^2/\varepsilon)$ continuous operations. In contrast to Kliuchnikov's results [Kli13], our constructive approach can build unitaries in arbitrary dimensions.

3.3 Lie-Trotter-Suzuki Formulas

Product formula approximations have been widely popular in studying quantum simulations due to their accurate approximation of an operator exponential as a product of operator exponentials which can be easily implemented in the laboratory. These formulas can provide a high degree of precision in approximating a desired unitary since they can approximate a unitary operation with a sequence of simpler unitary operations. This particular feature makes them an ideal candidate for quantum computing applications.

The Lie-Trotter-Suzuki (LTS) formula is the most accurate known product formula approximation. The LTS formula approximates the unitary e^{-iHt} for $H = \sum_{j=1}^{m} H_j$ as a sequence of simpler unitaries of the form $\{e^{-iH_j t}\}_j$. Consider the following sequence,

$$\mathcal{S}_1(t) = \prod_{j=1}^{m} e^{-iH_j t}. \tag{3.2}$$

The Lie-Trotter method provides the following approximation

$$e^{-iHt} \approx (\mathcal{S}_1(t/r))^r, \tag{3.3}$$

where the equality holds in the limit of $r \to \infty$. The approximation error for a finite value of r is $\| e^{-iHt} - \mathcal{S}_1(t/r)^r \| = O((mt)^2/r)$ where m denotes the number of local Hamiltonians H_j that generates the dynamics of the quantum system. Since $\mathcal{S}_1$ is accurate to first order in t, the Lie-Trotter method is a first-order formula. The goal of approximating a unitary operation is to make the value of r large enough to achieve a desired accuracy.

In order to obtain more accurate error bounds by improving the scaling of the time parameter t, we avail higher order product formulas namely the LTS formula. The $2k$-th order LTS formula [Suz90, Suz91] provides a sequence that achieves an error of $O(t^{2k+1})$ for $k \geq 1$. The systematic construction of $2k$-th order formulas is as follows

$$\begin{aligned} \mathcal{S}_2(t) &= \prod_{j=1}^{m} e^{-iH_j t/2} \prod_{j=m}^{1} e^{-iH_j t/2}, \\ \mathcal{S}_{2k}(t) &= \mathcal{S}_{2k-2}(p_k t)^2 \mathcal{S}_{2k-2}((1-4p_k)t) \mathcal{S}_{2k-2}(p_k t)^2, \end{aligned} \tag{3.4}$$

where $p_k = (4 - 4^{1/(2k+1)})^{-1}$. Rigorous error analysis indicates that to approximate the local Hamiltonian dynamics governed by $H = \sum_{j=1}^{m} H_j$ with a desired accuracy ε, the first-order formula requires $r = O((mt)^2/\epsilon)$ while the $2k$-th order formula requires $r_{2k} = O((mt)^{1+1/(2k)}/\varepsilon^{1/(2k)})$.

The main advantage of the product formulas is they can provide a very high degree of accuracy and approximate a unitary operation $U(t) = e^{-iHt}$ with a product of simple unitary operations which can be easily implemented using a quantum computer. The disadvantage is that the product formulas require $O(5^k)$ exponentials to build an $O(t^{2k+1})$ approximation. Thus the number of constituent operations in the approximating sequence exponentially increases with the order k of the LTS formula. A significant improvement over existing product formula based quantum simulation algorithms would be to develop a product formula based algorithm which requires substantially fewer exponentials in the approximating sequence.

3.4 General Approach

We begin this section by defining the Gaussian unitary ensemble. Consider the complex $d \times d$ Hermitian matrix H where the elements H_{pq} are sampled from Gaussian distributions, i.e., $H_{pq} \sim N(0, \frac{1}{2}) + iN(0, \frac{1}{2})$ and $H_{pp} \sim N(0, 1)$ for $1 \leq p, q \leq d$.

Definition 3.4.1 *The Gaussian unitary ensemble (GUE) on the space of $d \times d$ Hermitian matrices H is defined by the Gaussian measure with probability density*

$$P(H) = \frac{1}{Z_d} e^{-\frac{1}{2} Tr\, H^2}.$$

Here Z_d is a normalization factor such that the integral of the probability density $P(H)$ in the space of Hermitian matrices is equal to one. Note that for the rest of the chapter, we would be working with Hermitian matrices A, B that are sampled from the Gaussian unitary ensemble.

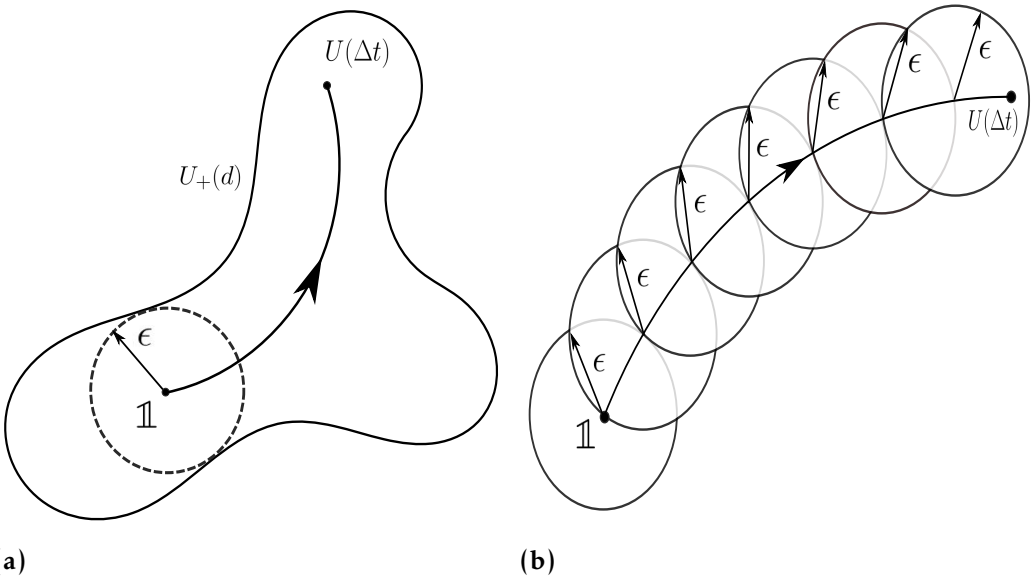


Figure 3.1: a) Largest ε-ball around the identity unitary $\mathbb{1}$ in the space of polynomial time reachable unitaries $U_+(d)$ in time Δt. b) Several ε-balls covering a path of finite length L in $U_+(d)$ from the identity unitary $\mathbb{1}$ to the final unitary $U(\Delta t)$.

In this section, we provide a sketch of our constructive approach for building an arbitrary unitary operation. Recall that we are building unitaries of the form $U(\vec{t}, \vec{\tau})$ by alternating applications of the exponentials of the matrices A, B given by

$$U(\vec{t}, \vec{\tau}) = e^{-iB\tau_N} e^{-iAt_N} \ldots e^{-iB\tau_1} e^{-iAt_1}. \tag{3.5}$$

Our results indicate that when $N = O(d^2)$, there exists an open ball of radius $\varepsilon > 0$ in the vicinity of the identity operator in the manifold $SU(d)$ such that one can construct a desired unitary e^{-iHt} where $\| H \|_1 = 1$ and $t \leq \varepsilon$, by a suitable choice of the parameters $\vec{t}, \vec{\tau}$ in Eq. (3.5). The size of the ε-ball depends on the non-commuting control Hamiltonians A, B that can be implemented using a quantum computer and the target Hamiltonian H. The essential point here is that the radius

of the ε-ball should be strictly bounded away from zero. In practice, because of the finite precision achievable in the parameters $\vec{t}, \vec{\tau}$, we can only reach the desired unitary approximately. The size of the ε-ball and the effects of such finite precision will be addressed below.

To be precise, we conjecture that it is possible to advance a non-zero distance along any direction in the algebra of Hamiltonians in $su(d)$. The target unitary that we aim to implement is given by $U = e^{-iHt}$ for $|t| \leq \pi$. We discuss two approaches for constructing an arbitrary unitary transformation, namely the non-infinitesimal construction and the infinitesimal construction. In order to approximate the target unitary $U \in U(d)$, we first find a sequence of gates or control parameters to realize $e^{-iH\varepsilon}$ using either the non-infinitesimal construction or the infinitesimal construction. We demonstrate how to construct $e^{-iH\varepsilon}$ when there is an explicit representation of the unitary U. Let $U_+(d)$ be the set of polynomial time (i.e., polynomial in the dimension d) reachable unitaries in $SU(d)$ starting from the identity. For elements belonging to the subgroup $U_+(d)$, there exists a path of finite length that connects the initial and final unitaries in finite time [LM14]. We can then simply repeat the sequence t/ϵ times to implement e^{-iHt}. In Fig. (3.1), we illustrate our general method applicable to both the non-infinitesimal and infinitesimal approaches to construct a desired unitary operation in the $SU(d)$ manifold.

In the non-infinitesimal method, as will be seen, we typically require $2d^2$ terms in the sequence in Eq. (3.1) to reach any unitary transformation in $SU(d)$ within a distance $O(\varepsilon)$ of the identity. In the infinitesimal case, we can build a specific set of unitaries (the infinitesimal approach does not enable to build any unitary transformation in time $O(d^2)$) in $SU(d)$ within $O(\varepsilon/\log d)$ of the identity using $N = O(d^2)$ terms. Note that ε is independent of the dimension d. Consequently, the non-infinitesimal method requires $N = O(d^2/\varepsilon)$ exponentials to construct a desired unitary U while in the infinitesimal case, we need $N = O(d^2 \log d/\varepsilon)$. Note that the non-infinitesimal method achieves the best possible scaling of the time complexity in the dimension d for constructing an arbitrary unitary operation.

3.5 Construction: Non-Infinitesimal Case

We begin this section by stating the assumptions required for constructing an arbitrary unitary transformation in $SU(d)$ via the non-infinitesimal method. The assumptions are the following: (1) we consider operators $A, B \in \mathcal{H}_d$ that are bounded

such that $\| A \|_1 = \| B \|_1 = 1$; (2) the operators A, B are drawn from the Gaussian Unitary Ensemble (GUE) so that the algebra generated by A, B via commutation is with probability one complete in $su(d)$ [RHR04, KRK$^+$05, RHR05, RHH$^+$06, CR07, MHR08, HDR09, BCR10, RTB$^+$15, RWSR17, RRW17]; (3) the operators $\pm A, \pm B$ can be implemented; equivalently, we can consider the parameters $\vec{t}, \vec{\tau}$ (in Eq. 3.1) to be positive or negative.

We aim to build an arbitrary unitary transformation in $SU(d)$ by a sequential application of the exponentials of the matrices A, B as in Eq. (3.1). The initial unitary can be determined by the identity transformation (where the parameters $\vec{t}, \vec{\tau}$ are set to zero in $U(\vec{t}, \vec{\tau})$) or a random point in the unitary manifold (for a random choice of the parameters $\vec{t}, \vec{\tau}$ in $U(\vec{t}, \vec{\tau})$). Note that in order to reach the target unitary transformation, one should be able to explore all linearly independent directions in the $SU(d)$ manifold.

We now state a conjecture for constructing a set of linearly independent basis elements in the Lie algebra $su(d)$ using d^2 parameters in the sequence $U(\vec{t}, \vec{\tau})$ in Eq. 3.1.

Conjecture 3.5.1 *Let $N = d^2/2$ and $k = \{1, 2, \ldots, N\}$. Let A and B be matrices in $\mathcal{H}_d$ drawn from GUE(d) and $U(\vec{t}, \vec{\tau}) = e^{-iB\tau_N} e^{-iAt_N} \ldots e^{-iB\tau_1} e^{-iAt_1}$. Then the set of elements $\{U^\dagger \partial U/\partial t_k, U^\dagger \partial U/\partial \tau_k\}_{\vec{t}, \vec{\tau}}$ are linearly independent and forms a basis in $su(d)$ for almost any $U(\vec{t}, \vec{\tau})$ in the unitary manifold.*

We make the following remark regarding the above conjecture. A random selection of the parameters $\vec{t}, \vec{\tau}$ for constructing the unitary $U(\vec{t}, \vec{\tau})$ (as in Eq. (3.1)) followed by a derivative with respect to one the parameters and then performing the inverse transformation allows us to move along any direction in the vicinity of the identity. However, there exists a set of parameters $\vec{t}, \vec{\tau}$ for which the sequence $U(\vec{t}, \vec{\tau})$ lives in a submanifold of dimension less than d^2. Such unitaries have less than d^2 independent parameters and the above conjecture cannot guarantee to build an arbitrary unitary via the sequence $U(\vec{t}, \vec{\tau})$ with less than d^2 parameters.

We have numerically verified Conjecture 3.5.1 for matrices A and B drawn from the Gaussian Unitary Ensemble (Definition 3.4.1) in $d = 2, 3, 4, 5$ dimensions. We discuss Conjecture 3.5.1 in the simplest case of $d = 2$ here. The numerical observations can be easily extended to higher dimensions. In $d = 2$, first we obtain the matrices A, B from the GUE(2) (such matrices can be obtained using in-built

commands in Mathematica to sample from the GUE distribution for arbitrary dimensions). Since $d^2 = 4$ parameters are required to describe any unitary matrix in $U(2)$, we select a random choice of the parameters $t_1, \tau_1, t_2, \tau_2 \in [-\pi, \pi]$. The unitary $U(\vec{t}, \vec{\tau}) = e^{-iB\tau_2} e^{-iAt_2} e^{-iB\tau_1} e^{-iAt_1}$ is then constructed using the matrices A, B and the parameters t_1, τ_1, t_2, τ_2. Using the sequence $U(\vec{t}, \vec{\tau})$, we can construct the basis elements of the Lie algebra $su(2)$ which are given by

$$e^{iAt_1} \cdot B \cdot e^{-iAt_1} \quad , \quad e^{iAt_1} e^{iB\tau_1} \cdot A \cdot e^{-iB\tau_1} e^{-iAt_1} \quad , \quad e^{iAt_1} e^{iB\tau_1} e^{iAt_2} \cdot B \cdot e^{-iAt_2} e^{-iB\tau_1} e^{-iAt_1}. \tag{3.6}$$

We have numerically verified the linear independence of the above basis elements in $su(2)$ for GUE matrices A, B. Further experiments were performed in higher dimensions to verify Conjecture 3.5.1 for constructing a spanning set of basis elements. We have been unable to numerically check the conjecture for dimensions $d \geq 6$ due to computer memory issues. However, we believe that Conjecture 3.5.1 holds true for dimensions $d \geq 6$.

One can also construct a spanning set of linearly independent elements in $su(d)$ using the partial derivatives of the sequence $U(\vec{t}, \vec{\tau})$ with respect to the parameters $\vec{t}, \vec{\tau}$. For matrices A, B drawn from the GUE(d) and $k = \{1, 2, \ldots, d^2/2\}$, the set of all partial derivatives $\{\partial U(\vec{t}, \vec{\tau})/\partial t_k, \partial U(\vec{t}, \vec{\tau})/\partial \tau_k\}_{\vec{t'}, \vec{\tau'}}$ spans the space of directions in the manifold of unitaries $SU(d)$ at the point $\vec{t'}, \vec{\tau'}$. We have

$$\frac{\partial U(\vec{t}, \vec{\tau})}{\partial t_k} = e^{-iB\tau_N} e^{-iAt_N} \ldots (-iA) \cdot e^{-iAt_k} \ldots e^{-iB\tau_1} e^{-iAt_1}. \tag{3.7}$$

A similar expression can be obtained for $\partial U(\vec{t}, \vec{\tau})/\partial \tau_k$. Thus, we can explore any direction in the space of unitaries in the neighbourhood of $U(\vec{t}, \vec{\tau})$ and hence in the neighbourhood of the identity after mapping back by exploiting the first order variation of $U(\vec{t}, \vec{\tau})$ with respect to $\vec{t}, \vec{\tau}$. If we have access to an explicit representation of a desired unitary $\widetilde{U}$, this technique allows us to move in the right direction towards the target unitary $\widetilde{U}$. However, because the gradients of $U(\vec{t}, \vec{\tau})$ form a spanning set in $su(d)$, we can also proceed along the right direction using gradient descent on the weight space $\vec{t}, \vec{\tau}$ to reduce the error function defined by the training set. This will be discussed in more details where the task is to learn a target unitary operator from a training set of input-output pairs.

Motivated by Conjecture 3.5.1, we ask how many linearly independent directions in the algebra of Hamiltonians in $\mathcal{H}_d$ can be explored when the matrices A and B correspond to nearest neighbour random local interactions. Specifically, for

a n-qubit physical system on a 1-dimensional lattice, we consider the following pair of Hamiltonians,

$$\mathcal{A} = A_{1,2} \otimes \mathcal{I}_{3\cdots n} + \mathcal{I}_{12} \otimes A_{3,4} \otimes \mathcal{I}_{5\cdots n} + \cdots + \mathcal{I}_{1\cdots n-2} \otimes A_{n-1,n}, \tag{3.8a}$$

$$\mathcal{B} = \mathcal{I}_1 \otimes B_{2,3} \otimes \mathcal{I}_{4\cdots n} + \mathcal{I}_{123} \otimes B_{4,5} \otimes \mathcal{I}_{6\cdots n} + \cdots + \mathcal{I}_{1\cdots n-3} \otimes B_{n-2,n-1} \otimes \mathcal{I}_n, \tag{3.8b}$$

where $A_{i,i+1}$, $B_{j,j+1}$ are two-qubit random traceless Hamiltonians sampled from the GUE and $\mathcal{I}$ is the identity matrix. Similar to the global random matrices discussed in Conjecture 3.5.1, local random matrices of the form $\mathcal{A}$ and $\mathcal{B}$ enables one to move along all $d^2 - 1$ linearly independent directions in the unitary manifold. We state the following conjecture regarding the local matrices $\mathcal{A}$ and $\mathcal{B}$.

Conjecture 3.5.2 *Let $d = 2^n$, $N = d^2/2$ and $k = \{1, 2, \ldots, N\}$. Let $A_{i,i+1}$ and $B_{j,j+1}$ be matrices drawn from GUE(4) for $1 \leq i, j \leq n$. Let $\mathcal{A} = \sum_{i=1}^{n-1} A_{i,i+1}$, $\mathcal{B} = \sum_{j=2}^{n-2} B_{j,j+1}$ and $U(\vec{t}, \vec{\tau}) = e^{-i\mathcal{B}\tau_N} e^{-i\mathcal{A}t_N} \ldots e^{-i\mathcal{B}\tau_1} e^{-i\mathcal{A}t_1}$. Then the set of elements $\{U^\dagger \partial U/\partial t_k\,,\, U^\dagger \partial U/\partial \tau_k\}_{\vec{t}',\vec{\tau}'}$ are linearly independent and forms a basis in $su(d)$ for almost any $U(\vec{t}', \vec{\tau}')$ in the unitary manifold.*

The above conjecture has been numerically verified for n = 2,3,4 qubits. We believe Conjecture 3.5.2 will also hold true for $n > 4$ qubits. We further remark that Conjecture 3.5.2 remains valid even when $\mathcal{A}$ and $\mathcal{B}$ are composed of local homogeneous Hamiltonians. By homogeneity, we mean that all local Hamiltonians $A_{i,i+1}$ and all $B_{j,j+1}$ represent one and the same two-qubit random local transformation.

We now elucidate the non-infinitesimal method for constructing an arbitrary unitary in $SU(d)$. Recall that we have access to a pair of non-commuting traceless Hamiltonians that can be implemented using a quantum computer. We then construct unitaries $U(\vec{t}, \vec{\tau})$ of the form in Eq. (3.1) by alternating applications of the Hamiltonians A, B. The key point of our argument is, generically $U(\vec{t}, \vec{\tau})$ explores a $2N$ dimensional manifold in the space of all unitaries for $2N \leq d^2$. When $2N \geq d^2$, there exists an ε-ball of reachable unitaries around a typical point in the manifold.

We conjecture that under certain assumptions, a random selection of the parameters $\vec{t}, \vec{\tau}$ can create any unitary within a distance ε of the identity using $N = O(d^2)$ steps. If an accidental choice of $\vec{t}, \vec{\tau}$ gives a lower dimensional manifold of transformations in the vicinity of $U(\vec{t}, \vec{\tau})$, then we discard this selection and perform a random selection of the parameters again. An ideal selection would be the parameters $\vec{t}, \vec{\tau}$ that maximizes the radius of the ε-ball of attainable transformations in the neighbourhood of the identity. In general, it is a hard problem to find

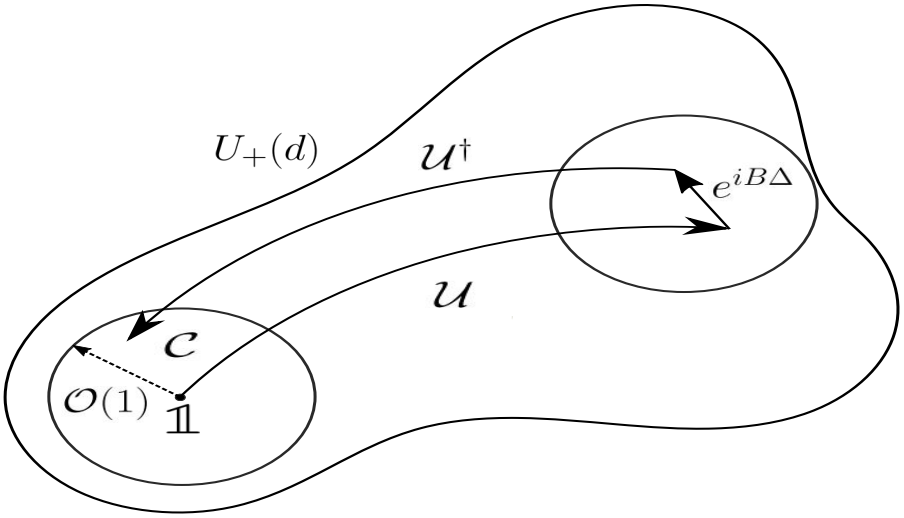


Figure 3.2: Non-infinitesimal case: Any unitary can be reached within an ε-ball around the identity where $\varepsilon = \mathcal{O}(1)$. First, we prefer a generic $\mathcal{U}(\vec{t},\vec{\tau})$. For a random choice of the parameters $\vec{t},\vec{\tau}$, the gradient of $\mathcal{U}(\vec{t},\vec{\tau})$ is typically non-zero in all directions. Consequently, we can reach an ε-ball of size $\mathcal{O}(1)$ in the vicinity of the unitary $\mathcal{U}(\vec{t},\vec{\tau})$. Mapping the ball back to the origin which is the identity unitary $\mathbb{1}$ allows us to attain any point within a distance $\varepsilon = \mathcal{O}(1)$ from the origin. The manifold $U_+(d)$ is the group of polynomial time reachable unitaries starting from the identity unitary $\mathbb{1}$.

the optimal choice of parameters $\vec{t},\vec{\tau}$. However, a random choice of the parameters $\vec{t},\vec{\tau}$ is adequate to move away from the origin. Now we move a distance $\Delta = O(1)$ along the direction of either A or B in the unitary manifold. In the final step, we map the reachable ε-ball back to the origin using the inverse transformation $U^\dagger(\vec{t},\vec{\tau}) = U(-\vec{t},-\vec{\tau}) = e^{iAt_1}e^{iB\tau_1}...e^{iAt_N}e^{iB\tau_N}$.

Conjecture 3.5.3 *Let $N = d^2/2$. Let A and B be matrices drawn from the GUE(d) and $U(\vec{t},\vec{\tau}) = e^{-iB\tau_N}e^{-iAt_N}\dots e^{-iB\tau_1}e^{-iAt_1}$. Furthermore, assume that we have access to a matrix B such that the spectrum of B is proportional to the logarithm of the spectrum of a desired unitary transformation. Then the sequence* $\exp(-i\mathcal{U}^\dagger(\vec{t},\vec{\tau})\cdot B\cdot\mathcal{U}(\vec{t},\vec{\tau})\Delta)$ *can realize any unitary at a non-zero distance Δ from the identity.*

The non-infinitesimal method enables us to realize an arbitrary unitary transformation $\mathcal{C}$ by implementing the following sequence

$$\mathcal{C} = \left(e^{iAt_1}e^{iB\tau_1}\dots e^{iAt_N}e^{iB\tau_N}\right)\cdot\left(e^{-iB\tau_N}e^{-iAt_N}\dots e^{-iB(\tau_k+\Delta)}e^{-iAt_k}\dots e^{-iB\tau_1}e^{-iAt_1}\right) \tag{3.9a}$$

$$= \underbrace{e^{iAt_1}e^{iB\tau_1}\dots e^{iAt_k}}_{\mathcal{U}^\dagger(\vec{t},\vec{\tau})}.e^{-iB\Delta}.\underbrace{e^{-iAt_k}\dots e^{-iB\tau_1}e^{-iAt_1}}_{\mathcal{U}(\vec{t},\vec{\tau})} \tag{3.9b}$$

$$= e^{-i\widetilde{B}\Delta} \text{ where } \widetilde{B} = \mathcal{U}^\dagger(\vec{t},\vec{\tau})\cdot B\cdot\mathcal{U}(\vec{t},\vec{\tau}). \tag{3.9c}$$

The non-infinitesimal method assumes that the we have access to a matrix with a spectrum that is a constant times the logarithm of the spectrum of the target unitary dynamics. The unitary $\mathcal{C}$ can reach any point in the unitary manifold within a distance $\Delta = O(1)$ of the identity using $2N = d^2$ exponentials in the sequence $U(\vec{t},\vec{\tau})$ of Eq. (3.1). The parameter Δ need not be small and can be less than or equal to π. This is because the generator $\widetilde{B}$ in Eq. (3.9c) can be expressed as a combination of d^2-1 linearly independent elements of the Lie algebra $su(d)$. In contrast to higher order product formulas or Trotterization, the non-infinitesimal method can explore all possible directions in the space of unitaries within the first order of the parameter Δ. Thus, by a suitable choice of the parameters $\vec{t},\vec{\tau}$ in the sequence $\mathcal{U}(\vec{t},\vec{\tau})$, the non-infinitesimal method allows us to move along any direction in the $SU(d)$ manifold towards the desired unitary. In Fig. (3.2), we have illustrated the details of each step in the non-infinitesimal technique.

An interesting feature of the unitary $\mathcal{C}$ in Eq. (3.9) is its resemblance with a physical quantity termed as the out-of-time-ordered correlator (OTOC). For hermitian or unitary operators V and W, the OTOC is defined as the the correlation function $F_t = \langle W_t^\dagger V^\dagger W_t V\rangle = \mathrm{Tr}(W_t^\dagger V^\dagger W_t V\rho)$ where ρ is the state of the physical system, $W_t = U^\dagger W U$ and U is the time evolution operator. OTOCs have been used to characterize the delocalization or scrambling of quantum information in strongly interacting many-body quantum systems via the exponential growth of the local Heisenberg operators. The ability to move along any direction in the Lie algebra $su(d)$ is equivalent to the ability in using our controls to perform scrambling or effective randomization of the many-body dynamics.

3.5.1 Learning unitaries from training data

In general, we do not have access to an explicit representation of the target unitary $\widetilde{U}$ that is approximated using a sequence $U(\vec{t},\vec{\tau})$ as in Eq. (3.1). Instead, we have access to a training set of input and output quantum states $\{|\psi_\ell\rangle, \widetilde{U}|\psi_\ell\rangle\}_{\ell=1}^M$ and the arbitrary target unitary $\widetilde{U}$ is considered as a black box. In such cases, one can perform gradient descent with the parameters $\vec{t},\vec{\tau}$ to optimize the error function $E(\vec{t},\vec{\tau}) = 1-(1/M)\sum_\ell \langle\psi_\ell|\widetilde{U}^\dagger U(\vec{t},\vec{\tau})|\psi_\ell\rangle$, where M is the number of training samples. Marvian and Lloyd [ML16] provides a general criterion for when such a training set can learn the target unitary $\widetilde{U}$ exactly. We emphasize that when there are $N = O(d^2)$ parameters in the sequence $U(\vec{t},\vec{\tau})$, then we can explore the full neighborhood of the identity to reduce the cost function $E(\vec{t},\vec{\tau})$.

Our aim is to minimize the cost function $E(\vec{t},\vec{\tau})$ by moving along the right direction towards the target unitary $\widetilde{U}$ such that $\| \widetilde{U}|\psi\rangle - U(\vec{t},\vec{\tau})|\psi\rangle \| \ll 1$. The first step is to randomly initialize the parameters $\vec{t},\vec{\tau}$ in the sequence $U(\vec{t},\vec{\tau})$ and perform gradient descent to optimize the error function $E(\vec{t},\vec{\tau})$. This method allows us to move a distance $O(\varepsilon)$ closer to the target unitary $\widetilde{U}$. There can be situations when for a given initial parameter choice, gradient descent optimization may not converge to the target unitary $\widetilde{U}$ because of the existence of saddle points in the weight space $\vec{t},\vec{\tau}$ of the error function $E(\vec{t},\vec{\tau})$. Having advanced as far as we can in the direction of $\widetilde{U}$ via the first sequence, we add another sequence $U(\vec{t},\vec{\tau})$ of depth $2N = O(d^2)$ by assigning a different initialization of the parameters and perform gradient descent again. Since we can explore any direction in the vicinity of the identity using the gradients $\{\partial U(\vec{t},\vec{\tau})/\partial t_k, \partial U(\vec{t},\vec{\tau})/\partial \tau_k\}_{\vec{t},\vec{\tau}}$ (which forms a basis in the Lie algebra $su(d)$ for random matrices A,B in the sequence $U((\vec{t},\vec{\tau}))$), this ensures that we can learn the target unitary $\widetilde{U}$ in $O(d^2/\varepsilon)$ time using gradient descent optimization.

3.6 Construction: Infinitesimal Case

The non-infinitesimal method elucidated in the previous section scales optimally with $N = O(d^2)$ exponentials required in the sequence $U(\vec{t},\vec{\tau})$. Traditional methods for building unitaries (for example, higher order product formulas) rely on infinitesimal methods using nested commutators [LB99, SL11, CW13]. Moreover, as noted before, the infinitesimal approach allows us to address the problem of power and band width limits in quantum control via the process of Trotterization. In this section, we demonstrate that such methods scale only slightly worse than the optimal which is $O(d^2)$. To be precise, the number of exponentials required in the sequence $U(\vec{t},\vec{\tau})$ to realize a unitary transformation near identity scales as $N = O(d^2 \log d)$.

Our assumptions for constructing unitary transformations using the infinitesimal method are similar to the non-infinitesimal method. For the clarity of the discussion, we restate the assumptions required for the infinitesimal method: (1) we consider operators $A,B \in \mathcal{H}_d$ that are bounded such that $\| A \|_1 = \| B \|_1 = 1$; (2) the operators A,B are drawn from the Gaussian Unitary Ensemble (GUE) so that the algebra generated by A,B via commutation is with probability one complete in $su(d)$ [RHR04, KRK$^+$05, RHR05, RHH$^+$06, CR07, MHR08, HDR09, BCR10,

RTB+15, RWSR17, RRW17]; (3) the operators $\pm A, \pm B$ can be implemented; equivalently, we can consider the parameters $\vec{t}, \vec{\tau}$ (in Eq. 3.1) to be positive or negative.

The basic method for constructing unitary transformations close to the identity is motivated by the Baker-Campbell-Hausdorff relation

$$e^{iBt}e^{iAt}e^{-iBt}e^{-iAt} = e^{[A,B]t^2} + O(t^3). \tag{3.10}$$

Repeated application of the above method enables the implementation of effective Hamiltonians which take the form of nested commutators of the operators A and B. For example, the kth order nested commutator is

$$[A,[A,[B,[A,[B,[\ldots[A,B]]\ldots], \tag{3.11}$$

where k is the number of commutators in the above $(k+1)$th degree homogeneous polynomial. The number of exponentials required in the sequence $U(\vec{t}, \vec{\tau})$ to implement a kth degree nested commutator is $N = O(2^k)$. Additionally, there are a total of $O(2^k)$ nested commutators till order k. Thus, when $2^k = O(d^2)$ for $k = 2\log d + c$ and $c = O(1)$, there are potentially $d^2 - 1$ linearly independent nested commutators to span all directions in the $su(d)$ algebra in the neighbourhood of the identity operator. The coefficient of the kth order nested commutator is t^k and the norm of the higher order terms determines the error in approximating the target unitary operator. In the rest of the chapter, the base of the logarithm is set to 2.

It is important to note that not all of the nested commutators are linearly independent. For example, we have the relation $[A,[B,[A,B]]] = -[B,[A,[A,B]]]$. There are two sources of redundancies (or linear dependence) in the kth order nested commutators. First, the redundancies that are due to the nested commutators of lower degrees. Second, there can be internal redundancies in between nested commutator of a certain order. An example of a internal redundancy at sixth order is $[B[A[A[B[A,B]]]]] = -[A[B[A[B[B,A]]]]] + \frac{1}{3}\Big([A[A[B[B[B,A]]]]] + [B[B[A[A[A,B]]]]]\Big)$, which is independent of redundancies due to nested commutators of lower orders. Further examples of higher order linearly independent nested commutators have been provided in Appendix B.

In light of the redundancies occurring at a certain degree, we are interested in finding the actual number of linearly independent nested commutators at order k. To this end, we use a theorem due to Witt.

Theorem 3.6.1 ([Wit56]) *Suppose $L(X)$ is a free Lie algebra on a q element set X and a_k be the dimension of the homogeneous part of degree k of $L(X)$. Then $a_k = \frac{1}{k}\sum_{\mathfrak{d}/k}\mu(k/\mathfrak{d})(q^{\mathfrak{d}}-1)$.*

The important point here is, a_k is the number of potentially linearly independent nested commutators of order k. Here μ is the Möbius function, q is the cardinal number of the generating set X, $\mathfrak{d}$ is the mth divisor of k and the summation is over all divisors $\mathfrak{d}$ of k. The interested reader can learn more about a free Lie algebra in Appendix A. A standard reference for free Lie algebras is the work of Reutenauer [Reu01, Bac01]. Although the expression in Witt's theorem is rather complicated, it immediately implies the following simple result.

Corollary 3.6.2 *Suppose $L(X)$ is a free Lie algebra on a q element set X. The cardinality of the spanning set of nested commutators of degree k of the free Lie algebra $L(X)$ scales as $O(q^k)$.*

Consequently, the total number of potentially linearly independent nested commutators till order k that can be constructed from the random matrices A, B scales as $O(2^k)$. We conjecture that for operators A, B sampled from the GUE(d), the nested commutators till degree $k = O(\log d)$ which forms the basis elements of the free Lie algebra L are actually linearly independent in the Lie algebra $su(d)$. Thus for random operators A and B, we can obtain all d^2-1 basis elements in $su(d)$ from the spanning set of the first kth degree nested commutators where $k = 2\log d + c$ and $c = O(1)$. We state the following conjecture below.

Conjecture 3.6.3 *Suppose A and B are matrices in $\mathcal{H}_d$ drawn from the GUE(d). Then the set of all linearly independent nested commutators in the free Lie algebra of A and B are linearly independent in $su(d)$.*

We have verified the above conjecture numerically for dimensions up to 20 and order k up to 11. The number of linearly independent nested commutators can be further reduced if the operators A and B have a specific form. For example, the operators A and B can be lower order polynomials of the Pauli operators. In all the cases that we have investigated, however, the number of linearly independent nested commutators of degree k scales as $O(2^k)$. Additional lack of linear independence simply increases the constant c. We have not come across an example of a set of matrices that generates the full Lie algebra by commutation but that fails to

generate a linearly independent spanning set of operators via nested commutators within order $k = O(\log d)$.

The constructive infinitesimal procedure for implementing a desired unitary is as follows. We choose the order k satisfying $2^k = O(d^2)$ such that a spanning set for the Lie algebra $su(d)$ is generated by commutators within order k of the sequence $U(\vec{t}, \vec{\tau})$. It takes $N = O(2^k)$ transformations to enact nested commutators till order $k = O(\log d)$. Recall that the number of exponentials required in the sequence $U(\vec{t}, \vec{\tau})$ to implement a kth degree nested commutator is $N = O(2^k)$. The nested commutators at order k occur with coefficients which scales as $O(t^k)$. The higher order terms beyond order k generate transformations that are spanned by the lower order nested commutators. That is, using the infinitesimal method, we can generate a specific set of unitaries in the vicinity of the identity. Accordingly, there exists an ε-ball in the neighbourhood of identity where a specific set of unitaries within the ball can be implemented using $N = O(d^2)$ transformations.

Note that the obtainable ε-ball is larger than t^k but the constructive method only allows us to build a particular unitary transformation explicitly with coefficient t^k. The size of the actual ε-ball reachable at order $k = O(\log d)$ is given by $\varepsilon_f = \varepsilon^k$. We rescale the size of the ε-ball by $\tilde{\varepsilon} = |\log \varepsilon|$ such that $\tilde{\varepsilon}$ is small. This gives the following relation $\tilde{\varepsilon} = \tilde{\varepsilon}_f / \log d$ which implies that a minimum time of $O(d^2 \log d / \tilde{\varepsilon}_f)$ is required to implement a finite unitary transformation. Thus we are able to build a subset of all possible unitaries in $SU(d)$. It would be interesting to quantify how does the measure of the space of unitaries generated via the infinitesimal approach in time $O(d^2 \log d / \tilde{\varepsilon}_f)$ scales with the dimension of the Hilbert space $\mathcal{H}_d$.

3.7 Applications

In quantum control, one considers a time-dependent Hamiltonian of the form $H = H_0 + \gamma(t) H_c$ where H_0 and H_c are the drift and control Hamiltonians, and $\gamma(t)$ is a time-dependent control field [LM14]. The objective is to implement unitaries in $SU(d)$ via the non-infinitesimal method using bounded Hamiltonians of the form $H_0 + \gamma(t) H_a$ and $H_0 + \gamma(t) H_b$. We make the following assumptions. First, the inverse transformation of the drift Hamiltonian H_0 (i.e., the inverse unitary $e^{iH_0 t}$) cannot be implemented. Second, $[H_0, H_a] \neq 0$, $[H_0, H_b] \neq 0$ and $[H_a, H_b] \neq 0$. Third, the

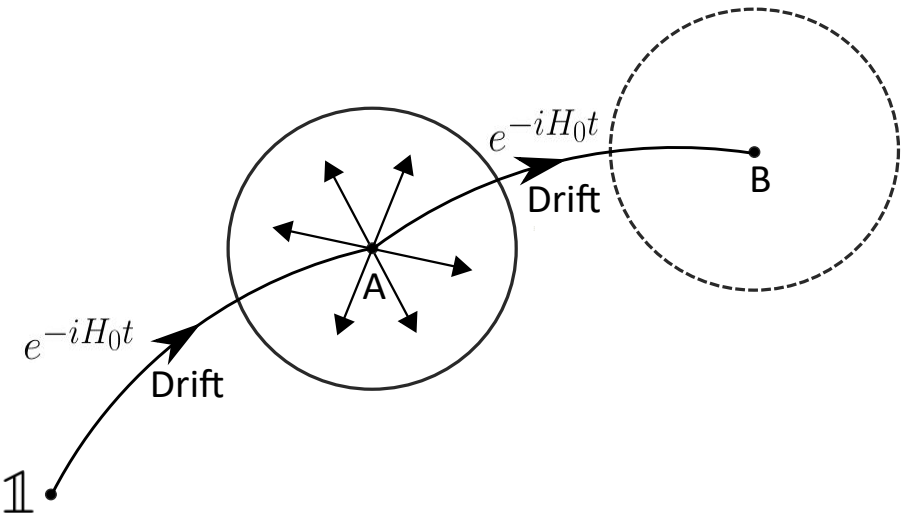


Figure 3.3: Unitary evolutions that can be realized in the presence of a drift Hamiltonian H_0.

control field parameter $\gamma(t)$ is fixed when the parameters $\vec{t}, \vec{\tau}$ are small. Fourth, the algebra generated by H_a and H_b via commutation is complete in $su(d)$.

We implement unitary transformations generated by the Hamiltonians $H_0 + \gamma(t)H_a$ and $H_0 + \gamma(t)H_b$ according to Eq. (3.9a). This technique generates $d^2 - 1$ directions in the first order such as

$$e^{i(H_0+\gamma_0 H_a)t_1} e^{i(H_0+\gamma_0 H_b)\tau_1} \dots e^{i(H_0+\gamma_0 H_a)t_N} H_b e^{-i(H_0+\gamma_0 H_a)t_N} \dots e^{-i(H_0+\gamma_0 H_b)\tau_1} e^{-i(H_0+\gamma_0 H_a)t_1}, \tag{3.12}$$

where for simplicity, we have set $\gamma(t) = \gamma_0$ for small times. The parameters $\vec{t}, \vec{\tau}$ need to be small for bounding the error terms such as $[H_0, H_a]$, $[H_0, H_b]$ and their higher order counterparts. Since the parameters $\vec{t}, \vec{\tau}$ are small, we can reach a specific set of unitaries in the vicinity of the identity in $O(d^2)$ steps. It would be interesting to classify the unitaries that are constructed by the Hamiltonians with a drift term H_0 in time $O(d^2)$. If the assumptions $[H_0, H_a] \neq 0$ and $[H_0, H_b] \neq 0$ are relaxed such that the drift term H_0 commutes with the control fields H_a and H_b, then we can explore any direction in SU(d) in the vicinity of the point A as illustrated in Figure 3.3. Note that when the parameters $\vec{t}, \vec{\tau}$ are not small, we need to implement the time ordered exponential

$$\mathcal{T} \exp\left\{ i \int_{t=0}^{t=T} (H_0 + \gamma(t)H_a) dt \right\} = \prod_{j=1}^{m} \exp\left\{ i\Big(H_0 + \gamma_j H_a\Big)\Delta t \right\}, \tag{3.13}$$

where $\mathcal{T}$ is the time-ordering operator and $T = m\Delta t$. Similarly for $H_0 + \gamma(t)H_b$. The number of steps now would scale as $O(md^2)$ to construct any unitary near the

identity. Since the $SU(d)$ group is compact, any desired unitary can be constructed using the non-infinitesimal approach.

Chapter 4

Learning unitary operations using gradient descent optimization

The efficient implementation of unitary operations discussed in Chapter 3 provides upper bounds on the minimum time required to implement a desired unitary transformation on a d dimensional Hilbert space when applying sequences of Hamiltonian transformations. The previous chapter discusses strategies of building a desired unitary operation from a sequence using the non-infinitesimal and infinitesimal approaches, or equivalently, using power and band limited controls, can yield the best possible scaling in time $O(d^2)$. However, it does not address the problem of finding the optimal controllable parameters in a non-infinitesimal or infinitesimal sequence to approximate the target unitary. The task of finding efficient optimization strategies to obtain the optimal parameters in alternating operator sequences such as the non-infinitesimal and infinitesimal approaches can pave the way for implementing a desired unitary operation using near term quantum computers.

A promising and widely studied optimization strategy is the gradient descent method. Gradient descent algorithms are first order optimization methods and they are of particular interest to the machine learning community. The primary reason for their prevalence is greater computational efficiency of gradient descent over second order techniques such as L-BFGS in achieving convergence in convex spaces. Gradient descent techniques have found numerous applications in the machine learning community to optimize a cost function by finding the optimal set of parameters using an iterative procedure. In quantum computation and quantum control, gradient descent optimization has been employed to find the optimal

control parameters in a sequence of simple parameterized unitaries that can approximate a desired unitary transformation.

A fundamental task in both quantum computation and quantum control is to determine the minimum amount of resources required to implement a desired unitary transformation. In this chapter, we present a simple model that allows us to analyze some of the key aspects of implementing unitaries in the context of both quantum circuits and quantum control. In particular, we implement unitaries using alternating operator sequences of the form $e^{-iAt_K}e^{-iB\tau_K}\dots e^{-iAt_1}e^{-iB\tau_1}$. A sequence is parameterized by the times $\{t_1,\tau_1,\dots,t_K,\tau_K\}$ which need not be small. The approach of considering sequences of parameterized unitaries to approximate a target unitary forms the basis of the quantum approximate optimization algorithm (QAOA) [FGG14a, FGG14b]. It represents one of the simplest possible settings for investigating problems of controllability using a quantum computer.

In the literature, the acronym QAOA is also used to refer to the phrase *Quantum Alternating Operator Ansatz*. Farhi and Harrow have demonstrated the quantum supremacy of constant depth QAOA unitaries [FH16] by arguing that the output distribution of even the simplest version of the QAOA cannot be efficiently simulated on a classical computer. Recently, it has been shown that quantum alternating operator unitaries can perform universal quantum computation [Llo18]. In the infinitesimal time setting, QAOA also encompasses the more general problem of the application of time varying quantum controls [RHR04, KRK$^+$05, RHR05, RHH$^+$06, CR07, MHR08, HDR09, BCR10, RTB$^+$15, RWSR17, RRW17]. The above results inspire the study of the quantum alternating operator formalism as a general framework for performing arbitrary unitary transformations.

4.1 Overview of our results

We study the hardness of learning unitary transformations in $U(d)$ by performing gradient descent on the time parameters of sequences of alternating operators. In general, the loss function landscape of alternating operator sequences in $U(d)$ is highly non-convex, and standard gradient descent can fail to converge to the global minimum in such spaces. We find that unsurprisingly, when the number of parameters in the sequence is less than d^2, gradient descent fails to learn an arbitrary unitary operation. Initially, we had expected that because of the highly non-convex nature of the loss landscape, when the number of parameters in the

sequence was greater than or equal to d^2 – the minimum number of parameters required to specify a $d \times d$ unitary matrix – gradient descent would sometimes fail to learn the target unitary. However, our numerical experiments reveal the opposite.

In this chapter, we provide numerical evidence that – despite the highly non-convex nature of the control landscape – when the alternating operator sequence contains d^2 or more parameters, gradient descent always converges to the target unitary. The rates of convergence provide evidence for a *computational phase transition* at the critical point (where the number of parameters in the sequence equals d^2) between the under-parameterized and over-parameterized domains. When the number of parameters is less than d^2, gradient descent converges to a sub-optimal solution. When the number of parameters is greater than d^2, gradient descent converges rapidly and exponentially to an optimal solution. At the computational critical point where the number of parameters in the alternating operator sequence equals d^2, the rate of convergence is polynomial with a critical exponent of approximately 1.25.

The learning paradigm for building a desired unitary is as follows. Suppose we have knowledge of the entries of a unitary $\mathcal{U} \in U(d)$ and access to the Hamiltonians $\pm A$ and $\pm B$. Recent work has provided a constructive approach to build a learning sequence $\mathcal{V}(\vec{t}, \vec{\tau}) = e^{-iAt_K} e^{-iB\tau_K} \dots e^{-iAt_1} e^{-iB\tau_1}$ that can perform any target unitary $\mathcal{U}$ where $K = O(d^2)$ [LM19]. In this chapter, we ask whether optimal learning sequences for performing the target unitary $\mathcal{U}$ can be obtained by using gradient descent optimization on the parameters $\vec{t}, \vec{\tau}$ of $\mathcal{V}(\vec{t}, \vec{\tau})$. The matrices A, B are sampled from the Gaussian Unitary Ensemble (GUE) so that the algebra generated by A, B via commutation is with probability one complete in $u(d)$, i.e., the system is controllable [RHR04, KRK$^+$05, RHR05, RHH$^+$06, CR07, MHR08, HDR09, BCR10, RTB$^+$15, RWSR17, RRW17]. The parameters $\vec{t}, \vec{\tau}$ represent the times for which the generators of $\mathcal{V}(\vec{t}, \vec{\tau})$ are applied. We assume the operators $\pm A, \pm B$ can be implemented; equivalently, we can consider $\vec{t}, \vec{\tau}$ to be positive or negative. Note that this problem formulation lies in the domain of quantum optimization algorithms such as the Quantum Approximate Optimization Algorithm [FH16, JRW17, ZWC$^+$18, GAW19], the Variational Quantum Eigensolver [MRBA16, PMS$^+$14a, KLP$^+$19, SKCC19], and the Variational Quantum Unsampling [CMO$^+$20] where one varies the classical parameters in a quantum circuit to minimize an objective function.

In general, the control landscape for learning the target unitary $\mathcal{U}$ is highly non-convex [RHR04, KRK+05, RHR05, RHH+06, CR07, MHR08, HDR09, BCR10, RTB+15, RWSR17, RRW17]. Gradient descent algorithms do not necessarily converge to a globally optimal solution in the parameters of a non-convex space [ZRS+18]. In non-convex loss landscapes, gradient descent frequently converges to some undesired critical point. We study how hard it is to learn an arbitrary unitary with the quantum alternating operator formalism via gradient descent. We quantify the hardness of learning a unitary with the minimum number of parameters required in the sequence $\mathcal{V}(\vec{t},\vec{\tau})$ to perform the unitary $\mathcal{U}$. Since $\mathcal{U}$ has d^2 independent parameters, in general, at least d^2 parameters in the sequence $\mathcal{V}(\vec{t},\vec{\tau})$ are required to learn a unitary $\mathcal{U} \in U(d)$ within a desired error. Nevertheless, the non-convex loss landscape suggests that it might not be possible to learn an arbitrary $\mathcal{U}$ with gradient descent using $O(d^2)$ parameters. Our work numerically shows that exactly d^2 parameters in the sequence $\mathcal{V}(\vec{t},\vec{\tau})$ suffice to learn an arbitrary unitary $\mathcal{U}$ to a desired accuracy.

We also study the case of learning *shallow* target unitaries of the form $\mathcal{U}(\vec{t},\vec{\tau}) = e^{-iAt_N}e^{-iB\tau_N}\dots e^{-iAt_1}e^{-iB\tau_1}$ where the number of parameters in the target unitary is $2N \ll d^2$. For example, the simplest such target unitary is a depth-1 sequence $\mathcal{U}(t,\tau) = e^{-iAt}e^{-iB\tau}$. Such unitaries are, by definition, attainable via a shallow depth alternating operator sequence, and we look to see if it is possible to use gradient descent to obtain a learning sequence $\mathcal{V}(\vec{t},\vec{\tau})$ of the same depth that approximates the target unitary $\mathcal{U}(\vec{t},\vec{\tau})$. That is, we study the alternating operator version of whether it is possible to learn the unitaries generated by shallow quantum circuits. We find that gradient descent typically requires d^2 parameters in the sequence $\mathcal{V}(\vec{t},\vec{\tau})$ to learn even a depth-1 unitary. This result suggests that gradient descent is not an efficient method to learn low depth unitaries.

4.2 Related works

Methods for parameterizing and learning unitary matrices have found applications in the deep learning literature. Unitary matrices have been proposed as a natural solution to a common challenge in deep neural networks associated with vanishing or exploding gradients, which arise when large or small eigenvalues of the matrices grow or diminish exponentially with increasing powers of the matrices. This problem is most notable in the case of recurrent neural networks where outputs are fed back into neurons as input [PMB13]. Recent studies

have introduced methods to parameterize and integrate unitary matrices in neural networks to more naturally avoid the issue of vanishing or exploding gradients [ASB16, JSD+17].

In quantum computation, the Quantum Approximate Optimization Algorithm (QAOA) [FGG14a, FGG14b] has sparked a wave of interest in developing quantum-classical algorithms to solve combinatorial problems in time that is polynomially better (might be exponentially better in some instances) than their classical counterparts. The goal of QAOA is to alternatingly apply a pair of non-commuting Hamiltonians on a quantum system and drive the system of interest to the ground state of one of the Hamiltonians by varying the associated parameters with each of the Hamiltonians. In other words, a depth-p QAOA consists of $2p$ interleaved unitary operations generated by a pair of non-commuting Hamiltonians. The optimal set of parameters is determined by a classical optimization algorithm such as gradient descent. It is interesting to note that the output of QAOA is not efficiently classically simulatable even for the lowest circuit depth with $p = 1$ [FH16].

A recent study by Zhou et al. [ZWC+18] considers the applicability of depth-p QAOA ($1 < p < \infty$) on near-term quantum machines to solve the MaxCut problem. The main hurdle in analyzing intermediate depth QAOA circuits lies in the difficulty to efficiently optimize in the non-convex and high dimensional loss landscape. The authors developed heuristic approaches to efficiently optimize the variational parameters of QAOA. Furthermore, they found that the heuristic optimization strategies can find the global optimal solution in the parameter landscape in time $O(\mathrm{poly}\, p)$ which cannot be efficiently surpassed by the best known classical techniques.

In quantum control, Rabitz et al. considered the case of controllable quantum systems with time varying controls, including systems with drift (i.e., the Hamiltonian driving the system dynamics consists of a drift term), and show that when the controls are unconstrained (the space of controls is essentially infinite dimensional), there are no sub-optimal local minima even for non-convex loss landscapes [RHR04, RHR05, RRW17]. For example, it has been shown by Ho et al. [HDR09] that non-convexity in the loss landscape of fully controllable quantum systems with infinite dimensional control fields is due to the presence of non-trapping saddle points in the loss landscape. When the sequence of controls is finite dimensional, prior studies involving systems with drift sometimes find traps in the

control landscape [MHR08, RTB+15, RWSR17]. Here, we look at the simplest possible case where the system does not have drift and the space of controls is finite dimensional. Our numerical results show that even in non-convex spaces where the dimension of the system (and the dimension of the loss landscape) is the minimum it can be to attain the desired unitary, the control landscape still contains no sub-optimal local minima and gradient descent obtains the global optimal solution.

4.3 Experiments for learning an arbitrary unitary

In this section, we are interested in the hardness of learnability of an arbitrary unitary with the quantum alternating operator formalism via gradient descent optimization. Before delving into the details of learning a general unitary transformation $\mathcal{U} \in U(d)$, we first define what does it mean to learn the target unitary $\mathcal{U}$ using a quantum alternating operator sequence.

Definition 4.3.1 (Learnability of unitaries) *Fix $d > 0$. Let $\{t_1, \tau_1, \cdots, t_K, \tau_K\} \in \mathbb{R}$ and $M(d)$ be the set of $d \times d$ complex valued matrices. Let $\mathcal{U}$ be a unitary in the manifold $U(d)$ and $\mathcal{V}(\vec{t}, \vec{\tau}) = e^{-iAt_K} e^{-iB\tau_K} \dots e^{-iAt_1} e^{-iB\tau_1}$ be a sequence generated by the matrices $A, B \in M(d)$. We say the sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ learns the unitary $\mathcal{U}$ with a desired accuracy ε if the optimal parameters $\vec{t'}, \vec{\tau'}$ can be obtained by gradient descent such that*

$$\| \mathcal{U} - \mathcal{V}(\vec{t'}, \vec{\tau'}) \|^2 \leq \varepsilon. \tag{4.1}$$

We present numerical experiments that aim to learn an arbitrary unitary $\mathcal{U} \in U(d)$ by constructing a sequence $\mathcal{V}(\vec{t}, \vec{\tau}) = e^{-iAt_K} e^{-iB\tau_K} \dots e^{-iAt_1} e^{-iB\tau_1}$ and performing gradient descent on all $2K$ parameters to minimize the loss function $L(\vec{t}, \vec{\tau}) = \| \mathcal{U} - \mathcal{V}(\vec{t}, \vec{\tau}) \|^2$. Here $\| \cdot \|$ denotes the Frobenius norm which will be used in the rest of the chapter. Given access to the entries of a Haar random target unitary $\mathcal{U}$, we fix the number of parameters $2K$ and ask how many gradient descent steps S are required to construct the sequence $\mathcal{V}(\vec{t}, \vec{\tau}) = e^{-iAt_K} e^{-iB\tau_K} \dots e^{-iAt_1} e^{-iB\tau_1}$ that can learn the target unitary $\mathcal{U}$ to a desired accuracy or loss.

Initially, we had expected that learning an arbitrary unitary in $U(d)$ with an alternating operator sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ would require $K = O(d^s)$ parameters where $s > 2$. However, we present numerical evidence that with d^2 parameters in the sequence $\mathcal{V}(\vec{t}, \vec{\tau})$, any selected Haar random unitary $\mathcal{U} \in U(d)$ can be learned using (vanilla) gradient descent. We emphasize this is not an obvious result since the loss

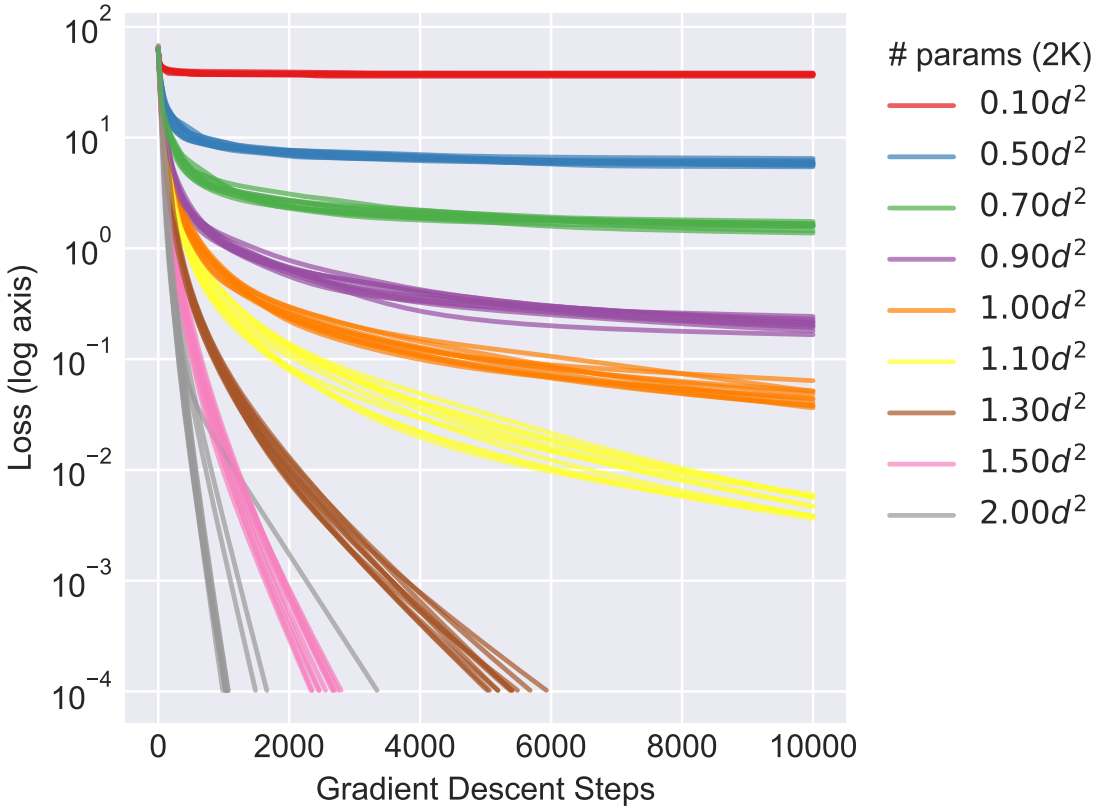


Figure 4.1: Gradient descent experiments for a Haar random target unitary $\mathcal{U}$ of dimension 32. The logarithm of the loss function $L(\vec{t},\vec{\tau})$ with increasing gradient descent steps for learning sequences $\mathcal{V}(\vec{t},\vec{\tau})$ with $2K$ parameters.

landscape over the control parameters can be highly non-convex and the ratio of saddle points to local minima increases exponentially with the dimension d. The details of the numerical analysis are provided below.

We ran experiments for a Haar random target unitary $\mathcal{U}$ of dimension 32 while varying the number of parameters $2K$ in $\mathcal{V}(\vec{t},\vec{\tau})$. At each step, we compute the gradients $\nabla_t L$, $\nabla_\tau L$ with respect to the $2K$ parameters in $\vec{t},\vec{\tau}$ and perform simple (or vanilla) gradient descent with a fixed learning rate of 0.001. Note that we use simple gradient descent optimization instead of the more sophisticated versions of gradient descent (which includes a momentum term). This is because we are interested in understanding the hardness of learning unitaries using the simplest form of first order optimization which is devoid of any hyperparameter.

In Fig. (4.1), we plot the loss $L(\vec{t},\vec{\tau})$ as a function of the number of gradient de-

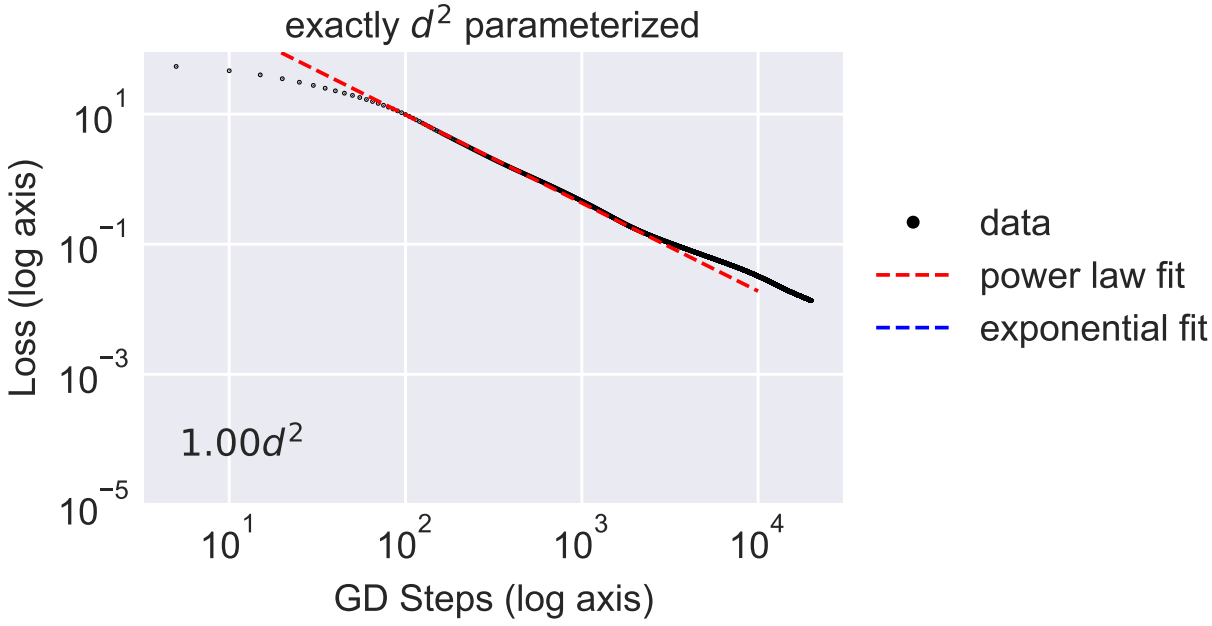


Figure 4.2: Gradient descent experiments for a Haar random target unitary $\mathcal{U}$ of dimension 32 exhibit a power law convergence in the first 1,000 gradient descent steps (best fit line shown in dashed red in the plot). In the critical case, $2K = d^2$, the power law persists throughout the gradient descent providing further evidence for a computational phase transition.

scent steps S for learning sequences $\mathcal{V}(\vec{t}, \vec{\tau})$ of varying depth K. When the sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ is under-parameterized with $2K < d^2$ parameters, we find that the loss function $L(\vec{t}, \vec{\tau})$ initially decreases but then plateaus. Thus, in the under-parameterized loss landscape, we find that as expected, with high probability, the gradient descent algorithm reaches a sub-optimal value of the loss which cannot be decreased by further increasing the number of gradient descent steps.

When the number of parameters $2K$ in $\mathcal{V}(\vec{t}, \vec{\tau})$ is equal to d^2 or more, we find that gradient descent *always* converges to the target unitary. This provides support to the claim that there are apparently no sub-optimal local minima in the loss landscape $L(\vec{t}, \vec{\tau})$ with more than d^2 parameters. As noted above, this result was unexpected given the non-convex nature of the loss landscape. We also find that the rate of convergence increases with the degree of over-parameterization as shown in Fig. (4.1). At the critical point where the number of parameters $2K = d^2$, we note the existence of a *computational phase transition*. When $2K = d^2$, the learning process converges to the desired target unitary, but the rate of convergence becomes very slow. For each parameter manifold of dimension $0.1d^2 \leq 2K \leq 2d^2$,

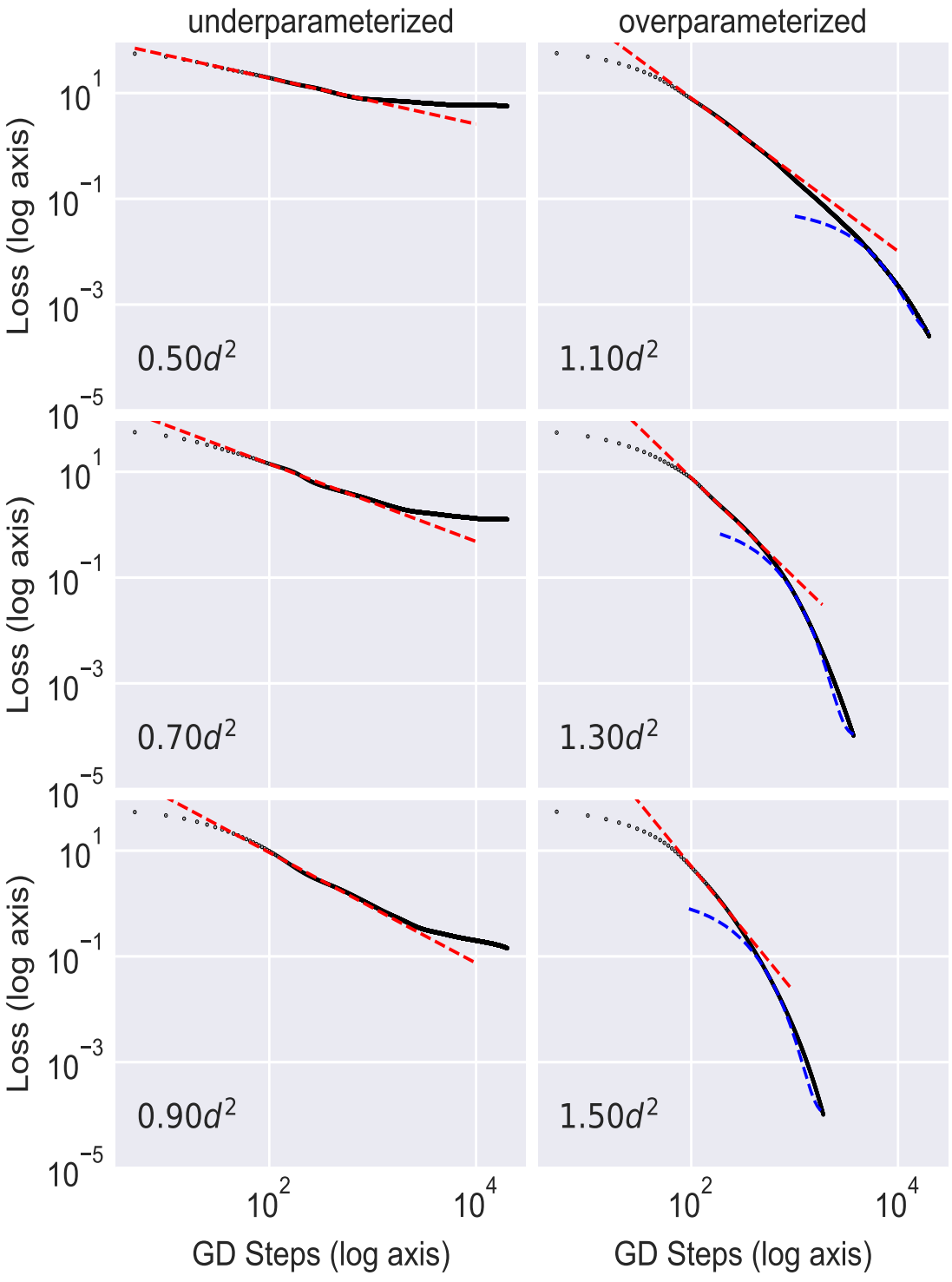


Figure 4.3: Gradient descent experiments for a Haar random target unitary $\mathcal{U}$ of dimension 32 exhibit a power law convergence in the first 1,000 gradient descent steps (best fit line shown in dashed red in the plot). In the under-parameterized case, at a certain point, the gradient descent plateaus at a sub-optimal local minimum. In the over-parameterized case, after the power law regime, the gradient descent enters an exponential regime consistent with a quadratic form for the loss function in the vicinity of the global minimum (best fit line shown in dashed blue in the plot).

Figure 4.4: The power law rate of gradient descent α for the initial 1000 gradient descent steps grows linearly with the number of parameters ($2K$). The first 50 steps have been excluded in the fit. The slope of the best fit line is 1.9. The computational phase transition takes place at a value of $\alpha \approx 1.25$.

we performed ten experiments and each experiment has been plotted in Fig. (4.1).

In Fig. (4.2), we fit the loss $L(\vec{t}, \vec{\tau})$ over the first 1000 gradient descent steps (the first 50 steps are excluded) to a power law

$$L = C_0 S^{-\alpha} + C_1, \tag{4.2}$$

where C_0 and C_1 are constants, $L = L(\vec{t}, \vec{\tau})$ and S is the number of gradient descent steps. As shown in Fig. (4.2), the data for the initial 1000 gradient descent steps fits closely to the power law in Eq. (4.2). However, with the exception of the critical learning sequence with $2K = d^2$ parameters, the performance of gradient descent deviates from a power law fit at later steps. For the under-parameterized case, the gradient descent plateaus at a sub-optimal value of the loss. For the over-parameterized case, the power law transitions to an exponential as the gradient

descent approaches the global minimum, which is consistent with the expected quadratic form of the loss function in the vicinity of the global minimum.

In Fig. (4.3), the exponential fit for the later stages of gradient descent in the over-parameterized setting is plotted. The exponential fit takes the form

$$L = C_0 e^{-r(S-S_0)} + C_1, \tag{4.3}$$

where C_0, C_1, r, and S_0 are constants (optimized during the fit), $L = L(\vec{t}, \vec{\tau})$ and S is the number of gradient descent steps. The critical case of the sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ with exactly d^2 parameters is consistent with a power law rate of convergence to the target unitary $\mathcal{U}$ during the entire gradient descent process.

The initial power law form of the gradient descent is consistent with a loss landscape that obeys the relation $\Delta L/\Delta S \propto -S^{-(\alpha+1)}$ and $\alpha \geq 0$. For example, the case $\alpha = 1$ corresponds to a power law of the form $\Delta L/\Delta S \propto -S^{-2}$. The final exponential form of convergence corresponds to the case $\alpha \to \infty$, and to a quadratic landscape where $\Delta L/\Delta S \propto -e^{-S} \propto -L$. The fitted value of α in the initial power law regime is plotted as a function of the number of parameters in Fig. (4.4). Here, we observe a linear relationship between the power law exponent α in Eq. (4.2) and the number of parameters $2K$ in $\mathcal{V}(\vec{t}, \vec{\tau})$. This indicates the following. The larger the degree of over-parameterization, the faster the rate of convergence, and the larger the exponent in the power law.

We make a remark here regarding the computational phase transition. Let $y = \log L$ and $x = \log S$. Beyond the first 1000 gradient descent steps in Fig. (4.3), observe that the sign of the second derivative, $\text{sign}(\partial^2 y/\partial x^2) = +1$ for the under-parameterized sequences while $\text{sign}(\partial^2 y/\partial x^2) = -1$ for the over-parameterized models. The learning sequence with exactly d^2 parameters grazes along the power law curve where $\text{sign}(\partial^2 y/\partial x^2) = 0$. This signals the onset of a *phase transition* from the under-parameterized to the over-parameterized sequences in learning an arbitrary $d \times d$-dimensional unitary $\mathcal{U}$.

Note that the non-positive eigenvalues of the Hessian matrix (the elements of the Hessian are $\partial^2 L/\partial t_i^2, \partial^2 L/\partial \tau_j^2$ and $\partial^2 L/\partial t_i \partial \tau_j$ for $1 \leq i, j \leq K$ and $L = L(\vec{t}, \vec{\tau})$) at loss L correspond to directions in the landscape along which gradient descent can further decrease the loss function. Let $\gamma = 2K/d^2$. We assume that the gradients of the loss function $L(\vec{t}, \vec{\tau})$ at the the value L is proportional to the fraction $\beta(\gamma, L)$ of non-positive eigenvalues of the Hessian at L, i.e., we have $\frac{dL}{dt} \propto -\beta(\gamma, L)L$ or $\frac{dL}{d\tau} \propto -\beta(\gamma, L)L$. We make the following conjecture,

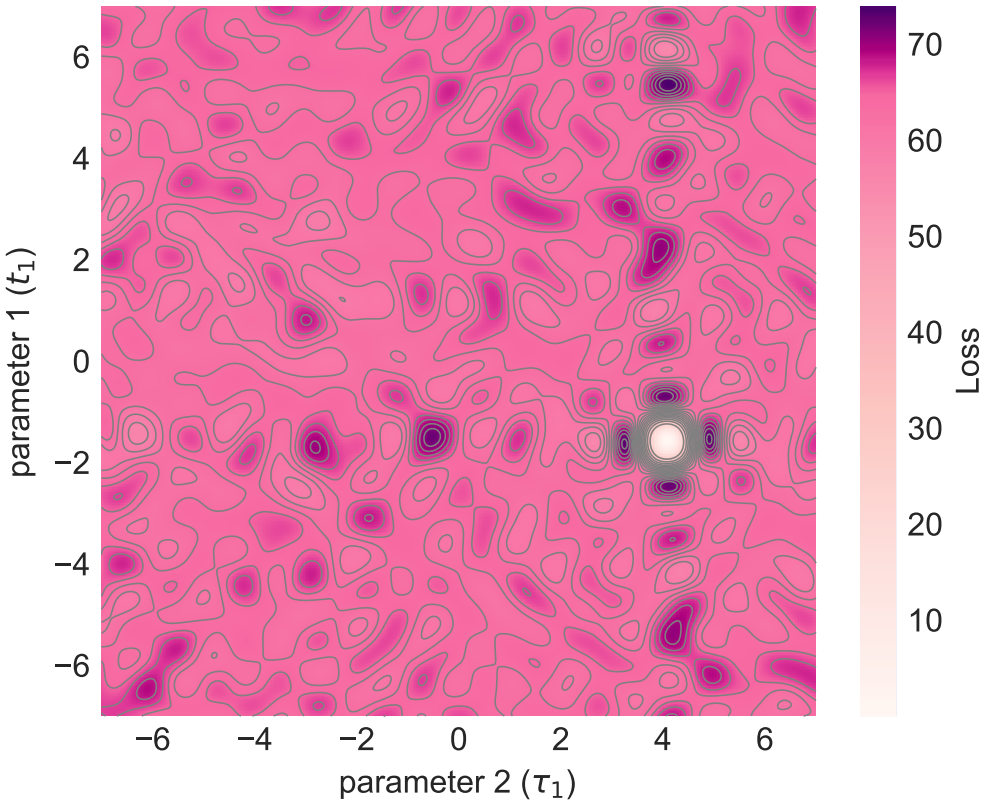


Figure 4.5: Loss function landscape when the target unitary is $e^{-iAt^*}e^{-iB\tau^*}$ where $t^* = -1.59$ and $\tau^* = 4.08$. The landscape is highly non-convex with many local minima, indicating that it is difficult to learn the target unitary with first order optimization methods such as gradient descent unless the starting point of optimization lies in the neighbourhood of the global minimum.

Conjecture 4.3.2 *Let $a, b, c > 0$ and $\gamma = 2K/d^2$. Let L denote the loss function. In the limit of $L \to 0$, the fraction $\beta(\gamma, L)$ of non-positive eigenvalues of the Hessian of the loss landscape is*

$$\beta(\gamma, L) = \begin{cases} (\gamma - 1)/\gamma & \text{if } \gamma > 1 \\ 0 & \text{if } \gamma < 1 \\ aL + bL^2 + cL^3 & \text{otherwise.} \end{cases} \tag{4.4}$$

Observe that from the empirical results in Fig. (4.1), in the the under-parameterized domain, we have $\beta(\gamma, L) = 0$ when $L(\vec{t}, \vec{\tau}) < L_0(\gamma)$ where $L_0(\gamma)$ is the attainable loss by an under-parameterized model with γd^2 parameters.

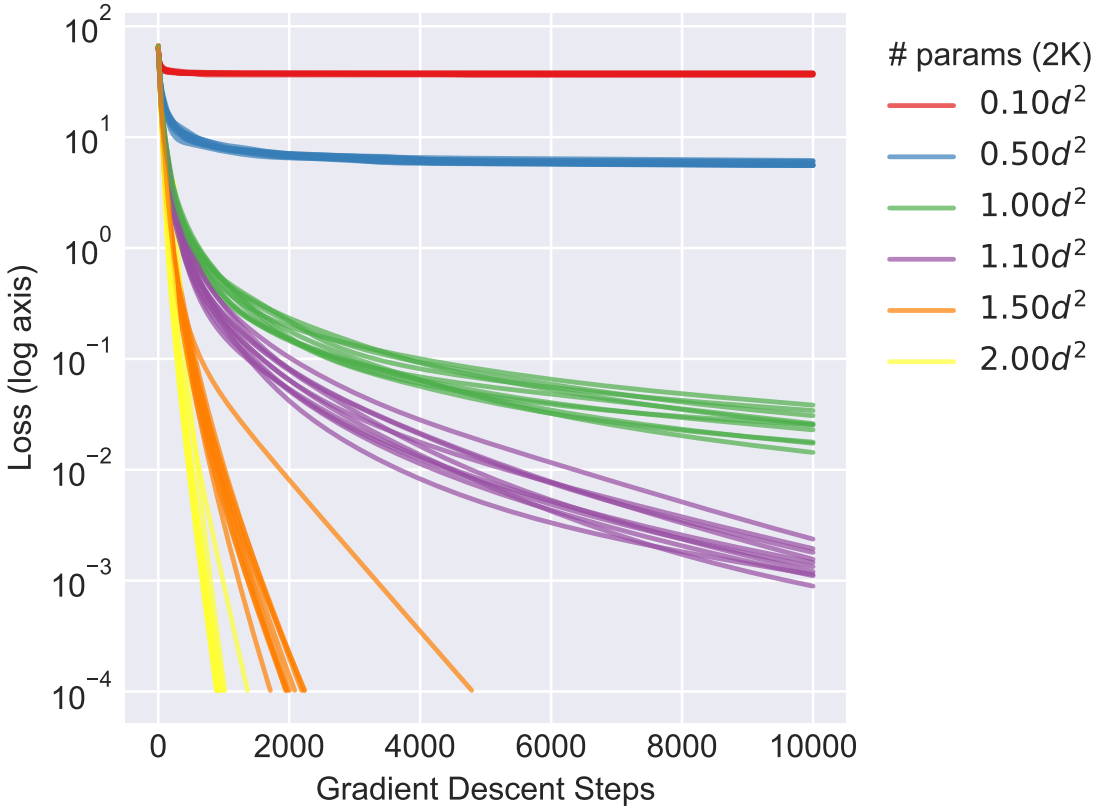


Figure 4.6: Gradient descent experiments for a low-depth unitary $\mathcal{U}(t_1^*, \tau_1^*, t_2^*, \tau_2^*)$ of dimension 32 with 4 parameters (N=2) where $t_1^*, t_2^*, \tau_1^*, \tau_2^* \in [-2, 2]$.

4.4 Learning shallow-depth unitaries

In this section, we study the learnability of low-depth alternating operator unitaries $\mathcal{U}(\vec{t}, \vec{\tau}) = e^{-iAt_N} e^{-iB\tau_N} \dots e^{-iAt_1} e^{-iB\tau_1}$ where $2N \ll d^2$. Such unitaries are the alternating operator analogue of shallow depth quantum circuits. As noted above, unitaries of this form are by definition, obtainable by a learning sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ with depth $K \geq N$. We wish to investigate for which values of K, it is possible to learn the target unitary $\mathcal{U}(\vec{t}, \vec{\tau})$ of depth $N \ll d^2$. We could reasonably hope that such a shallow depth unitary could be learned by performing gradient descent over sequences $\mathcal{V}(\vec{t}, \vec{\tau})$ of depth $K = N$. We find that this is not the case. Indeed, we find that even to learn a unitary $\mathcal{U}(\vec{t}, \vec{\tau})$ of depth $N = 1$, with high probability, we require a full depth learning sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ of depth $K \geq d^2/2$ or $2K \geq d^2$ parameters in the sequence $\mathcal{V}(\vec{t}, \vec{\tau})$.

The target unitaries with depth $N = 1$ take the form $\mathcal{U}(t^*, \tau^*) = e^{-iAt^*}e^{-iB\tau^*}$. In Fig. (4.5), we present the landscape of the loss function $L(t, \tau) = \| e^{-iAt^*}e^{-iB\tau^*} - e^{-iAt}e^{-iB\tau} \|^2$ which is a two dimensional parametric manifold. Here we attempt to learn the target unitary $\mathcal{U}(t^*, \tau^*)$ via a sequence $\mathcal{V}(t, \tau)$ also with two parameters. The loss function landscape is highly non-convex and contains many local sub-optimal traps. Learning the target unitary with much less than d^2 parameters using gradient descent is guaranteed only when the initial values of the parameters t, τ lie in the neighbourhood of the global minimum at $t^* = -1.59$ and $\tau^* = 4.08$. In unbounded parametric manifolds, such an optimal initialization is generally hard to achieve.

Next, we consider a target unitary $\mathcal{U}(\vec{t^*}, \vec{\tau^*})$ with four parameters ($N = 2$). In Fig. (4.6), we find that when the sequence $\mathcal{V}(\vec{t}, \vec{\tau})$ has $2K < d^2$ parameters, the loss function plateaus with increasing gradient descent steps. This indicates that gradient descent halts at a local minimum of the loss function landscape. The rate of learning improves when $2K = d^2$ or $2K > d^2$ as in the over-parameterized domain. In this setting, the loss function rapidly converges towards the global minimum of the landscape, and the rate of convergence to the target unitary $\mathcal{U}(\vec{t^*}, \vec{\tau^*})$ is similar to the over-parameterized case shown in Fig. (4.3).

Surjectivity in the map from the control parameters to the tangent space of the unitary manifold has been shown to be a sufficient condition for constructing loss landscapes with no poor local minima in the quantum control settings [RRW17]. This criteria implies that complete freedom of movement at any point in the unitary manifold is sufficient to guarantee convergence to a global minimum. The under-parameterized setting does not meet this criteria of surjectivity, since infinitesimal variations in the $2K < d^2$ parameters $\{t_1, \tau_1, \ldots, t_K, \tau_K\}$ are not sufficient to generate any local infinitesimal change in the unitary manifold of dimension d^2. When the number of control parameters is d^2 or greater, the map from the controls to unitaries is locally surjective at almost every point in the control space, which implies that any direction in the space of unitaries can be explored. Our numerical results suggest that when there are a sufficient number of control parameters to render the system controllable, the control map is locally surjective along the entire path of gradient descent all the way to the global optimum.

4.5 A greedy algorithm

As noted above, we find that gradient descent algorithms require d^2 parameters in the sequence $\mathcal{V}(\vec{t},\vec{\tau})$ to learn a low-depth unitary $\mathcal{U}(\vec{t^*},\vec{\tau^*}) = e^{-iAt_N^*}e^{-iB\tau_N^*}\dots e^{-iAt_1^*}e^{-iB\tau_1^*}$ where $N \ll d^2$. This suggests that such low-depth unitaries are intrinsically hard to learn with less than d^2 parameters using gradient descent. We also considered a simple greedy algorithm for performing a low-depth target unitary $\mathcal{U}(\vec{t^*},\vec{\tau^*})$. Let $\mathcal{V}_q(\vec{t},\vec{\tau}) = e^{-iAt_q}e^{-iB\tau_q}\dots e^{-iAt_1}e^{-iB\tau_1}$. The first step of the greedy algorithm begins with $q = 1$ and uses gradient descent to optimize the parameters t_1 and τ_1. The next step of the algorithm at $q = 2$ performs gradient descent starting from the initial values $t_2 = \tau_2 = 0$ and t_1, τ_1 which are the optimal values obtained in the previous step. The greedy algorithm then continues, and at each step, q is incremented by 1. At the qth step, the initial starting points for gradient descent are $t_q = \tau_q = 0$ and the remaining parameters $\{t_i,\tau_i\}_{1\le i\le q-1}$ are the optimal values obtained at the end of the previous step. We present the pseudocode of the greedy algorithm below.

Algorithm 3 Greedy Algorithm

Input: Low-depth target unitary $\mathcal{U}(\vec{t^*},\vec{\tau^*}) \in U(d)$ and GUE matrices $\pm A, \pm B$.

Initialize: Parameters $t_1, \tau_1 = 0$. $\mathcal{V}_0(\vec{t},\vec{\tau}) = I$. Loss $= a_0 = \| \mathcal{U}(\vec{t^*},\vec{\tau^*}) - I \|^2$.

1: **while** $q \le d^2/2$ **do**
2: Construct the unitary $\mathcal{V}_{q-1}e^{-iAt_q}e^{-iB\tau_q}$ with parameters t_q, τ_q. Loss $= a_q = \| \mathcal{U}(\vec{t^*},\vec{\tau^*}) - \mathcal{V}_{q-1}e^{-iAt_q}e^{-iB\tau_q} \|^2$.
3: Perform gradient descent on $\{t_i,\tau_i\}_{1\le i\le q}$ to minimize a_q starting from $\{t_i',\tau_i'\}_{1\le i\le q-1}$ in $\mathcal{V}_{q-1}$ and $t_q = \tau_q = 0$.
4: Let $\{t_i',\tau_i'\}_{1\le i\le q}$ be the optimal parameters. Updated loss $= a_q' = \| \mathcal{U}(\vec{t^*},\vec{\tau^*}) - e^{-iAt_q'}e^{-iB\tau_q'}\cdots e^{-iAt_1'}e^{-iB\tau_1'} \|^2$.
5: **if** $a_q' \le \varepsilon$, the sequence $\mathcal{V}_q(\vec{t'},\vec{\tau'})$ converges to $\mathcal{U}(\vec{t^*},\vec{\tau^*})$ and let Q = q.
6: **else** continue.
7: **end while**

Output: $\mathcal{V}_Q(\vec{t'},\vec{\tau'})$ such that $\| \mathcal{U}(\vec{t^*},\vec{\tau^*}) - \mathcal{V}_Q(\vec{t'},\vec{\tau'}) \|^2 \le \varepsilon$.

We investigated the performance of the greedy algorithm for systems of up to five qubits in a restricted space of the loss function landscape. In particular, we considered the parameters $\vec{t^*},\vec{\tau^*} \in [-2,2]$ in a low-depth target unitary $\mathcal{U}(\vec{t^*},\vec{\tau^*})$. In this setting, we find that with a probability that decreases as a function of the

number of qubits, the greedy algorithm can construct a sequence $\mathcal{V}_Q(\vec{t},\vec{\tau})$ where $Q \ll d^2/2$. In contrast, gradient descent experiments require d^2 parameters in $\mathcal{V}(\vec{t},\vec{\tau})$ to learn $\mathcal{U}(\vec{t}^*,\vec{\tau}^*)$. That is, the greedy algorithm does indeed sometimes approximate low-depth target unitaries even in cases where simple (or vanilla) gradient descent on under-parameterized sequences fails. For a system of five qubits, the success probability of the greedy algorithm to learn a target unitary $\mathcal{U}(\vec{t}^*,\vec{\tau}^*)$ of depth $N = 2$ (i.e., with 4 parameters) using less than 50 parameters in $\mathcal{V}_Q(\vec{t},\vec{\tau})$ is around 0.1.

Chapter 5

Boosting quantum machine learning algorithms

In the previous chapters, we studied the learnability of unitary transformations using quantum alternating operator sequences. In this chapter, we will study the learnability of functions with quantum algorithms under the assumption that a quantum computer has access to a uniform superposition of the training samples. The primary interest lies in determining the tasks which can be learned by quantum computers operating on quantum data faster than their classical counterparts. To be precise, the goal is to develop quantum algorithms that can provide a polynomial or exponential speedup in learning certain concept classes. We do not discuss here the broader practical question of whether quantum algorithms can more efficiently solve learning problems for which there already exists polynomial time classical algorithms.

In the past decade, Machine Learning (ML) has received a tremendous surge of interest due to its success in solving problems of practical interest. Given the broad applications of ML, there has been a lot of interest in understanding what are the learning tasks for which quantum computers could provide a speedup. In this direction, there has been a spate of quantum algorithms for practically relevant machine learning tasks that theoretically promise either exponential or polynomial quantum speedups over classical computers. In the past, theoretical works on quantum machine learning (QML) have focused on developing efficient quantum algorithms with favourable quantum complexities to solve interesting learning problems. More recently, there have been efforts in understanding the interplay between quantum machine learning algorithms and small noisy quantum

devices.

The field of QML has given us algorithms for various quantum and classical learning tasks such as (i) quantum improvements to classical algorithms for practically-motivated machine learning tasks such as support vector machines [RML14], linear algebra [Pra14], perceptron learning [KWS16], kernel-based classifiers [HCT+19, LCW19], algorithms to compute gradients [RSW+19, GAW19], clustering [ABG07, KLLP19]; (ii) arbitrary quantum states in the PAC setting [Aar07], shadow tomography of quantum states [Aar18, AG19], learnability of *quantum objects* such as the class of stabilizer states [Roc18], low-entanglement states [Yog19]; (iii) a quantum framework for learning Boolean-valued concept classes [BV93, BJ99, AS05, ACL+19]; (iv) quantum algorithms for optimization [HHL09, CCLW20, AGGW20]; (v) quantum algorithms for machine learning based on generative models [GZD17, LW18a].

While these results seem promising and establish that quantum computers can indeed provide an improvement for interesting machine learning tasks, there are still several practically motivated challenges that remain to be addressed. One important question is whether the assumptions made in some quantum machine learning algorithms are practically feasible? Recently, a couple of works [CGL+19, JGS19] demonstrated that under certain assumptions QML algorithms can be dequantized. In other words, they showed the existence of *efficient classical* algorithms for machine learning tasks which were previously believed to provide exponential quantum speedups. In this chapter, we address another important question which is motivated by practical implementation of QML algorithms:

> Suppose $\mathcal{A}$ is a QML algorithm that is theoretically designed to perform *very well*. However, when implemented on a noisy quantum computer, the performance of $\mathcal{A}$ is *weak*, i.e., the output of $\mathcal{A}$ is correct on a *slightly* better-than-half fraction of the inputs. Can we *boost* the performance of $\mathcal{A}$ so that $\mathcal{A}$'s output is correct on 2/3 of the inputs?

The classical *Adaptive Boosting* algorithm (also referred to as AdaBoost) due to Freund and Schapire [FS99] can be immediately used to convert a weak quantum learning algorithm to a strong algorithm. We provide a *quantum boosting* algorithm that quadratically improves upon the classical AdaBoost algorithm. Using our quantum boosting algorithm, not only can we convert a weak and inaccurate QML algorithm into a strong accurate algorithm, but we can do it in time that

is quadratically faster than classical boosting techniques in terms of some of the parameters of the algorithm.

5.1 Overview of our results

Our main contribution is a quantum boosting algorithm that quadratically improves upon AdaBoost in the VC dimension $\mathsf{VC}(\mathcal{C})$ of a concept class $\mathcal{C}$.

Theorem 5.1.1 (Informal) *For $n \geq 1$, let $\mathcal{C}_n \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$ and $\mathcal{C} = \bigcup_{n\geq 1} \mathcal{C}_n$. Let $\mathcal{A}$ be a γ-weak quantum PAC learner for $\mathcal{C}$ that takes time $Q(\mathcal{C})$. Then the quantum time complexity of converting $\mathcal{A}$ to a strong PAC learner is*

$$T_Q = \widetilde{O}\left(\sqrt{\mathsf{VC}(\mathcal{C})} \cdot Q(\mathcal{C})^{3/2} \cdot \frac{n^2}{\gamma^{11}}\right).$$

The classical complexity of AdaBoost scales as $\widetilde{O}(\mathsf{VC}(\mathcal{C}) \cdot R(\mathcal{C}) \cdot n/\gamma^4)$ where $R(\mathcal{C})$ is the time complexity of a classical PAC learner. Comparing this bound with our main result, we get a quadratic improvement in terms of $\mathsf{VC}(\mathcal{C})$ and also observe that the time complexity of quantum PAC learning $Q(\mathcal{C})$ could be polynomially or even exponentially smaller than classical PAC learning time complexity $R(\mathcal{C})$.[1]

There have been a few prior works [NDRM12, SP18, WMHY19] which touch upon AdaBoost but none of them rigorously prove that quantum techniques can improve boosting. As far as we are aware, ours is the *first work* that proves quantum algorithms can quadratically improve the complexity of classical AdaBoost. Given the importance of AdaBoost in classical machine learning, our quadratic quantum improvement could potentially have various applications in QML. We believe that the $(1/\gamma)$-dependence on our complexity can be further improved using quantum techniques (and we leave it as an open question). Although our complexity is weaker than the classical complexity in terms of $1/\gamma = \mathrm{poly}(n)$, observe that many concept classes have $\mathsf{VC}(\mathcal{C})$ that scales *exponentially* with n, in which case our quadratic improvement in terms of $\mathsf{VC}(\mathcal{C})$ *beats* the *polynomial loss* (in terms of $1/\gamma$) in the complexity of our quantum boosting algorithm.

[1] In [AW18], the authors prove that the *sample* complexity of classical and quantum PAC learning is the same up to constant factors, but there exist concept classes demonstrated by [SG04] for which there could be exponential separations in *time* complexity between quantum and classical learning (under complexity theoretic assumptions).

Applications to NISQ algorithms. We now consider the scenario where QML algorithms are implemented on a *noisy-intermediate scale quantum* computer (often referred to as NISQ [Pre18]). Suppose $\mathcal{A}$ is a quantum learner for the following well-known classification problem: given a set S of training points $x \in \mathbb{R}^d$ labelled as either 0 or 1, decide if the label of an $x \notin S$ is 0 or 1. Let us assume $\mathcal{A}$ is *designed* to be a strong learner, i.e., $\mathcal{A}$ outputs a separating hyperplane $H : \mathbb{R}^d \to \{0,1\}$ for the classification problem such that, at most a constant $\leq 1/3$-fraction of the points in $\mathbb{R}^d$ are misclassified by H. However when implementing $\mathcal{A}$ on a NISQ machine, suppose the errors in the quantum device can only guarantee that H classifies a $(1/2+\gamma)$-fraction of the points correctly (think of $\gamma = 1/\mathrm{poly}(d)$, i.e., H does barely better than random guessing). Then how do we use the NISQ machine to produce a hyperplane that correctly classifies a 2/3-fraction of the inputs? Our quantum boosting algorithm can be used here to find a good hyperplane using multiple invocations of the NISQ device. Although our quantum boosting algorithm uses quantum phase estimation as a subroutine, which isn't a NISQ-friendly quantum algorithm, we leave it as an open question if one could use variational techniques as proposed by Peruzzo et al. [PMS+14b] to replace the quantum phase estimation step.

We now provide more details. Let N be a power of 2, $n = \log_2 N$ and $\mathcal{X} \subseteq \{0,1\}^N$. Let $x \in \mathcal{X}$ be an unknown string, which can be thought of as a set of N points labelled 0 or 1. Furthermore, let us make the assumption that for a set of M uniformly random $i_1, \ldots, i_M \in [N]$, our algorithm has knowledge of $S = \{(i_1, x_{i_1}), \ldots, (i_M, x_{i_M})\}$. We believe that this is a realistic assumption, since in most learning algorithms we often have prior knowledge of the unknown string x at uniformly random coordinates $\{i_1, \ldots, i_M\}$. Let $\mathcal{A}$ be a quantum algorithm that, on input a training sample S drawn from a distribution $\mathcal{D} : [N] \to [0,1]$, takes time at most Q to output a string $y \in \{0,1\}^N$ that satisfies $\Pr_{i\sim\mathcal{D}}[y_i = x_i] \geq \frac{1}{2} + \gamma$. Our quantum boosting algorithm can be used to perform the following: suppose $|S| \geq \widetilde{\Omega}\left(\mathsf{VC}(\mathcal{X})/\gamma^2\right)$, then in time

$$\widetilde{O}\left(\sqrt{\mathsf{VC}(\mathcal{X})} \cdot \frac{Q^{3/2}}{\gamma^{11}} \cdot N^2\right),$$

our quantum algorithm can produce a string $\widetilde{y}$ such that $\Pr[\widetilde{y}_i = x_i] \geq 2/3$ (i.e., the Hamming distance between $\widetilde{y}$ and x is at most $N/3$), where the probability is taken

over uniformly random $i \in [N]$.[2] Classically, one could have used the AdaBoost algorithm to perform the same task, which would have taken time that depends linearly on $\mathsf{VC}(\mathcal{X})$.

5.2 Proof sketch

We now give a sketch of our quantum boosting algorithm. The quantum algorithm follows the structure of the classical AdaBoost algorithm. On a very high level, our quantum speedup is obtained by using quantum techniques to estimate the quantity

$$\varepsilon_t = \Pr_{x \sim D^t}[h_t(x) \neq c(x)] = \sum_{x \in S} D_x^t \cdot [h_t(x) \neq c(x)],$$

quadratically faster than classical methods. In order to do so, we use the quantum algorithm for mean estimation, which given a set of numbers $\alpha_1, \ldots, \alpha_M \in [0,1]$, produces an approximation of $\frac{1}{M}\sum_{i \in [M]} \alpha_i$ up to an additive error δ in time $\Theta(\sqrt{M}/\delta)$ [NW99, BDGT11],[3] whereas classical methods take time $\Theta(M)$.

5.2.1 Why a quantum speedup to AdaBoost is not trivial ?

At first glance, our quantum speedup might seem like an immediate application of the quantum mean estimation algorithm. However, as we discuss below, using the mean estimation subroutine to improve classical AdaBoost comes with various issues.

1. **Errors while computing ε_ts**: Quantumly, the mean estimation subroutine *approximates* ε_t up to an additive error δ in time $O(\sqrt{M}/\delta)$. Suppose we obtain ε_t' satisfying $|\varepsilon_t' - \varepsilon_t| \leq \delta$. Recall that the distribution update in the tth step of AdaBoost is given by

$$D_x^{t+1} = \frac{D_x^t}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t} & \text{otherwise}, \end{cases} \tag{5.1}$$

   where $Z_t = \sum_{x \in S} D_x^t \exp(-c(x)\alpha_t h_t(x))$ and $\alpha_t = \frac{1}{2}\ln((1-\varepsilon_t)/\varepsilon_t)$. Given an additive approximation ε_t' of ε_t, first note that the approximate weights $\alpha_t' =$

[2] Note that we use the uniform distribution here because we started with the assumption that $S = \{(i_1, x_{i_1}), \ldots, (i_M, x_{i_M})\}$ was obtained by uniformly sampling $i_1, \ldots, i_M \in [N]$. Our quantum boosting algorithm works equally well in case obtain indices $i_1, \ldots, i_M$ from an arbitrary (possibly unknown) distribution D instead of the uniform distribution, in which case we obtain a $\widetilde{y}$ such that $\Pr_{i \sim D}[\widetilde{y_i} = x_i] \geq 2/3$.

[3] In this section we omit poly-logarithmic factors in the complexity for simplicity.

$\frac{1}{2}\ln((1-\varepsilon'_t)/\varepsilon'_t)$ could be very far from α_t. Moreover, it is not clear why $\widetilde{D}^{t+1}$ defined as

$$\widetilde{D}_x^{t+1} = \frac{1}{Z_t} \cdot D_x^t \exp(\alpha'_t c(x) \cdot h_t(x))$$

is *even close* to a distribution. Another possible way to update our distribution would be

$$\widetilde{D}_x^{t+1} = \frac{1}{Z'_t} \cdot D_x^t \exp(-\alpha'_t c(x) \cdot h_t(x)), \tag{5.2}$$

where $Z'_t = \sum_{x \in S} D_x^t \exp(-c(x)\alpha'_t h_t(x))$, so by definition $\widetilde{D}^{t+1}$ in Eq. (5.2) is a distribution. However, in this case note that a quantum learner cannot exactly compute Z'_t in time $O(\sqrt{M})$ but instead can *approximate* Z'_t and we face the same issue as mentioned above.[4]

2. **Strong approximation of ε_t:** One possible way to get around this would be to estimate ε_t *very well* so that one could potentially show that $\widetilde{D}^{t+1}$ is close to a distribution. However, it is not too hard to see that if $\widetilde{D}^{t+1}$ should be close to a distribution, then we require a $\delta = 1/\sqrt{M}$-approximation of ε_t. Such a strong approximation increases the complexity from $O(\sqrt{M})$ to $O(M)$ which removes the entire quantum speedup.

3. **Noisy inputs to a quantum learner:** Let us further assume that we could spend time $O(M)$ as mentioned above to estimate ε_t very well (instead of using classical techniques to compute ε_t). Suppose we obtain $\widetilde{D}^{t+1}$ which is close to a distribution. Recall that the input to a quantum learner should be copies of a quantum state $|\psi_t\rangle = \sum_{x \in S} \sqrt{D_x^t}|x, c(x)\rangle$. However, we only have access to a quantum state $|\phi_t\rangle = \sum_{x \in S} \sqrt{\widetilde{D}_x^t}|x, c(x)\rangle + |\chi_t\rangle$, where $|\chi_t\rangle$ is orthogonal to the first part of $|\phi_t\rangle$ (note that $|\phi_t\rangle$ is no longer a quantum state without the additional quantum register $|\chi_t\rangle$). Now it is unclear what will be the output of a quantum learner on input $|\phi_t\rangle$ instead of $|\psi_t\rangle$.

4. **Why is the final hypothesis good:** Assume for now that we are able to show that the learner on input copies of $|\phi_t\rangle$ produces a weak hypothesis h_t for the target concept c. Then, after T steps of the quantum boosting algorithm, the final hypothesis would be $H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha'_t h_t(x)\Big)$. It is not at all clear why H should satisfy $H(x) = c(x)$ for even a constant fraction of the xs in S.

[4]Note that computing Z'_t would take time $O(M)$ classically (to be precise, the time taken to compute Z'_t classically would be M where M is the size of the training set) even though we have knowledge of α'_t since Z'_t involves a summation of M terms. Hence, we need to approximate Z'_t again using a mean estimation algorithm.

Observe that the analysis of classical AdaBoost crucially used that $H(x) = c(x)$ for almost every $x \in S$ in order to conclude that the generalization error is small, i.e., $\Pr_{x\sim\mathcal{D}}[H(x) \neq c(x)] \leq 1/3$ where $\mathcal{D}$ is an unknown distribution over $\{0,1\}^n$.

Our main contribution is a *quantum boosting algorithm* that overcomes all the issues mentioned above.

5.2.2 Quantum boosting algorithm

We now give more details of our quantum boosting algorithm. In order to avoid the issues mentioned in the previous section, our main technical contribution is the following: we provide a quantum algorithm that modifies the standard distribution update rule of classical AdaBoost in order to take care of the *approximations* of ε_ts. We show that the output of our modified quantum boosting algorithm has the same guarantees as classical AdaBoost.

Before we elaborate more on the quantum boosting algorithm, we remark that the modified distribution update rule is also applicable in classical AdaBoost. Suppose in classical AdaBoost, we obtain the approximations ε_t's instead of the *exact* weighted errors ε_ts in time P. Then our *robust classical AdaBoost* algorithm (i.e., AdaBoost with modified distribution update) can still produce a hypothesis H that has small training error and the complexity of such a robust classical AdaBoost algorithm will be $O(P)$. Clearly, it is possible that P could be *much smaller* than M (which is the time taken by classical AdaBoost to compute ε_t exactly) in which case the robust classical AdaBoost algorithm is faster than standard classical AdaBoost (in fact we are not aware if the classical AdaBoost or the MMUW algorithm is robust to errors).

We now discuss the important modification in our quantum boosting algorithm: the distribution update step. As mentioned before, classically one can compute the quantity $\varepsilon = \Pr_{x\sim D}[h(x) \neq c(x)]$ in time $O(M)$. Recall that Q is the time complexity of a weak quantum learner to learn the concept class $\mathcal{C}$ and T is the number of iterations in the quantum boosting algorithm. Quantumly, we describe a subroutine that for a fixed δ, performs the following: outputs 'yes' if $\varepsilon \geq \Omega((1-\delta)/(QT^2))$ and 'no' otherwise. In the 'yes' instance when ε is large, the algorithm also outputs an approximation ε' that satisfies $|\varepsilon' - \varepsilon| \leq \delta\varepsilon'$ and in the 'no' instance, the algorithm outputs an ε' that satisfies $|\varepsilon' - \varepsilon| \leq 1/(QT^2)$. The essential

point here is the subroutine takes time $O(\sqrt{M})$.[5] The subroutine crucially uses the fact that in the 'yes' instance, the complexity of the standard quantum mean estimation algorithm scales as $O(\sqrt{M})$. However, in the 'no' instance when ε is "small", obtaining a good multiplicative approximation of ε' using the quantum mean estimation algorithm could potentially take time $O(M)$. In this case, we observe that we *do not* need a good approximation of ε and instead we set $\varepsilon' = \tau = 1/(QT^2)$. We justify this shortly.

Depending on whether we are in the 'yes' instance or 'no' instance of the subroutine, we update the distribution differently. In the 'yes' instance, we make a distribution update that resembles the standard AdaBoost update using the approximation ε'_t instead of ε_t. We let $Z_t = 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)}$, $\alpha'_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon'_t}{\varepsilon'_t}\right)$ and update $\widetilde{D}^t_x$ as follows:

$$\widetilde{D}^{t+1}_x = \frac{\widetilde{D}^t_x}{(1+2\delta)Z_t} \times \begin{cases} e^{-\alpha'_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha'_t} & \text{otherwise .} \end{cases} \tag{5.3}$$

However, in the 'no' instance when ε_t is small, we cannot hope to get a good multiplicative approximation in time $O(\sqrt{M})$. In this case, we crucially observe that $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ is large, hence with a *worse approximation* ε'_t and the corresponding $\alpha'_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon'_t}{\varepsilon'_t}\right)$, we can still show that the hypothesis $H(x) = \text{sign}\left(\sum_{t=1}^{T}\alpha'_t h_t(x)\right)$ has small training error. As a result, in the 'no' instance, we simply let $\varepsilon'_t = \tau$, $Z_t = 2\sqrt{\tau(1-\tau)}$ and $\alpha'_t = \frac{1}{2}\ln\left(\frac{1-\tau}{\tau}\right)$ and update $\widetilde{D}^t_x$ as follows:

$$\widetilde{D}^{t+1}_x = \frac{\widetilde{D}^t_x}{(1+2/(QT^2))Z_t} \times \begin{cases} (2-1/(QT^2))e^{-\alpha'_t} & \text{if } h_t(x) = c(x) \\ (1/(QT^2))e^{\alpha'_t} & \text{otherwise .} \end{cases} \tag{5.4}$$

Note that the distribution update in Eq. (5.4) is *not* the standard boosting distribution update and differs from it by assigning higher weights to the correctly classified training examples and lower weights to the misclassified ones. In both cases of the distribution update in Eq. (5.3), (5.4), observe that $\widetilde{D}$ need not be a true distribution. However, we are able to show that $\widetilde{D}$ is very close to a distribution, i.e., we argue that $\sum_{x\in S}\widetilde{D}_x \in [1-30\delta, 1]$. This aspect is very crucial because,

[5] We remark that the subroutine outputs 'yes' or 'no' with high probability (here, for simplicity in exposition, we assume that the subroutine always correctly outputs 'yes' or 'no').

in every iteration of the quantum boosting algorithm, we will pass copies of

$$|\Phi'\rangle = \sum_{x \in S} \sqrt{\widetilde{D}_x}|x, c(x)\rangle + |\chi\rangle,^{6}$$

to the quantum learner instead of the ideal quantum state

$$|\Phi\rangle = \sum_{x \in S} \sqrt{D_x}|x, c(x)\rangle.$$

A priori it is not clear, what will be the output of the weak quantum learner on the input $|\Phi'\rangle$. However, we show that the state $|\Phi'\rangle$ is close to $|\Phi\rangle$, in particular we show that $|\langle\Phi'|\Phi\rangle| \geq 1 - \delta$. Suppose a weak quantum learner outputs a weak hypothesis h when given copies of $|\Phi\rangle$ (with probability at least $1 - 1/T$). Using the properties of $\widetilde{D}$ and the lower bound on $|\langle\Phi'|\Phi\rangle|$, we show that the same quantum learner will output a weak hypothesis h when given copies of the state $|\Phi'\rangle$, with probability at least $1 - 2/T$. A union bound over the T iterations of the algorithm shows that with probability at least $2/3$, we obtain a strong hypothesis H after the T iterations. Finally, after T rounds, our quantum boosting algorithm outputs the hypothesis $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha'_t h_t(x)\right)$ for all $x \in \{0,1\}^n$.

It remains to show that the final hypothesis H has small training error. We remark that the calculations to prove this are mathematically technical and are the non-trivial aspects of our quantum algorithm. Crucially, we use the structure of the modified distribution updates to show that H has small training error. In order to go from small training error to small generalization error, we use the same ideas as in classical AdaBoost to show that, if the number of classical labelled examples M is at least $O(\mathsf{VC}(\mathcal{C}))$, then H has generalization error at most $1/3$. The overall time complexity of our quantum boosting algorithm is dominated by the subroutine for estimating ε in every iteration, which scales as $O(\sqrt{M})$. The remaining part of the quantum boosting algorithm invokes the weak quantum learner which takes time $Q(\mathcal{C})$ and performs arithmetic operations in the distribution update state which takes time $O(n^2)$. So the overall complexity of our quantum boosting algorithm scales as $\widetilde{O}(n^2 \cdot \sqrt{\mathsf{VC}(\mathcal{C})} \cdot Q(\mathcal{C})^{3/2})$, which is quadratically better than the classical AdaBoost complexity in terms of $\mathsf{VC}(\mathcal{C})$.

[6] We need $|\chi\rangle$ because $\widetilde{D}$ is not a distribution, and $\sum_{x \in S} \sqrt{\widetilde{D}_x}|x, c(x)\rangle$ need not be a valid quantum state.

5.3 Related works

As we mentioned earlier, there are only a few works that consider boosting in the quantum setting. Neven et al. [NDRM12] considered a heuristic variant of the AdaBoost algorithm and showed how to implement it using the adiabatic quantum hardware on a D-Wave machine. They give numerical evidence that quantum computers should give a speedup to AdaBoost. In particular, the authors considered the framework of adiabatic quantum optimization to boost the performance of large scale binary classifiers. They were able to cast the training of the weak classifiers into a structure that is compatible with existing adiabatic quantum architecture. They proposed an iterative training quantum algorithm, named *QBoost* where a set of weak classifiers is sampled by solving a hard optimization problem in each iteration. Finally, a strong classifier is constructed from the sample of weak classifiers using the standard AdaBoost distribution update rule. The study provided numerical results which indicated that *QBoost* might gain advantage over AdaBoost by achieving smaller generalization error and requiring fewer weak classifiers than AdaBoost.

Schuld and Petruccione [SP18] studied the problem of *boosting* the performance of quantum classifiers and use AdaBoost as a subroutine. The aim of their work was to construct an ensemble of quantum classifiers that can enhance the predictive power of a single quantum classifier. The standard approach to constructing a strong classifier is to perform an optimization in a convex parameter space. In order to avoid the problem of optimization in a large parameter space, the authors discuss how techniques from non-parametric Bayesian learning theory can aid in building a strong quantum classifier from an ensemble of weak quantum classifiers. They further emphasize that the Bayesian learning framework for constructing a strong classifier can be applied to an exponentially large weak quantum ensemble which cannot otherwise be performed with standard AdaBoost.

Finally, in a recent work Wang et al. [WMHY19] proposed a quantum algorithm to improve the performance of weak learning algorithms.[7] Their quantum algorithm departs from standard boosting algorithms in several ways since they make various assumptions in their work. With reference to Section 5.2.1: (1) they assume knowledge of ε_ts *exactly*: note that this is not possible in $o(M)$ time and standard quantum mean estimation takes time $O(\sqrt{M})$ in order to *approximate* the

[7] In their work, they consider the setting where the learners are *probabilistic*. We do not discuss that setting here, but our analysis works exactly the same in case the learners are probabilistic.

mean; (2) their quantum algorithm only approximates $\alpha_1, \ldots, \alpha_T$ additively; (3) given such additive approximations, it is not clear why the final output hypothesis H has small training error. We believe that in order to prove their quantum algorithm outputs a *strong* hypothesis, the time complexity should be $O(M \cdot T^2)$ instead of the claimed $O(\sqrt{M} \cdot T^2)$. However using our techniques, we can improve their complexity to $O(\sqrt{M}T^5)$. Although this complexity might seem worse than the classical AdaBoost complexity of $O(MT)$, note that we set $T = O(\log M)$ in AdaBoost for the convergence analysis. Hence, the overall quantum complexity is quadratically better than classical AdaBoost in terms of M, which is fixed to be $\mathsf{VC}(\mathcal{C})$ in order to have small generalization error.

5.4 Classically boosting quantum machine learning algorithms

In this section, we describe the classical AdaBoost algorithm and explain how one can use AdaBoost to improve a weak quantum learner to a strong quantum learner.

5.4.1 Classical AdaBoost

We begin with presenting the classical AdaBoost algorithm. AdaBoost achieves the following goal: suppose $\mathcal{A}$ is a γ-weak PAC learner for a concept class $\mathcal{C}$ (think of $\gamma = 1/\mathrm{poly}(n)$). Then, for a fixed unknown distribution $\mathcal{D}$, can we use $\mathcal{A}$ multiple times to output a hypothesis H such that $\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq 2/3$? Freund and Schapire [FS99] provided a simple algorithm that outputs such a strong hypothesis.

We do not prove why the output hypothesis H is a strong hypothesis and simply state the main result of Freund and Schapire [FS99, SF12]. Suppose $T \geq (\log M)/\gamma^2$, then the output H of Algorithm 4 with high probability (over the randomness of the algorithm and the training set S) has *zero training error* (i.e., $H(x_i) = c(x_i)$ for every $i \in [M]$). A priori, it might seem that this task is easy to achieve, since we could simply construct a function g that satisfies $g(x_i) = c(x_i)$ for every $i \in [M]$ given explicit access to $S = \{(x_i, c(x_i))\}_{i\in[M]}$. However, recall that the goal of AdaBoost is to output a strong hypothesis H that satisfies $\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq 2/3$ (where $\mathcal{D}$ is the unknown distribution according to which S is generated in Algorithm 4) and it is not clear whether g satisfies this condition. Freund and Schapire [FS99] showed the surprising property that the output of classical

Algorithm 4 Classical AdaBoost

Input: Classical weak learner $\mathcal{A}$, query access to training samples $S = \{(x_1, c(x_1)), \ldots, (x_M, c(x_M))\}$, where $x_i \sim \mathcal{D}$ and $\mathcal{D} : \{0,1\}^n \to [0,1]$ is an *unknown* distribution.

Initialize: Let D^1 be the uniform distribution over S.

1: **for** t = 1 to T **do**

2: Train a weak learner $\mathcal{A}$ on the distribution D^t using the labelled examples S. Suppose we obtain a hypothesis h_t.

3: Compute the weighted error $\varepsilon_t = \sum_{x \in S} D_x^t[h_t(x) \neq c(x)]$, and let $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.

4: Update the distribution D_x^t as follows:

$$D_x^{t+1} = \frac{D_x^t}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t} & \text{otherwise} \end{cases}$$
$$= \frac{D_x^t \exp\left(-c(x)\alpha_t h_t(x)\right)}{Z_t},$$

where $Z_t = \sum_{x \in S} D_x^t \exp\left(-c(x)\alpha_t h_t(x)\right)$.

Output: Hypothesis H defined as $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$ for all $x \in \{0,1\}^n$.

AdaBoost H (i.e., a weighted combination of the hypotheses generated in each iteration) is in fact a strong hypothesis, provided M is *sufficiently large*. In particular, Freund and Schapire [FS99] proved the following theorem.

Theorem 5.4.1 ([SF12, Theorems 4.3 and 4.6]) *Fix $\eta, \gamma > 0$. Let $\mathcal{C} = \cup_{n \geq 1} \mathcal{C}_n$ be a concept class and $\mathcal{A}$ be a γ-weak PAC learner for $\mathcal{C}$ that takes time $R(\mathcal{C})$. Let $n \geq 1$, $\mathcal{D} : \{0,1\}^n \to [0,1]$ be an unknown distribution, $c \in \mathcal{C}_n$ be the unknown target concept and*

$$M = \left\lceil \frac{\mathsf{VC}(\mathcal{C})}{\gamma^2} \cdot \frac{\log(\mathsf{VC}(\mathcal{C})/\gamma^2)}{\eta^2} \right\rceil.$$

Suppose we run Algorithm 4 for $T \geq ((\log M) \cdot \log(1/\delta))/(2\gamma^2)$ rounds, then with probability $\geq 1 - \delta$ (over the randomness of the algorithm and the random examples $\{(x_i, c(x_i))\}_{i \in [M]}$ where $(x_i, c(x_i)) \sim \mathcal{D}$), we obtain a hypothesis H that has zero training error[8] and small generalization error

$$\Pr_{x \sim \mathcal{D}}[H(x) = c(x)] \geq 1 - \eta.$$

[8] Suppose H has training error δ, then the generalization error becomes $\Pr_{x \sim \mathcal{D}}[H(x) = c(x)] \geq 1 - \delta - \eta$.

Moreover the time complexity of the classical AdaBoost algorithm is

$$\widetilde{O}(R(\mathcal{C}) \cdot TMn) = \widetilde{O}\left(\frac{\mathsf{VC}(\mathcal{C})}{\eta^2} \cdot R(\mathcal{C}) \cdot \frac{n}{\gamma^4} \cdot \log(1/\delta)\right).$$

5.4.2 Boosting quantum machine learners

We now describe how classical AdaBoost can improve the performance of *quantum* machine learning algorithms. Suppose $\mathcal{A}$ is a quantum PAC learner that learns a concept class $\mathcal{C}$ in time $Q(\mathcal{C})$. Can we use $\mathcal{A}$ multiple times to construct a strong quantum learner? Indeed, this is possible using classical AdaBoost which we describe below.

Algorithm 5 Classical boosting of weak quantum learners

Input: *Quantum* weak learner $\mathcal{A}$, *quantum* query access to training samples $S = \{(x_1, c(x_1)), \ldots, (x_M, c(x_M))\}$ where $x_i \sim \mathcal{D}$ and $\mathcal{D} : \{0,1\}^n \to [0,1]$ is an unknown distribution.

Initialize: Let D^1 be the uniform distribution over S.

1: **for** t = 1 to T (assume classical query access to $h_1, \ldots, h_{t-1}$ and knowledge of $D^1, \ldots, D^{t-1}$). **do**

2: Prepare $Q(\mathcal{C})$ copies of $|\psi_t\rangle = \sum_{x \in S} \sqrt{D_x^t}|x\rangle$.

3: For every $|\psi_t\rangle$, make a quantum query to S to produce $|\psi_t'\rangle = \sum_{x \in S} \sqrt{D_x^t}|x, c(x)\rangle$.

4: Pass $|\psi_t'\rangle^{\otimes Q(\mathcal{C})}$ to $\mathcal{A}$ and suppose $\mathcal{A}$ produces a hypothesis h_t.

5: Compute the weighted error $\varepsilon_t = \sum_{x \in S} D_x^t \cdot [h_t(x) \neq c(x)]$, and let $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.

6: Update the distribution D_x^t as follows:

$$D_x^{t+1} = \frac{D_x^t}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t} & \text{otherwise} \end{cases}$$
$$= \frac{D_x^t \exp\left(-c(x)\alpha_t h_t(x)\right)}{Z_t},$$

where $Z_t = \sum_{x \in S} D_x^t \exp\left(-c(x)\alpha_t h_t(x)\right)$.

Output: Hypothesis H defined as $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$ for all $x \in \{0,1\}^n$.

The correctness of this algorithm follows from the classical AdaBoost algorithm, since at each iteration, the output h_t of the quantum learner is the same as classical AdaBoost. We simply analyze the time complexity of the algorithm here: in the worst-case, step 2 takes time $O(n^2 M Q(\mathcal{C}))$ (note that one could potentially

use tricks by Grover-Rudolph [GR02], Kaye-Mosca [KM01] to prepare $|\psi_t\rangle$ more efficiently), step 3 takes $Q(\mathcal{C})$ quantum queries and time $O(n \log M Q(\mathcal{C}))$ (assuming we have an efficiently implementable quantum random-access-memory (QRAM)) for each state $|\psi_t\rangle$, step 4 takes time $Q(\mathcal{C})$ (by assumption of $\mathcal{A}$), step 5 involves making M queries to h_t and $O(n)$ operations to compute ε_t, α_t and step 6 takes time $O(nM)$ in order to update the distribution. Overall Algorithm 5 takes time $O(n^2 \cdot MQ(\mathcal{C}) \cdot T)$ time. Crucially, in Algorithm 5 a quantum computer is required only to prepare the state $|\psi_t\rangle$ in every step and run the weak quantum learner. The remaining computation can be done on a classical device. Putting everything together, we obtain the following theorem whose proof follows from Theorem 5.4.1.

Theorem 5.4.2 *Fix $\eta, \gamma > 0$. Let $\mathcal{C} = \cup_{n \geq 1} \mathcal{C}_n$ be a concept class and $\mathcal{A}$ be a γ-weak quantum PAC learner for $\mathcal{C}$ that takes time $Q(\mathcal{C})$. Let $n \geq 1$ and $c \in \mathcal{C}_n$ be the unknown target concept. Then the time complexity of Algorithm 5 to produce a strong hypothesis is*

$$\widetilde{O}\left(\frac{\mathsf{VC}(\mathcal{C})}{\eta^2} \cdot Q(\mathcal{C}) \cdot \frac{n^2}{\gamma^4}\right),$$

Moreover, Algorithm 5 uses the quantum computer only for state preparation (in Step 2) and to run the weak-quantum PAC learning algorithm $\mathcal{A}$ (in Step 4) and the remaining operations can be performed on a classical device.

5.5 Quantum Boosting

In the previous section, we used *classical* ML techniques to boost a weak quantum learner to a strong quantum learner. In this section, we use *quantum* techniques in order to improve the time complexity of AdaBoost. Like in classical AdaBoost, we break the analysis into two stages. Stage (1) is a quantum algorithm that reduces training error: produces a hypothesis that does well on the training set and Stage (2) reduces generalization error: we show that for a sufficiently large training set, not only does the hypothesis output in Stage (1) performs well on the training set, but also has a small generalization error. In Section 5.5.1, we first present a quantum algorithm for boosting and in Section 5.5.6, we show why the hypothesis generated in Section 5.5.1 is a strong hypothesis.

5.5.1 Reducing the training error

The bulk of the technical work in our quantum boosting algorithm lies in reducing the training error and we devote this entire section to proving it. We now state the

main theorem for Stage (1) of our quantum boosting algorithm.

Theorem 5.5.1 *Let $\gamma > 0$. Let $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$ be a concept class and $\mathcal{A}$ be a γ-weak quantum PAC learner for $\mathcal{C}$ that takes time $Q(\mathcal{C})$. Let $n \geq 1$, $\mathcal{D} : \{0,1\}^n \to [0,1]$ be an unknown distribution, $c \in \mathcal{C}_n$ be the unknown target concept and M be sufficiently large.*[9] *Given a training set $S = \{(x_i, c(x_i))\}_{i\in[M]}$ where $x_i \sim \mathcal{D}$ and $c \in \mathcal{C}_n$, our quantum boosting algorithm takes time $\widetilde{O}(n^2\sqrt{M}\cdot Q(\mathcal{C})^{3/2})$,*[10] *and with probability $\geq 2/3$, outputs a hypothesis H that has training error at most $1/10$.*

Since our quantum algorithm is fairly involved to describe and analyze, we break down this section into four subsections, in order to separate the technicalities from our quantum algorithm. In Section 5.5.2, we describe our quantum boosting algorithm. In Section 5.5.3, we prove a few claims which are unproven in Section 5.5.2. In Section 5.5.4, we analyze the correctness of the algorithm and finally in Section 5.5.5, we analyze the time complexity of our quantum boosting algorithm. Putting together these four subsections gives a proof of Theorem 5.5.1.

5.5.2 Quantum boosting algorithm for reducing training error

We begin by presenting our quantum algorithm. For simplicity in notation, we first denote $Q(\mathcal{C})$ as Q and secondly we assume that Q is the time it takes for $\mathcal{A}$ to output a weak hypothesis with probability at least $1 - O(1/T)$ (note that this only increases the complexity by a multiplicative $O(\log T)$-factor). Since the sample complexity of $\mathcal{A}$ is at most the time complexity, we will assume that it suffices to provide $\mathcal{A}$ with Q quantum examples. Our quantum algorithm is a T-round iterative algorithm similar to classical AdaBoost and in each round, our quantum algorithm produces a distribution $\widetilde{D}$. In the tth round, our quantum algorithm follows a three-step process:

1. Invoke the weak quantum learner $\mathcal{A}$ to produce a weak hypothesis h_t under an *approximate* distribution $\widetilde{D}^t$ over the training set S.

2. By making quantum queries to h_t, our algorithm computes ε_t', an approximation to $\widetilde{\varepsilon_t} = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$. We then use ε_t' to update the distribution $\widetilde{D}^t$ to $\widetilde{D}^{t+1}$. In this step, we depart from standard AdaBoost.

[9] We quantify what we mean by *sufficiently large* in the next section, in particular in Theorem 5.5.6.

[10] Here $\widetilde{O}(\cdot)$ hides poly-logarithmic factors in M.

3. Using ε'_t compute a *weight* α'_t. After T steps, output a hypothesis $H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha'_t h_t(x)\Big)$.

Before we describe our quantum algorithm, we first describe a subroutine which will be useful in performing step (2) in the 3-step procedure above. This subroutine uses ideas developed by Ambainis [Amb10] and the proof uses ideas required to prove Theorem 2.5.3. Ambainis proposed a modified version (Theorem 2.5.3) of the quantum amplitude estimation algorithm where the aim is to estimate the probability up to a multiplicative error instead of an additive error as is the case in standard amplitude estimation.

Let $\widetilde{\varepsilon}, M > 0$. We describe the modified amplitude estimation below.

Algorithm 6 Modified Amplitude Estimation

Input: The state $|\psi\rangle = \sqrt{\widetilde{\varepsilon}/M}|\phi_1\rangle|1\rangle + \sqrt{1-\widetilde{\varepsilon}/M}|\phi_0\rangle|0\rangle$ and the unitary U such that $U|0\rangle = |\psi\rangle$.

1: **for** $J = \frac{2\pi\sqrt{M}}{\delta}$ to $\frac{16\sqrt{2}\pi\sqrt{M}}{\delta} \cdot \sqrt{QT^2}\log(MT/\delta)$ **do**

2: Let ε'/M be the output after performing amplitude estimation (in Theorem 2.5.2) to estimate $\widetilde{\varepsilon}/M$ using J queries to U and U^{-1}.

3: Check if $\frac{2\sqrt{2}\pi\sqrt{(1-\delta)\varepsilon'}}{J\sqrt{M}} + \frac{\pi^2}{J^2} \leq \frac{\delta\varepsilon'}{M}$. If yes, then output ε' and quit the loop. Else, let $J = 2 \cdot J$.

Output: $\{\varepsilon', \text{yes}\}$ if there exists ε' in step (3), else output $\{\varepsilon' = 1/(QT^2), \text{no}\}$.

Lemma 5.5.2 *Let $\delta = 1/(10QT^2)$. Algorithm 6 satisfies the following: with probability $\geq 1 - 10\delta/T$, if the output is $\{\varepsilon', yes\}$, then $|\widetilde{\varepsilon} - \varepsilon'| \leq \delta\varepsilon'$; and if the output is $\{\varepsilon' = 1/(QT^2), no\}$, then $|\widetilde{\varepsilon} - \varepsilon'| \leq 1/(QT^2)$. The total number queries to U and U^{-1} used by Algorithm 6 is $O(\sqrt{M}Q^{3/2}T^3)$.*

Proof. We first consider the case when Algorithm 6 outputs $\{\varepsilon', \text{yes}\}$. In this case, there exists a J and ε' which satisfies the relation in step (3) of the algorithm. First observe that, since ε'/M was obtained by amplitude amplification in step (2), we have

$$\left|\frac{\varepsilon'}{M} - \frac{\widetilde{\varepsilon}}{M}\right| \leq \left|\frac{\varepsilon'}{M} - \frac{(1-\delta)\widetilde{\varepsilon}}{M}\right| \leq \frac{2\pi\sqrt{(1-\delta)\widetilde{\varepsilon}}}{J\sqrt{M}} + \frac{\pi^2}{J^2} \leq \frac{2\sqrt{2}\pi\sqrt{(1-\delta)\varepsilon'}}{J\sqrt{M}} + \frac{\pi^2}{J^2}, \tag{5.5}$$

where the second inequality used Eq. (2.17) in Theorem 2.5.2 and the third inequality used the first and second inequalities to conclude $|\widetilde{\varepsilon} - \varepsilon'| \leq \frac{2\pi\sqrt{(1-\delta)M\widetilde{\varepsilon}}}{J} + \frac{\pi^2 M}{J^2}$.

This implies

$$\widetilde{\varepsilon} \leq \varepsilon' + \frac{2\pi\sqrt{(1-\delta)M\widetilde{\varepsilon}}}{J} + \frac{\pi^2 M}{J^2} \leq 2\varepsilon',$$

where we used $J \geq (2\pi\sqrt{M})/\delta$. Putting together the upper bound in Eq. (5.5) along with the upper bound in Step (3) of the algorithm, we get $|\widetilde{\varepsilon}-\varepsilon'| \leq \delta\varepsilon'$. Furthermore, we also show that $\widetilde{\varepsilon} \geq (1-2\delta)/(64QT^2)$. Recall that J, ε' satisfies step (3) of the algorithm. In particular, this implies that

$$\frac{2\sqrt{2}\pi\sqrt{(1-\delta)\varepsilon'}}{J_{max}\sqrt{M}} + \frac{\pi^2}{J_{max}^2} \leq \frac{\delta\varepsilon'}{M},$$

since $J \leq J_{max}$. Substituting the value of J_{max} in the inequality above gives $\frac{\sqrt{(1-\delta)\varepsilon'}}{8\sqrt{QT^2}} + \frac{\delta}{512QT^2} \leq \varepsilon'$. Solving for the above equation, we obtain $\varepsilon' \geq 1/(64QT^2)\cdot(1-\delta)$ (we ignore the other solution for ε' since $\varepsilon' \geq 0$). Using $|\widetilde{\varepsilon}-\varepsilon'| \leq \delta\varepsilon'$, we get $\widetilde{\varepsilon} \geq (1-\delta)\varepsilon' \geq (1-2\delta)/(64QT^2)$.

Now we consider the case when Algorithm 6 outputs $\{\varepsilon' = 1/(QT^2), \text{no}\}$, and we argue that $|\widetilde{\varepsilon}-\varepsilon'| < 1/(QT^2)$. In order to see this, first observe that

$$\left|\frac{\varepsilon'}{M} - \frac{\widetilde{\varepsilon}}{M}\right| \leq \frac{2\sqrt{2}\pi\sqrt{(1-\delta)\varepsilon'}}{J\sqrt{M}} + \frac{\pi^2}{J^2} \leq \frac{\delta\sqrt{2\varepsilon'}}{M} + \frac{\delta^2}{4M} \leq \frac{10\delta}{M}, \tag{5.6}$$

where the first inequality used Eq. (5.5), the second inequality used $J \geq (2\pi\sqrt{M})/\delta$ and the third inequality used $\varepsilon' < 1$. Using $\delta = 1/(10QT^2)$, we obtain $|\widetilde{\varepsilon}-\varepsilon'| \leq 1/(QT^2)$. Furthermore, we show that in the 'no' instance, we have $\widetilde{\varepsilon} < 1/(QT^2)$. We prove this by a contrapositive argument: suppose $\widetilde{\varepsilon} \geq 1/(QT^2)$, there exists a $J' \in \left[J^*, J_{max}\right]$, where $J^* = \frac{8\pi\sqrt{M}}{\delta\sqrt{(1-\delta)\widetilde{\varepsilon}}}$, for which the inequality in step (3) of Algorithm 6 is satisfied with probability at least $1 - 10\delta/T$.[11] In order to see this, first observe that

$$\frac{2\sqrt{2}\pi\sqrt{(1-\delta)\varepsilon'}}{J\sqrt{M}} + \frac{\pi^2}{J^2} \leq \frac{4\pi\sqrt{(1-\delta)\widetilde{\varepsilon}}}{J\sqrt{M}} + \frac{\pi^2}{J^2} \leq \frac{\delta(1-\delta)\widetilde{\varepsilon}}{2M} + \frac{\delta^2(1-\delta)\widetilde{\varepsilon}}{64M} \leq \frac{\delta(1-\delta)\widetilde{\varepsilon}}{M}, \tag{5.7}$$

where the second inequality used $J \geq J^*$ and the remaining inequalities are straightforward. Using Eq. (5.6) and Eq. (5.7), we have $|\varepsilon' - \widetilde{\varepsilon}| \leq \delta(1-\delta)\widetilde{\varepsilon}$. Moreover, using $|\widetilde{\varepsilon}-\varepsilon'| \leq \delta(1-\delta)\widetilde{\varepsilon}$, we can further upper bound Eq. (5.7) by $\delta\varepsilon'/M$, which implies step (3) of Algorithm 6 is satisfied, in which case the algorithm would have output

[11] Note that $J^* \leq J_{max}$ follows immediately by using the lower bound $\widetilde{\varepsilon} \geq 1/QT^2$.

'yes' with probability $\geq 1-10\delta/T$. Hence, by the contrapositive argument, if Algorithm 6 outputs 'no' with probability at least $1-10\delta/T$, then we have $\widetilde{\varepsilon} < 1/QT^2$.

Finally, we bound the total number of queries made to U and U^{-1} in Algorithm 6. Given that J is doubled in every round and $J \leq J_{max} = O(\sqrt{MQT^2}/\delta)$, the total number of queries is

$$J_{max} + \frac{J_{max}}{2} + \frac{J_{max}}{4} + \cdots + \left\lceil \frac{2\pi\sqrt{M}}{\delta} \right\rceil < 2J_{max} = O\left(\sqrt{MQT^2}/\delta \right) = O(\sqrt{M}Q^{3/2}T^3) \quad (5.8)$$

using $\delta = O(1/QT^2)$ (for simplicity, we assume that all these terms are powers of 2). This concludes the proof of the lemma. □

We are now ready to describe our quantum boosting algorithm. Before describing the state of the quantum boosting algorithm in every step, we make a couple of remarks. We use the notation $\widetilde{D}_x^t$ in the quantum boosting algorithm because $\{\widetilde{D}_x^t\}_x$ is not a true distribution since it satisfies $\sum_{x\in S} \widetilde{D}_x^t \leq 1$ (this is also the reason for incorporating the state $|\chi_t\rangle$ in step (4)). In addition to $\widetilde{D}_x^t$, we also define the *true* updated distribution D_x^{t+1} as follows

$$D_x^{t+1} = \frac{\widetilde{D}_x^t}{Z_t} \times \begin{cases} e^{-\alpha_t'} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t'} & \text{otherwise} , \end{cases} \quad (5.9)$$

where $\alpha_t' = \ln\left(\sqrt{(1-\varepsilon_t')/\varepsilon_t'}\right)$ and $\varepsilon_t', \widetilde{\varepsilon_t}$ are defined in step (8) of Algorithm 7, and

$$\begin{aligned} Z_t = \sum_{i=1}^{M} \widetilde{D}_t(x_i) \exp\left(-\alpha_t' h_t(x_i) c(x_i)\right) &= \sum_{i:h_t(x_i)=c(x_i)} \widetilde{D}_t(x_i) \cdot e^{-\alpha_t'} + \sum_{i:h_t(x_i)\neq c(x_i)} \widetilde{D}_t(x_i) \cdot e^{\alpha_t'} \\ &= (1-\widetilde{\varepsilon_t}) \cdot e^{-\alpha_t'} + \widetilde{\varepsilon_t} \cdot e^{\alpha_t'}. \end{aligned} \quad (5.10)$$

Note that $\sum_{x\in S} D_x^{t+1} = 1$ and observe that our quantum boosting algorithm cannot make the distribution update Eq. (5.9) since the value of $\widetilde{\varepsilon_t}$ in Eq. (5.10) is unknown to the quantum algorithm. Let ε_t be the weighted error given by $\varepsilon_t = \Pr_{x\sim D^t}[h_t(x) \neq c(x)]$ corresponding to the true distribution $\{D_x^t\}_x$.

This is one of the main differences between standard AdaBoost and our quantum boosting Algorithm 7. In standard AdaBoost, one assumes that the εs can be computed exactly by spending time $O(M)$. However, a quantum algorithm can only approximate the εs in time $O(\sqrt{M})$. Hence the distribution update from $D^t \to D^{t+1}$ in classical AdaBoost might result in a sub-normalized distribution

Algorithm 7 Quantum boosting algorithm

Input: Quantum weak learner $\mathcal{A}$ with time complexity Q, a training sample $S = \{(x_i, c(x_i))\}_{i\in[M]}$, where x_i is sampled from an unknonwn distribution $\mathcal{D}$.

Initialize: Let $\widetilde{D}^1 = D^1$ be the uniform distribution on S. Let h_0 be the constant function,[12] $T = O((\log M)/\gamma^2)$ and $\delta = 1/(10QT^2)$. $\varepsilon'_0 = 1/2$.

1: **for** t = 1 to T (assume quantum query access to $h_1, \ldots, h_{t-1}$ and knowledge of $\varepsilon'_1, \ldots, \varepsilon'_{t-1}$.) **do**

2: Prepare $Q+1$ many copies of $|\psi_1\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x), \widetilde{D}^1_x\rangle$. Let $|\Phi_1\rangle = |\psi_1\rangle$.

Phase (1): Obtaining hypothesis h_t

3: Using quantum queries to $\{h_1, \ldots, h_{t-1}\}$ and knowledge of $\{\varepsilon'_1, \ldots, \varepsilon'_{t-1}\}$, prepare the state

$$|\Phi_3\rangle = \Big(\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x), \widetilde{D}^t_x\rangle\Big).$$

4: Apply amplitude amplification to prepare $|\Phi_6\rangle = \Big(\sum_{x\in S}\sqrt{\widetilde{D}^t_x}|x, c(x)\rangle + |\chi_t\rangle\Big)$.

5: Pass $|\Phi_6\rangle^{\otimes Q}$ to the quantum learner $\mathcal{A}$ to obtain a hypothesis h_t.

Phase (2): Estimating weighted errors $\widetilde{\varepsilon_t}$

6: Using quantum queries to h_t, prepare $|\psi_5\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x), \widetilde{D}^t_x \cdot [h_t(x) \neq c(x)]\rangle$.

7: Let $\widetilde{\varepsilon_t} = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$. Prepare $|\psi_6\rangle = \sqrt{1-\widetilde{\varepsilon_t}/M}|\phi_0\rangle|0\rangle + \sqrt{\widetilde{\varepsilon_t}/M}|\phi_1\rangle|1\rangle$.

8: Invoke subroutine 6 to estimate $\widetilde{\varepsilon_t}$ with ε'_t.

Phase (3): Updating distributions

9: **If** subroutine 6 outputs 'yes': let $Z_t = 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)}$, $\alpha'_t = \ln\Big(\sqrt{(1-\varepsilon'_t)/\varepsilon'_t}\Big)$ and update $\widetilde{D}^t_x$:

$$\widetilde{D}^{t+1}_x = \frac{\widetilde{D}^t_x}{(1+2\delta)Z_t} \times \begin{cases} e^{-\alpha'_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha'_t} & \text{otherwise .} \end{cases} \tag{5.11}$$

10: **If** subroutine 6 outputs 'no': let $Z_t = \Big(2\sqrt{QT^2-1}\Big)/(QT^2)$, $\alpha'_t = \ln\Big(\sqrt{QT^2-1}\Big)$ and update $\widetilde{D}^t_x$:

$$\widetilde{D}^{t+1}_x = \frac{\widetilde{D}^t_x}{(1+2/(QT^2))Z_t} \times \begin{cases} (2-1/(QT^2))e^{-\alpha'_t} & \text{if } h_t(x) = c(x) \\ (1/(QT^2))e^{\alpha'_t} & \text{otherwise .} \end{cases} \tag{5.12}$$

Output: Hypothesis H defined as $H(x) = \text{sign}\Big(\sum_{t=1}^{T}\alpha'_t h_t(x)\Big)$ for all $x \in \{0,1\}^n$.

[12] To be precise, we let the query operation $\mathcal{O}_{h_0}$ corresponding to h_0 be the identity map.

$\widetilde{D}^{t+1}$ in the quantum case. Moreover even if a learner could produce a hypothesis h_t that is $(1/2+\gamma)$-close to the target concept c *under* $\widetilde{D}^{t+1}$, it is not clear if the final hypothesis H has small training error. In order to overcome this, we split the distribution update step into two cases (steps (9,10) in Algorithm 7) depending on whether ε is *large* or *small*. Using the structure of the distribution update we show that the resulting hypothesis H has small training error. Before we proceed with the proof of the main theorem, we state the following properties about the distributions $\widetilde{D}s$.

Claim 5.5.3 *Let $t \geq 1$, $\widetilde{D}^t : \{0,1\}^n \to [0,1]$ be defined as in Eq. (5.11), (5.12). Then $\sum_{x\in S} \widetilde{D}^t_x \in [1-30\delta, 1]$.*

Claim 5.5.4 *Let $t \geq 1$, $\widetilde{\varepsilon_t} = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$ be the weighted error corresponding to the approximate distribution $\widetilde{D}^t$ and $\varepsilon_t = \Pr_{x\sim D^t}[h_t(x) \neq c(x)]$ correspond to the true distribution D^t. Then $|\widetilde{\varepsilon_t} - \varepsilon_t| \leq 50\delta$.*

We prove these claims later. Observe that our quantum boosting algorithm will run on the sub-normalized distribution $\widetilde{D}^t$ instead of the ideal distribution D^t, since we do not have knowledge of $\widetilde{\varepsilon_t}$ in Eq. (5.10) and instead have an estimate ε'_t of $\widetilde{\varepsilon_t}$. We now describe the unitary operation that updates $\widetilde{D}_1$ (in step (2)) to $\widetilde{D}_t$ (in step (3)). For $t \in \{1,\ldots,T\}$, let $\mathcal{G}_t$ be the quantum circuit that makes the distribution update from $\widetilde{D}^1 \to \widetilde{D}^t$. Given access to $h_1,\ldots,h_{t-1}, c : \{0,1\}^n \to \{-1,1\}$ and knowledge of $\varepsilon'_1,\ldots,\varepsilon'_{t-1}$, define $\mathcal{G}_t$ as the map:

$$\begin{aligned}\mathcal{G}_t : \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle \otimes |\widetilde{D}^1_x\rangle \otimes |[h_1(x)\neq c(x)],\ldots,[h_{t-1}(x)\neq c(x)]\rangle \\ \to \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle \otimes |\widetilde{D}^t_x\rangle \otimes |[h_1(x)\neq c(x)],\ldots,[h_{t-1}(x)\neq c(x)]\rangle.\end{aligned} \tag{5.13}$$

Details of Algorithm 7. We are now ready to describe our quantum boosting algorithm in more details. In the tth step, we have quantum query access to the hypotheses $\{h_1,\ldots,h_{t-1}\}$ and knowledge of the approximate weighted errors $\{\varepsilon'_1,\ldots,\varepsilon'_{t-1}\}$. Let U_1, U_2 be unitaries that satisfy $U_1 : |0\rangle \to |\psi_1\rangle$ and $U_2 : |0\rangle \to |\Phi_1\rangle$, where

$$|\psi_1\rangle = \left(\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle \otimes |\widetilde{D}^1_x\rangle \otimes |0\rangle^{\otimes t+1}\right), \quad |\Phi_1\rangle = \left(\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle \otimes |\widetilde{D}^1_x\rangle \otimes |0\rangle^{\otimes t}\right).$$

Recall that $\widetilde{D}^1$ is the uniform distribution over the training set S. We assume that theoretically one could use a quantum random access memory (QRAM) to prepare

the state $|\psi_0\rangle = \frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle$ in time $O(\log M)$. By *use a QRAM to prepare* we mean, we can perform the operation that takes the training set $S = \{(x_i, c(x_i))\}_{i \in [M]}$ stored in a classical data structure and prepare the *uniform* quantum state $|\psi_0\rangle$ in time $O(\log M)$.

We digress briefly to discuss about the QRAM. A QRAM is a quantum device that encodes classical data $\{i, z_i\}_{i \in [n]}$ and allows one to perform queries of the form $|i, 0\rangle \to |i, z_i\rangle$ to be made in superposition into polylog(n) qubits in time $O(\text{polylog}(n))$. More specifically, the action of a QRAM on a state $\sum_{i \in [n]} \alpha_i |i, 0\rangle$ is

$$\text{QRAM} : \sum_{i \in [n]} \alpha_i |i, 0\rangle \to \sum_{i \in [n]} \alpha_i |i, z_i\rangle.$$

Since a QRAM can encode classical data in poly-logarithmic time, it is possible to construct a superposition of quantum states which encodes a Boolean function in polynomial time. Note that, in the absence of a QRAM, queries of the form $|i, 0\rangle \to |i, z_i\rangle$ can be made efficiently only if $z_i = f(i)$ and there exists a quantum circuit of depth poly-logarithmic in n that can compute z_i for all $i \in [n]$. A QRAM can provide the advantage of performing the oracle for arbitrary data. A potential implementation of a QRAM has been discussed in [GLM08]. However, the exponential scaling in terms of the physical resources required to build a QRAM has raised doubts regarding its practical applications in quantum computing [Aar15].

Given the QRAM assumption has been debatable and seems strong in quantum machine learning, we make a couple of remarks: (i) our quantum boosting algorithm *only* requires a QRAM to prepare the *uniform* superposition over classical data S. Also, our quantum algorithm does not use QRAM as an oracle for Grover-like algorithms, so the negative results of [AGJO$^+$15] do not apply to our algorithm; (ii) we use the QRAM at the beginning of $T = O(\log M)$ iterations of our algorithm to prepare the uniform superposition $|\psi_0\rangle$, so even if the quantum time complexity of preparing $|\psi_0\rangle$ is $O(\sqrt{M})$, then our complexity increases by an *additive* $O(\sqrt{M} \log M)$ term and we still do not lose our quantum speedup; (iii) of course if QRAM is infeasible then we can also assume that a quantum learner has access to uniform quantum examples $|\psi_0\rangle$ or has *quantum query access* to the training examples in S (i.e., can perform the map $|x, b\rangle \to |x, b \cdot c(x)\rangle$ for $x \in S$). In both cases we do not need a QRAM.

We now describe the quantum boosting algorithm. The algorithm begins by first preparing $U_1|0\rangle \otimes U_2^{\otimes Q}|0\rangle = |\psi_1\rangle \otimes |\Phi_1\rangle^{\otimes Q}$. We then apply $(Q+1)(t-1)$ quantum

queries to the oracles $\{O_{h_1}, \ldots, O_{h_{t-1}}\}$ to obtain[13]

$$|\psi_2\rangle \otimes |\Phi_2\rangle^{\otimes Q} = \left(\frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle \otimes |\widetilde{D}_x^1\rangle \otimes |[h_1(x) \neq c(x)], \ldots, [h_{t-1}(x) \neq c(x)]\rangle |0\rangle^2\right) \otimes \left(\frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle |\widetilde{D}_x^1\rangle |[h_1(x) \neq c(x)], \ldots, [h_{t-1}(x) \neq c(x)], 0\rangle\right)^{\otimes Q}. \tag{5.14}$$

We then apply the unitary $\mathcal{G}_t$ on $|\psi_2\rangle$ and each of the Q copies of $|\Phi_2\rangle$ to update $\widetilde{D}^1$. The resulting state is

$$|\psi_3\rangle \otimes |\Phi_3\rangle^{\otimes Q} = \left(\frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle \otimes |\widetilde{D}_x^t\rangle \otimes |[h_1(x) \neq c(x)], \ldots, [h_{t-1}(x) \neq c(x)]\rangle |0\rangle^2\right) \otimes \left(\frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle |\widetilde{D}_x^t\rangle |[h_1(x) \neq c(x)], \ldots, [h_{t-1}(x) \neq c(x)], 0\rangle\right)^{\otimes Q}. \tag{5.15}$$

We now break down the algorithm into two steps, the first phase uses $|\Phi_3\rangle^{\otimes Q}$ to obtain h_t and the second phase uses h_t and $|\psi_3\rangle$ to compute ε_t'.

Phase (1): Obtaining hypothesis h_t. Let $V : |p\rangle|0\rangle|0\rangle \to |p\rangle\Big(\sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle\Big)|\sin^{-1}(\sqrt{p})\rangle$. For each of the Q copies of $|\Phi_3\rangle$, append an auxiliary $|0\rangle$ and apply V to obtain

$$|\Phi_4\rangle = \frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle |\widetilde{D}_x^t\rangle |[h_1(x) \neq c(x)], \ldots, [h_{t-1}(x) \neq c(x)]\rangle \Big(\sqrt{\widetilde{D}_x^t}|0\rangle + \sqrt{1 - \widetilde{D}_x^t}|1\rangle\Big). \tag{5.16}$$

Let W_t be the unitary that performs $W_t : |\Phi_1\rangle \to |\Phi_4\rangle$ and $\widetilde{W_t} = W_t U_2$ be the map $\widetilde{W_t} : |0\rangle \to |\Phi_4\rangle$. Let Y_t be the unitary that uses an expected $O(\sqrt{M} \log T)$ invocations of $\widetilde{W_t}$ and $\widetilde{W_t}^{-1}$ to perform amplitude amplification (in Theorem 2.5.1) on the state $|\Phi_4\rangle$ and with probability at least $1 - O(1/T)$ produce the state $|\Phi_5\rangle$

$$|\Phi_5\rangle = Y_t |\Phi_4\rangle = \sum_{x \in S} \sqrt{\widetilde{D}_x^t} |x, c(x)\rangle |\widetilde{D}_x^t\rangle |[h_1(x) \neq c(x)], \ldots, [h_{t-1}(x) \neq c(x)], 0\rangle + |\Psi\rangle, \tag{5.17}$$

[13] To be precise, we obtain the tth indicator function $[h_t(x) \neq c(x)]$ as follows: first use O_{h_t} to perform the map $\frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle |\widetilde{D}_x^1\rangle |0\rangle = \frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle |\widetilde{D}_x^1\rangle |h_t(x)\rangle$, next apply the CNOT gate between the second and fourth register to produce $\frac{1}{\sqrt{M}} \sum_{x \in S} |x, c(x)\rangle |\widetilde{D}_x^1\rangle |h_t(x) \cdot c(x)\rangle$

where $|\Psi\rangle$ is orthogonal to the first register. Observe that without $|\Psi\rangle$, the state $|\Phi_5\rangle$ is no longer a quantum state because $\{\widetilde{D}_x^t\}_x$ is a sub-normalized distribution and note that the complexity of amplitude amplification is $\Theta\Big(\sqrt{M/\sum_{x\in S}\widetilde{D}_x^t}\Big) = \Theta(\sqrt{M})$ using Claim5.5.3. Moreover, using Claim 5.5.3, we have

$$\||\Psi\rangle\| \leq 1 - \sum_{x\in S}\widetilde{D}_x^t \leq 30\delta. \tag{5.18}$$

Observe that if we had run the boosting algorithm with the ideal distribution D^t, we would have obtained the state

$$|\Phi_5'\rangle = \sum_{x\in S}\sqrt{D_x^t}|x, c(x)\rangle|D_x^t\rangle|[h_1(x)\neq c(x)],\ldots,[h_{t-1}(x)\neq c(x)], 0\rangle, \tag{5.19}$$

instead of $|\Phi_5\rangle$. We now uncompute the auxiliary registers $|\widetilde{D}_x^t\rangle|[h_1(x)\neq c(x)],\ldots,[h_{t-1}(x)\neq c(x)], 0\rangle$ in $|\Phi_5\rangle$ as follows: let $\mathcal{G}_t^{-1}$ be the unitary which maps $|\widetilde{D}_x^t\rangle \to |\widetilde{D}_x^1\rangle$ and let $O_{h_1},\ldots,O_{h_{t-1}}$ be the query operations that uncompute the $\{[h_i(x)\neq c(x)]\}_{i\in[t-1]}$ registers. Applying $\mathcal{G}_t^{-1}$ on the *actual* state $|\Phi_5\rangle$ (instead of the *ideal* state $|\Phi_5'\rangle$) gives

$$\mathcal{G}_t^{-1}|\Phi_5\rangle = \sum_{x\in S}\sqrt{\widetilde{D}_x^t}|x, c(x)\rangle|0\rangle|[h_1(x)\neq c(x)],\ldots,[h_{t-1}(x)\neq c(x)], 0\rangle + \mathcal{G}_t^{-1}|\Psi\rangle,$$

and then performing $O_{h_1}^{-1},\ldots,O_{h_{t-1}}^{-1}$ gives us

$$|\Phi_6\rangle = \sum_{x\in S}\sqrt{\widetilde{D}_x^t}|x, c(x)\rangle|0\rangle|0\rangle^t + O_{h_{t-1}}\cdots O_{h_1}\cdot\mathcal{G}_t^{-1}|\Psi\rangle. \tag{5.20}$$

By performing the operations $\mathcal{G}_t^{-1}$, $O_{h_1}^{-1},\ldots,O_{h_{t-1}}^{-1}$ on the *ideal* state $|\Phi_5'\rangle$, we would have

$$|\Phi_6'\rangle = \sum_{x\in S}\sqrt{D_x^t}|x, c(x)\rangle|0\rangle|0\rangle^t. \tag{5.21}$$

Ideally, our goal would be to pass Q copies of $|\Phi_6'\rangle$ to a quantum learner in order to obtain a hypothesis h_t. Although, we do not have access to $|\Phi_6'\rangle$, we continue our quantum boosting algorithm by passing Q copies of $|\Phi_6\rangle$ to a quantum learner (instead of Q copies of $|\Phi_6'\rangle$). A priori, it is not clear what will be the output of the quantum learner on input $|\Phi_6\rangle^{\otimes Q}$. In order to understand this, we first show that $|\Phi_6\rangle$ and $|\Phi_6'\rangle$ are close. Using this, it is not hard to see that a quantum learner would *behave similarly* when given $|\Phi_6\rangle^{\otimes Q}$ instead of $|\Phi_6'\rangle^{\otimes Q}$. In order to formalize this, we first state the following claim which we prove later.

Claim 5.5.5 *Let* $|\Phi_6\rangle$ *and* $|\Phi_6'\rangle$ *be as defined in Eq.* (5.20), (5.21). *Then we have* $|\langle\Phi_6|\Phi_6'\rangle| \geq 1 - 50\delta$.

Recall that $|\Phi_6'\rangle$ is the ideal state that satisfies the following: suppose Q copies of $|\Phi_6'\rangle$ are given to a weak quantum learner, then with probability at least $1 - 1/T$, the learner outputs a weak hypothesis h_t. We now show that the same learner, when fed Q copies of $|\Phi_6\rangle$ (instead of $|\Phi_6'\rangle$) will output h_t with probability at least $1 - 17/T$. In order to see this, let $p' = \mathsf{Pr}[\mathcal{A}$ outputs h_t given $|\Phi_6'\rangle^{\otimes Q}] \geq 1 - 1/T$ and $p = \mathsf{Pr}[\mathcal{A}$ outputs h_t given $|\Phi_6\rangle^{\otimes Q}]$. Let $\mathcal{H}$ be the hypothesis class and suppose $\{\mathsf{E}_{h_t}\}_{h_t \in \mathcal{H}}$ (satisfying $\sum_{h_t \in \mathcal{H}} \mathsf{E}_{h_t} = I$) is the final POVM performed by $\mathcal{A}$. Then we have the following,

$$\begin{aligned}
|p' - p| &= \left|\mathrm{Tr}(\mathsf{E}_{h_t}|\Phi_6'\rangle\langle\Phi_6'|^{\otimes Q}) - \mathrm{Tr}(\mathsf{E}_{h_t}|\Phi_6\rangle\langle\Phi_6|^{\otimes Q})\right| \\
&\leq \sum_{h_t \in \mathcal{H}} \left|\mathrm{Tr}(\mathsf{E}_{h_t}|\Phi_6'\rangle\langle\Phi_6'|^{\otimes Q}) - \mathrm{Tr}(\mathsf{E}_{h_t}|\Phi_6\rangle\langle\Phi_6|^{\otimes Q})\right| \\
&\leq 2\| \, (|\Phi_6'\rangle\langle\Phi_6'|)^{\otimes Q} - (|\Phi_6\rangle\langle\Phi_6|)^{\otimes Q} \, \|_1 \\
&= 4(1 - \langle\Phi_6|\Phi_6'\rangle^{2Q})^{1/2} \\
&\leq 4(1 - (1 - 50\delta)^{2Q})^{1/2} \leq 4(1 - (1 - 50Q\delta)^2)^{1/2} \leq 16/T,
\end{aligned} \tag{5.22}$$

where we have used the definition of trace distance in the second equality, Claim 5.5.5 in the third inequality, Bernoulli's inequality $(1 - x)^t \geq 1 - xt$ (for $x \leq 1$ and $t \geq 0$) in the penultimate inequality and $\delta = 1/(10QT^2)$ in the final inequality. Additionally, the second inequality follows from the definition of the trace distance between quantum states

$$\| \, \rho - \sigma \, \|_1 = \max_{\{\mathsf{E}_m\}} \frac{1}{2} \sum_m |\mathrm{Tr}(E_m(\rho - \sigma))|.$$

In particular, suppose we have a weak learner $\mathcal{A}$ that outputs h_t with probability at least p', then for every $t \in [T]$, we have

$$p = \mathsf{Pr}[\mathcal{A} \text{ outputs } h_t \text{ given } |\Phi_6\rangle^{\otimes Q}] \geq p' - 16/T \geq 1 - 1/T - 16/T = 1 - 17/T. \tag{5.23}$$

Hence, on passing $|\Phi_6\rangle^{\otimes Q}$ to a quantum learner $\mathcal{A}$, it outputs a weak hypothesis h_t with probability at least $1 - 17/T$ using Eq. (5.22). We assume that the output h_t is presented in terms of an oracle O_{h_t} (which on query $|x, b\rangle$ outputs $|x, b \cdot h_t(x)\rangle$ for all $b \in \{-1, 1\}, x \in \{0, 1\}^n$).

Phase (2): Computing ε'_t. Using the oracle O_{h_t} produced in Phase (1), we now perform the query operation O_{h_t} on $|\psi_3\rangle$ (defined in Eq. (5.15)) and obtain

$$|\psi_4\rangle = O_{h_t}|\psi_3\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle \otimes |\widetilde{D}^t_x\rangle \otimes |[h_1(x) \neq c(x)], \ldots, [h_t(x) \neq c(x)], 0\rangle. \quad (5.24)$$

Using arithmetic operations, one can additionally produce the following state

$$|\psi_5\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle \otimes |\widetilde{D}^t_x \cdot [h_t(x) \neq c(x)]\rangle \otimes |[h_1(x) \neq c(x)], \ldots, [h_t(x) \neq c(x)], 0\rangle. \quad (5.25)$$

We now apply the controlled reflection operator $V : |p\rangle|0\rangle|0\rangle \to |p\rangle\Big(\sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle\Big)|\sin^{-1}(\sqrt{p})\rangle$ (where we store $\sin^{-1}(\cdot)$ up to n bits of accuracy) to $|\psi_5\rangle$ to obtain

$$|\psi_6\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle\otimes|\beta^t_x\rangle\otimes|[h_1(x) \neq c(x)], \ldots, [h_t(x) \neq c(x)]\rangle\otimes\Big(\sqrt{1-\beta^t_x}|0\rangle+\sqrt{\beta^t_x}|1\rangle\Big)\otimes|\sin^{-1}(\sqrt{\beta^t_x})\rangle,$$

where $\beta^t_x = \widetilde{D}^t_x[h_t(x) \neq c(x)]$. We can rewrite the above equation as

$$|\psi_6\rangle = \sqrt{\widetilde{\varepsilon}_t/M}|\phi_1\rangle|1\rangle + \sqrt{1-\widetilde{\varepsilon}_t/M}|\phi_0\rangle|0\rangle, \quad (5.26)$$

where $\widetilde{\varepsilon}_t = \sum_{x\in S}\beta^t_x = \sum_{x\in S}\widetilde{D}^t_x[h_t(x) \neq c(x)]$ and $|\phi_0\rangle, |\phi_1\rangle$ are defined as

$$|\phi_0\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}\frac{\sqrt{1-\beta^t_x}}{\sqrt{1-\widetilde{\varepsilon}_t/M}}|x, c(x)\rangle \otimes |\beta^t_x\rangle \otimes |[h_1(x) \neq c(x)], \ldots, [h_t(x) \neq c(x)]\rangle \otimes |\sin^{-1}(\sqrt{\beta^t_x})\rangle, \quad (5.27a)$$

$$|\phi_1\rangle = \frac{1}{\sqrt{M}}\sum_{x\in S}\frac{\sqrt{\beta^t_x}}{\sqrt{\widetilde{\varepsilon}_t/M}}|x, c(x)\rangle \otimes |\beta^t_x\rangle \otimes |[h_1(x) \neq c(x)], \ldots, [h_t(x) \neq c(x)]\rangle \otimes |\sin^{-1}(\sqrt{\beta^t_x})\rangle. \quad (5.27b)$$

Let F_t be the unitary given by the map $F_t : |\psi_1\rangle \to |\psi_6\rangle$ and $\widetilde{F}_t = F_t U_1$ be the map $\widetilde{F}_t : |0\rangle \to |\psi_6\rangle$. Let P_t be the unitary that implements amplitude estimation using J_t invocations of $\widetilde{F}_t$ and $\widetilde{F}_t^{-1}$. In order to approximate $\widetilde{\varepsilon}_t$, we run Algorithm 6 on the state $|\psi_6\rangle$ assuming unitary access to $\widetilde{F}_t$. Depending on the output ε'_t of Algorithm 6, we compute $\alpha'_t = \frac{1}{2}\ln\Big(\frac{1-\varepsilon'_t}{\varepsilon'_t}\Big)$. Using ε'_t and α'_t, we now update the distribution from $\widetilde{D}^t$ to $\widetilde{D}^{t+1}$,

5.5.3 Proof of claims

In this section, we state and prove a few claims from the previous section. We restate these claims for convenience of the reader. Additionally, we will crucially

use the following relations multiple times in this section: for every $t \geq 1$, let $\widetilde{D}^t$ be the sub-normalized distribution defined in Algorithm 7 (in particular Eq. (5.11)) when Algorithm 6 outputs 'yes', then

$$\begin{aligned}\sum_{x\in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x)) &= \sum_{i:h_t(x_i)=c(x_i)} \widetilde{D}^t(x_i)\cdot e^{-\alpha_t'} + \sum_{i:h_t(x_i)\neq c(x_i)} \widetilde{D}^t(x_i)\cdot e^{\alpha_t'} \\ &= (1-\widetilde{\varepsilon}_t)\cdot e^{-\alpha_t'} + \widetilde{\varepsilon}_t \cdot e^{\alpha_t'},\end{aligned} \tag{5.28}$$

where $\widetilde{\varepsilon}_t = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$. Also let $\widetilde{D}^t$ be the sub-normalized distribution defined in Algorithm 7 (in particular Eq. (5.12)) when Algorithm 6 outputs 'no', then

$$\begin{aligned}\sum_{x\in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x)\neq c(x)]}) &= \sum_{i:h_t(x_i)=c(x_i)} \widetilde{D}^t(x_i)\cdot e^{-\alpha_t'+\kappa_0} + \sum_{i:h_t(x_i)\neq c(x_i)} \widetilde{D}^t(x_i)\cdot e^{\alpha_t'+\kappa_1} \\ &= (1-\widetilde{\varepsilon}_t)\cdot e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t \cdot e^{\alpha_t'+\kappa_1}.\end{aligned} \tag{5.29}$$

Claim 4.3 *Let $t \geq 1$, $\widetilde{D}^t : \{0,1\}^n \to [0,1]$ be as defined in Eq. (5.11), (5.12). Then $\sum_{x\in S} \widetilde{D}_x^t \in [1-30\delta, 1]$.*

Proof. We divide the proof of the claim into two cases. Recall $\delta = 1/(10QT^2)$.

Case I: Suppose Algorithm 6 outputs 'yes' in the tth iteration. Recall the definition of $\widetilde{D}^{t+1}$.

$$\widetilde{D}_x^{t+1} = \frac{\widetilde{D}_x^t}{2(1+2\delta)\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \times \begin{cases} e^{-\alpha_t'} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t'} & \text{otherwise}, \end{cases} \tag{5.30}$$

where $\alpha_t' = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t'}{\varepsilon_t'}\right)$ and $|\widetilde{\varepsilon}_t - \varepsilon_t'| \leq \delta\varepsilon_t'$. In order to prove the lower bound, observe that

$$\begin{aligned}\sum_{x\in S} \widetilde{D}_x^{t+1} &= \frac{1}{2(1+2\delta)\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \sum_{x\in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x)) \\ &= \frac{\sum_{x\in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x))}{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'} + \widetilde{\varepsilon}_t e^{\alpha_t'}} \cdot \frac{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'} + \widetilde{\varepsilon}_t e^{\alpha_t'}}{2(1+2\delta)\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \\ &= \frac{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'} + \widetilde{\varepsilon}_t e^{\alpha_t'}}{2(1+2\delta)\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \qquad \text{(using Eq. (5.28))} \\ &= \frac{1}{2(1+2\delta)} \frac{1}{\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \cdot \left((1-\widetilde{\varepsilon}_t)\sqrt{\frac{\varepsilon_t'}{1-\varepsilon_t'}} + \widetilde{\varepsilon}_t\sqrt{\frac{1-\varepsilon_t'}{\varepsilon_t'}}\right) \\ &\qquad \text{(using the definition of } \alpha_t') \\ &= \frac{1}{2(1+2\delta)}\left(\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} + \frac{\widetilde{\varepsilon}_t}{\varepsilon_t'}\right),\end{aligned}$$

where the second equality used $\sum_{x\in S}\widetilde{D}_x^t \exp(-\alpha_t' c(x)h_t(x)) = (1-\widetilde{\varepsilon}_t)e^{-\alpha_t'}+\widetilde{\varepsilon}_t e^{\alpha_t'}$ (which follows from Eq. (5.10)). Since $|\widetilde{\varepsilon}_t - \varepsilon_t'| \le \delta\varepsilon_t'$, we have

$$\frac{\widetilde{\varepsilon}_t}{\varepsilon_t'} \ge \frac{\varepsilon_t'(1-\delta)}{\varepsilon_t'} = 1-\delta. \tag{5.31}$$

Additionally,

$$\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} \ge \frac{1-\varepsilon_t'(1+\delta)}{1-\varepsilon_t'} = 1 - \frac{\delta\varepsilon_t'}{1-\varepsilon_t'} \ge 1-2\delta, \tag{5.32}$$

where the second inequality uses $\varepsilon_t' \le 2/3$ (since we assume $\varepsilon_t \le 1/2$). Putting together Eq. (5.31) and Eq. (5.32) into the expression for $\sum_{x\in S}\widetilde{D}_x^{t+1}$, we get

$$\sum_{x\in S}\widetilde{D}_x^{t+1} \ge \frac{2-3\delta}{2(1+2\delta)} \ge 1-4\delta \ge 1-30\delta.$$

Next, we prove the upper bound. Note that

$$\frac{\widetilde{\varepsilon}_t}{\varepsilon_t'} \le \frac{\varepsilon_t'(1+\delta)}{\varepsilon_t'} = 1+\delta, \tag{5.33}$$

and

$$\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} \le \frac{1-\varepsilon_t'(1-\delta)}{1-\varepsilon_t'} = 1 + \frac{\delta\varepsilon_t'}{1-\varepsilon_t'} \le 1+2\delta, \tag{5.34}$$

where the second inequality uses $\varepsilon_t' \le 2/3$. Using Eq. (5.33), (5.34), we have

$$\sum_{x\in S}\widetilde{D}_x^{t+1} = \frac{1}{2(1+2\delta)}\left(\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} + \frac{\widetilde{\varepsilon}_t}{\varepsilon_t'}\right) \le \frac{1}{2(1+2\delta)}\cdot\Big((1+2\delta)+(1+\delta)\Big) \le \frac{2+4\delta}{2(1+2\delta)} = 1.$$

Case II: Suppose Algorithm 6 outputs 'no' in the tth iteration. The distribution $\widetilde{D}^{t+1}$ is then updated according to

$$\widetilde{D}_x^{t+1} = \frac{\widetilde{D}_x^t}{(1+2/(QT^2))Z_t} \times \begin{cases} (2-1/(QT^2))e^{-\alpha_t'} & \text{if } h_t(x) = c(x) \\ (1/(QT^2))e^{\alpha_t'} & \text{otherwise ,} \end{cases} \tag{5.35}$$

where we use $\varepsilon_t' = 1/(QT^2)$, $\alpha_t' = \ln\sqrt{(1-\varepsilon_t')/\varepsilon_t'}$ and $Z_t = 2\sqrt{\varepsilon_t'(1-\varepsilon_t')}$. Let $\kappa_0 = \ln(2-1/(QT^2))$ and $\kappa_1 = \ln(1/(QT^2))$. In order to prove the upper and lower bounds of the claim, we first observe that

$$\begin{aligned}
\sum_{x\in S}\widetilde{D}_x^{t+1} &= \frac{1}{(1+2/(QT^2))\cdot 2\sqrt{\varepsilon_t'(1-\varepsilon_t')}}\sum_{x\in S}\widetilde{D}_x^t \exp\big(-\alpha_t' c(x)h_t(x) + \kappa_{[h_t(x)\neq c(x)]}\big) \\
&= \frac{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t'+\kappa_1}}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}} && \text{(using Eq. (5.29))} \\
&= \frac{(2-1/(QT^2))(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'} + (1/(QT^2))\widetilde{\varepsilon}_t e^{\alpha_t'}}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \\
&= \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} + \frac{1}{2QT^2}\cdot\frac{\widetilde{\varepsilon}_t}{\varepsilon_t'}\right),
\end{aligned}$$

where the second equality used $\widetilde{\varepsilon}_t = \sum_{x:h_t(x)\neq c(x)} \widetilde{D}^t_x$, third equality follows by the definition of κ_0, κ_1 and the final equality used $\alpha'_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon'_t}{\varepsilon'_t}\right)$. We now prove the lower bound in the claim:

$$\begin{aligned}
\sum_{x\in S} \widetilde{D}^{t+1}_x &= \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t} + \frac{1}{2QT^2}\cdot\frac{\widetilde{\varepsilon}_t}{\varepsilon'_t}\right) \\
&\geq \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t}\right) && \text{(using } \widetilde{\varepsilon}_t > 0\text{)} \\
&\geq \frac{1}{(1+2/(QT^2))}\cdot\left(1-\frac{1}{2QT^2}\right) && \text{(using } 1-\widetilde{\varepsilon}_t \geq 1-\varepsilon'_t\text{)} \\
&\geq 1-\frac{3}{QT^2} = 1-30\delta, && \text{(since } \delta = \tfrac{1}{10QT^2}\text{)}
\end{aligned}$$

where we used $\widetilde{\varepsilon}_t \leq \varepsilon'_t$ in the penultimate inequality because we are in the 'no' instance of Lemma 5.5.2 in Case II of our proof. We finally get the desired upper bound in the claim as follows

$$\begin{aligned}
\sum_{x\in S} \widetilde{D}^{t+1}_x &= \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t} + \frac{1}{2QT^2}\cdot\frac{\widetilde{\varepsilon}_t}{\varepsilon'_t}\right) \\
&\leq \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\frac{1-\widetilde{\varepsilon}_t}{1-1/(QT^2)} + \frac{1}{2QT^2}\right) \\
& && \text{(using } \widetilde{\varepsilon}_t \leq \varepsilon'_t = 1/(QT^2)\text{)} \\
&\leq \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\frac{1}{1-1/(QT^2)} + \frac{1}{2QT^2}\right) && \text{(using } 1-\widetilde{\varepsilon}_t \leq 1\text{)} \\
&\leq \frac{1}{(1+2/(QT^2))}\left(\left(1-\frac{1}{2QT^2}\right)\cdot\left(1+\frac{2}{QT^2}\right) + \frac{1}{2QT^2}\right) \leq 1.
\end{aligned}$$

□

Claim 4.4 *Let $t \geq 1$, $\widetilde{\varepsilon}_t = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$ be the weighted error corresponding to the sub-normalized distribution $\widetilde{D}^t$ and $\varepsilon_t = \Pr_{x\sim D^t}[h_t(x) \neq c(x)]$ correspond to the true distribution D^t. Then $|\widetilde{\varepsilon}_t - \varepsilon_t| \leq 50\delta$.*

Proof. We break down the proof of the claim into two cases.

Case I: Algorithm 6 outputs 'yes' in the tth iteration. Recall the definition of the sub-normalized distribution $\widetilde{D}^t$ in Eq. (5.11) and the true distribution D^t in

Eq. (5.9). We then have

$$
\begin{aligned}
|\widetilde{\varepsilon}_{t+1} - \varepsilon_{t+1}| &= \left| \sum_{x \in S} \widetilde{D}_x^{t+1} [h_{t+1}(x) \neq c(x)] - \sum_{x \in S} D_x^{t+1} [h_{t+1}(x) \neq c(x)] \right| \\
&\leq \sum_{x \in S} \left| \widetilde{D}_x^{t+1} - D_x^{t+1} \right| \cdot \left| [h_{t+1}(x) \neq c(x)] \right| \\
&\leq \sum_{x \in S} \left| \widetilde{D}_x^{t+1} - D_x^{t+1} \right| && \text{(since } [h_{t+1}(x) \neq c(x)] \leq 1\text{)} \\
&= \sum_{x \in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x)) \left| \frac{1}{2(1+2\delta)\sqrt{\varepsilon_t'(1-\varepsilon_t')}} - \frac{1}{(1-\widetilde{\varepsilon}_t) \cdot e^{-\alpha_t'} + \widetilde{\varepsilon}_t \cdot e^{\alpha_t'}} \right| \\
&= \frac{\sum_{x \in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x))}{(1-\widetilde{\varepsilon}_t) e^{-\alpha_t'} + \widetilde{\varepsilon}_t e^{\alpha_t'}} \left| \frac{(1-\widetilde{\varepsilon}_t) e^{-\alpha_t'} + \widetilde{\varepsilon}_t e^{\alpha_t'}}{2(1+2\delta)\sqrt{\varepsilon_t'(1-\varepsilon_t')}} - 1 \right| \\
&= \frac{1}{2(1+2\delta)} \left| \frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} + \frac{\widetilde{\varepsilon}_t}{\varepsilon_t'} - 2(1+2\delta) \right| && \text{(using Eq. (5.28))} \\
&\leq \frac{1}{2(1+2\delta)} \left(\left| \frac{\widetilde{\varepsilon}_t - \varepsilon_t'}{1-\varepsilon_t'} \right| + \left| \frac{\widetilde{\varepsilon}_t - \varepsilon_t'}{\varepsilon_t'} \right| + 4\delta \right) && \text{(using triangle inequality)} \\
&\leq \frac{\delta}{2(1+2\delta)} \left(\left| \frac{\varepsilon_t'}{1-\varepsilon_t'} \right| + 5 \right) && \text{(using } |\widetilde{\varepsilon}_t - \varepsilon_t'| \leq \delta \varepsilon_t'\text{)} \\
&\leq \frac{7\delta}{2(1+2\delta)} \leq 4\delta. && \text{(using } \varepsilon_t' \leq 2/3\text{)}
\end{aligned}
$$

Case II: Algorithm 6 outputs 'no' in the tth iteration. Recall the definition of the sub-normalized distribution $\widetilde{D}^t$ in Eq. (5.12). Let $\kappa_0 = \ln(2 - 1/(QT^2))$ and $\kappa_1 = \ln(1/(QT^2))$. We have

$$
\begin{aligned}
&|\widetilde{\varepsilon}_{t+1} - \varepsilon_{t+1}| \\
&= \left| \sum_{x \in S} \widetilde{D}_x^{t+1} [h_{t+1}(x) \neq c(x)] - \sum_{x \in S} D_x^{t+1} [h_{t+1}(x) \neq c(x)] \right| \\
&\leq \sum_{x \in S} |\widetilde{D}_x^{t+1} - D_x^{t+1}| \\
&= \sum_{x \in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x) \neq c(x)]}) \cdot \left| \frac{1}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}} - \frac{1}{(1-\widetilde{\varepsilon}_t) e^{-\alpha_t' + \kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t' + \kappa_1}} \right| \\
&= \frac{\sum_{x \in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x) \neq c(x)]})}{(1-\widetilde{\varepsilon}_t) e^{-\alpha_t' + \kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t' + \kappa_1}} \cdot \left| \frac{(1-\widetilde{\varepsilon}_t) e^{-\alpha_t' + \kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t' + \kappa_1}}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}} - 1 \right| \\
&= \frac{1}{2(1+2/(QT^2))} \left| \left(2 - \frac{1}{QT^2}\right) \cdot \frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} + \frac{1}{QT^2} \cdot \frac{\widetilde{\varepsilon}_t}{\varepsilon_t'} - 2\left(1 + \frac{2}{QT^2}\right) \right| && \text{(using Eq. (5.29))}
\end{aligned}
$$

$$\leq \frac{1}{2(1+2(QT^2))}\left(2\cdot\left|\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t}-1\right|+\frac{1}{QT^2}\cdot\left(\left|\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t}\right|+\left|\frac{\widetilde{\varepsilon}_t}{\varepsilon'_t}\right|\right)+\frac{4}{QT^2}\right)$$

(using triangle inequality)

$$\leq \frac{1}{2(1+2/(QT^2))}\left(\frac{2}{QT^2}\cdot\left(\frac{1}{1-1/(QT^2)}\right)+\frac{1}{QT^2}\cdot\left(\frac{1}{1-1/(QT^2)}+1\right)+\frac{4}{QT^2}\right)$$

$$\leq \frac{5}{QT^2}=50\delta, \qquad \text{(using } \delta=1/(10QT^2)\text{)}$$

where the second last inequality used $0\leq\widetilde{\varepsilon}_t\leq\varepsilon'_t=1/(QT^2)$ and $|\widetilde{\varepsilon}_t-\varepsilon'_t|\leq 1/(QT^2)$. □

Claim 4.5 *Let $t\geq 1$, $|\Phi_6\rangle=\sum_{x\in S}\sqrt{\widetilde{D}_x^t}|x,c(x)\rangle|0\rangle|0\rangle^t+O_{h_{t-1}}\cdots O_{h_1}\cdot\mathcal{G}_t^{-1}|\Psi\rangle$ and $|\Phi'_6\rangle=\sum_{x\in S}\sqrt{D_x^t}|x,c(x)\rangle|0\rangle|0\rangle^t$ be defined as in Eq.* (5.20), (5.21) *respectively. Then we have $|\langle\Phi_6|\Phi'_6\rangle|\geq 1-50\delta$.*

Proof. We break down the proof into two cases.

Case I: Algorithm 6 outputs 'yes' in the tth iteration. We now lower bound the inner product between

$$|\Phi_6\rangle=\sum_{x\in S}\sqrt{\widetilde{D}_x^t}|x,c(x)\rangle|0\rangle|0\rangle^t+\underbrace{O_{h_{t-1}}\cdots O_{h_1}\cdot\mathcal{G}_t^{-1}|\Psi\rangle}_{:=|\Psi'\rangle}, \qquad |\Phi'_6\rangle=\sum_{x\in S}\sqrt{D_x^t}|x,c(x)\rangle|0\rangle|0\rangle^t.$$

In order to do so, we first lower bound the following quantity

$$\begin{aligned}\sum_{x\in S}\sqrt{\widetilde{D}_x^{t+1}D_x^{t+1}} &= \sum_{x\in S}\sqrt{\frac{\widetilde{D}_x^t\exp(-\alpha'_t c(x)h_t(x))}{2(1+2\delta)\sqrt{\varepsilon'_t(1-\varepsilon'_t)}}}\cdot\sqrt{\frac{\widetilde{D}_x^t\exp(-\alpha'_t c(x)h_t(x))}{(1-\widetilde{\varepsilon}_t)e^{-\alpha'_t}+\widetilde{\varepsilon}_t e^{\alpha'_t}}}\\ &=\left(\frac{1}{2(1+2\delta)\sqrt{\varepsilon'_t(1-\varepsilon'_t)}}\cdot\frac{1}{(1-\widetilde{\varepsilon}_t)e^{-\alpha'_t}+\widetilde{\varepsilon}_t e^{\alpha'_t}}\right)^{1/2}\sum_{x\in S}\widetilde{D}_x^t\exp(-\alpha'_t c(x)h_t(x))\\ &=\left(\frac{(1-\widetilde{\varepsilon}_t)e^{-\alpha'_t}+\widetilde{\varepsilon}_t e^{\alpha'_t}}{2(1+2\delta)\sqrt{\varepsilon'_t(1-\varepsilon'_t)}}\right)^{1/2}\cdot\frac{\sum_{x\in S}\widetilde{D}_x^t\exp(-\alpha'_t c(x)h_t(x))}{(1-\widetilde{\varepsilon}_t)e^{-\alpha'_t}+\widetilde{\varepsilon}_t e^{\alpha'_t}}\\ &=\left(\frac{1}{2(1+2\delta)}\left(\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t}+\frac{\widetilde{\varepsilon}_t}{\varepsilon'_t}\right)\right)^{1/2}\cdot 1\geq 1-2\delta,\end{aligned} \tag{5.36}$$

where the first equality used Eq. (5.11) and (5.9), the final equality used Eq. (5.28), and the final inequality used Eq. (5.31) and Eq. (5.32) to conclude

$$\frac{1}{2(1+2\delta)}\left(\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon'_t}+\frac{\widetilde{\varepsilon}_t}{\varepsilon'_t}\right)\geq 1-4\delta.$$

We are now ready to prove the claim

$$|\langle\Phi_6|\Phi_6'\rangle| = \Big|\sum_{x\in S}\sqrt{\widetilde{D}_x^t D_x^t} + \langle\Psi'|\Phi_6\rangle\Big| \geq \Big|\Big|\sum_{x\in S}\sqrt{\widetilde{D}_x^t D_x^t}\Big| - \big|\langle\Psi'|\Phi_6\rangle\big|\Big|$$

(by reverse triangle inequality)

$$\geq \Big|\sum_{x\in S}\sqrt{\widetilde{D}_x^t D_x^t}\Big| - \big|\langle\Psi'|\Phi_6\rangle\big|$$

$$\geq 1 - 2\delta - \big|\langle\Psi'|\Phi_6\rangle\big| \qquad \text{(using Eq.(5.36))}$$

$$\geq 1 - 2\delta - \big\||\Psi'\rangle\big\| = 1 - 2\delta - \big\||\Psi\rangle\big\| \geq 1 - 50\delta$$

(using Eq. (5.18))

where the penultimate inequality used $\langle\Psi'|\Phi_6\rangle \leq \||\Psi'\rangle\| \leq 30\delta$ from Eq. (5.18).

Case II: Algorithm 6 outputs 'no' in the tth iteration. Recall that $\kappa_0 = \ln(2 - 1/(QT^2))$ and $\kappa_1 = \ln(1/(QT^2))$. Using Eq. (5.12) and $0 \leq \widetilde{\varepsilon}_t \leq \varepsilon_t' = 1/(QT^2)$, we have

$$\begin{aligned}
&\sum_{x\in S}\sqrt{\widetilde{D}_x^{t+1} D_x^{t+1}} \\
&= \sum_{x\in S}\sqrt{\frac{\widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x)\neq c(x)]})}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}}} \cdot \sqrt{\frac{\widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x)\neq c(x)]})}{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t'+\kappa_1}}} \\
&= \left(\frac{1}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}} \cdot \frac{1}{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t'+\kappa_1}}\right)^{1/2} \cdot \sum_{x\in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x)\neq c(x)]}) \\
&= \left(\frac{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t'+\kappa_1}}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}}\right)^{1/2} \cdot \frac{\sum_{x\in S} \widetilde{D}_x^t \exp(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x)\neq c(x)]})}{(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t e^{\alpha_t'+\kappa_1}} \\
&= \left(\frac{(2-1/(QT^2))\cdot(1-\widetilde{\varepsilon}_t)e^{-\alpha_t'} + (1/(QT^2))\cdot\widetilde{\varepsilon}_t e^{\alpha_t'}}{2(1+2/(QT^2))\sqrt{\varepsilon_t'(1-\varepsilon_t')}}\right)^{1/2} \\
&= \left(\frac{1}{(1+2/(QT^2))} \cdot \left(\left(1 - \frac{1}{2QT^2}\right)\cdot\frac{1-\widetilde{\varepsilon}_t}{1-\varepsilon_t'} + \left(\frac{1}{2QT^2}\right)\cdot\frac{\widetilde{\varepsilon}_t}{\varepsilon_t'}\right)\right)^{1/2} \geq 1 - \frac{3}{2QT^2}.
\end{aligned} \tag{5.37}$$

The first equality used Eq. (5.12) and Eq. (5.29), the fourth equality used the modified distribution update in Eq. (5.12) to conclude

$$\begin{aligned}
\sum_{x\in S}\widetilde{D}_x^t \exp\big(-\alpha_t' c(x) h_t(x) + \kappa_{[h_t(x)\neq c(x)]}\big) &= \sum_{x:h_t(x)=c(x)} \widetilde{D}_x^t \exp(-\alpha_t' + \kappa_1) + \sum_{x:h_t(x)\neq c(x)} \widetilde{D}_x^t \exp(\alpha_t' + \kappa_0) \\
&= (1-\widetilde{\varepsilon}_t)\cdot e^{-\alpha_t'+\kappa_0} + \widetilde{\varepsilon}_t \cdot e^{\alpha_t'+\kappa_1}.
\end{aligned}$$

and the last inequality used the lower bound on $\sum_{x\in S}\widetilde{D}_x^t$ in Claim 4.3 (Case II). We

are now ready to prove the claim

$$\begin{aligned}
|\langle\Phi_6|\Phi_6'\rangle| &\geq \Big|\sum_{x\in S}\sqrt{\widetilde{D}_x^t D_x^t}\Big| - \Big|\langle\Psi'|\Phi_6\rangle\Big| \\
&\geq 1 - \frac{3}{2QT^2} - \Big|\langle\Psi'|\Phi_6\rangle\Big| && \text{(using Eq.(5.37))} \\
&\geq 1 - \frac{3}{2QT^2} - \Big\||\Psi'\rangle\Big\| \\
&\geq 1 - \frac{3}{2QT^2} - \frac{3}{QT^2} \geq 1 - \frac{5}{QT^2} = 1 - 50\delta. \\
& && \text{(using Eq. (5.18) and } \delta = 1/(10QT^2))
\end{aligned}$$

This concludes the proof of the claim. □

The proofs of these three claims conclude the description of the quantum boosting algorithm. It remains to prove the correctness of the algorithm.

5.5.4 Proof of correctness

Output of quantum algorithm. After T rounds, the algorithm computes $\varepsilon_1', \ldots, \varepsilon_T'$ and outputs a hypothesis H which is a weighted combination of the weak hypotheses $\{h_1, \cdots, h_T\}$ obtained after T rounds of boosting. The final hypotheses H is then given by

$$H(x) = \operatorname{sign}\Big(\sum_{t=1}^{T} \alpha_t' h_t(x)\Big),$$

where $\alpha_t' = \frac{1}{2}\ln\Big(\frac{1-\varepsilon_t'}{\varepsilon_t'}\Big)$ is the weight obtained in the tth iteration.

Probability of outputting H. We now bound the probability of failure of the quantum boosting algorithm in obtaining the strong hypothesis H. The first source of error is due to amplitude amplification in step (4) of the boosting algorithm, which fails with probability $\leq \frac{1}{3T}$. The second error is due to the quantum weak learner failing to output a weak hypothesis in step (5), whose probability is $\leq \frac{1}{3T}$. The third source of error is in estimating $\widetilde{\varepsilon_t}$ in step (8), the probability of failure in estimating $\widetilde{\varepsilon_t}$ is $\leq 10\delta/T = O(1/(QT^3))$ (since we set $\delta = 1/(10QT^2)$). By applying a union bound over the T iterations and all three failure events, we ensure that the overall failure probability of outputting H at the end of our quantum boosting algorithm is an arbitrary constant at most 1/3 (with a constant overhead in the complexity).

It remains to argue that the training error of H is $\leq 1/10$, i.e., $H(x) = c(x)$ for 9/10 of the $(x, c(x))$s in S. To prove this, we analyze the training error of H with respect to the uniform distribution $\widetilde{D}^1$ as follows. We break the proof of correctness into two cases and argue separately. In fact in the first case we will argue that H has zero training error and in the second case we will show the training error of H is at most 1/10.

Case I: Suppose Algorithm 6 outputs 'yes' for every $t \in [T]$. This case corresponds to the setting where each weighted error $\widetilde{\varepsilon}_t$ is estimated by an ε'_t such that $|\varepsilon'_t - \widetilde{\varepsilon}_t| \leq \delta\varepsilon'_t$ for every iteration of the quantum boosting algorithm. In this case

$$\widetilde{D}^{t+1}(x) = \frac{\widetilde{D}^t(x)}{Z'_t} \times \begin{cases} e^{-\alpha'_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha'_t} & \text{otherwise} \end{cases} = \frac{\widetilde{D}^t(x)\exp\left(-c(x)\alpha'_t h_t(x)\right)}{Z'_t}. \tag{5.38}$$

where $Z'_t = 2(1+2\delta)\sqrt{\varepsilon'_t(1-\varepsilon'_t)}$. By definition, we obtain

$$\widetilde{D}^{T+1}(x) = \widetilde{D}^1(x) \cdot \prod_{t=1}^{T} \frac{\exp\left(-c(x)\alpha'_t h_t(x)\right)}{Z'_t} = \frac{D^1(x)\exp\left(-c(x)\cdot\sum_{t=1}^{T}\alpha'_t h_t(x)\right)}{\Pi_{t=1}^{T} Z'_t}, \tag{5.39}$$

where the second equality used $\widetilde{D}^1 = D^1$ which is the uniform distribution. We now upper bound the training error under the distribution D^1

$$\begin{aligned} \Pr_{x\sim D^1}[H(x) \neq c(x)] &= \Pr_{x\sim D^1}\Big[\operatorname{sign}\Big(\sum_{t=1}^{T}\alpha'_t h_t(x)\Big) \neq c(x)\Big] \\ &\leq \Pr_{x\sim D^1}\Big[\exp\Big(-\sum_{t=1}^{T}\alpha'_t h_t(x)\cdot c(x)\Big)\Big] \\ &= \sum_{i=1}^{M} D^1(x_i)\exp\Big(-c(x_i)\sum_{t=1}^{T}\alpha'_t h_t(x_i)\Big) = \sum_{i=1}^{M}\widetilde{D}^{T+1}(x_i)\Pi_{t=1}^{T}Z'_t \leq \Pi_{t=1}^{T}Z'_t, \end{aligned} \tag{5.40}$$

where the first equality used the definition of $H(x) = \operatorname{sign}(\sum_{t=1}^{T}\alpha'_t h_t(x))$, the first inequality used $[\operatorname{sign}(z) \neq y] \leq e^{-z\cdot y}$ for $z \in \mathbb{R}, y \in \{-1, 1\}$, the final equality used Eq. (5.39) and the final inequality used the fact that $\widetilde{D}^{T+1}$ is a sub-normalized distribution ($\sum_{x\in S}\widetilde{D}^{T+1}_x \leq 1$). We are now in a stage to analyze the training error

of H on D^1,

$$\begin{aligned}
\Pr_{x\sim D^1}[H(x)\neq c(x)] &\leq \prod_{t=1}^{T} Z'_t = (1+2\delta)^T \prod_{t=1}^{T} 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)} \\
&\qquad\qquad \text{(using Eq. (5.40) and definition of } Z'_t) \\
&\leq e^{2\delta T} \prod_{t=1}^{T} 2\sqrt{\frac{\widetilde{\varepsilon}_t}{1-\delta}\cdot\Big(1-\frac{\widetilde{\varepsilon}_t}{1+\delta}\Big)} \qquad \text{(since } |\widetilde{\varepsilon}_t-\varepsilon'_t|\leq \delta\varepsilon'_t) \\
&\leq e^{2\delta T} \prod_{t=1}^{T} 2\sqrt{\widetilde{\varepsilon}_t(1+2\delta)(1-\widetilde{\varepsilon}_t(1-\delta))} \\
&\leq e^{2\delta T} \prod_{t=1}^{T} 2\sqrt{(\varepsilon_t+4\delta)(1+2\delta)(1-(\varepsilon_t-4\delta)(1-\delta))} \\
&\qquad\qquad \text{(using } |\widetilde{\varepsilon}_t-\varepsilon_t|\leq 4\delta) \\
&\leq e^{2\delta T} \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)+75\delta} \\
&\leq e^{2\delta T} \prod_{t=1}^{T} 2\sqrt{1/4-\gamma_t^2+75\delta} \qquad \text{(since } \varepsilon_t\leq 1/2-\gamma_t) \\
&= e^{2\delta T} \prod_{t=1}^{T} \sqrt{1-4(\gamma_t^2-75\delta)} \\
&\leq e^{2\delta T} \prod_{t=1}^{T} \sqrt{1-4(\gamma^2-75\delta)} \qquad \text{(since } \gamma\leq\gamma_t \text{ for all } t) \\
&\leq \exp\Big(2\delta T - 2\sum_{t=1}^{T}\Big(\gamma^2-75\delta\Big)\Big) \\
&\qquad\qquad \text{(since } 1+x\leq e^x \text{ for } x\in\mathbb{R}) \\
&\leq \exp\big(-2T\gamma^2+16/(QT)\big), \qquad \text{(since } \delta=1/(10QT^2))
\end{aligned}$$

where we used Claim 4.4 (Case I) in the third inequality to conclude $|\widetilde{\varepsilon}_t-\varepsilon_t|\leq 4\delta$.

In order to conclude the proof-of-correctness, note that for $T=O((\log M)/\gamma^2)$ and for a sufficiently large constant in the $O(\cdot)$, the final upper bound on the expression is

$$\Pr_{x\sim D^1}[H(x)\neq c(x)] < 1/M.$$

Since D^1 is the uniform distribution over S, i.e., $D^1_x = 1/M$ for $(x,c(x))\in S$, this implies that $\Pr_{x\sim D^1}[H(x)\neq c(x)]=0$. Hence H has *zero training error*.

Case II: In this case, we assume that Algorithm 6 outputs 'no' in the first $\ell\in[T]$

rounds of the quantum boosting Algorithm 7.[14] We additionally assume that $\ell \leq T/\log(2\sqrt{Q}T)-1$, which is standard in AdaBoost for the following reason: suppose the weighted errors of each of the first $t \in [\ell]$ hypotheses satisfies $\varepsilon_t \leq 1/QT^2 \ll 1/3$ (which is the 'no' instance of Algorithm 6), then observe that the resulting learner is *strong* and we need not do boosting in the first place. Moreover, suppose $\ell \geq T/\log(2\sqrt{Q}T)$, then observe that the final hypothesis after the T rounds of AdaBoost has training error at most 1/10 and we are again done:

$$\Pr_{x\sim D^1}[H(x)\neq c(x)] = \Pr_{x\sim D^1}\Big[\operatorname{sign}\Big(\sum_{t=1}^{T}\alpha_t h_t(x)\Big)\neq c(x)\Big] \leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$
$$\leq \prod_{t=1}^{\ell} 2\sqrt{\varepsilon_t} \leq \left(\frac{2}{\sqrt{QT}}\right)^{\ell} \leq \frac{1}{10},$$

where the last equality used $\ell \geq T/\log(2\sqrt{Q}T)$ and $T \geq \log M$.

So from here onwards we will assume $\ell \leq T/\log(2\sqrt{Q}T)$ and still show that the training error is at most 1/10. Note that for the first ℓ iterations, the distribution follows the update rule, which defers from the standard AdaBoost update: for every $k \in [\ell]$,

$$\begin{aligned}\widetilde{D}_x^{k+1} &= \frac{\widetilde{D}_x^k}{2(1+2/(QT^2))\sqrt{\varepsilon'_k(1-\varepsilon'_k)}} \times \begin{cases}(2-1/(QT^2))e^{-\alpha'_k} & \text{if } h_k(x)=c(x)\\ (1/(QT^2))e^{\alpha'_k} & \text{otherwise}\end{cases}\\ &= \frac{\widetilde{D}^k(x)\exp\big(-\alpha'_k\cdot c(x)h_k(x)+\kappa_{[h_k(x)\neq c(x)]}\big)}{Z'_k},\end{aligned} \tag{5.41}$$

where $\varepsilon'_k = 1/(QT^2)$ and $Z'_k = 2(1+2/(QT^2))\sqrt{\varepsilon'_k(1-\varepsilon'_k)}$. Let $\kappa_0 = \ln(2-1/(QT^2))$ and $\kappa_1 = \ln(1/(QT^2))$. In particular, observe that for every $\ell \geq 1$, we have

$$\widetilde{D}_x^{\ell+1} = \frac{D_x^1}{\Pi_{i=1}^{\ell} Z'_i}\cdot\exp\Big(-c(x)\cdot\sum_{i=1}^{\ell}\alpha'_i h_i(x)\Big)\cdot\exp\Big(\sum_{i=1}^{\ell}\kappa_{[h_i(x)\neq c(x)]}\Big). \tag{5.42}$$

We bound the training error as follows:

$$\begin{aligned}\Pr_{x\sim D^1}[H(x)\neq c(x)] &\leq \sum_{x\in S} D_x^1 \exp\Big(-c(x)\sum_{t=1}^{T}\alpha'_t h_t(x)\Big)\\ &= \sum_{x\in S} D_x^1 \exp\Big(-c(x)\sum_{t=1}^{\ell}\alpha'_t h_t(x)\Big)\cdot\exp\Big(-c(x)\sum_{t=\ell+1}^{T}\alpha'_t h_t(x)\Big)\end{aligned}$$

[14] Our analysis also works when Algorithm 6 outputs 'no' for arbitrary ℓ rounds of the quantum boosting algorithm instead of the first ℓ rounds.

$$
\begin{aligned}
&= \prod_{t=1}^{\ell} Z'_t \sum_{x\in S} \widetilde{D}_x^{\ell+1} \exp\Big(-c(x)\sum_{t=\ell+1}^{T} \alpha'_t h_t(x)\Big)\cdot \exp\Big(-\sum_{i=1}^{\ell} \kappa_{[h_i(x)\neq c(x)]}\Big)\\
&= \prod_{t=1}^{T} Z'_t \sum_{x\in S} \widetilde{D}_x^{T+1} \exp\Big(-\sum_{i=1}^{\ell} \kappa_{[h_i(x)\neq c(x)]}\Big)\\
&\leq \prod_{t=1}^{T} Z'_t \cdot (QT^2)^{\ell} \cdot \sum_{x\in S} \widetilde{D}_x^{T+1} \leq \prod_{t=1}^{T} Z'_t \cdot (QT^2)^{\ell},
\end{aligned}
$$

where the second equality uses Eq. (5.42) (i.e., distribution update for 'no' instances) and third equality uses Eq. (5.39) (i.e., distribution update for the 'yes' instances), the penultimate inequality uses $\exp(-\kappa_0) \leq \exp(-\kappa_1) \leq QT^2$ (we remark that this bound is very loose, since $\exp(\kappa_0) = O(1)$) and the final inequality uses the fact that $\widetilde{D}$ is a sub-normalized distribution by Claim 4.3. Continuing to upper bound the above expression, we get

$$
\begin{aligned}
\Pr_{x\sim D^1}[H(x)\neq c(x)] &\leq (QT^2)^{\ell} \prod_{t=1}^{T} Z'_t\\
&= \left((QT^2)^{\ell}\prod_{t=1}^{\ell} Z'_t\right)\cdot\left(\prod_{t=\ell+1}^{T} Z'_t\right)\\
&= \left((QT^2)^{\ell}(1+2/(QT^2))^{\ell}\prod_{t=1}^{\ell} 2\sqrt{1/(QT^2)\cdot(1-1/(QT^2))}\right)\\
&\qquad\qquad \cdot\left((1+2\delta)^{T-\ell}\prod_{t=\ell+1}^{T} 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)}\right)\\
&\leq \left((QT^2)^{\ell}(1+2/(QT^2))^{\ell}\prod_{t=1}^{\ell} 2/\sqrt{QT^2}\right)\cdot\left((1+2\delta)^{T-\ell}\prod_{t=\ell+1}^{T} 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)}\right)\\
&\leq \left((2\sqrt{Q}T)^{\ell}\exp\Big((2\ell)/(QT^2)\Big) - 2(T-\ell)\gamma^2 + (16T-16\ell)/(QT^2)\right)\\
&\leq (2\sqrt{Q}T)^{\ell}\exp\Big(-2(T-\ell)\gamma^2 + 16/(QT)\Big)\\
&\leq \exp\Big(2\ell(\ln(2\sqrt{Q}T)+\gamma^2) - 2T\gamma^2 + 1\Big),
\end{aligned}
$$

where the second equality used

$$
Z'_t = \begin{cases} 2(1+2/(QT^2))\sqrt{1/(QT^2)\cdot(1-1/(QT^2))} & \text{for } t\leq \ell\\ 2(1+2\delta)\cdot\sqrt{\varepsilon'_t(1-\varepsilon'_t)} & \text{for } t\geq \ell+1, \end{cases}
$$

the third inequality used $1+x\leq e^x$ for $x\in\mathbb{R}$ and the second factor

$$
\exp\Big(-2(T-\ell)\gamma^2 + (16T-16\ell)/(QT^2)\Big),
$$

came from the upper bound of the training error derived in Case I with T replaced by $T-\ell$ (recall that we had showed $\Pr_{x\sim D^1}[H(x)\neq c(x)] \leq \prod_{t=1}^{T} Z'_t \leq \exp(-2T\gamma^2 + 16/(QT))$). Finally, using $\ell \leq T/\ln(2\sqrt{Q}T) - 1$, we have

$$\begin{aligned}\Pr_{x\sim D^1}[H(x)\neq c(x)] &\leq \exp\Big(2\ell(\ln(2\sqrt{Q}T)+\gamma^2) - 2T\gamma^2 + 1\Big)\\ &\leq \exp\Big(\Big(\frac{2T}{\ln(2\sqrt{Q}T)} - 2\Big)\cdot(\ln(2\sqrt{Q}T)+\gamma^2) - 2T\gamma^2 + 1\Big)\\ &= \exp\Big(2T - 2\gamma^2 - 2\ln(2\sqrt{Q}T) + \frac{2T\gamma^2}{\ln(2\sqrt{Q}T)} - 2T\gamma^2 + 1\Big) \leq \frac{e}{4QT^2} \leq \frac{1}{10},\end{aligned}$$

where the final inequality used that $Q, T = O(\log M)$ are sufficiently large. Hence, we have shown that H has training error at most $1/10$.

5.5.5 Complexity of the algorithm

First we analyze the *query* complexity of the quantum boosting algorithm (where the query complexity refers to the total number of queries made to the hypothesis-oracles $\{O_{h_1},\ldots,O_{h_T}\}$). We consider the complexity of the tth iteration: in *phase 1*, the number of queries made to $\{h_1,\ldots,h_{t-1}\}$ in order for the quantum weak learner $\mathcal{A}$ to output the hypothesis h_t is at most $\sqrt{M}Q\cdot t$: the $\sqrt{M}$-factor comes from amplitude amplification and the application of the unitary $\widetilde{W}_t : |0\rangle \to |\Phi_4\rangle$ involves $Q(t-1)$ queries for the Q copies of the input to the weak learner. An additional $Q(t-1)$ queries to $\{h_1,\ldots,h_{t-1}\}$ are required while applying $O_{h_1},\ldots,O_{h_{t-1}}$ to uncompute the queries. In *phase 2*, the number of queries made during multiplicative amplitude estimation in order to compute ε'_t is $\sqrt{M}Q^{3/2}T^3\cdot t$: the $\sqrt{M}Q^{3/2}T^3$-factor is due to multiplicative amplitude estimation (in Lemma 5.5.2). Furthermore, each application of $\widetilde{F}_t : |0\rangle \to |\psi_6\rangle$ involves making t queries. Putting together the contribution from both phases, the total query complexity of the quantum boosting algorithm is

$$\sum_{t=1}^{T} \sqrt{M}Q(t-1) + Q(t-1) + \sqrt{M}Q^{3/2}T^3 t = O(\sqrt{M}Q^{3/2}T^5 + \sqrt{M}QT^2) = \widetilde{O}(\sqrt{M}Q^{3/2}T^5).$$

We now discuss the time complexity of the quantum boosting algorithm. We begin by analyzing the time complexity of the tth iteration. Assuming that a quantum RAM can prepare a *uniform* superposition $\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle$ using $O(n\log M)$

gates, the time complexity of preparing the initial state $|\psi_1\rangle \otimes |\Phi_1\rangle^{\otimes Q}$ is $O(nQ)$.[15] In the second step, in order to prepare $|\psi_2\rangle \otimes |\Phi_2\rangle^{\otimes Q}$, our quantum algorithm uses $O(Qt)$ quantum queries to $\{h_1, \ldots, h_{t-1}\}$ and this can be performed in time $O(Qt)$. The third step involves updating the registers from $\widetilde{D}^1$ to $\widetilde{D}^t$ which requires $Q+1$ applications of the control unitary $\mathcal{G}_t$. Since there are $t-1$ control qubits and updating the distribution register is an arithmetic operation, the third step for implementing $O(Q)$ operations of $\mathcal{G}_t$ can be performed in $O(n^2Qt)$ time.[16]

In phase 1 of the quantum algorithm, we perform amplitude amplification with the unitary $Y_t^{\otimes Q}$ which makes $O(\sqrt{M}Q)$ calls to $\widetilde{W}_t$ and $\widetilde{W}_t^{-1}$. This takes time $O(n^2\sqrt{M}Qt)$. Next, in order to uncompute the $t-1$ quantum queries in the Q copies, our algorithm uses $O(nQt)$ time. The weak learner $\mathcal{A}$ takes as input Q samples and outputs a hypothesis h_t in time $O(n^2Q)$. Note that we require the quantum learner to output an oracle for h_t instead of explicitly outputting a circuit for h_t.

In phase 2, the algorithm initially performs an arithmetic operation $\sum_{x\in S}|x\rangle|\widetilde{D}_x^t\rangle|[h_t(x) \neq c(x)]\rangle \rightarrow \sum_{x\in S}|x\rangle|\widetilde{D}_x^t[h_t(x) \neq c(x)]\rangle|[h_t(x) \neq c(x)]\rangle$ using $O(n)$ gates. Then a controlled reflection operator $V : |p\rangle|0\rangle|0\rangle \rightarrow |p\rangle\Big(\sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle\Big)|\sin^{-1}(\sqrt{p})\rangle$ is applied where the operation $|p\rangle|0\rangle|0\rangle \rightarrow |p\rangle|0\rangle|\sin^{-1}(\sqrt{p})\rangle$ is an arithmetic process and uses $O(n)$ gates while the operation $|p\rangle|0\rangle|\sin^{-1}(\sqrt{p})\rangle \rightarrow |p\rangle\Big(\sqrt{1-p}|0\rangle + \sqrt{p}|1\rangle\Big)|\sin^{-1}(\sqrt{p})\rangle$ uses one controlled rotation gate. The next step is phase estimation which involves applying QFT using $O(n \cdot \log n)$ gates. The time required for amplitude estimation in order to compute ε_t' is $O(\sqrt{M}Q^{3/2}T^3 \cdot tn^2)$: the $\sqrt{M}Q^{3/2}T^3$ calls are made to the unitaries $\widetilde{F}_t, \widetilde{F}_t^{-1}$ and each application of $\widetilde{F}_t : |0\rangle \rightarrow |\psi_6\rangle$ requires $O(n^2 \cdot t)$ time. The overall time complexity of the quantum algorithm is

$$\sum_{t=1}^{T} O\Big(n^2\sqrt{M}Q^{3/2}T^3t + n^2\sqrt{M}Qt + n^2Qt\Big) = \widetilde{O}(n^2\sqrt{M}Q^{3/2}T^5).$$

5.5.6 Reducing generalization error

In the previous section, we showed that our quantum boosting algorithm produces a hypothesis H that has training error 1/10 over the training set $\{(x_i, c(x_i))\}_{i\in[M]}$,

[15] As we mentioned earlier, we could also assume that a quantum learner has access to the uniform quantum examples $\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle$, in which case we do not need to assume a quantum RAM.

[16] A quantum circuit can perform arithmetic operations with the same time complexity as a Boolean circuit [Kit95].

where $(x_i, c(x_i))$ was sampled according to the unknown distribution $\mathcal{D}$. Now we consider Stage (2) of the algorithm. Recall that the goal of our quantum boosting algorithm is to output a hypothesis $H : \{0,1\}^n \to \{-1,1\}$ that satisfies

$$\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq 1 - \eta. \tag{5.43}$$

A priori, it is unclear if the output of the quantum boosting algorithm H satisfies Eq. (5.43). However, we saw in Theorem 5.4.1 that as long as M, i.e., the number of training examples (given as input to the quantum boosting algorithm) is *large enough*, then not only does H has *zero* training error, but it also ensures small *generalization error*, i.e., H also satisfies Eq. (5.43). In particular, Stage (2) of classical AdaBoost simply uses Theorem 5.4.1 to argue that: suppose the training error of H is 0, then the *generalization error* of H is at most η as long as $M \geq O(\mathsf{VC}(\mathcal{C})/\eta^2)$. Using Theorem 5.4.1, we can now prove our main theorem:

Theorem 5.5.6 (Complexity of Quantum Boosting algorithm) *Fix $\eta > 0$ and $\gamma > 0$. Let $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$ be a concept class and $\mathcal{A}$ be a γ-weak quantum PAC algorithm for $\mathcal{C}$ that takes time $Q(\mathcal{C})$. Let $n \geq 1$, $\mathcal{D} : \{0,1\}^n \to [0,1]$ be an unknown distribution and $c \in \mathcal{C}_n$ be the unknown target concept. Let*

$$M = \left\lceil \frac{\mathsf{VC}(\mathcal{C})}{\gamma^2} \cdot \frac{\log(\mathsf{VC}(\mathcal{C})/\gamma^2)}{\eta^2} \right\rceil.$$

Suppose we run Algorithm 7 for $T \geq ((\log M)\cdot\log(1/\delta))/(2\gamma^2)$ rounds, then with probability $\geq 1-\delta$ (over the randomness of the algorithm), we obtain a hypothesis H that has training error $1/10$ and generalization error

$$\Pr_{x\sim\mathcal{D}}[H(x) \neq c(x)] \leq 1/10 + \eta.$$

Moreover, the time complexity of the quantum boosting algorithm is

$$T_Q = O(n^2\sqrt{M}Q(\mathcal{C})^{3/2}T^5) = \widetilde{O}\left(\frac{\sqrt{\mathsf{VC}(\mathcal{C})}}{\eta}\cdot Q(\mathcal{C})^{3/2}\cdot\frac{n^2}{\gamma^{11}}\cdot\mathrm{polylog}(1/\delta)\right). \tag{5.44}$$

Picking $\eta = 1/10$ we get that H has generalization error at most 1/5. Recall that the complexity of classical AdaBoost using Theorem 5.4.1 is

$$T_C = \widetilde{O}\left(\frac{\mathsf{VC}(\mathcal{C})}{\eta^2}\cdot R(\mathcal{C})\cdot\frac{n}{\gamma^4}\log(1/\delta)\right).$$

In comparison, T_Q is quadratically better than T_C in terms of the VC dimension of the concept class $\mathcal{C}$ and $1/\eta$. Additionally, we could potentially have $Q(\mathcal{C}) \ll R(\mathcal{C})$

since the the quantum time complexity of a weak learner can be much less than the classical time complexity of learning a concept class as exhibited by [SG04] (under computational assumptions).

Chapter 6

Conclusions

We have studied three questions that are important for understanding the advantages and limitations of learning models in quantum computation and quantum control. First, in Chapter 3, we have investigated the problem of finding the optimal time required to implement an arbitrary unitary transformation, and provided a constructive approach for building a general unitary operation. The ability to build low-depth unitary operations in polynomial time can be a stepping stone towards developing NISQ devices [Pre18]. We have demonstrated two distinct approaches based on a sequential application of Hamiltonian operations for realizing unitary transformations in $SU(d)$. In the non-infinitesimal method under certain assumptions, the evolution times of the Hamiltonians are systematically controlled in order to reach any unitary with a desired accuracy in time $O(d^2/\varepsilon)$ while the complexity is slightly worse for the infinitesimal approach. The Solovay-Kitaev theorem shows that any unitary in $SU(d)$ can be approximated to a desired accuracy $\tilde{\varepsilon}$ with $O(d^2 \cdot \text{polylog}(1/\tilde{\varepsilon}))$ elementary one and two qubit gates [DN05]. We have demonstrated that the non-infinitesimal technique achieves the same time complexity $O(d^2)$ as the Solovay-Kitaev algorithm to construct an arbitrary unitary operation. We remark there is no relation in between ε and $\tilde{\varepsilon}$. The primary difference between the infinitesimal and non-infinitesimal approaches is that the infinitesimal method requires the construction of higher-order nested commutators to build a unitary operation in $SU(d)$. In contrast, the non-infinitesimal approach generates all basis elements in the Lie algebra $su(d)$ within the first order. This implies we can follow any direction in the space of unitaries by varying the parameters $\vec{t}, \vec{\tau}$ to the first order. A drawback of the infinitesimal approach is one needs to escape higher-order saddle points of a complex landscape to achieve a de-

sired unitary transformation. In contrast, the non-infinitesimal method provides a direct way for finding optimal solutions in the vicinity of the identity transformation. A further remark is the non-infinitesimal method can build any unitary in $SU(d)$ which is not possible with the infinitesimal approach.

Second, in Chapter 4, we follow up on the results of Chapter 3 by applying methods of gradient descent to learn the optimal alternating operator sequences required to implement a given unitary. Where in Chapter 3, we argued that the number of required terms in that sequence was $O(d^2)$, in Chapter 4, our numerical results indicate that the number of terms required is *exactly* d^2. We have investigated the difficulty of learning Haar random unitaries in $U(d)$ using parameterized alternating operator sequences. We have provided a detailed numerical analysis of the hardness of obtaining the optimal control parameters in an alternating operator sequence for learning both arbitrary unitaries and shallow depth unitaries using gradient descent optimization. For learning a Haar random target unitary in d dimensions to a desired accuracy, we find that gradient descent requires at least d^2 parameters in an alternating operator sequence. When there are fewer than d^2 parameters in the sequence, gradient descent converges to an undesirable local minimum of the loss function landscape which cannot be escaped with further gradient descent steps. This is true even for learning shallow-depth alternating operator target unitaries which are the alternating operator analogue of shallow depth quantum circuits. Gradient descent methods generally guarantee convergence only in convex spaces. The loss function landscape for unitaries is highly non-convex, and when we began this investigation, we did not know whether gradient descent on $2K \geq d^2$ parameters in the landscape would succeed in the search for a global minimum. Indeed, we expected that gradient descent would not always converge. However, in contrast to our initial expectations, we find that when the number of parameters in the loss function landscape is $2K \geq d^2$, gradient descent always converges to an optimal global minimum in the landscape. At the critical value of $2K = d^2$ parameters, we observe a *computational phase transition* characterized by a power law convergence to the global optimum.

In Chapter 5, we asked whether a quantum computer can convert a weak learning quantum algorithm into a strong learning quantum algorithm, and showed that a quantum computer can perform quadratically better (in some parameters of the algorithm including the VC dimension) than classical AdaBoost. We remark that the VC dimension could potentially be exponential in n for some concept

classes whereas $1/\gamma$ is poly(n) in AdaBoost. Additionally, our quantum boosting algorithm provides a quadratic improvement in the parameter η which is the generalization error of the final strong hypothesis. We have introduced a classical robust boosting algorithm which can be faster than AdaBoost. To the best of our knowledge, our robust boosting framework is the *first boosting algorithm* that can beat the time complexity of AdaBoost. We conclude by providing a discussion on future work in Section 6.1 and discuss the outlook of learning models in paving the way for new directions in the interface of physics and computer science in Section 6.2.

6.1 Future Work

A general future goal would be to use the techniques introduced in the above three chapters and develop quantum learning algorithms for a noisy quantum computer that can accurately compute a physical quantity, with high probability, from the training data. For example, can a quantum computer be trained to classify different topological phases of quantum matter? Can a quantum learner be trained with input data corrupted with noise and still perform well on test data? In order to address these questions, we need to focus on resolving the issues which can provide insights about problems of more general interest.

- We found that both the infinitesimal and non-infinitesimal methods can generate directions in the unitary manifold that are either nested commutators or their linear combinations which is a Lie polynomial. An interesting question in this direction is to determine the hardness of building unitary operations generated by the polynomials $A^m B^n \pm B^n A^m$ where A, B are bounded hermitian matrices and $m, n > 0$. This is intrinsically related to the question of Hamiltonian complexity for simulating time-evolution of physical systems where the dynamics is governed by the operator $A^m B^n \pm B^n A^m$. We have also left open the analysis of the sensitivity of our constructive approaches in the presence of experimental noise. This is important since we can implement the Hamiltonians A, B and their time evolutions $\vec{t}, \vec{\tau}$ with a finite precision in the laboratory. A more ambitious project is to develop a constructive approach (using a sequential application of non-commuting Hamiltonians) to build any unitary in time $O(d^2 \cdot \text{polylog}(1/\varepsilon))$. Can we improve the complexity of the accuracy ε in the infinitesimal or non-infinitesimal approach?

- Near-term quantum devices can perform computations using shallow depth quantum circuits. This stimulates the study of learning shallow depth target unitaries with parameterized circuits composed of local operations using gradient descent. A quantum circuit composed of Clifford (CNOT, Hadamard and phase gates) and T gates can perform an arbitrary unitary operation. It would be interesting to study the hardness of the learnability of target unitaries consisting of Clifford gates and a small number of T gates with an alternating operator sequence using gradient descent. Another promising avenue is to investigate the distribution of local critical points and saddle points in the landscape of the loss function of unitary manifolds [CHM+15].

- Recently, there has been a lot of interest in understanding the time complexity or the circuit depth (where the circuit is composed of a pair of quantum alternating operators applied sequentially) of simulating *physical states* such as ground states of a Hamiltonian [WMS20]. A comprehensive study of simulating symmetry constrained physical states using low depth circuits is of central interest to the quantum information community. For example, instead of considering the learnability of a Haar random target unitary as discussed in the thesis, the goal is to study the learnability of matrix product states or tensor network states (such states have entanglement constraints and symmetry constraints as well) [Orú19]. The immediate question that arises is, what is the minimum number of parameters required in an alternating operator sequence for learning matrix product states or tensor network states? Can we learn local unitaries using poly-logarithmic number of parameters in an alternating operator sequence where the generators A, B comprises of one and two-qubit local Hamiltonians? In particular, can we drive an initial state to a desired physical state using alternating applications of the Ising and single qubit Pauli σ_x Hamiltonians?

- There are a few interesting questions that need to be addressed to improve the performance of our quantum boosting algorithm. We believe that there should be a $\Omega(\sqrt{VC})$ lower bound on the time complexity of quantum boosting. We have left open the search for a lower bound on the quantum time complexity in this thesis and further work is required to prove that $\Omega(\sqrt{VC})$ is the optimal lower bound. Furthermore, it would be interesting to find the optimal lower bound on the time complexity in terms of the factor $1/\gamma$. The dependence on $1/\gamma$ in the quantum time complexity is not favorable. Although,

this is a shortcoming of our techniques, we strongly believe (like many other optimization algorithms in quantum computing) that future works can improve the dependence and maybe even improve upon the $1/\gamma^4$ classical complexity. There is indeed a large body of work left to be done in this direction since it would be interesting to see if recent results of Alon et al. [AGHM20] can be used to improve the $1/\gamma$ dependence. This stimulates the question, can we improve the polynomial dependence on $1/\gamma$ in the quantum time complexity of boosting? Can we use the quantum boosting algorithm to improve the complexities of various quantum algorithms that use *classical* AdaBoost to amplify the performance of a weak quantum algorithm? Is it possible to find *practically relevant* concept classes which have large VC dimension for which our quantum boosting algorithm gives a large quantum speedup? Since near term quantum computers would be noisy and error prone, is it possible to replace the quantum phase estimation step in our quantum boosting algorithm by variational techniques developed by Peruzzo et al. [PMS+14b]? Is there an application to bandit optimization [Sli19] using our techniques? Our work is the first step in this direction.

- Our quantum boosting algorithm is designed to learn a concept class comprising of Boolean valued functions, i.e., our work shows how to convert a weak Boolean learner into a strong Boolean learner. In future works, we aim to extend our quantum boosting algorithm to quantize real AdaBoost [FS97, Kég03], multi-class AdaBoost [DB95, FS97] and probabilistic Ababoost [Gro06]. Suppose the goal is to learn the expectation values of measurements performed on a quantum state, such as in quantum state tomography. The problem is to show that the real quantum boosting algorithm (or quantum real AbaBoost) can improve the time complexity of learning a quantum state in Aaronson's state tomography setting [Aar07]. Another interesting avenue to explore is when the training examples in AdaBoost are corrupted with classification noise. A no-go theorem in machine learning states that a weak learner cannot be boosted to a strong learner using noisy training data [LS08]. In the quantum setting, the task would be to show that, given noisy quantum examples, one can *still* convert a quantum weak learner into a strong learner.

6.2 Outlook

Learning models in computer science have enjoyed tremendous success in the past couple of decades. The PAC learning framework of Valiant [Val84] conclusively settled the question, "what does it mean for a machine to learn?". Further progress in machine learning such as Empirical Risk Minimization (ERM) and Structural Risk Minimization (SRM) provided a thorough understanding of the question, "how can a machine learn?". The task of a learning model or a learning algorithm is to enable a machine detect interesting patterns in data. In the real world, we often have to extract information from large data sets which might not be feasible for a human programmer. It is important to note that the patterns in large data sets can be extremely complicated and a human or a traditional computer would require enormous amounts of resources (which is impractical) to accomplish the tasks. However, the works of Valiant and other researchers have shown that machine learning techniques can equip computer programs with the ability to *learn and adapt* and *adapt and learn* with favorable amount of resources. In this age, our lives are deeply associated with machine learning based technology: our smartphones are endowed with face detection tools and intelligent personal assistance applications, cars are facilitated with safe driving machine learning softwares, credit cards are enabled with algorithms that learn to detect fraudulent activities and many more.

The work of Bshouty and Jackson [BJ99] commenced the study of learning models in the context of a quantum advantage. Their work sparked a lot of interest in how quantum principles can be utilized to obtain a *speedup* over classical machine learning models. Recent results in machine learning have been promising to understand the properties of quantum many-body systems [GPA$^+$18], analyze data in particle physics [Bou19] and astronomy [Bar19]. The union of computer science and physics has opened new directions of research and it is important to develop a rigorous understanding of the advantages and limitations of various quantum and classical learning models. In the near future, machine learning techniques will have a deeper impact in how we study basic sciences, either theoretical or applied.

Appendix A

Free Lie Algebra

The contents of this appendix is standard literature [Reu01, Bac01, BP07, GK17]. For the sake of completeness, we define a free Lie algebra that has been previously studied in the literature in the context of out-of-time-ordered correlators (OTOCs) [HLNR17] and in relation to Maxwell algebra [GK17].

A free Lie algebra $L(X)$ is the maximal Lie algebra that can be constructed over a generating set $\{X_a\}$ where $a = 1, 2, .., q$ such that skew symmetry and Jacobi identity holds. Since the generators do not satisfy any additional imposed relations, the free Lie algebra is infinite dimensional and it is the linear space spanned by nested commutators of the form $[X_{a_i}, [X_{a_j}, [\ldots [X_{a_l}, X_{a_m}]]\ldots]$. The homogeneous part of degree k of the free Lie algebra refers to a free Lie subalgebra that is spanned by k-th degree nested commutators. For example, the free Lie subalgebra spanned by all commutators of the form $[X_a, X_b]$ is a space of dimension $q(q-1)/2$.

Definition. A free Lie algebra L on a set X is a Lie algebra with a mapping $i : X \to L$ that satisfies the following universal property. For every Lie algebra M with the mapping $g : X \to M$, there exists a unique Lie algebra homomorphism $G : L \to M$ such that $g = G \circ i$.

A general property of the map G when $|X| = \dim(M)$, G is surjective. One can prove that there exists a unique free Lie algebra L generated by a set X. The basis elements of a free Lie algebra L can be constructed in terms of Lyndon words which gives the Lyndon basis. Lyndon words have wide ranging applications in algebra and combinatorics.

Definition. A Lyndon word is a primitive word which is strictly smaller than all

its non trivial cyclic rotations.

For example, the Lyndon words for the two-symbol binary alphabet $\{a, b\}$ sorted by length and then lexicographically forms an infinite sequence given by,

$$\{a,b\},\{ab\},\{aab,abb\},\{aaab,aabb,abbb\},\{aaaab,aaabb,aabab,aabbb,ababb,abbbb\},\ldots \tag{A.1}$$

The number of Lyndon words of length k on q symbols is given by the Witt formula. A Lyndon word u that is not a letter has the following property of being expressed as $u = vw$ where v, w are Lyndon words with $v < w$ lexicographically. In general, this is not a unique factorisation since, for example, $(a)(abb) = (aab)(b)$. However there is a unique factorisation of a Lyndon word u as a product of two Lyndon words v, w with $v < w$ termed as the standard factorisation. An important theorem in this context is given below.

Theorem.(Chen-Fox-Lyndon) If w is lexicographically the smallest proper suffix of a Lyndon word $u = vw$, then v and w are also Lyndon words such that $v < w$.

The standard factorisation of a Lyndon word is obtained by selecting w to be the lexicographically least proper suffix of u which is also the longest proper suffix of u that is a Lyndon word. In the above example, $(a)(abb)$ is the standard factorisation of $u = aabb$.

There exists a bijection $\mathcal{M}$ from the set of Lyndon words to the basis elements of a free Lie algebra. $\mathcal{M}$ is defined as follows. If the word u is a letter, then $\mathcal{M}(u) = u$. When the length of u is greater than or equal to two, then by standard factorisation, $u = vw$ for Lyndon words v, w and w being the longest possible suffix. Thus $\mathcal{M}(u) = [\mathcal{M}(v), \mathcal{M}(w)]$. For example, the standard factorisation $(a)(ababb)$ of the Lyndon word $aababb$ can be mapped to the commutator $[a,[[a,b],[[a,b],b]]]$.

It is evident that the basis elements of free Lie algebra L are homogeneous polynomials in the elements of the generating set X. The free Lie algebra vector space can be expressed as a direct sum of graded Lie algebras given by,

$$L = \bigoplus_{k>0} L_k = L_1 \oplus L_2 \oplus \ldots \tag{A.2}$$

where L_1 is a q-dimensional vector space spanned by the elements of X with $|X|$=q. The vector space L_2 is spanned by commutators of the form $[X_a, X_b]$ with the corresponding dimension equals $q(q-1)/2$. L_2 can also be expressed as $[L_1, L_1]$. Technically, L_2 is referred to as the exterior square of L_1. Notice that this holds true for

any arbitrary $k > 0$,

$$L_{k+1} = [L_k, L_1]. \tag{A.3}$$

By the definition of graded Lie algebras, we have the following relation $[L_k, L_{k'}] \subseteq L_{k+k'}$. An important property of any subalgebra of a free Lie algebra is due to Shirsov and Witt.

Theorem.(Shirshov–Witt) Any Lie subalgebra of a free Lie algebra is a free Lie algebra.

This is an analogue of the Nielsen-Schreier theorem in group theory which states that every subgroup of a free group is free.

Appendix B

PLI Nested Commutators

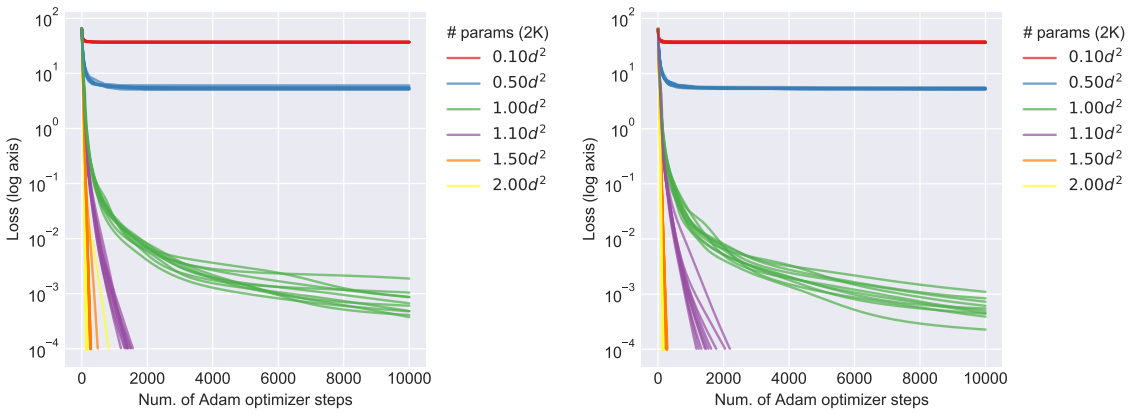
In this appendix, we have tabulated the linearly independent nested commutators for random matrices A and B in $su(d)$.

Order	Commutators	Linearly Independent	N_{min}
1	A, B	A, B	
2	[A,B]	[A,B]	4
3	[A,[A,B]], [B,[A,B]]	[A,[A,B]], [B,[A,B]]	8
4	[A,[A,[A,B]]], [B,[B,[A,B]]], [A,[B,[A,B]]], [B,[A,[A,B]]]	[A,[A,[A,B]]], [B,[B,[A,B]]], [A,[B,[A,B]]]	12
5	[A,[A,[A,[A,B]]]], [B,[B,[B,[A,B]]]], [A,[B,[A,[A,B]]]], [B,[A,[B,[A,B]]]], [A,[A,[B,[A,B]]]], [B,[B,[A,[A,B]]]], [B,[A,[A,[A,B]]]], [A,[B,[B,[A,B]]]], [[A,B],[A,[A,B]]], [[A,B],[B,[A,B]]]	[A,[A,[A,[A,B]]]], [B,[B,[B,[A,B]]]], [A,[A,[B,[A,B]]]], [B,[B,[A,[A,B]]]], [B,[A,[A,[A,B]]]], [A,[B,[B,[A,B]]]]	~ 18

Table B.1: The first column represents the order of the Taylor expansion of Eq. (3.1), the second column includes all possible commutators for a given order k, the third column indicates the linearly independent commutators and the fourth column specifies the minimum number of parameters N_{min} required to exponentiate a k-th degree nested commutator.

Appendix C

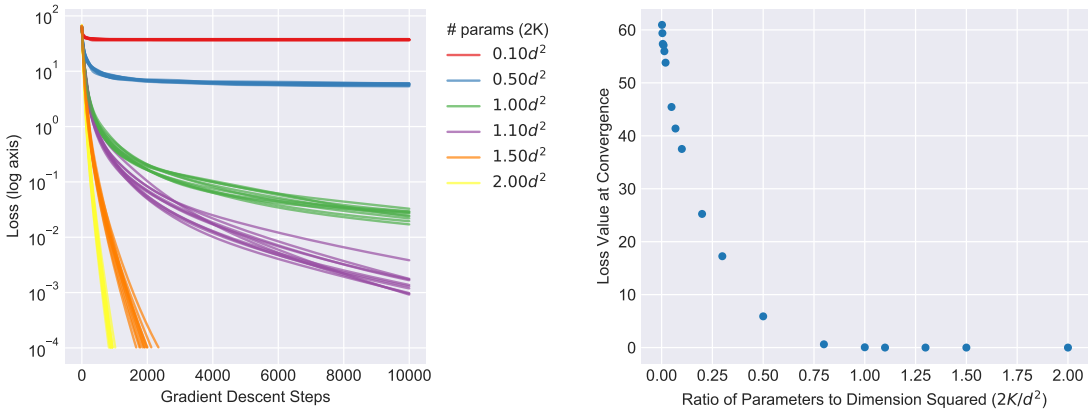
Experiments using Adam optimizer



(a) **(b)**

Figure C.1: a) Experiments using Adam gradient descent for a Haar random target unitary $\mathcal{U}$. b) Experiments using the Adam optimizer for a low-depth target unitary $\mathcal{U}$ of dimension 32 with 8 parameters (N=4).

In addition to performing optimization using simple (vanilla) gradient descent, we performed optimization using the Adam optimizer [KB15], a common optimization method used in deep learning. Adaptive Moment Estimation or Adam is an upgrade of the simple gradient descent algorithm where parameters are assigned different learning rate which are adaptively computed in every iteration of the algorithm. These updates are solely computed from first order gradients. In contrast, the learning rate is fixed for each parameter in simple gradient descent. For more on the Adam optimizer, the reader is referred to [KB15]. The final loss obtained for learning unitary matrices using the Adam optimizer was consistent with those obtained from simple gradient descent. However, the Adam



(a) **(b)**

Figure C.2: a) Simple gradient descent experiments for a target unitary $\mathcal{U}$ of dimension 32 with 8 parameters (N=4). b) Value of the loss function after convergence with 10,000 steps of simple gradient descent. Experiments were performed for $\mathcal{U}$ of dimension 32 with various number of parameters.

optimizer appears to converge to a final outcome in far fewer steps. The results of our experiments are provided in Fig. (C.1). A comparison between the performance of simple and Adam gradient descent can be observed from Fig. (C.1) and Fig. (C.2).

C.1 Critical points in the under-parameterized models

When learning target unitaries using alternating operator sequences with d^2 parameters or more, gradient descent converges to a global minimum of the loss function landscape. When learning with under-parameterized models, we find that gradient descent plateaus at a non-zero loss function value. In the under-parameterized setting, we further explore how the loss function changes over the course of gradient descent by investigating the magnitude of the gradients. In the under-parameterized setting, we find that the magnitude of the gradients can both increase and decrease over the course of gradient descent, suggesting that the path of gradient descent passes in the vicinity of saddle points in the loss landscape. In the over-parameterized setting, the magnitudes of the gradients monotonically decrease with increasing gradient descent steps, suggesting that in this case, the path of gradient descent does not explore saddle points. The results of our findings are presented in Fig. (C.3).

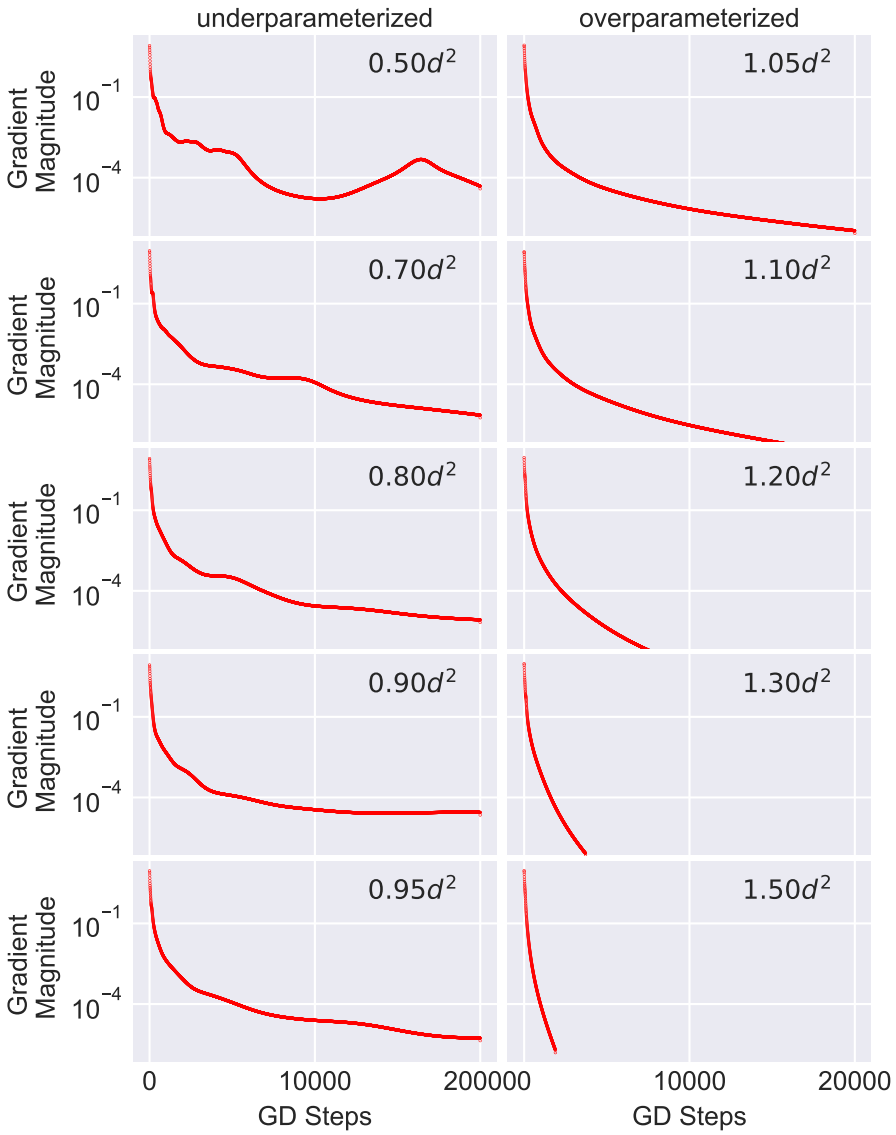


Figure C.3: Magnitude of the gradient at each step of gradient descent. In the under-parameterized setting, we find that the magnitudes can increase and decrease over the course of gradient descent. In the over-parameterized setting, we find that the magnitudes decrease, often rapidly, over the course of gradient descent. For each parameter setting, a single experiment was performed which has been plotted here.

Bibliography

[Aar07] S. Aaronson. The learnability of quantum states. *Proceedings of the Royal Society of London*, 463(2088), 2007. arXiv:quant-ph/0608142. 5, 61, 104

[Aar15] S. Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015. 1, 80

[Aar18] S. Aaronson. Shadow tomography of quantum states. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 325–338, 2018. 5, 61

[ABG07] E. Aïmeur, G. Brassard, and S. Gambs. Quantum clustering algorithms. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML)*, pages 1–8, 2007. 5, 61

[ACL+19] S. Arunachalam, S. Chakraborty, T. Lee, M. Parashar, and R. de Wolf. Two new results about quantum exact learning. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 16:1–16:15, 2019. arXiv:1810.00481. 5, 61

[AG19] J. van Apeldoorn and A. Gilyén. Improvements in quantum SDP-solving with applications. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 99:1–99:15, 2019. arXiv:1804.05058. 61

[AGGW20] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020. arXiv:1809.00643. 61

[AGHM20] N. Alon, A. Gonen, E. Hazan, and S. Moran. Boosting simple learners, 2020. arXiv:2001.11704. 104

[AGJO+15] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, and P. V. Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015. arXiv:1502.03450. 80

[AHK12] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. 13

[Amb10] A. Ambainis. Quantum search with variable times. *Theory of Computing Systems*, 47(3):786–807, 2010. 22, 75

[AS05] A. Atıcı and R. Servedio. Improved bounds on quantum learning algorithms. *Quantum Information Processing*, 4(5):355–386, 2005. arXiv:quant-ph/0411140. 5, 61

[ASB16] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, 2016. 48

[AT03] D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 20–29, 2003. 27

[AW17] S. Arunachalam and R. de Wolf. Guest column: A survey of quantum learning theory. *SIGACT News*, 48(2):41–67, 2017. arXiv:1701.06806. 11

[AW18] S. Arunachalam and R. de Wolf. Optimal quantum sample complexity of learning algorithms. *Journal of Machine Learning Research*, 19(71):1–36, 2018. Earlier version in CCC'17. arXiv:1607.00932. 62

[Bac01] R. Bacher. The on-line encyclopedia of integer sequences (OEIS). http://oeis.org/A059966, 2001. A059966. 40, 106

[BACS07] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007. 27

[Bar19] D. Baron. Machine learning in astronomy: A practical overview. 2019. arXiv:1904.07248. 105

[BC12] D. W. Berry and A. M. Childs. Black-box hamiltonian simulation and unitary implementation. *Quantum Information & Computation*, 12(1-2):29–62, 2012. 27

[BCC+14] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Exponential improvement in precision for simulating sparse hamiltonians. In *Symposium on Theory of Computing, STOC*, pages 283–292. ACM, 2014. 25, 28

[BCC+15] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Physical review letters*, 114(9):090502, 2015. 25, 28

[BCR10] C. Brif, R. Chakrabarti, and H. Rabitz. Control of quantum phenomena: past, present and future. *New Journal of Physics*, 12(7):075008, 2010. 23, 26, 33, 39, 45, 46, 47

[BDC+18] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018. 1

[BDGT11] G. Brassard, F. Dupuis, S. Gambs, and A. Tapp. An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance, 2011. arXiv:1106.4267. 64

[BHMT02] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. 21, 22

[BJ99] N. H. Bshouty and J. C. Jackson. Learning DNF over the uniform distribution using a quantum example oracle. *SIAM Journal on Computing*, 28(3):1136–1153, 1999. 5, 61, 105

[Bou19] D. Bourilkov. Machine and deep learning applications in particle physics, 2019. arXiv:1912.08245. 105

[BP07] J. Berstel and D. Perrin. The origins of combinatorics on words. *European Journal of Combinatorics*, 28(3):996–1022, 2007. 106

[BV93] E. Bernstein and U. V. Vazirani. Quantum complexity theory. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 11–20, 1993. 5, 61

[CCC+19] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019. 1

[CCLW20] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:220, 2020. arXiv:1809.01731. 61

[CGL+19] N. Chia, A. Gilyén, T. Li, H. Lin, E. Tang, and C. Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning, 2019. arXiv:1910.06151. 61

[Chi10] A. M. Childs. On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603, 2010. 27

[CHKN17] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson. Machine learning in the string landscape. *Journal of High Energy Physics*, 2017(9):157, 2017. 1

[CHM+15] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, 2015. 103

[CM17] J. Carrasquilla and R. G. Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, 2017. 1

[CMO+20] J. Carolan, M. Mohseni, J. P. Olson, M. Prabhu, C. Chen, D. Bunandar, M. Y. Niu, N. C. Harris, F. Wong, M. Hochberg, et al. Variational quantum unsampling on a quantum photonic processor. *Nature Physics*, pages 1–6, 2020. 46

[CR07] R. Chakrabarti and H. Rabitz. Quantum control landscapes. *International Reviews in Physical Chemistry*, 26(4):671–735, 2007. 4, 23, 26, 33, 39, 45, 46, 47

[CW12] A. M. Childs and N. Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Information & Computation*, 12(11-12):901–924, 2012. 28

[CW13] A. M. Childs and N. Wiebe. Product formulas for exponentials of commutators. *Journal of Mathematical Physics*, 54(6):062202, 2013. 38

[DB95] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal Artificial Intelligence Research*, 2:263–286, 1995. 104

[DN05] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev algorithm, 2005. arXiv: quant-ph/0505030. 19, 20, 28, 100

[DR93] M. Demiralp and H. Rabitz. Optimally controlled quantum molecular dynamics: A perturbation formulation and the existence of multiple solutions. *Physical Review A*, 47(2):809, 1993. 23

[Fey82] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. 24, 25, 27

[FGG14a] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm, 2014. arXiv:1411.4028. 45, 48

[FGG14b] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, 2014. arXiv:1412.6062. 45, 48

[FGGS00] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution, 2000. arXiv:quant-ph/0001106. 27

[FH16] E. Farhi and A. W. Harrow. Quantum supremacy through the quantum approximate optimization algorithm, 2016. arXiv:1602.07674. 45, 46, 48

[FHT01] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. 2001. 1

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995. Earlier in COLT'90. 12

[FS97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. Earlier version in EuroCOLT'95. 5, 104

[FS99] Y. Freund and R. Schapire. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14:771–780, 1999. 12, 61, 70, 71

[GAN14] I. M. Georgescu, S. Ashhab, and F. Nori. Quantum simulation. *Reviews of Modern Physics*, 86(1):153, 2014. 28

[GAW19] A. Gilyén, S. Arunachalam, and N. Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2019. 46, 61

[GCS17] J. M. Gambetta, J. M. Chow, and M. Steffen. Building logical qubits in a superconducting quantum computing system. *npj Quantum Information*, 3(1):1–7, 2017. 28

[GK17] J. Gomis and A. Kleinschmidt. On free lie algebras and particles in electro-magnetic fields. *Journal of High Energy Physics*, 2017(7):85, 2017. 106

[GLM08] V. Giovannetti, S. Lloyd, and L. Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008. 80

[GPA+18] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac. Neural-network quantum states, string-bond states, and chiral topological states. *Physical Review X*, 8(1):011006, 2018. 105

[GR02] L. K. Grover and T. Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. arXiv:quant-ph/0208112. 73

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 212–219, 1996. arXiv:quant-ph/9605043. 21

[Gro06] E. Grossmann. A theory of probabilistic boosting, decision trees and matryoshki, 2006. arXiv:cs/0607110. 104

[GZD17] X. Gao, Z. Zhang, and L. Duan. An efficient quantum algorithm for generative machine learning, 2017. arXiv:1711.02038. 61

[HCT+19] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209, 2019. arXiv:1804.11326. 61

[HDR09] T.-S. Ho, J. Dominy, and H. Rabitz. Landscape of unitary transformations in controlled quantum dynamics. *Physical Review A*, 79(1):013422, 2009. 4, 26, 33, 39, 45, 46, 47, 48

[HG01] D. Horn and A. Gottlieb. Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Physical review letters*, 88(1):018702, 2001. 1

[HHJ+10] D. Hanneke, J. P. Home, J. D. Jost, J. M. Amini, D. Leibfried, and D. J. Wineland. Realization of a programmable two-qubit quantum processor. *Nature Physics*, 6(1):13–16, 2010. 28

[HHKL18] J. Haah, M. B. Hastings, R. Kothari, and G. H. Low. Quantum algorithm for simulating real time evolution of lattice hamiltonians. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 350–360. IEEE Computer Society, 2018. 28

[HHL09] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 15:150502, 2009. arXiv:0811.3171v3. 61

[HLNR17] F. M. Haehl, R. Loganayagam, P. Narayan, and M. Rangamani. Classification of out-of-time-order correlators, 2017. arXiv:1701.02820. 106

[JGS19] D. Jethwani, F. Le Gall, and S. Singh. Quantum-inspired classical algorithms for singular value transformation. 2019. arXiv:1910.05699. 61

[JRW17] Z. Jiang, E. Rieffel, and Z. Wang. A QAOA-inspired circuit for Grover's unstructured search using a transverse field, 2017. arXiv:1702.0257. 46

[JSD+17] L. Jing, Y. Shen, T. Dubcek, J. Peurifoy, S. Skirlo, Y. LeCun, M. Tegmark, and M. Soljacic. Tunable efficient unitary neural networks and their

application to rnns. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017. 48

[KB15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015. 16, 110

[Kég03] B. Kégl. Robust regression by boosting the median. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory*, pages 258–272, 2003. 104

[Kit95] A. Y. Kitaev. Quantum measurements and the Abelian stabilizer problem, 1995. arXiv:quant-ph/9511026. 97

[Kit97] A. Y. Kitaev. Quantum computations: algorithms and error correction. *Uspekhi Matematicheskikh Nauk*, 52(6):53–112, 1997. 3, 19, 28

[Kli13] V. Kliuchnikov. Synthesis of unitaries with clifford+t circuits, 2013. arXiv:1306.3200. 29

[KLLP19] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash. q-means: A quantum algorithm for unsupervised machine learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pages 4136–4146, 2019. arXiv: 1812.03584. 5, 61

[KLM20] B. T. Kiani, S. Lloyd, and R. Maity. Learning unitaries by gradient descent, 2020. arXiv:2001.11897. 6

[KLP+19] S. Khatri, R. LaRose, A. Poremba, L. Cincio, A. T. Sornborger, and P. J. Coles. Quantum-assisted quantum compiling. *Quantum*, 3:140, 2019. 46

[KLS19] C. P. Koch, M. Lemeshko, and D. Sugny. Quantum control of molecular rotation. *Reviews of Modern Physics*, 91(3):035005, 2019. 23

[KM01] P. Kaye and M. Mosca. Quantum networks for generating arbitrary quantum states. In *International Conference on Quantum Information Optical Society of America*, 2001. arXiv:quant-ph/0407102. 73

[KMM16] V. Kliuchnikov, D. Maslov, and M. Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum clifford and t circuits. *IEEE Transactions on Computers*, 65(1):161–172, 2016. 29

[KRK+05] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser. Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296–305, 2005. 4, 26, 33, 39, 45, 46, 47

[KV94] M. J. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT press, 1994. 7

[KWS16] A. Kapoor, N. Wiebe, and K. Svore. Quantum perceptron models. In *Proceedings of Neural Information Processing Systems'16*, pages 3999–4007, 2016. arxiv:1602.04799. 61

[LB99] S. Lloyd and S. L. Braunstein. Quantum computation over continuous variables. In *Quantum information with continuous variables*, pages 9–17. Springer, 1999. 38

[LC17] G. H. Low and I. L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical review letters*, 118(1):010501, 2017. 25, 28

[LC19] G. H. Low and I. L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019. 25

[LCW19] T. Li, S. Chakrabarti, and X. Wu. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 3815–3824, 2019. arXiv:1904.02276. 61

[Llo96] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996. 24, 25, 26, 27

[Llo18] S. Lloyd. Quantum approximate optimization is computationally universal, 2018. arXiv:1812.11075. 45

[LM14] S. Lloyd and S. Montangero. Information theoretical analysis of quantum optimal control. *Physical review letters*, 113(1):010502, 2014. 32, 41

[LM19] S. Lloyd and R. Maity. Efficient implementation of unitary transformations, 2019. arXiv:1901.03431. 6, 46

[LMR14] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014. 28

[LS08] P. M. Long and R. A. Servedio. Random classification noise defeats all convex potential boosters. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, volume 307, pages 608–615. ACM, 2008. 104

[LV01] S. Lloyd and L. Viola. Engineering quantum dynamics. *Physical Review A*, 65(1):010101, 2001. 26

[LW18a] S. Lloyd and C. Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018. 61

[LW18b] G. H. Low and N. Wiebe. Hamiltonian simulation in the interaction picture, 2018. arXiv:1805.00675. 28

[MHR08] K. Moore, M. Hsieh, and H. Rabitz. On the relationship between quantum control landscape structure and optimization complexity. *The Journal of chemical physics*, 128(15):154117, 2008. 4, 26, 33, 39, 45, 46, 47, 49

[ML16] I. Marvian and S. Lloyd. Universal quantum emulator, 2016. arXiv:1606.02734. 37

[MRBA16] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016. 46

[MRS17] R. Maity, P. Roy, and T. Sarkar. Black hole phase transitions and the chemical potential. *Physics Letters B*, 765:386–394, 2017. 6

[MVBS04] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa. Quantum circuits for general multiqubit gates. *Physical review letters*, 93(13):130502, 2004. 28

[NC16] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. 7, 26, 28

[NDRM12] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready. Qboost: Large scale classifier training with adiabatic quantum optimization. In *ACML*, pages 333–348, 2012. 62, 69

[NW99] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the Thirty-First*

Annual ACM Symposium on Theory of Computing, pages 384–393, 1999. 64

[Orú19] R. Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, 2019. 103

[PISR06] A. Pechen, N. Il'in, F. Shuang, and H. Rabitz. Quantum control by von neumann measurements. *Physical Review A*, 74(5):052102, 2006. 28

[PMB13] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, 2013. 47

[PMS+14a] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014. 46

[PMS+14b] A. Peruzzo, J. McClean, P. Shadbolt, M-H. Yung, X-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and Jeremy L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014. arXiv:1304.3061v1. 63, 104

[Pra14] A. Prakash. Quantum algorithms for linear algebra and machine learning, 2014. PhD Thesis. 5, 61

[Pre18] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018. arXiv:1801.00862. 63, 100

[qua] 6

[Reu01] C. Reutenauer. Free lie algebras. Technical report, 2001. 40, 106

[RHH+06] H. Rabitz, T.-S. Ho, M. Hsieh, R. Kosut, and M. Demiralp. Topology of optimally controlled quantum mechanical transition probability landscapes. *Physical Review A*, 74(1):012721, 2006. 4, 26, 33, 39, 45, 46, 47

[RHR04] H. Rabitz, M. Hsieh, and C. Rosenthal. Quantum optimally controlled transition landscapes. *Science*, 303(5666):1998–2001, 2004. 4, 26, 33, 39, 45, 46, 47, 48

[RHR05] H. Rabitz, M. Hsieh, and C. Rosenthal. Landscape for optimal control of quantum-mechanical unitary transformations. *Physical Review A*, 72(5):052337, 2005. 4, 26, 33, 39, 45, 46, 47, 48

[RML14] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014. 5, 61

[Roc18] A. Rocchetto. Stabiliser states are efficiently PAC-learnable. *Quantum Information & Computation*, 18(7&8):541–552, 2018. 5, 61

[RRW17] B. Russell, H. Rabitz, and R.-B. Wu. Quantum control landscapes are almost always trap free. *Journal of Physics A: Mathematical and Theoretical*, 50(20):205302, 2017. 23, 26, 33, 39, 45, 46, 47, 48, 57

[RS16] N. J. Ross and P. Selinger. Optimal ancilla-free clifford+t approximation of z-rotations. *Quantum Information & Computation*, 16(11-12):901–953, 2016. 29

[RSW+19] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd. Quantum gradient descent and Newton's method for constrained polynomial optimization. *New Journal of Physics*, 2019. 61

[RTB+15] G. Riviello, K. M. Tibbetts, C. Brif, R. Long, R.-B. Wu, T.-S. Ho, and H. Rabitz. Searching for quantum optimal controls under severe constraints. *Physical Review A*, 91(4):043401, 2015. 23, 26, 33, 39, 45, 46, 47, 49

[Rud16] S. Ruder. An overview of gradient descent optimization algorithms, 2016. arXiv:1609.04747. 16

[RWSR17] G. Riviello, R.-B. Wu, Q. Sun, and H. Rabitz. Searching for an optimal control in the presence of saddles on the quantum-mechanical observable landscape. *Physical Review A*, 95(6):063418, 2017. 23, 26, 33, 39, 45, 46, 47, 49

[SBM06] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006. 28

[SBT+18] V. M. Schäfer, C. J. Ballance, K. Thirumalai, L. J. Stephenson, T. G. Ballance, A. M. Steane, and D. M. Lucas. Fast quantum logic gates with trapped-ion qubits. *Nature*, 555(7694):75–78, 2018. 28

[Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. Earlier in FOCS'89. 12

[Sel12] P. Selinger. Efficient clifford+t approximation of single-qubit operators, 2012. arXiv:1212.6253. 29

[SF12] R.E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012. 5, 12, 70, 71

[SG04] R. A. Servedio and S. J. Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004. 62, 99

[SGHZ19] H. Shen, D. George, E. A. Huerta, and Z. Zhao. Denoising gravitational waves with enhanced deep recurrent denoising auto-encoders. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3237–3241. IEEE, 2019. 1

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society, 1994. 21

[SKCC19] K. Sharma, S. Khatri, M. Cerezo, and P. J. Coles. Noise resilience of variational quantum compiling, 2019. arXiv:1908.04416. 46

[SL11] S. Sefi and P. V. Loock. How to decompose arbitrary continuous-variable quantum operations. *Physical review letters*, 107(17):170501, 2011. 38

[Sli19] A. Slivkins. Introduction to multi-armed bandits, 2019. arXiv:1904.07272. 104

[SP18] M. Schuld and F. Petruccione. Quantum ensembles of quantum classifiers. *Scientific reports*, 8(1):2772, 2018. 62, 69

[Suz90] M. Suzuki. Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations. *Physics Letters A*, 146(6):319–323, 1990. 25, 30

[Suz91] M. Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991. 25, 30

[Tro59] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959. 25

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 10, 105

[VC71] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. English translation of 1968 Russian paper in *Dokl. Akad. Nauk.* 181(4). 11

[VC15] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015. 9

[WBHS10] N. Wiebe, D. W. Berry, P. Høyer, and B. C. Sanders. Higher order decompositions of ordered operator exponentials. *Journal of Physics A: Mathematical and Theoretical*, 43(6):065203, 2010. 26, 27

[WBHS11] N. Wiebe, D. W. Berry, P. Høyer, and B. C. Sanders. Simulating quantum dynamics on a quantum computer. *Journal of Physics A: Mathematical and Theoretical*, 44(44):445308, 2011. 27

[WD13] H.-R. Wei and F.-G. Deng. Universal quantum gates for hybrid systems assisted by quantum dots inside double-sided optical microcavities. *Physical Review A*, 87(2):022305, 2013. 28

[Wit56] E. Witt. Die unterringe der freien lieschen ringe. *Mathematische Zeitschrift*, 64(1):195–216, 1956. 40

[WMHY19] X. Wang, Y-C. Ma, M-H. Hsieh, and M-H. Yung. Quantum speedup in adaptive boosting of binary classification. 2019. arXiv: 1902.00869. 62, 69

[WMS20] M. M. Wauters, G. B. Mbeng, and G. E. Santoro. Polynomial scaling of qaoa for ground-state preparation: taming first-order phase transitions, 2020. arXiv:2003.07419. 103

[Wol19] R. de Wolf. Quantum computing: Lecture notes, 2019. arXiv:1907.09415. 7

[Yog19] M. Yoganathan. A condition under which classical simulability implies efficient state learnability, 2019. arXiv:1907.08163. 5, 61

[ZLW+17] J. Zhang, Y. Liu, R.-B. Wu, K. Jacobs, and F. Nori. Quantum feedback: theory, experiments, and applications. *Physics Reports*, 679:1–60, 2017. 28

[ZRS+18] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 9793–9803, 2018. 47

[ZWC+18] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin. Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices, 2018. arXiv:1812.01041. 46, 48

[ZYH+17] C. Zhang, B. Yadin, Z.-B. Hou, H. Cao, B.-H. Liu, Y.-F. Huang, R. Maity, V. Vedral, C.-F. Li, G.-C. Guo, et al. Detecting metrologically useful asymmetry and entanglement by a few local measurements. *Physical Review A*, 96(4):042327, 2017. 6