Sublinear Quantum Algorithms for Training Linear and Kernel-based Classifiers

Tongyang Li [1] Shouvanik Chakrabarti [1] Xiaodi Wu [1]

Abstract

We investigate quantum algorithms for classification, a fundamental problem in machine learning, with provable guarantees. Given n d-dimensional data points, the state-of-the-art (and optimal) classical algorithm for training classifiers with constant margin (Clarkson et al., 2012) runs in $\tilde{O}(n+d)$[1]. We design sublinear quantum algorithms for the same task running in $\tilde{O}(\sqrt{n}+\sqrt{d})$, a quadratic improvement in both n and d. Moreover, our algorithms use the standard quantization of the classical input and generate the same classical output, suggesting minimal overheads when used as subroutines for end-to-end applications. We also demonstrate a tight lower bound (up to poly-log factors) and discuss the possibility of implementation on near-term quantum machines.

1. Introduction

Motivations. Classification is a fundamental problem of supervised learning, which takes a training set of data points of known classes as inputs and aims to training a model for predicting the classes of future data points. It is also ubiquitous due to its broad connections and applications to computer vision, natural language processing, statistics, etc.

A fundamental case of classification is *linear classification*, where we are given n data points $X_1, \ldots, X_n$ in $\mathbb{R}^d$ and a label vector $y \in \{1,-1\}^n$. The goal is to find a separating hyperplane, i.e., a unit vector w in $\mathbb{R}^d$, such that

$$y_i \cdot X_i^\top w \geq 0 \quad \forall\, i \in [n]. \tag{1.1}$$

By taking $X_i \leftarrow (-1)^{y_i} X_i$, it reduces to a *maximin* problem $\max_w \min_i X_i^\top w \geq 0$. The approximation version of linear classification is to find a unit vector $\bar{w} \in \mathbb{R}^d$ so that

$$X_i^\top \bar{w} \geq \max_{w\in\mathbb{R}_d} \min_{i'\in[n]} X_{i'}^\top w - \epsilon \quad \forall\, i \in [n], \tag{1.2}$$

i.e., $\bar{w}$ approximately solves the maximin problem. More generally, we can regard a (nonlinear) classifier as a *kernel-based* classifier by replacing X_i by $\Psi(X_i)$ (Ψ being a kernel function). We will focus on algorithms finding approximate classifiers (in the sense of (1.2)) with *provable guarantees*.

The Perceptron Algorithm for linear classification is one of the oldest algorithms studied in machine learning (Novikoff, 1963; Minsky & Papert, 1988), which runs in time $O(nd/\epsilon^2)$ for finding an $\bar{w} \in \mathbb{R}^d$ satisfying (1.2). The state-of-the-art classical result along this line (Clarkson et al., 2012) solves linear classification in time $\tilde{O}((n+d)/\epsilon^2)$. A careful reader might notice that the input to linear classification is n d-dimensional vectors with total size $O(nd)$. Hence, the result of Clarkson et al. (2012) is *sub-linear* in its input size. To make it possible, Clarkson et al. (2012) assumes the following entry-wise input model:

Input model: given any $i \in [n]$ and $j \in [d]$, the j-th entry of X_i can be recovered in $O(1)$ time.

The output of Clarkson et al. (2012) is an *efficient* classical representation of $\bar{w}$ in the sense that every entry of $\bar{w}$ can be recovered with $\tilde{O}(1)$ cost. It is no surprise that $\bar{w}$ per se gives such a representation. However, there could be more *succinct and efficient representations* of $\bar{w}$, which could be reasonable alternatives of $\bar{w}$ for sub-linear algorithms that run in time less the dimension of $\bar{w}$ (as we will see in the quantum case). The complexity of Clarkson et al. (2012) is also optimal (up to poly-logarithmic factors) in the above input/output model as shown by the same paper.

Recent developments in quantum computation, especially in the emerging topic of "quantum machine learning" (see the surveys Biamonte et al. (2017); Arunachalam & de Wolf (2017); Schuld et al. (2015)), suggest that quantum algorithms might offer significant speed-ups for optimization and machine learning problems. In particular, a quantum counterpart of the Perceptron algorithm has been proposed in Kapoor et al. (2016) with improved time complexity from $O(nd/\epsilon^2)$ to $\tilde{O}(\sqrt{n}d/\epsilon^2)$ (details in related works). Motivated both by the significance of classification and the

[1]Department of Computer Science, UMIACS, and Joint Center for Quantum Information and Computer Science, University of Maryland. Correspondence to: Tongyang Li <tongyang@cs.umd.edu>, Xiaodi Wu <xwu@cs.umd.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]$\tilde{O}(\cdot)$, $\tilde{\Omega}(\cdot)$, and $\tilde{\Theta}(\cdot)$ notations hide poly-logarithmic factors.

promise of quantum algorithms, we investigate the *optimal* quantum algorithm for classification. Specifically, we aim to design a quantum counterpart of Clarkson et al. (2012).

It is natural to require that quantum algorithms make use of the classical input/output model as much as possible to make the comparison fair. In particular, it is favorable to avoid the use of too powerful input data structure which might render any finding of quantum speedup inconclusive, especially in light of a recent development of quantum-inspired classical machine learning algorithms (e.g., Tang (2018a)). Our choice of input/output models for quantum algorithms is hence almost the same as the classical one, except we allow *coherent* queries to the entries of X_i:

Quantum input model: given any $i \in [n]$ and $j \in [d]$, the j-th entry of X_i can be recovered in $O(1)$ time *coherently*.

Coherent queries allow the quantum algorithm to query many locations in super-position, which is a *standard* assumption that accounts for many quantum speed-ups (e.g., Grover's algorithm (Grover, 1996)). A more precise definition is given in Section 2.

On the other side, our output is exactly the same as classical algorithms, which guarantees no overhead when using our quantum algorithms as subroutines for any applications.

Contributions. Inspired by Clarkson et al. (2012), our main contribution is a *tight* characterization (up to poly-log factors) of quantum algorithms for various classification problems in the aforementioned input/output model.

Theorem 1.1 (Main theorem). *Given $\epsilon = \Theta(1)$, we have quantum algorithms that return an efficient representation of $\bar{w} \in \mathbb{B}_d$ for the following problems*[2]*, respectively, with complexity $\tilde{O}(\sqrt{n} + \sqrt{d})$ and high success probability:*

- *Linear classification (Theorem 3.2):*
$$\min_{i\in[n]} X_i^\top \bar{w} \geq \max_{w\in\mathbb{B}_d} \min_{i\in[n]} X_i^\top w - \epsilon. \tag{1.3}$$
- *Kernel-based classification:*
$$\min_{i\in[n]} \langle \Psi(X_i), \bar{w}\rangle \geq \max_{w\in\mathbb{B}_d} \min_{i\in[n]} \langle \Psi(X_i), w\rangle - \epsilon, \tag{1.4}$$
where $k(a, b) := \langle \Psi(a), \Psi(b)\rangle$ can be the polynomial kernel $k_q(a, b) = (a^\top b)^q$ (Corollary 4.1) or the Gaussian kernel $k_{Gauss}(a, b) = \exp\bigl(-\|a - b\|^2\bigr)$ (Corollary 4.2).
- *Minimum enclosing ball (Theorem 4.2):*
$$\max_{i\in[n]} \|\bar{w} - X_i\|^2 \leq \min_{w\in\mathbb{R}^d} \max_{i\in[n]} \|w - X_i\|^2 + \epsilon. \tag{1.5}$$
- *ℓ_2-margin SVM (Corollary 4.3):*
$$\min_{i\in[n]} (X_i^\top \bar{w})^2 \geq \max_{w\in\mathbb{R}^d} \min_{i\in[n]} 2X_i^\top w - \|w\|^2 - \epsilon. \tag{1.6}$$

[2]Here $\mathbb{B}_d$ is the unit ball in $\mathbb{R}^d$.

On the other hand, we show that it requires $\Omega(\sqrt{n} + \sqrt{d})$ queries to the quantum input model to prepare such $\bar{w}$ for these classification problems (Theorem 5.1, Theorem 5.2).

Our matching upper and lower bounds $\sqrt{n} + \sqrt{d}$ give a *quadratic* improvement in both n and d comparing to the classical state-of-the-art results in Clarkson et al. (2012).

Technically, our result is also inspired by the recent development of quantum semidefinite program (SDP) solvers (e.g., Brandão et al. (2017)) which provide quantum speed-ups for approximating zero-sum games for the purpose of solving SDPs. Note that such a connection was leveraged classically in another direction in a follow-up work of Clarkson et al. (2012) for solving SDPs (Garber & Hazan, 2011). However, our algorithm is even simpler because we only use simple quantum state preparation instead of complicated quantum operations in quantum SDP solvers; this is because quantum state preparation is a direct counterpart of the ℓ_2 sampling used in Clarkson et al. (2012) (see Section 3.1 for details). In a nutshell, our result is a demonstration of quantum speed-ups for sampling-based classical algorithms.

Moreover, our algorithms are hybrid classical-quantum algorithms where the quantum part is isolated pieces of state preparation connected by classical processing. In addition, special instances of these state preparation might be physically realizable as suggested by some work-in-progress in Google (Brandão, 2018). All of the above suggest the possibility of implementing these algorithms on near-term quantum machines (Preskill, 2018).

In general, we deem our result as a proposal of one end-to-end quantum application in machine learning, with both provable guarantees and the perspective of implementation (at least in prototype) on near-term quantum machines.

Related works. We make the following comparisons with existing literatures in quantum machine learning.

- The most relevant result is the quantum perceptron models in Kapoor et al. (2016). The classical perceptron method (Novikoff, 1963; Minsky & Papert, 1988) is a pivotal linear classification algorithm. In each iteration, it checks whether (1.1) holds; if not, then it searches for a violated constraint i_0 (i.e., $y_{i_0} X_{i_0}^\top \bar{w} < 0$) and update $\bar{w} \leftarrow \bar{w} + X_{i_0}$ (up to normalization). This classical perceptron method has complexity $\tilde{O}(nd/\epsilon^2)$; the quantum counterpart in Kapoor et al. (2016) improved the complexity to $\tilde{O}(\sqrt{n}d/\epsilon^2)$ by applying Grover search (Grover, 1996) to find a violated constraint. In contrast, we quantize the sublinear algorithm for linear classification in Clarkson et al. (2012) with techniques inspired by quantum SDP solvers (Brandão et al., 2017). As a result, we establish a better quantum complexity $\tilde{O}(\sqrt{n} + \sqrt{d})$.

  In addition, Kapoor et al. (2016) relies on an unusual input model where a data point in $\mathbb{R}^d$ is represented by concatenating the the binary representations of the d floating point numbers; if we were only given standard inputs with entry-wise queries to the coordinates of data points, we need a cost of $\Omega(d)$ to transform the data into their input form, giving the total complexity $\tilde{O}(\sqrt{n}d)$.

The same group of authors also gave a quantum algorithm for nearest-neighbor classification with complexity $\tilde{O}(\sqrt{n})$ (Wiebe et al., 2015). This complexity also depends on the sparsity of the input data; in the worst case where every data point has $\Theta(d)$ nonzero entries, the complexity becomes $\tilde{O}(\sqrt{n}d^2)$.

- There have been rich developments on quantum algorithms for linear algebraic problems. One prominent example is the quantum algorithm for solving linear systems (Harrow et al., 2009; Childs et al., 2017); in particular, they run in time $\mathrm{poly}(\log d)$ for any sparse d-dimensional linear systems. These linear system solvers are subsequently applied to machine learning applications such as cluster assignment (Lloyd et al., 2013), support vector machine (SVM) (Rebentrost et al., 2014), etc.

  However, these quantum algorithms have two drawbacks. First, they require the input matrix to be *sparse* with efficient access to nonzero elements, i.e., every row/column of the matrix has at most $\mathrm{poly}(\log d)$ nonzero elements and their indexes can be queried in $\mathrm{poly}(\log d)$ time. Second, the outputs of these algorithms are quantum states instead of classical vectors, and it takes $\Omega(d)$ copies of the quantum state to reveal one entry of the output in the worst case. More caveats are listed in Aaronson (2015).

  In contrast, our quantum algorithms do not have the sparsity constraint and work for arbitrary input data, and the outputs of our quantum algorithms are succinct but efficient classical representations of vectors in $\mathbb{R}^d$, which can be directly used for classical applications.

- There are two lines of quantum machine learning algorithms with different input requirements. One of them is based on quantum principal component analysis (Lloyd et al., 2014) and requires purely quantum inputs.

  Another line is the recent development of quantum-inspired classical poly-logarithmic time algorithms for various machine learning tasks such as recommendation systems (Tang, 2018a), principal component analysis (Tang, 2018b), solving linear systems (Chia et al., 2018; Gilyén et al., 2018), SDPs (Chia et al., 2019), and so on. These algorithms follow a Monte-Carlo approach for low-rank matrix approximation (Frieze et al., 2004) and assume the ability to take samples according to the spectral norms of all rows. In other words, these results enforce additional requirements on their input: the input matrix should not only be low-rank but also be preprocessed as the sampling data structure.

- There are also a few heuristic quantum machine learning approaches for classification (Havlicek et al., 2018; Farhi & Neven, 2018; Kerenidis & Luongo, 2018) without theoretical guarantees. We, however, look forward to further experiments based on their proposals.

2. Preliminaries

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

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

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

$$|u\rangle \otimes |v\rangle = (u_0 v_0, u_0 v_1, \ldots, u_{d_1-1} v_{d_2-1})^\top. \tag{2.1}$$

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

Operations in quantum computation are *unitary transformations* and can be stated in the circuit model[3] where a k*-qubit gate* is a unitary matrix in $\mathbb{C}^{2^k}$. It is known that two-qubit gates are *universal*, i.e., every n-qubit gate can be written as composition of a sequence of two-qubit gates. Thus, one usually refers to the *number of two-qubit gates* as the *gate complexity* of quantum algorithms.

Quantum oracle. Quantum access to the input data (referred as quantum oracles) needs to be reversible and allows access to different parts of the input data in *superposition* (the essence of quantum speed-ups). Specifically, to access elements in an $n \times d$ matrix X, we exploit an oracle O_X (a unitary on $\mathbb{C}^n \otimes \mathbb{C}^d \otimes \mathbb{C}^{d_{\text{acc}}}$) such that

$$O_X(|i\rangle \otimes |j\rangle \otimes |z\rangle) = |i\rangle \otimes |j\rangle \otimes |z \oplus X_{ij}\rangle \tag{2.2}$$

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

for any $i \in [n]$, $j \in [d]$ and $z \in \mathbb{C}^{d_{\text{acc}}}$ such that X_{ij} can be represented in $\mathbb{C}^{d_{\text{acc}}}$. Intuitively, O_X reads the entry X_{ij} and stores it in the third register. However, to make O_X reversible (and unitary), O_X applies the XOR operation ($\oplus$) on the third register. Note that O_X is a natural unitary generalization of classical random access to X, or in cases when any entry of X can be efficiently read. However, it is potentially stronger when queries become linear combinations of basis vectors, e.g., $\sum_k \alpha_k|i_k\rangle \otimes |j_k\rangle$. This is technically how to make superposition of different queries in quantum.

We summarize the quantum notations as follows.

	Classical	Quantum
Ket and bra	$\vec{e}_i$ and $\vec{e}_i^{\top}$	$\lvert i\rangle$ and $\langle i\rvert$
Basis	$\{\vec{e}_0,\ldots,\vec{e}_{d-1}\}$	$\{\lvert 0\rangle,\ldots,\lvert d-1\rangle\}$
State	$\vec{v}=(v_0,\ldots,v_{d-1})^{\top}$	$\lvert v\rangle=\sum_{i=0}^{d-1} v_i\lvert i\rangle$
Tensor	$\vec{u}\otimes\vec{v}$	$\lvert u\rangle\otimes\lvert v\rangle$ or $\lvert u\rangle\lvert v\rangle$
Oracle	$w=(X_{ij})_{i,j=1}^{n}$	$O_X\lvert i\rangle\lvert j\rangle\lvert z\rangle=\lvert i\rangle\lvert j\rangle\lvert z\oplus X_{ij}\rangle$

Table 1. Summary of quantum notations used in this paper.

Quantum complexity measure. We assume that a single query to the oracle O_X has a unit cost. *Quantum query complexity* is defined as the total counts of oracle queries, and *quantum gate complexity* is defined as the total counts of oracle queries and two-qubit gates.

Notations. In this paper, we denote $\mathbf{1}_n$ to be the n-dimensional all-one vector and $\mathbb{B}_d := \left\{a \in \mathbb{R}^d \mid \sum_{i\in[d]} |X_i|^2 \leq 1\right\}$ (the unit ball). Throughout the paper, we denote $X \in \mathbb{R}^{n\times d}$ to be the matrix whose ith row is $X_i^{\top}$ for all $i \in [n]$. Without loss of generality, we assume $X_1,\ldots,X_n \in \mathbb{B}_d$, i.e., all the n data points are normalized to have ℓ_2-norm at most 1.

3. Linear Classification

3.1. Techniques

At a high level, our quantum algorithm leverages ideas from both classical and quantum algorithm design. We use a primal-dual approach under the multiplicative weight framework (Freund & Schapire, 1999), in particular its improved version in Clarkson et al. (2012) by sampling the update of weight vectors. An important observation of ours is that such classical algorithms can be accelerated significantly in quantum computation, which relies on a seminal technique in quantum algorithm design: amplitude amplification and estimation (Grover, 1996; Brassard et al., 2002).

Multiplicative weight under a primal-dual approach. Note that linear classification is essentially a minimax problem (zero-sum game); by strong duality, we have

$$\sigma = \max_{w\in\mathbb{R}_d}\min_{p\in\Delta_n} p^{\top}Xw = \min_{p\in\Delta_n}\max_{w\in\mathbb{R}_d} p^{\top}Xw. \quad (3.1)$$

To find its equilibrium point, we adopt an online primal-dual approach with T rounds; at round $t \in [T]$, the primal computes $p_t \in \Delta_n$ and the dual computes $w_t \in \mathbb{R}_d$, both based on p_τ and w_τ for all $\tau \in [t-1]$. After T rounds, the average solution $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ approximately solves the zero-sum game with high probability, i.e., $\min_{p\in\Delta_n} p^{\top}X\bar{w} \geq \sigma - \epsilon$.

For the primal problem, we pick p_t by the *multiplicative weight* (MW) method. Given a sequence of vectors $r_1,\ldots,r_T \in \mathbb{R}^n$, MW sets $w_1 := \mathbf{1}_n$ and for all $t \in [T]$, $p_t := w_t/\|w_t\|_1$ and $w_{t+1}(i) := w_t(i)f_{\text{w}}(-\eta r_t(i))$ for all $i \in [n]$, where f_{w} is a weight function and η is the parameter representing the step size. MW promises an upper bound on $\sum_{t=1}^{T} p_t^{\top} r_t$, whose precise form depends on the choice of the weight function f_{w}. The most common update is the *exponential weight update*: $f_{\text{w}}(x) = e^{-x}$ (Freund & Schapire, 1999), but in this paper we use a quadratic weight update suggested by Clarkson et al. (2012), where $w_{t+1}(i) := w_t(i)(1 - \eta r_t(i) + \eta^2 r_t(i)^2)$. In our primal problem, we set $r_t = Xw_t$ for all $t \in [T]$ to find p_t.

For the dual problem, we pick w_t by the *online gradient descent* method (Zinkevich, 2003). Given a set of vectors $q_1,\ldots,q_T \in \mathbb{R}^d$ such that $\|q_i\|_2 \leq 1$. Let $w_0 := \mathbf{0}_d$, and $y_{t+1} := w_t + \frac{1}{\sqrt{T}}q_t$, $w_{t+1} := \frac{y_{t+1}}{\max\{1,\|y_{t+1}\|\}}$. Then

$$\max_{w\in\mathbb{B}_d}\sum_{t=1}^{T} q_t^{\top}w - \sum_{t=1}^{T} q_t^{\top}w_t \leq 2\sqrt{T}. \quad (3.2)$$

This can be regarded as a *regret* bound, i.e., $\sum_{t=1}^{T} q_t^{\top}w_t$ has at most a regret of $2\sqrt{T}$ compared to the best possible choice of w. In our dual problem, we set q_t as a sample of rows of X following the distribution p_t.

This primal-dual approach gives a correct algorithm with only $T = \tilde{O}(1/\epsilon^2)$ iterations. However, the primal step runs in $\Theta(nd)$ time to compute Xw_t. To obtain an algorithm that is *sublinear* in the size of X, a key observation by Clarkson et al. (2012) is to replace the precise computation of Xw_t by an unbiased random variable. This is achieved via ℓ_2 sampling of w: we pick $j_t \in [d]$ by $j_t = j$ with probability $w_t(j)^2/\|w_t\|^2$, and for all $i \in [n]$ we take $\tilde{v}_t(i) = X_i(j_t)\|w_t\|^2/w_t(j_t)$. The expectation of the random variable $\tilde{v}_t(i)$ satisfies

$$\mathbb{E}[\tilde{v}_t(i)] = \sum_{j=1}^{d}\frac{w_t(j)^2}{\|w_t\|^2}\frac{X_i(j)\|w_t\|^2}{w_t(j)} = X_i w_t. \quad (3.3)$$

In a nutshell, the update of weight vectors in each iteration need not to be precisely computed because an ℓ_2 sample from w suffices to promise the provable guarantee of the framework. This trick improves the running time of MW to $O(n)$ and online gradient descent to $O(d)$; since there are $\tilde{O}(1/\epsilon^2)$ iterations, the total complexity is $\tilde{O}(\frac{n+d}{\epsilon^2})$ as claimed in Clarkson et al. (2012).

Amplitude amplification and estimation. Consider a search problem where we are given a function $f_\omega\colon [n] \to \{-1, 1\}$ such that $f_\omega(i) = 1$ iff $i \neq \omega$. To search for ω, classically we need $\Omega(n)$ queries to f_ω as checking all n positions is the only method.

Quantumly, given a unitary U_ω such that $U_\omega|i\rangle = |i\rangle$ for all $i \neq \omega$ and $U_\omega|\omega\rangle = -|\omega\rangle$, Grover's algorithm (Grover, 1996) finds ω with complexity $\tilde{O}(\sqrt{n})$. Denote $|s\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$ (the uniform superposition), $|s'\rangle = \frac{1}{\sqrt{n-1}}\sum_{i\in[n]/\{\omega\}}|i\rangle$, and $U_s = 2|s\rangle\langle s| - I$, the unitary U_ω reflects a state with respect to $|s'\rangle$ and the unitary U_s reflects a state with respect to $|s\rangle$. If we start with $|s\rangle$ and denote $\theta = 2\arcsin(1/\sqrt{n})$ (the angle between $U_\omega|s\rangle$ and $|s\rangle$), then the angle between $U_\omega|s\rangle$ and $U_sU_\omega|s\rangle$ is *amplified* to 2θ, and in general the angle between $U_\omega|s\rangle$ and $(U_sU_\omega)^k|s\rangle$ is $2k\theta$. To find ω, it suffices to take $k = \Theta(\sqrt{n})$ in this quantum algorithm. See Figure 1 for an illustration.

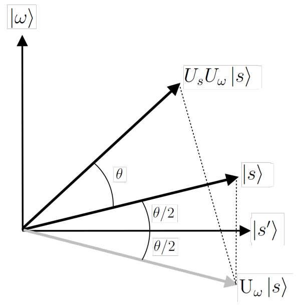

Figure 1. Geometric interpretation of Grover's algorithm. This figure is copied from Wikipedia.

This trick of alternatively applying two unitaries is called *amplitude amplification*; in general, this provides a quadratic speedup for search. For the quantitative version of estimating θ (not only finding ω), quadratic quantum speedup also holds via an improved version of amplitude amplification called *amplitude estimation* (Brassard et al., 2002).

Our **main technical contribution** is the implementations of amplitude amplification and estimation in the primal-dual approach for solving minimax problems. On the one hand, we achieve quadratic quantum speedup for multiplicative weight update, i.e., we improve the complexity from $\tilde{O}(n)$ to $\tilde{O}(\sqrt{n})$. This is because the ℓ_2 sampling of w is identical to measuring the quantum state $|w\rangle$ in the computational basis, which is prepared by amplitude amplification.

On the other hand, we also achieve quadratic quantum speedup for online gradient descent (improving $\tilde{O}(d)$ to $\tilde{O}(\sqrt{d})$). This is because the main cost of online gradient descent comes from estimating the norms $\|y_t\|$, which can be regarded as an amplitude estimation problem.

Comparison between classical and quantum results. Although our quantum algorithms enjoy quadratic speedups in n and d, their executions incur a larger dependence in ϵ: we have worst case $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$ compared to the classical $\tilde{O}\big(\frac{n}{\epsilon^2} + \frac{d}{\epsilon^2}\big)$ in Clarkson et al. (2012). The main reason of having a larger ϵ-dependence in quantum is because we cannot prepare the weight states in MW via those in previous iterations (i.e., the quantum state $|w_t\rangle$ cannot be prepared by $|w_{t-1}\rangle$), and we have to start over every time; this is an intrinsic difficulty due to quantum state preparation.

Therefore, there is a trade-off between Clarkson et al. (2012) and our results for arbitrary ϵ: we provide faster training of the classifiers if we allow a constant error, while the classical algorithms in Clarkson et al. (2012) might work better if we require high-accuracy classifiers.

3.2. Quantum speedup for multiplicative weights

First, we give a quantum algorithm for linear classification with complexity $\tilde{O}(\sqrt{n})$:

Theorem 3.1. *With success probability at least $2/3$, Algorithm 1 returns a succinct classical representation of a vector $\bar{w} \in \mathbb{R}^d$ such that*

$$X_i\bar{w} \geq \max_{w\in\mathbb{B}_d}\min_{i'\in[n]} X_{i'}w - \epsilon \quad \forall\, i \in [n], \tag{3.4}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon^2}\big)$ quantum gates.

Note that Algorithm 1 is inspired by the classical algorithm (Clarkson et al., 2012) by using online gradient descent in Line 5 and ℓ_2 sampling in Line 6 and Line 7. We defer the proof of correctness to the supplementary material; here, we focus on its gate complexity to see our quantum speedup.

Gate complexity of Algorithm 1. To run Line 3 and Line 5, we need d time and space to compute and store w_t and y_{t+1}; for all $t \in [T]$, this takes total complexity $O(dT) = \tilde{O}(\frac{d}{\epsilon^2})$. It takes another $O(dT) = \tilde{O}(\frac{d}{\epsilon^2})$ cost to compute j_t for all $t \in [T]$ in Line 6.

The quantum part of Algorithm 1 mainly happens at Line 7 and Line 8, where we prepare the quantum state $|p_{t+1}\rangle$ instead of computing the coefficients $u_{t+1}(i)$ one by one for all $i \in [n]$. To be more specific, we construct an oracle O_t such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$ for all $i \in [n]$. This is

[4]By defining w_t here, we do not write down the whole vector but we construct any query to its entries in $O(1)$ time. For example, the i^{th} coordinate of w_t is $w_t(i) = \frac{y_t(i)}{\max\{1,\|y_t\|\}}$, constructed by one query to $y_t(i)$. The y_{t+1} in Line 5 is defined in the same sense.

Algorithm 1: Quantum linear classification algorithm.

Input: $\epsilon > 0$, a quantum oracle O_X for $X \in \mathbb{R}^{n\times d}$.
Output: $\bar{w}$ that satisfies (3.4).

```
1 Let $T = 23^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$,
    $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
2 for $t = 1$ to $T$ do
3     Define[4] $w_t := \frac{y_t}{\max\{1,\|y_t\|\}}$;
4     Measure the state $|p_t\rangle$ in the computational basis
        and denote the output as $i_t \in [n]$;
5     Define $y_{t+1} := y_t + \frac{1}{\sqrt{2T}}X_{i_t}$;
6     Choose $j_t \in [d]$ by $j_t = j$ with probability $\frac{w_t(j)^2}{\|w_t\|^2}$;
7     For all $i \in [n]$, denote $\tilde{v}_t(i) = X_i(j_t)\frac{\|w_t\|^2}{w_t(j_t)}$,
        $v_t(i) = \min\{1/\eta, \max\{-1/\eta, \tilde{v}_t(i)\}\}$, and
        $u_{t+1}(i) = u_t(i)(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$.
        Implement a quantum oracle $O_t$ such that for all
        $i \in [n]$, $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$;
8     Prepare $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]}u_{t+1}(i)|i\rangle$;
9 Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T}w_t$;
```

achieved iteratively, i.e., at iteration s we map $|i\rangle|u_s(i)\rangle$ to $|i\rangle|u_{s+1}(i)\rangle$. In total, one query to O_t is implemented by $2t$ queries to O_X and $\tilde{O}(t)$ additional arithmetic computations.

Finally, we prepare the state $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\cdot\sum_{i\in[n]}u_{t+1}(i)|i\rangle$ in Line 8 using $O(\sqrt{n})$ calls to O_t, which are equivalent to $O(\sqrt{n}t)$ calls to O_X by Line 7 and $\tilde{O}(\sqrt{n}t)$ additional arithmetic computations. Therefore, the total complexity of Line 8 for all $t \in [T]$ is

$$\sum_{t=1}^{T}\tilde{O}(\sqrt{n}t) = \tilde{O}(\sqrt{n}T^2) = \tilde{O}\Big(\frac{\sqrt{n}}{\epsilon^4}\Big). \tag{3.5}$$

In all, the total complexity of Algorithm 1 is $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{d}{\epsilon^2}\big)$, establishing our statement.

Finally, the output $\bar{w}$ has a succinct classical representation with space complexity $O(\log n/\epsilon^2)$. To achieve this, we save $2T = O(\log n/\epsilon^2)$ values in Algorithm 1: $i_1, \ldots, i_T$ and $\|y_1\|, \ldots, \|y_T\|$; it then only takes $O(\log n/\epsilon^2)$ cost to recover any coordinate of $\bar{w}$ by Line 3 and Line 5. $\square$

3.3. Quantum speedup for online gradient descent

We further improve the dependence in d to $\tilde{O}(\sqrt{d})$. To achieve this, we cannot update w_t and y_t in Line 3 and Line 5 by each coordinate because storing w_t or y_t would already take cost at least d. We solve this issue by not updating w_t and y_t explicitly and instead only computing $\|y_t\|$ for all $i \in [T]$. This norm estimation is achieved by the following lemma:

Lemma 3.1. *Assume that $F\colon [d] \to [0,1]$ with a quantum oracle $O_F|i\rangle|0\rangle = |i\rangle|F(i)\rangle$ for all $i \in [d]$. Denote $m = \frac{1}{d}\sum_{i=1}^{d}F(i)$. Then for any $\delta > 0$, there is a quantum algorithm that uses $O(\sqrt{d}/\delta)$ queries to O_F and returns an $\tilde{m}$ such that $|\tilde{m} - m| \le \delta m$ with probability at least 2/3.*

Instead of updating y_t explicitly in Line 5 of Algorithm 1, we save the i_t for all $t \in [T]$ in Line 4, which only takes $\tilde{O}(1/\epsilon^2)$ cost in total but we can directly generate y_t given $i_1, \ldots, i_t$. Furthermore, notice that the probabilities in the ℓ_2 sampling in Line 6 do not change because $w_t(j)^2/\|w_t\|^2 = y_t(j)^2/\|y_t\|^2$; it suffices to replace $\tilde{v}_t(i) = X_i(j_t)\|w_t\|^2/w_t(j_t)$ by $\tilde{v}_t(i) = X_i(j_t)\|y_t\|^2/(y_t(j_t)\max\{1, \|y_t\|\})$ in Line 7. These observations imply the following result:

Theorem 3.2. *With success probability at least $2/3$, there is a quantum algorithm that returns a succinct classical representation of a vector $\bar{w} \in \mathbb{R}^d$ such that*

$$X_i\bar{w} \ge \max_{w\in\mathbb{B}_d}\min_{i'\in[n]} X_{i'}w - \epsilon \quad \forall\, i \in [n], \tag{3.6}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8}\big)$ quantum gates.

Remark 3.1. *In real applications, typically the number of data points n is larger than the dimension d, so Theorem 3.1 might perform better than Theorem 3.2 in practice. Nevertheless, the $\tilde{O}(\sqrt{d})$ complexity in Theorem 3.2 matches our quantum lower bound (see Theorem 5.1).*

4. Applications

As introduced in Section 3.1, the ℓ_2 sampling of w picks $j_t \in [d]$ by $j_t = j$ with probability $w(j)^2/\|w\|^2$, and the expectation of the random variable $X_i(j_t)\|w\|^2/w(j_t)$ is X_iw. Here, if we consider some alternate random variables, we could give unbiased estimators of nonlinear functions of X. We first look at the general case of applying kernel functions (Schölkopf & Smola, 2002) in Section 4.1. We then look at the special case of quadratic problems in Section 4.2 as they enjoy simple forms that can be applied to finding minimum enclosing balls (Saha et al., 2011) and ℓ_2-margin support vector machines (Suykens & Vandewalle, 1999).

4.1. Kernel methods

Having quantum algorithms for solving linear classification at hand, it is natural to consider linear classification under kernels. Let $\Psi\colon \mathbb{R}^d \mapsto \mathcal{H}$ be a mapping into a reproducing kernel Hilbert space (RKHS), and the problem is to find the classifier $h \in \mathcal{H}$ that solves the maximin problem

$$\sigma = \max_{h\in\mathcal{H}}\min_{i\in[n]}\langle h, \Psi(X_i)\rangle, \tag{4.1}$$

where the kernel is defined as $k(a,b) := \langle\Psi(a), \Psi(b)\rangle$ for all $a, b \in \mathbb{R}^d$.

Classically, Clarkson et al. (2012) gave the following result for classification under efficiently-computable kernels, following the linear classification algorithm therein:

Theorem 4.1 (Lemma 5.3 of Clarkson et al. (2012))**.** *Denote T_k as the time cost for computing $k(X_i, X_j)$ for some $i, j \in [n]$, and denote L_k as the time cost for computing a random variable $\tilde{k}(X_i, X_j)$ for some $i, j \in [n]$ such that $\mathbb{E}[\tilde{k}(X_i, X_j)] = k(X_i, X_j)$ and $\mathrm{Var}[k(X_i, X_j)] \leq 1$. Then there is a classical algorithm that runs in time*

$$\tilde{O}\Big(\frac{L_k n + d}{\epsilon^2} + \min\Big\{\frac{T_k}{\epsilon^4}, \frac{L_k}{\epsilon^6}\Big\}\Big) \tag{4.2}$$

and returns a vector $\bar{h} \in \mathcal{H}$ such that with high success probability $\langle \bar{h}, \Psi(X_i)\rangle \geq \sigma - \epsilon$ for all $i \in [n]$.

Quantumly, we give the following algorithm for classification under kernels based on Algorithm 1:

Algorithm 2: Quantum kernel-based classification.

```
Input: $\epsilon > 0$, a quantum oracle $O_X$ for $X \in \mathbb{R}^{n\times d}$.
Output: $\bar{w}$ that satisfies (3.4).
Let $T = 27^2\epsilon^{-2}\log n$, $y_1 = \mathbf{0}_d$, $\eta = \sqrt{\frac{\log n}{T}}$, $u_1 = \mathbf{1}_n$,
  $|p_1\rangle = \frac{1}{\sqrt{n}}\sum_{i\in[n]}|i\rangle$;
for $t = 1$ to $T$ do
    Measure the state $|p_t\rangle$ in the computational basis and
      denote the output as $i_t \in [n]$;
    Define $y_{t+1} := y_t + \frac{1}{\sqrt{2T}}\Psi(X_{i_t})$;
    Apply Lemma 3.1 for $2\lceil\log T\rceil$ times to estimate $\|y_t\|^2$
      with precision $\delta = \eta^2$, and take the median of all the
      $2\lceil\log T\rceil$ outputs, denoted $\widetilde{\|y_t\|}^2$;
    Choose $j_t \in [d]$ by $j_t = j$ with probability
      $y_t(j)^2/\|y_t\|^2$, which is achieved by applying
      Algorithm 3 to prepare the quantum state $|y_t\rangle$ and
      measure in the computational basis;
    For all $i \in [n]$, denote $\tilde{v}_t(i) = \frac{\Psi(X_i)(j_t)\widetilde{\|y_t\|}^2}{y_t(j_t)\max\{1,\widetilde{\|y_t\|}\}}$,
      $v_t(i) = \mathrm{clip}(\tilde{v}_t(i), 1/\eta)$, and $u_{t+1}(i) = u_t(i)\cdot$
      $(1 - \eta v_t(i) + \eta^2 v_t(i)^2)$. Apply Algorithm 4 to
      prepare an oracle $O_t$ such that $O_t|i\rangle|0\rangle = |i\rangle|u_{t+1}(i)\rangle$
      for all $i \in [n]$, using $2t$ queries to $O_X$ and $\tilde{O}(t)$
      additional arithmetic computations;
    Prepare the state $|p_{t+1}\rangle = \frac{1}{\|u_{t+1}\|_2}\sum_{i\in[n]}u_{t+1}(i)|i\rangle$
      using Algorithm 3 and $O_t$;
Return $\bar{w} = \frac{1}{T}\sum_{t=1}^{T}\frac{y_t}{\max\{1,\widetilde{\|y_t\|}\}}$;
```

Theorem 3.2 and Theorem 4.1 imply that our quantum kernel-based classifier has time complexity

$$\tilde{O}\Big(\frac{L_k\sqrt{n}}{\epsilon^4} + \frac{\sqrt{d}}{\epsilon^8} + \min\Big\{\frac{T_k}{\epsilon^4}, \frac{L_k}{\epsilon^6}\Big\}\Big). \tag{4.3}$$

For polynomial kernels of degree q, i.e., $k_q(x, y) = (x^\top y)^q$, we have $L_{k_q} = q$ by taking the product of q independent ℓ_2 samples (this is an unbiased estimator of $(x^\top y)^q$ and the variance of each sample is at most 1). As a result of (4.3),

Corollary 4.1. *For the polynomial kernel of degree q, there is a quantum algorithm that solves the classification task within precision ϵ with gate complexity $\tilde{O}\big(\frac{q\sqrt{n}}{\epsilon^4} + \frac{q\sqrt{d}}{\epsilon^8}\big)$.*

Compared to the classical complexity $\tilde{O}\big(\frac{q(n+d)}{\epsilon^2} + \min\big\{\frac{d\log q}{\epsilon^4}, \frac{q}{\epsilon^6}\big\}\big)$ in Corollary 5.4 of Clarkson et al. (2012), our quantum algorithm gives quadratic speedups in n and d.

For Gaussian kernels, i.e., $k_{\text{Gauss}}(x, y) = \exp(-\|x - y\|^2)$, Corollary 5.5 of Clarkson et al. (2012) proved that $L_{k_{\text{Gauss}}} = 1/s^4$ if the Gaussian has standard deviation s. As a result,

Corollary 4.2. *For the polynomial kernel of degree q, there is a quantum algorithm that solves the classification task within precision ϵ with gate complexity $\tilde{O}\big(\frac{\sqrt{n}}{s^4\epsilon^4} + \frac{\sqrt{d}}{s^4\epsilon^8}\big)$.*

This still gives quadratic speedups in n and d compared to the classical complexity $\tilde{O}\big(\frac{n+d}{s^4\epsilon^2} + \min\big\{\frac{d}{\epsilon^4}, \frac{1}{s^4\epsilon^6}\big\}\big)$ in Corollary 5.5 of Clarkson et al. (2012).

4.2. Quadratic machine learning problems

We consider the maximin problem of a quadratic function:

$$\begin{aligned}&\max_{w\in\mathbb{R}^d}\min_{p\in\Delta_n} p^\top(b + 2Xw - \mathbf{1}_n\|w\|^2)\\ &= \max_{w\in\mathbb{R}^d}\min_{i\in[n]} b_i + 2X_i w - \|w\|^2,\end{aligned} \tag{4.4}$$

where $b \in \mathbb{R}^n$ and $X \in \mathbb{R}^{n\times d}$. Note that the function $b_i + 2X_i w - \|w\|^2$ in Eq. (4.4) is 2-strongly convex; as a result, the regret of the online gradient descent after T rounds can be improved to $O(\log T)$ by Sra et al. (2012) instead of $O(\sqrt{T})$ as in Eq. (3.2). In addition, ℓ_2 sampling of the w in Algorithm 1 still works: consider the random variable $w = b_i + \frac{2X_i(j)\|w\|^2}{w(j)} - \|w\|^2$ where $j = k$ with probability $\frac{w(k)^2}{\|w\|^2}$. Then the expectation of w is

$$\begin{aligned}\mathbb{E}[X] &= \sum_{j=1}^{d}\frac{w(j)^2}{\|w\|^2}\Big(b_i + \frac{2X_i(j)\|w\|^2}{w(j)} - \|w\|^2\Big)\\ &= b_i + 2X_i w - \|w\|^2,\end{aligned} \tag{4.5}$$

i.e., w is an unbiased estimator of the quadratic form in (4.4). As a result, given the quantum oracle O_X in (2.2), we could give sublinear quantum algorithms for such problems; these include two important problems: minimum enclosing balls (MEB) and ℓ_2-margin supper vector machines (SVM).

4.2.1. Minimum enclosing ball

In the minimum enclosing ball (MEB) problem we have $b_i = -\|X_i\|^2$ for all $i \in [n]$; Eq. (4.4) then becomes $\max_{w\in\mathbb{R}^d}\min_{i\in[n]} -\|X_i\|^2 + 2X_i w - \|w\|^2 =$

$-\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w-X_i\|^2$, which is the smallest radius of the balls that contain all the data points $X_1,\dots,X_n$.

Denote $\sigma_{\text{MEB}}=\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w-X_i\|^2$, the following theorem is a direct consequence of Theorem 3.1 (see also Theorem 3.1 in Clarkson et al. (2012)) and Theorem 3.2:

Theorem 4.2. *There is a quantum algorithm that returns a vector $\bar{w}\in\mathbb{R}^d$ such that with probability at least $2/3$,*

$$\max_{i\in[n]}\|\bar{w}-X_i\|^2\leq\sigma_{\text{MEB}}+\epsilon, \tag{4.6}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{d}{\epsilon}\big)$ quantum gates; the quantum gate complexity can also be improved to $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^7}\big)$.

4.2.2. ℓ_2-MARGIN SVM

To estimate the margin of a support vector machine (SVM) in ℓ_2-norm, we take $b_i=0$ for all $i\in[n]$; Eq. (4.4) then becomes solving $\sigma_{\text{SVM}}:=\max_{w\in\mathbb{R}^d}\min_{i\in[n]}2X_iw-\|w\|^2$.

Notice that $\sigma_{\text{SVM}}\geq 0$ because $2X_iw-\|w\|^2=0$ for all $i\in[n]$ when $w=0$. For the case $\sigma_{\text{SVM}}>0$ and taking $0<\epsilon<\sigma_{\text{SVM}}$, similar to Theorem 4.2 we have:

Corollary 4.3. *There is a quantum algorithm that returns a vector $\bar{w}\in\mathbb{R}^d$ such that with probability at least $2/3$,*

$$\min_{i\in[n]}2X_i\bar{w}-\|\bar{w}\|^2\geq\sigma_{\text{SVM}}-\epsilon>0, \tag{4.7}$$

using $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{d}{\epsilon}\big)$ quantum gates; the quantum gate complexity can also be improved to $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^7}\big)$.

Note that (4.7) implies that $X_i\bar{w}>0$ for all $i\in[n]$; furthermore, by the AM-GM inequality we have $\frac{(X_i\bar{w})^2}{\|\bar{w}\|^2}+\|\bar{w}\|^2\geq 2X_i\bar{w}$, and hence

$$\min_{i\in[n]}\Big(\frac{X_i\bar{w}}{\|\bar{w}\|}\Big)^2\geq\min_{i\in[n]}2X_i\bar{w}-\|\bar{w}\|^2\geq\sigma_{\text{SVM}}-\epsilon. \tag{4.8}$$

If we denote $\hat{w}=\bar{w}/\|\bar{w}\|$, then $X_i\hat{w}\geq\sqrt{\sigma_{\text{SVM}}-\epsilon}>0$ for all $i\in[n]$. Consequently, if the data X is from an SVM, we obtain a normalized direction $\hat{w}$ (in ℓ_2-norm) such that all data points have a margin of at least $\sqrt{\sigma_{\text{SVM}}-\epsilon}$. Classically, this task takes time $\tilde{O}(n+d)$ for constant σ_{SVM} by Clarkson et al. (2012), but our quantum algorithm only takes time $\tilde{O}(\sqrt{n}+\sqrt{d})$.

5. Quantum Lower Bounds

All quantum algorithms (upper bounds) above have matching lower bounds in n and d. In particular, assuming $\epsilon=\Theta(1)$ and given the oracle O_X in (2.2), we have:

Theorem 5.1. *To return an $\bar{w}\in\mathbb{B}_d$ satisfying*

$$X_j\bar{w}\geq\max_{w\in\mathbb{B}_d}\min_{i\in[n]}X_iw-\epsilon\quad\forall j\in[n] \tag{5.1}$$

with probability at least $2/3$, we need $\Omega(\sqrt{n}+\sqrt{d})$ quantum queries to O_X.

Theorem 5.2. *To return an $\bar{w}\in\mathbb{R}_d$ satisfying*

$$\max_{i\in[n]}\|\bar{w}-X_i\|^2\leq\min_{w\in\mathbb{R}^d}\max_{i\in[n]}\|w-X_i\|^2+\epsilon \tag{5.2}$$

with probability at least $2/3$, we need $\Omega(\sqrt{n}+\sqrt{d})$ quantum queries to O_X.

Because the kernel-based classifier in Section 4.1 contains the linear classification in Section 3 as a special case, Theorem 5.1 implies an $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bound on the kernel method. Similarly, Theorem 5.2 implies an $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bound on the ℓ_2-margin SVM. Therefore, we establish tight quantum bounds for all the classification problems in Section 3 and Section 4.

We prove both theorems by constructing reductions to the quantum search lower bound (Bennett et al., 1997); their full proofs are deferred to the supplementary material.

6. Conclusion

We give quantum algorithms for training linear and kernel-based classifiers with complexity $\tilde{O}(\sqrt{n}+\sqrt{d})$, where n and d are the number and dimension of data points, respectively; furthermore, our quantum algorithms are optimal as we prove matching $\Omega(\sqrt{n}+\sqrt{d})$ quantum lower bounds. Our quantum algorithms take standard entry-wise inputs and give classical outputs with succinct representations. Technically, our result is a demonstration of quantum speed-ups for sampling-based classical algorithms using the technique of amplitude amplification and estimation.

Our paper raises a couple of natural open questions for future work. For instance:

- Can we improve the dependence in ϵ? Recall our quantum algorithms have worst-case complexity $\tilde{O}\big(\frac{\sqrt{n}}{\epsilon^4}+\frac{\sqrt{d}}{\epsilon^8}\big)$ whereas the classical complexities in Clarkson et al. (2012) are $\tilde{O}\big(\frac{n}{\epsilon^2}+\frac{d}{\epsilon^2}\big)$; as a result, the quantum algorithms perform better only when we tolerate a significant error. It would be interesting to check if some clever tricks could be applied to circumventing some dependence in ϵ.
- Can we solve equilibrium point problems other than classification? Recall that our results in Theorem 1.1 are all formulated as maximin problems where the minimum is taken over $[n]$ and the maximum is taken over $\mathbb{B}_d$ or $\mathbb{R}_d$. It would be interesting to study other type of equilibrium point problems in game theory, learning theory, etc.
- What happens if we work with more sophisticated data structures such as QRAM or its augmented variants? Their preprocessing time will likely be at least linear. However, it might be still advantageous to do so, e.g., to reduce the amortized complexity when one needs to perform multiple classification tasks on the same data set.

Acknowledgements

We thank the anonymous reviewers who provide helpful suggestions on a preliminary version of this paper. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams program. TL also received support from IBM PhD Fellowship and QISE-NET Triplet Award (NSF grant DMR-1747426). XW also received support from the National Science Foundation (grants CCF-1755800 and CCF-1816695).

References

Aaronson, S. Read the fine print. *Nature Physics*, 11(4): 291, 2015.

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

Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. arXiv:quant-ph/9701001.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195, 2017. arXiv:1611.09347.

Brandão, F. Quantum speed-up for sdps and kernel learning. QuICS quantum machine learning workshop, 2018. Also available at http://qml2018.umiacs.io, 2018.

Brandão, F. G., Kalev, A., Li, T., Lin, C. Y.-Y., Svore, K. M., and Wu, X. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. *arXiv:1710.02581*, 2017.

Brassard, G., Høyer, P., Mosca, M., and Tapp, A. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. arXiv:quant-ph/0005055.

Chia, N.-H., Lin, H.-H., and Wang, C. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *arXiv:1811.04852*, 2018.

Chia, N.-H., Li, T., Lin, H.-H., and Wang, C. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches. *arXiv:1901.03254*, 2019.

Childs, A. M., Kothari, R., and Somma, R. D. Quantum linear systems algorithm with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017. arXiv:1511.02306.

Clarkson, K. L., Hazan, E., and Woodruff, D. P. Sublinear optimization for machine learning. *Journal of the ACM (JACM)*, 59(5):23, 2012. arXiv:1010.4408.

Dürr, C. and Høyer, P. A quantum algorithm for finding the minimum. 1996. arXiv:quant-ph/9607014.

Farhi, E. and Neven, H. Classification with quantum neural networks on near term processors. *arXiv:1802.06002*, 2018.

Freund, Y. and Schapire, R. E. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

Frieze, A., Kannan, R., and Vempala, S. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

Garber, D. and Hazan, E. Approximating semidefinite programs in sublinear time. In *Advances in Neural Information Processing Systems 24*, pp. 1080–1088. Curran Associates, Inc., 2011.

Gilyén, A., Lloyd, S., and Tang, E. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv:1811.04909*, 2018.

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

Grover, L. K. Synthesis of quantum superpositions by quantum computation. *Physical Review Letters*, 85(6):1334, 2000.

Harrow, A. W., Hassidim, A., and Lloyd, S. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009. arXiv:0811.3171.

Havlicek, V., Córcoles, A. D., Temme, K., Harrow, A. W., Chow, J. M., and Gambetta, J. M. Supervised learning with quantum enhanced feature spaces. *arXiv:1804.11326*, 2018.

Kapoor, A., Wiebe, N., and Svore, K. Quantum perceptron models. In *Proceedings of the 30th Conference on Neural Information Processing Systems*, pp. 3999–4007, 2016. arXiv:1602.04799.

Kerenidis, I. and Luongo, A. Quantum classification of the mnist dataset via slow feature analysis. arxiv:1805.08837, 2018.

Kerenidis, I. and Prakash, A. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, pp. 49:1–49:21, 2017. arXiv:1603.08675.

Lloyd, S., Mohseni, M., and Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv:1307.0411*, 2013.

Lloyd, S., Mohseni, M., and Rebentrost, P. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014. arXiv:1307.0401.

Minsky, M. and Papert, S. A. *Perceptrons: An introduction to computational geometry*. MIT Press, 1988.

Novikoff, A. B. On convergence proofs for perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pp. 615–622, 1963.

Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018. ISSN 2521-327X. arXiv:1801.00862.

Rebentrost, P., Mohseni, M., and Lloyd, S. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014. arXiv:1307.0471.

Saha, A., Vishwanathan, S., and Zhang, X. New approximation algorithms for minimum enclosing convex shapes. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1146–1160. Society for Industrial and Applied Mathematics, 2011. arXiv:0909.1062.

Schölkopf, B. and Smola, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

Schuld, M., Sinayskiy, I., and Petruccione, F. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

Sra, S., Nowozin, S., and Wright, S. J. *Optimization for machine learning*. MIT Press, 2012.

Suykens, J. A. and Vandewalle, J. Least squares support vector machine classifiers. *Neural Processing Letters*, 9 (3):293–300, 1999.

Tang, E. A quantum-inspired classical algorithm for recommendation systems. *arXiv:1807.04271*, 2018a.

Tang, E. Quantum-inspired classical algorithms for principal component analysis and supervised clustering. *arXiv:1811.00414*, 2018b.

Wiebe, N., Kapoor, A., and Svore, K. M. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Information & Computation*, 15(3-4):316–356, 2015. arXiv:1401.2142.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 928–936, 2003.