nature portfolio

https://doi.org/10.1038/s41567-021-01287-z

Supplementary information

A rigorous and robust quantum speed-up in supervised machine learning

In the format provided by the authors and unedited

Supplementary Information: A rigorous and robust quantum speed-up in supervised machine learning

Yunchao Liu,[1,2] Srinivasan Arunachalam,[2] and Kristan Temme[2,*]

[1] *Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720*

[2] *IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598*

(Dated: May 25, 2021)

CONTENTS

I. Supervised learning and the discrete log problem 1

II. A concept class reducible to discrete log 2

III. Support vector machine and the quantum kernel estimation algorithm 4
A. Support vector machines 4
B. Non-linear classification 7
C. Quantum kernel estimation 7

IV. Efficient learnability with quantum kernel estimation 9
A. Quantum feature map 9
B. Mapping to high dimensional Euclidean space 10
C. Generalization of the noisy classifier 13

V. Generalization bound for soft margin SVM 18

References 19

I. SUPERVISED LEARNING AND THE DISCRETE LOG PROBLEM

We work in the same setting as in standard computational learning theory [1, 2]. The *data space* $\mathcal{X} \subseteq \mathbb{R}^d$ is a fixed subset of d-dimensional Euclidean space. In this paper we will be concerned with *distribution-dependent* learning, i.e., we fix our *data distribution* to be the uniform distribution over $\mathcal{X}$. A *concept class* $\mathcal{C}$ is a set of functions that maps data vectors to binary labels, i.e., every $f \in \mathcal{C}$ is a function $f : \mathcal{X} \to \{-1, 1\}$.

A learning algorithm is given a set of *training samples* $S = \{(x_i, f(x_i))\}_{i=1}^m$, where each x_i is independently drawn from the uniform distribution over $\mathcal{X}$, and $f \in \mathcal{C}$ is an unknown concept in the concept class. The goal of the learning algorithm is to return a classifier f^* that runs in polynomial time. The *test accuracy* of the learned classifier is defined as the probability of agreeing with the unknown concept,

$$\mathrm{acc}_f(f^*) = \Pr_{x\sim\mathcal{X}}\left[f^*(x) = f(x)\right]. \tag{1}$$

* kptemme@ibm.com

Definition 1 (Efficient learning of $\mathcal{C}$)**.** *Let $\mathcal{X} \subseteq \mathbb{R}^d$. A concept class $\mathcal{C} \subseteq \{f : \mathcal{X} \to \{-1, 1\}\}$ is* efficiently learnable*, if there exists a learning algorithm $\mathcal{A}$ that satisfies the following: for every $f \in \mathcal{C}$, algorithm $\mathcal{A}$ takes as input $\mathsf{poly}(d)$ many training samples S and with probability at least $2/3$ (over the choice of random training samples and randomness of the algorithm), outputs a classifier in time $\mathsf{poly}(d)$ that achieves 99% test accuracy.*

The concept class we construct for showing our quantum advantage is based on the *discrete log problem* (DLP) which we define first:

> DLP: given a prime p, a primitive element g of $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, and $y \in \mathbb{Z}_p^*$, find $x \in \mathbb{Z}_p^*$ such that $g^x \equiv y \pmod{p}$.

For a fixed p, g, we let $\mathsf{DLP}(p, g)$ be the discrete log problem with inputs $y \in \mathbb{Z}_p^*$. The input to the DLP problem can be described by $n = \lceil \log_2 p \rceil$ bits. It is shown [3] that DLP is reducible to the following *decision problem* $\mathsf{DLP}_{\frac{1}{2}}$:

- **Input:** prime p, generator g of $\mathbb{Z}_p^*$, $y \in \mathbb{Z}_p^*$.
- **Output:** 1 if $\log_g y \leq \frac{p-1}{2}$ and 0 otherwise.

Lemma 2 ([3, Theorem 3])**.** *For every prime p and generator g, if there exists a polynomial time algorithm that correctly decides $\mathsf{DLP}_{\frac{1}{2}}(p, g)$ for at least $\frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$ fraction of the inputs $y \in \mathbb{Z}_p^*$, then there exists a polynomial time algorithm for $\mathsf{DLP}(p, g)$.*

Furthermore, [3] showed that DLP can be further reduced to the following promise discrete logarithm problem DLP_c for $\frac{1}{\mathsf{poly}(n)} \leq c \leq \frac{1}{2}$:

- **Input:** prime p, generator g of $\mathbb{Z}_p^*$, $y \in \mathbb{Z}_p^*$.
- **Promise:** $\log_g y \in [1, c(p-1)]$ or $[\frac{p-1}{2} + 1, \frac{p-1}{2} + c(p-1)]$
- **Output:** -1 if $\log_g y \in [1, c(p-1)]$ and $+1$ if $\log_g y \in [\frac{p-1}{2} + 1, \frac{p-1}{2} + c(p-1)]$.

Lemma 3 ([3])**.** *For every prime p, generator g and $\frac{1}{\mathsf{poly}(n)} \leq c \leq \frac{1}{2}$, if there exists a polynomial time algorithm for $\mathsf{DLP}_c(p, g)$, then there exists a polynomial time algorithm for $\mathsf{DLP}(p, g)$.*

The proof of this fact is implicitly implied by the proof of Lemma 3 and Theorem 3 in [3].

II. A CONCEPT CLASS REDUCIBLE TO DISCRETE LOG

In this section we construct a concept class, wherein learning the concept class is as hard as solving $\mathsf{DLP}_{\frac{1}{2}}(p, g)$. Therefore, assuming the hardness of $\mathsf{DLP}(p, g)$, no classical polynomial time algorithm can learn this concept class. On the other hand, the learning problem is a simple clustering problem in 1D after taking the discrete logarithm, and therefore is easy to learn using a quantum computer.

We work in the setting introduced in the previous section, where we use standard definitions (see Definition 1) from computational learning theory, and we assume a fixed p, g such that computing discrete log in $\mathbb{Z}_p^*$ is classically hard. Our concept class is defined as follows.

Definition 4 (Concept class)**.** *We define a concept class over the data space $\mathcal{X} = \mathbb{Z}_p^* \subseteq \{0,1\}^n$, where $n = \lceil \log_2 p \rceil$. For any $s \in \mathbb{Z}_p^*$, define a concept $f_s : \mathbb{Z}_p^* \to \{-1, 1\}$ as*

$$f_s(x) = \begin{cases} +1, & \text{if } \log_g x \in [s, s + \frac{p-3}{2}], \\ -1, & \text{else.} \end{cases} \tag{2}$$

Note that in interval $[s, s + \frac{p-3}{2}]$, $s + i$ denotes addition within $\mathbb{Z}_p^*$. By definition, f_s maps half the elements in $\mathbb{Z}_p^*$ to $+1$ and half of them to -1. The concept class is defined as $\mathcal{C} = \{f_s\}_{s \in \mathbb{Z}_p^*}$.

A target concept in $\mathcal{C}$ can be specified by choosing an element $s \in \mathbb{Z}_p^*$, which can be understood as the "secret key" for the concept f_s. We can also efficiently generate training samples for every concept.

Lemma 5. *For every concept $f \in \mathcal{C}$, there exists an efficient classical algorithm that can generate samples $(x, f(x))$, where x is uniformly random in $\mathbb{Z}_p^*$.*

Proof. To generate a sample $(x, f_s(x))$ for a concept f_s, first generate a random $y \sim \mathbb{Z}_p^*$, and let

$$b = \begin{cases} +1, & \text{if } y \in [s, s + \frac{p-3}{2}], \\ -1, & \text{else.} \end{cases} \tag{3}$$

Then return (g^y, b). Since g^y is also uniformly distributed in $\mathbb{Z}_p^*$, this procedure correctly generates a sample from the data distribution. □

Using the quantum algorithm for discrete logarithm problem [4, 5], the concept class $\mathcal{C}$ is polynomially learnable in BQP (in fact with probability 1). On the other hand, the result of Blum and Micali [3] implies that no efficient classical algorithm can do better than random guessing.

Theorem 6. *The concept class $\mathcal{C}$ is efficiently learnable in* BQP*. On the other hand, suppose there exists an efficient classical algorithm that, for every concept $f \in \mathcal{C}$, can achieve $\frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$ test accuracy, with probability at least $2/3$ over the choice of random training samples and randomness of the algorithm. Then there exists an efficient classical algorithm for* DLP*.*

Remark 1. *In the following we prove a stronger statement for classical hardness. We show that assuming the classical hardness of* DLP*, no efficient classical algorithm can achieve $\frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$ test accuracy with probability $\frac{2}{3}$ for* any *concept in the concept class $\mathcal{C}$.*

Proof. We first show quantum learnability. For every concept $f_s \in \mathcal{C}$, use a quantum computer to take the discrete logarithm of the classical training data samples. Then, after taking the discrete logarithm, the training data samples are clustered in two intervals: $[s, s + \frac{p-3}{2}]$ with label $+1$, and $[s + \frac{p-1}{2}, s + p - 2]$ with label -1, for the unknown $s \in \mathbb{Z}_p^*$ (which defines the unknown concept f_s). For a new data sample x, use a quantum computer to take its discrete log. Then, compute the average distance d_+/d_- between $\log_g x$ and the $+1/-1$ labeled clusters, respectively. Assign label to x based on which cluster is closer to $\log_g x$. This algorithm can achieve 99% accuracy for any concept f_s, with high probability over random training samples. We omit the detailed proof here.

To show classical hardness as stated in Remark 1, consider an arbitrary concept f_s and polynomially many training samples. By Lemma 5 this can be generated classically in polynomial time. By assumption, an efficient classical algorithm $\mathcal{A}$ can learn this concept with $\frac{1}{2} + \frac{1}{\mathsf{poly}(n)}$ accuracy (call $\mathcal{A}$ a good classifier if it satisfies this), with probability at least $2/3$. We use this learned classifier to solve $\mathsf{DLP}_{\frac{1}{2}}(p, g)$.

Algorithm $\mathcal{A}'$ for $\mathsf{DLP}_{\frac{1}{2}}(p, g)$:

1. On input $y \in \mathbb{Z}_p^*$ (the goal is to decide that whether $\log_g y \in [1, \frac{p-1}{2}]$ or $[\frac{p-1}{2}+1, p-1]$)

2. Send $y \cdot g^{s-1}$ to the classifier $\mathcal{A}$, decide $\log_g y \in [1, \frac{p-1}{2}]$ if the classifier returns $+1$, and decide $\log_g y \in [\frac{p-1}{2}+1, p-1]$ if the classifier returns -1.

To see that this procedure correctly decides $\mathsf{DLP}_{\frac{1}{2}}(p,g)$ with a non-trivial bias, for a good classifier $\mathcal{A}$ we have

$$\begin{aligned}
&\Pr_{y\sim\mathbb{Z}_p^*}\left[\mathcal{A}' \text{ correctly decides } \mathsf{DLP}_{\frac{1}{2}}(p,g) \text{ on input } y\right] \\
&= \Pr_{y\sim\mathbb{Z}_p^*}\left[\mathcal{A} \text{ correctly classifies } y\cdot g^{s-1} \text{ for concept } f_s\right] \\
&= \Pr_{y\sim\mathbb{Z}_p^*}\left[\mathcal{A} \text{ correctly classifies } y \text{ for concept } f_s\right] \\
&= \frac{1}{2} + \frac{1}{\mathsf{poly}(n)}.
\end{aligned} \tag{4}$$

By Lemma 2, once we have an algorithm that can correctly decide $\mathsf{DLP}_{\frac{1}{2}}(p,g)$ on $\frac{1}{2}+\frac{1}{\mathsf{poly}(n)}$ fraction of the inputs, it can be used to solve $\mathsf{DLP}(p,g)$ with high success probability. Finally by a simple union bound, we have a polynomial time algorithm that with high probability solves $\mathsf{DLP}(p,g)$. □

An advantage of our supervised learning task is that it is efficiently verifiable by a classical verifier. Consider the following challenge:

1. A classical verifier picks a random concept $f_s \sim \mathcal{C}$ (it can do so by choosing a uniformly random $s \sim \mathbb{Z}_p^*$). Then, generate polynomial-sized samples (S,T) where the data labels in T are removed.

2. The verifier sends (S,T) to a prover, and the prover returns a set of $\{-1,1\}$ labels for T.

3. The verifier accepts if more than 99% of the labels returned by the prover are correct.

Say a prover passes the challenge if the verifier accepts with probability at least 2/3. Our hardness result implies the following Corollary:

Corollary 7. *There exists a* BQP *prover that can pass the above challenge. Assuming the classical hardness of* DLP*, no polynomial-time classical prover can pass the above challenge.*

III. SUPPORT VECTOR MACHINE AND THE QUANTUM KERNEL ESTIMATION ALGORITHM

A. Support vector machines

We give a brief overview of support vector machine and the quantum kernel estimation algorithm [6, 7]. Along the way, we also establish properties that are useful for the analysis of our algorithm in the next section. We refer to Ref. [8, 9] for a more detailed introduction to support vector machines.

A support vector machine (SVM) is a classification algorithm that takes as input a set of training samples $S = \{(x_1,y_1),\ldots,(x_m,y_m)\}$ where $x_i \in \mathbb{R}^d$, $y \in \{-1,1\}$ and in time $\mathsf{poly}(d,m)$ (assume that the training set has polynomial size, $m = \mathsf{poly}(d)$) returns a set of parameters $(w,b) \in \mathbb{R}^d \times \mathbb{R}$ which define a linear classifier $f^* : \mathcal{X} \to \{-1,1\}$ as follows

$$y_{\text{pred}} = f^*(x) = \text{sign}\left(\langle w, x\rangle_E + b\right), \tag{5}$$

where $\langle w, x\rangle_E = w^T x$ denotes the inner product in Euclidean space. For a data vector x with true label y, it is easy to see that the classifier is correct on this point *if and only if* $y\left(\langle w, x\rangle_E + b\right) > 0$. We say a training set S is *linearly separable* if there exists (w, b) such that

$$y_i\left(\langle w, x_i\rangle_E + b\right) > 0, \quad \text{for every } (x_i, y_i) \in S, \tag{6}$$

and such a (w, b) is called a *separating hyperplane* for S in $\mathbb{R}^d$. When the training set is linearly separable, the SVM algorithm can efficiently find a separating hyperplane by running the so-called *hard margin primal program*

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|_2^2 \\ \text{s.t.} \quad & y_i\left(\langle w, x_i\rangle_E + b\right) \geq 1, \end{aligned} \tag{7}$$

a convex quadratic program whose optimal solution can be obtained in polynomial time. One important property of SVM is that it is a maximum margin classifier. For a general unnormalized hyperplane (w, b), define its *normalized margin* on a training data (x, y) as

$$\hat{\gamma}_{(w,b)}(x, y) = \frac{1}{\|w\|_2} y\left(\langle w, x\rangle_E + b\right) \tag{8}$$

and let $\gamma_{(w,b)}(x, y) = y\left(\langle w, x\rangle_E + b\right)$ denote the unnormalized margin. It is easy to see that Eq. (7) returns a classifier that maximizes

$$\min_{(x,y)\in S} \hat{\gamma}_{(w,b)}(x, y), \tag{9}$$

which is the minimum distance from any training point to the hyperplane. The general intuition that SVM maximizes the margin is useful for understanding the generalization bounds that we will prove in the next section.

However, for most "practical" purposes, assuming S is linearly separable is a strong assumption. Additionally, when the training set S is not linearly separable, Eq. (7) does not have a feasible solution. To find a good linear classifier with the presence of outliers, we introduce the *soft margin primal program*

$$\begin{aligned} \min_{w,\xi,b} \quad & \frac{1}{2}\|w\|_2^2 + \frac{\lambda}{2}\sum_i \xi_i^p \\ \text{s.t.} \quad & y_i\left(\langle x_i, w\rangle_E + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \end{aligned} \tag{10}$$

where ξ_i are the *slack variables* introduced to relax the margin constraints, with an additional penalty term $\frac{\lambda}{2}\sum_i \xi_i^p$. For any positive integer p, Eq. (10) is a convex program and is feasible. In this work, we focus on choosing $p = 2$, which becomes a quadratic program. In practice it is also common to use $p = 1$.

For the $p = 2$ case, we further derive the Wolfe dual program of (10) based on Lagrangian duality, resulting in the *L2 soft margin dual program*

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \ \langle x_i, x_j\rangle_E - \frac{1}{2\lambda}\sum_i \alpha_i^2 \\ \text{s.t.} \quad & \alpha_i \geq 0 \\ & \sum_i \alpha_i y_i = 0. \end{aligned} \tag{11}$$

Here the primal Lagrangian is given by

$$L_P = \frac{1}{2}\|w\|_2^2 + \frac{\lambda}{2}\sum_i \xi_i^2 - \sum_i \alpha_i \left(y_i \left(\langle x_i, w\rangle_E + b\right) - 1 + \xi_i\right) - \sum_i \mu_i \xi_i, \tag{12}$$

and the primal and dual optimal solutions can be connected via the Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned}
&w = \sum_i \alpha_i y_i x_i \\
&\sum_i \alpha_i y_i = 0 \\
&\lambda\xi_i - \alpha_i - \mu_i = 0 \\
&\alpha_i \left(y_i \left(\langle x_i, w\rangle_E + b\right) - 1 + \xi_i\right) = 0 \\
&\mu_i \xi_i = 0 \\
&\alpha_i \geq 0 \\
&\mu_i \geq 0 \\
&\xi_i \geq 0 \\
&y_i \left(\langle x_i, w\rangle_E + b\right) - 1 + \xi_i \geq 0.
\end{aligned} \tag{13}$$

An immediate corollary of Eq. (13) is

$$\alpha_i = \lambda\xi_i \tag{14}$$

at the optimal solution. This means that when λ is a constant, the Lagrangian multipliers α_i is proportional to the slack variables ξ_i. This is a useful property for our analysis later. In addition, the bias parameter b can be determined by the equality $y_i \left(\langle x_i, w\rangle_E + b\right) - 1 + \xi_i = 0$ for any $\alpha_i \neq 0$.

Finally, it will be convenient for us to work with optimizations without the bias parameter $b \in \mathbb{R}$. We show that we can assume this is without loss of generality, as we can add one extra dimension $\tilde{x} = (x, 1)/\sqrt{2}$ to the data vectors so that the bias parameter is absorbed into w. In this case, the $L2$ soft margin dual program becomes

$$\begin{aligned}
\max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{4}\sum_{i,j} \alpha_i \alpha_j y_i y_j \left(\langle x_i, x_j\rangle_E + 1\right) - \frac{1}{2\lambda}\sum_i \alpha_i^2 \\
\text{s.t.} \quad & \alpha_i \geq 0.
\end{aligned} \tag{15}$$

Here we used the fact that $\langle \tilde{x}_i, \tilde{x}_j\rangle_E = \frac{1}{2}\left(\langle x_i, x_j\rangle_E + 1\right)$, and notice that the equality constraint $\sum_i \alpha_i y_i = 0$ is removed. This is because of the new KKT conditions

$$\begin{aligned}
&w = \sum_i \alpha_i y_i x_i \\
&\lambda\xi_i - \alpha_i - \mu_i = 0 \\
&\alpha_i \left(y_i \left(\langle x_i, w\rangle_E\right) - 1 + \xi_i\right) = 0 \\
&\mu_i \xi_i = 0 \\
&\alpha_i \geq 0 \\
&\mu_i \geq 0 \\
&\xi_i \geq 0 \\
&y_i \left(\langle x_i, w\rangle_E\right) - 1 + \xi_i \geq 0,
\end{aligned} \tag{16}$$

where $\alpha_i = \lambda\xi_i$ still hold at optimality.

B. Non-linear classification

In this section we generalize support vector machines for *non-linear* classification, i.e., we map the d-dimensional data vectors into a n-dimensional feature space ($n \gg d$) via a *feature map*:

$$\Phi : \mathcal{X} \to \mathcal{H} \subseteq \mathbb{R}^n, \tag{17}$$

where we assume that Φ is normalized, i.e., it maps a unit vector to a unit vector. The feature map is chosen prior to seeing the training data. Notice that in the dual program, training data is only accessed via the *kernel* matrix $K \in \mathbb{R}^{m\times m}$ (recall that m denotes the number of training samples) defined as

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}}. \tag{18}$$

Therefore, it's possible to work with an exponentially large (or even infinite-dimensional) feature space (i.e., when n is large), as long as the kernel is computable in $\mathsf{poly}(d)$ time.

In addition, for any feature map $\Phi_0 : \mathcal{X} \to \mathbb{R}^n$, we can always use a new feature map $\Phi : \mathcal{X} \to \mathbb{R}^{n+1}$ such that $\Phi(x) = (\Phi_0(x), 1)/\sqrt{2}$, which allows us to remove the bias parameter b. This can be done via changing the kernel as $K(x_i, x_j) = \frac{1}{2}\left(K_0(x_i, x_j) + 1\right)$ as shown in the previous section. Therefore, with a suitable kernel transformation, we can run the following dual program without loss of generality:

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \frac{1}{2\lambda}\sum_i \alpha_i^2 \\ \text{s.t.} \quad & \alpha_i \geq 0. \end{aligned} \tag{19}$$

Let $Q \in \mathbb{R}^{m\times m}$ be a matrix such that $Q_{ij} = y_i y_j K(x_i, x_j)$. Then we can write the dual program in vectorized form

$$\begin{aligned} \max_{\alpha} \quad & 1^T \alpha - \frac{1}{2}\alpha^T \left(Q + \frac{1}{\lambda}\mathbb{I}\right)\alpha \\ \text{s.t.} \quad & \alpha \geq 0. \end{aligned} \tag{20}$$

It is easy to see that for every $\lambda > 0$, Eq. (20) is a strongly convex quadratic program and has a unique optimal solution. After training is finished (i.e., we obtain training samples, compute the kernel K and solve Eq. (20)), when a learner is presented with a new test example x, the classifier returns

$$y_{\text{pred}} = \text{sign}\left(\langle w, \Phi(x)\rangle_{\mathcal{H}}\right) = \text{sign}\left(\sum_i \alpha_i y_i K(x_i, x)\right), \tag{21}$$

where we used the KKT condition $w = \sum_i \alpha_i y_i \Phi(x_i)$ (which can be derived by simply replacing x_i with $\Phi(x_i)$ in Eq. (16)). So a classifier needs to evaluate the kernel function on the new test example and output y_{pred}.

C. Quantum kernel estimation

Different from the above standard approaches, the kernel used in our quantum algorithm is constructed by a quantum feature map. The main idea in the *quantum kernel estimation* algorithm [6, 7] is to map classical data vectors into quantum states:

$$x \to \Phi(x) = |\phi(x)\rangle\langle\phi(x)|, \tag{22}$$

where we use the density matrix representation to avoid global phase. Then, the kernel function is the Hilbert-Schmidt inner product between density matrices,

$$K(x_i, x_j) = \mathrm{Tr}\left[\Phi(x_i)^\dagger \Phi(x_j)\right] = \left|\langle\phi(x_i)|\phi(x_j)\rangle\right|^2 . \tag{23}$$

This *quantum feature map* is implemented via a quantum circuit parameterized by x,

$$|\phi(x)\rangle = U(x)\,|0^n\rangle\,, \tag{24}$$

where we assume the feature map uses n qubits. Therefore, to obtain the kernel function

$$K(x_i, x_j) = \left|\langle 0^n|\, U^\dagger(x_i)U(x_j)\,|0^n\rangle\right|^2 , \tag{25}$$

we can run the quantum circuit $U^\dagger(x_i)U(x_j)$ on input $|0^n\rangle$, and measure the probability of the 0^n output.

Algorithm 1 Support vector machine with quantum kernel estimation (SVM-QKE training)

Input: a training set $S = \{(x_i, y_i)\}_{i=1}^m$
Output: $\alpha_1, \ldots, \alpha_m$ (solution to the $L2$ soft margin dual program (20))

for $i = 1 \ldots m$ **do**
  $K_0(x_i, x_i) := 1$
end for
for $i = 1 \ldots m$ **do** ▷ quantum kernel estimation
  for $j = i + 1 \ldots m$ **do**
    Apply $U^\dagger(x_i)U(x_j)$ on input $|0^n\rangle$
    Measure the output probability of 0^n with R shots, denoted as p
    $K_0(x_i, x_j) = K_0(x_j, x_i) := p$
  end for
end for
$K := \frac{1}{2}\left(K_0 + \mathbf{1}_{m\times m}\right)$ ▷ SVM training
Run the dual program (20), record solution as α
Return α

Algorithm 2 Support vector machine with quantum kernel estimation (SVM-QKE testing)

Input: a new example $x \in \mathcal{X}$, a training set $S = \{(x_i, y_i)\}_{i=1}^m$, training parameters $\alpha_1, \ldots, \alpha_m$
Output: $y \in \{-1, 1\}$

$t := 0$
for $i = 1 \ldots m$ **do** ▷ quantum kernel estimation
  Apply $U^\dagger(x)U(x_i)$ on input $|0^n\rangle$
  Measure the output probability of 0^n with R shots, denoted as p
  $t := t + \alpha_i y_i \cdot \frac{p+1}{2}$
end for
Return $\mathrm{sign}(t)$

We describe the full support vector machine algorithm with quantum kernel estimation (SVM-QKE), with training (Algorithm 1) and testing (Algorithm 2) phases. Here, $\mathbf{1}$ denotes the all-one matrix, and R denotes the number of measurement shots for each kernel estimation circuit.

The main differences between QKE and classical kernels are two-fold:

- On the one hand, quantum feature maps are more expressive than classical feature maps. Therefore, a SVM trained with a quantum kernel may achieve better performance than with classical kernels.

- On the other hand, a fundamental feature in QKE is that we only have a noisy estimate of the quantum kernel entries in both training and testing, due to finite sampling error in experiment. More specifically, for each $K_0(x_i, x_j)$ as defined in Algorithm 1, we have access to a noisy estimator p with mean equals $K_0(x_i, x_j)$ and variance $\frac{1}{R}$.

Therefore, noise robustness is an important property for provable quantum advantage with QKE. We will formally prove this in the next section.

IV. EFFICIENT LEARNABILITY WITH QUANTUM KERNEL ESTIMATION

A. Quantum feature map

We now define our quantum feature map for learning the concept class $\mathcal{C}$ based on the discrete logarithm problem (recall Definition 4). This family of states, whose construction is based on the discrete logarithm problem, was first introduced in Ref. [10] to study the complexity of quantum state generation and statistical zero knowledge.

For a prime p, let $n = \lceil \log_2 p \rceil$ be the number of bits needed to represent $\{0, 1, \ldots, p-1\}$. For $y \in \mathbb{Z}_p^*$, $k \in \{1, 2, \ldots, n-1\}$, define a polynomial-sized classical circuit family $\{C_{y,k}\}$ as follows:

$$\begin{aligned} &C_{y,k} : \{0,1\}^k \to \{0,1\}^n, \\ &C_{y,k}(i) = y \cdot g^i \pmod p, \forall i \in \{0,1\}^k. \end{aligned} \tag{26}$$

It's easy to see that $C_{y,k}$ is injective, i.e., for all $i \neq j$, we have $C_{y,k}(i) \neq C_{y,k}(j)$. Furthermore, $C_{y,k}(i)$ can be computed using $\mathcal{O}(n)$ multiplications within $\mathbb{Z}_p^*$. We now show how to prepare a uniform superposition over the elements of $C_{y,k}$ on a quantum computer, which we refer to as the n-qubit feature state

$$|C_{y,k}\rangle = \frac{1}{\sqrt{2^k}} \sum_{i \in \{0,1\}^k} \left| y \cdot g^i \right\rangle. \tag{27}$$

First construct the reversible extension of $C_{y,k}$ as

$$\begin{aligned} &\tilde{C}_{y,k} : \{0,1\}^{2n} \to \{0,1\}^{2n}, \\ &\tilde{C}_{y,k} |i\rangle |0^n\rangle = |i\rangle |C_{y,k}(i)\rangle, \end{aligned} \tag{28}$$

where $|i\rangle$ uses the least significant bits of the n-bit register. Then, construct a quantum circuit U_y using the quantum algorithm for discrete log [4, 5, 11] which uses $\tilde{\mathcal{O}}(n^3)$ gates (we use $\tilde{\mathcal{O}}(\cdot)$ to hide polylog factors),

$$U_y |C_{y,k}(i)\rangle |0\rangle = |i\rangle |C_{y,k}(i)\rangle. \tag{29}$$

The overall procedure for preparing $|C_{y,k}\rangle$ is as follows, up to adding/discarding auxiliary qubits:

$$|0^n\rangle \xrightarrow{H^{\otimes k}} \frac{1}{\sqrt{2^k}} \sum_{i \in \{0,1\}^k} |i\rangle \xrightarrow{\tilde{C}_{y,k}} \frac{1}{\sqrt{2^k}} \sum_{i \in \{0,1\}^k} |i\rangle |C_{y,k}(i)\rangle \xrightarrow{U_y^\dagger} \frac{1}{\sqrt{2^k}} \sum_{i \in \{0,1\}^k} |C_{y,k}(i)\rangle. \tag{30}$$

Definition 8. *Define the family of* feature states *via the map* $(y, k) \to |C_{y,k}\rangle$ *that takes classical data* $y \in \mathbb{Z}_p^*, k \in \{1, 2, \ldots, n-1\}$ *and maps it to a* n*-qubit feature state*

$$|C_{y,k}\rangle = \frac{1}{\sqrt{2^k}} \sum_{i \in \{0,1\}^k} \left| y \cdot g^i \right\rangle. \tag{31}$$

Such a procedure can be implemented in BQP *using* $\tilde{\mathcal{O}}(n^3)$ *gates.*

In Ref. [10], it was proven that constructing these feature states is as hard as solving discrete log, where they show that $\mathsf{DLP}_{1/6}$ can be reduced to estimating the inner product between $|C_{y,k}\rangle$ with different y and k. In this work, our quantum kernel is constructed via the feature map with a fixed k, which is chosen prior to running the quantum kernel estimation algorithm. More specifically, for different training samples $y, y' \in \mathbb{Z}_p^*$, the corresponding kernel entry is given by

$$K_0(y, y') = \left|\langle C_{y,k} | C_{y',k}\rangle\right|^2 . \tag{32}$$

We note that these feature states have a special structure: after taking the discrete log for each basis state, the feature state becomes the superposition of an interval. Therefore, computing the inner product between feature states is equivalent to computing the intersection between their corresponding intervals. We provide more details in the following definition.

Definition 9 (Interval states)**.** *For a fixed g, p, suppose $y = g^x$. The feature state can be written as $|C_{y,k}\rangle = \frac{1}{\sqrt{2^k}} \sum_{i=0}^{2^k-1} |g^{x+i}\rangle$. This can be understood as "interval states" in the log space, where the exponent spans an interval $[x, \ldots, x + 2^k - 1]$ of length 2^k. As a consequence, since $y = g^x$ is a one-to-one mapping, computing the inner products between feature states is equivalent to computing intersection of the corresponding intervals.*

By definition, our kernel K_0 is constructed using interval states with a fixed length. In order for our quantum algorithm to solve a classically hard problem, a necessary condition is that the kernel entries $K_0(y, y')$ cannot be efficiently estimated up to additive error. Otherwise, the quantum kernel estimation procedure can be efficiently simulated classically. Next we show that estimating the kernel entries is as hard as solving the discrete log problem. Although this is implied by our main results (Theorem 6 and 20), here we give a direct proof which is a generalization of Ref. [10].

Lemma 10. *For an arbitrary (fixed) prime p and generator g, if there exists a polynomial time algorithm such that, on input $y, y' \in \mathbb{Z}_p^*$, computes $K_0(y, y')$ up to 0.01 additive error, then there exists a polynomial time algorithm for $\mathsf{DLP}(p, g)$.*

Proof. We show this lemma by using an algorithm that estimates the kernel entries well to solve $\mathsf{DLP}_{\frac{1}{16}}(p, g)$ which in turn (by Lemma 3) implies an efficient algorithm for $\mathsf{DLP}(p, g)$. In the following we assume $k = n - 3$, but the proof can be generalized to any $k = n - t \log n$ for some constant t.

Consider an input $y = g^x$ for the problem $\mathsf{DLP}_{\frac{1}{16}}(p, g)$, where we are promised that either $x \in [1, \frac{p-1}{16}]$ or $x \in [\frac{p-1}{2} + 1, \frac{p-1}{2} + \frac{p-1}{16}]$. Let $y' = g^{(p+1)/2}$ and consider feature states $|C_y\rangle$ and $|C_{y'}\rangle$. Then for the two cases,

1. If $x \in [1, \frac{p-1}{16}]$, $|C_y\rangle$ corresponds to a subinterval of $[1, \frac{p-1}{2}]$ and therefore $K_0(y, y') = 0$.

2. If $x \in [\frac{p-1}{2} + 1, \frac{p-1}{2} + \frac{p-1}{16}]$, the intersection of the corresponding intervals is at least $\frac{p}{16}$, so $K_0(y, y') \geq \frac{1}{16}$.

Therefore, an algorithm that can approximate $K_0(y, y')$ within 0.01 additive error can decide the promise problem $\mathsf{DLP}_{\frac{1}{16}}(p, g)$. Lemma 3 now shows the lemma statement. □

B. Mapping to high dimensional Euclidean space

Now we are ready to apply the feature map in our support vector machine algorithm using quantum kernel estimation. We recall the definition of $\mathcal{C}$ (Definition 4) here for convenience:

$\mathcal{C} = \{f_s\}_{s\in\mathbb{Z}_p^*}$, where

$$f_s(x) = \begin{cases} +1, & \text{if } \log_g x \in [s, s+\frac{p-3}{2}], \\ -1, & \text{else.} \end{cases} \tag{33}$$

Consider the mapping from $x \in \mathbb{Z}_p^*$ to the quantum feature states described in the previous section (renamed here as $|\phi(x)\rangle$), $x \to |\phi(x)\rangle\langle\phi(x)|$ with

$$|\phi(x)\rangle = \frac{1}{\sqrt{2^k}} \sum_{i=0}^{2^k-1} \left|x \cdot g^i\right\rangle, \tag{34}$$

where $k = n - t\log n$ for some constant t to be specified later (recall that $n = \lceil \log_2 p \rceil$). It was shown in Definition 8 that $|\phi(x)\rangle$ can be prepared in BQP. Let $\Delta = \frac{2^{k+1}}{p} = \mathcal{O}(n^{-t})$. Then the feature states span a $\frac{\Delta}{2} = \mathcal{O}(n^{-t})$ fraction of the elements in $\mathbb{Z}_p^*$.

Also define the *halfspace state* $|\phi_s\rangle$ corresponding to every concept $f_s \in \mathcal{C}$ as follows

$$|\phi_s\rangle = \frac{1}{\sqrt{(p-1)/2}} \sum_{i=0}^{(p-3)/2} \left|g^{s+i}\right\rangle, \quad \text{for every } s \in \mathbb{Z}_p^*. \tag{35}$$

Observe that the halfspace state spans a $\frac{1}{2}$-fraction of the full space $\mathbb{Z}_p^*$. The following property shows that $W_s = |\phi_s\rangle\langle\phi_s|$ is a separating hyperplane in the feature space.

- $|\langle\phi_s|\phi(x)\rangle|^2 = \Delta$, for $1-\Delta$ fraction of x in $\{x : f_s(x) = +1\}$.
- $|\langle\phi_s|\phi(x)\rangle|^2 = 0$, for $1-\Delta$ fraction of x in $\{x : f_s(x) = -1\}$.

Notice that $|\phi_s\rangle$ has a *large margin* property: it separates training samples with label $+1$ from those with label -1. The probability of having an outlier (a data point that lies inside the margin or on the wrong side of the hyperplane) is small, which equals $\Delta = 1/\mathsf{poly}(n)$. Recall that the goal of an SVM algorithm is to find a hyperplane that maximizes the margin on the training set, and the above property shows that such a good hyperplane exists.

In general, learning a separating hyperplane that separates $+1/-1$ examples is called a *halfspace learning* problem. Rigorously speaking, our data vectors are represented by quantum states with the Hilbert-Schmidt inner product. For simplicity, we now show that our learning problem is equivalent to learning a halfspace in 4^n-dimensional Euclidean space. For that, we first express a Hermitian matrix $W \in \mathbb{C}^{2^n\times 2^n}$ uniquely in terms of the orthonormal Pauli basis as follows

$$W = \frac{1}{\sqrt{2^n}} \sum_{p\in\{0,1,2,3\}^n} w_p \sigma_p, \tag{36}$$

where $\sigma_p \in \{\mathsf{I},\mathsf{X},\mathsf{Y},\mathsf{Z}\}^{\otimes n}$ are n-qubit Pauli operators and $w_p = \frac{1}{\sqrt{2^n}}\mathrm{Tr}[\sigma_p W] \in \mathbb{R}$ are the *Fourier coefficients*. We can use the 4^n dimensional *Pauli vector* $w = (w_p)$ to represent W, as the Hilbert-Schmidt inner product $\mathrm{Tr}\left[W^\dagger W'\right] = \langle w, w'\rangle_E$ is equivalent to the Euclidean inner product. Also note that a pure quantum state in Hilbert space corresponds to a unit Pauli vector in Euclidean space.

The large margin property can be recast in Euclidean space with the Pauli basis representation.

Lemma 11. *For any concept $f_s \in \mathcal{C}$, let w_s be the Pauli vector of $|\phi_s\rangle\langle\phi_s|$, $b = -\frac{\Delta}{2}$, and $\hat{x}$ be the Pauli vector of $|\phi(x)\rangle\langle\phi(x)|$. Then*

- $\langle w_s, \hat{x} \rangle_E + b = \frac{\Delta}{2}$, *for* $1 - \Delta$ *fraction of* x *in* $\{x : f_s(x) = +1\}$,
- $\langle w_s, \hat{x} \rangle_E + b = -\frac{\Delta}{2}$, *for* $1 - \Delta$ *fraction of* x *in* $\{x : f_s(x) = -1\}$,

where $\langle \cdot, \cdot \rangle_E$ *denotes Euclidean inner product.*

Our SVM algorithm that uses the kernel $K_0(x, x') = |\langle \phi(x) | \phi(x') \rangle|^2$ can be equivalently understood as using the kernel $K_0(x, x') = \langle \hat{x}, \hat{x}' \rangle_E$ based on a feature map that maps $x \in \mathbb{Z}_p^*$ to $\hat{x}$, a 4^n dimensional vector in Euclidean space. Finally, recall that we only have access to a noisy estimate of $K_0(x, x')$ using quantum kernel estimation. The noisy estimator that we obtain from a quantum computer has mean $K_0(x, x')$ and variance $\frac{1}{R}$, where R denotes the number of measurement shots for each kernel estimation circuit.

To summarize, here we have shown that the original problem of learning the concept class $\mathcal{C}$ can be mapped to a noisy halfspace learning problem [12] in high dimensional Euclidean space with the following properties. For simplicity, below we do not specify the concept, since everything holds equivalently for each concept in $\mathcal{C}$. From now on our analysis is restricted to the high dimensional Euclidean space, and for notation simplicity we overload x to represent the Pauli vector $\hat{x}$.

Properties of noisy halfspace learning.

1. *Data space*: $\mathcal{X} \subseteq \mathbb{R}^{4^n}$ with unit length $\|x\|_2 = 1$ for every $x \in \mathcal{X}$. Each x is associated with a label $y \in \{-1, 1\}$.

2. *Separability*: the data points lie outside a margin of $\frac{\Delta}{2}$ with high probability over the uniform distribution on $\mathcal{X}$. That is, there exists a hyperplane (w, b) where $w \in \mathbb{R}^{4^n}$, $\|w\|_2 = 1$, and $b \in \mathbb{R}$, such that

$$\Pr_{x \sim \mathcal{X}} \left[y(\langle w, x \rangle_E + b) \geq \frac{\Delta}{2} \right] = 1 - \Delta. \tag{37}$$

3. *Bounded distance*: all data points are close to the above hyperplane:

$$|y(\langle w, x \rangle_E + b)| \leq \frac{\Delta}{2}, \quad \text{for every } x \in \mathcal{X}. \tag{38}$$

4. *Noisy kernel*: instead of having the ideal kernel $K_0(x_i, x_j) = \langle x_i, x_j \rangle_E$, we have access to a noisy kernel K_0', where $K_0'(x_i, x_j) = K_0(x_i, x_j) + e_{ij}$. Here, e_{ij} are independent random variables satisfying

   - $e_{ij} \in [-1, 1]$
   - $\mathbb{E}[e_{ij}] = 0$
   - $\mathrm{Var}[e_{ij}] \leq \frac{1}{R}$, where R denotes the number of measurement shots.

These properties are simple corollaries of the definition of $|\phi(x)\rangle$, $|\phi_s\rangle$ and Lemma 11.

In order to further simplify our analysis, we perform an additional transform which allows us to remove the bias parameter b without loss of generality. This will help us simplify our notations in later proofs. More specifically, we replace x with $(x, 1)/\sqrt{2}$ and w with $(w, b)/\sqrt{w^2 + b^2}$. This corresponds to replacing the original kernel K_0 with a new kernel $K = \frac{1}{2}(K_0 + \mathbf{1}_{m \times m})$ where $\mathbf{1}$ denotes the all-one matrix. These steps are explained in more detail in Section III. The final form of our halfspace learning problem is given below.

Lemma 12. *We have mapped the original problem of learning the concept class $\mathcal{C}$ into the following noisy halfspace learning problem. Below we do not specify the concept, as these properties hold for every concept in $\mathcal{C}$.*

1. Data space: $\mathcal{X} \subseteq \mathbb{R}^{4^n+1}$ *with unit length* $\|x\|_2 = 1$, $\forall x \in \mathcal{X}$. *Each x is associated with a label* $y \in \{-1, 1\}$.

2. Separability: *the data points lie outside a $\mathcal{O}(\Delta)$ margin with high probability over the uniform distribution. That is, there exists a hyperplane w where $w \in \mathbb{R}^{4^n+1}$, $\|w\|_2 = 1$, such that*

$$\Pr_{x\sim\mathcal{X}}\left[y\langle w, x\rangle_E \geq \frac{\Delta}{\sqrt{8+2\Delta^2}}\right] = 1 - \Delta. \tag{39}$$

3. Bounded distance: *all data points are close to the above hyperplane:*

$$|y\langle w, x\rangle_E| \leq \frac{\Delta}{\sqrt{8+2\Delta^2}}, \quad \textit{for every } x \in \mathcal{X}. \tag{40}$$

4. Noisy kernel: *let* $K_{ij} = \frac{1}{2}(1 + K_0(x_i, x_j))$. *We have access to a noisy kernel K', where* $K'_{ij} = K_{ij} + e_{ij}$. *Here, e_{ij} are independent random variables satisfying*

   - $e_{ij} \in [-1/2, 1/2]$
   - $\mathbb{E}[e_{ij}] = 0$
   - $\mathrm{Var}[e_{ij}] \leq \frac{1}{R}$, *where R denotes the number of measurement shots.*

The hyperplane specified in the above lemma is particularly useful for our analysis later. We define its unnormalized version as the "ground truth hyperplane" as follows.

Definition 13. *Consider the halfspace learning problem defined in Lemma 12. Define $w^* \in \mathbb{R}^{4^n+1}$ as the (unnormalized) ground truth hyperplane as given in Lemma 12, that satisfies the following properties:*

$$\begin{aligned} &\Pr_{x\sim\mathcal{X}}\left[y\langle w^*, x\rangle_E \geq 1\right] = 1 - \Delta, \\ &|y\langle w^*, x\rangle_E| \leq 1, \quad \textit{for every } x \in \mathcal{X}. \end{aligned} \tag{41}$$

Note that the norm of w^* is $\|w^*\|_2 = \mathcal{O}(\Delta^{-1})$.

C. Generalization of the noisy classifier

Next, we focus on the noisy halfspace learning problem as given by Lemma 12. We show that the four properties established in Lemma 12 are sufficient for formally proving the efficient learnability of the concept class $\mathcal{C}$ using our quantum algorithm.

Consider the primal optimization problem in the support vector machine used by Algorithm 1:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\|w\|_2^2 + \frac{\lambda}{2}\sum_i \xi_i^2 \\ \text{s.t.} \quad & y_i\langle x_i, w\rangle_E \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{42}$$

with the dual form

$$\begin{aligned} \max_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} - \frac{1}{2\lambda}\sum_i \alpha_i^2 \\ \text{s.t.} \quad & \alpha_i \geq 0. \end{aligned} \tag{43}$$

The above duality follows from the KKT conditions $w = \sum_i \alpha_i y_i x_i$ and $\alpha_i = \lambda \xi_i$. The kernel matrix K is a positive semidefinite matrix. Let Q be a matrix such that $Q_{ij} = y_i y_j K_{ij}$, which is also positive semidefinite. Then the dual program (43) is equivalent to the following convex quadratic program:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T \left(Q + \frac{1}{\lambda}\mathbb{I}\right)\alpha - 1^T\alpha \\ \text{s.t.} \quad & \alpha \geq 0. \end{aligned} \tag{44}$$

Recall that we have small additive perturbations in K, which in turn gives additive perturbations in Q. One useful property of the dual program (44) is that it is robust to perturbations in Q. More specifically, we use the following lemma from standard perturbation analysis.

Lemma 14 ([13, Theorem 2.1])**.** *Let x_0 be the solution to the quadratic program*

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T K x - c^T x \\ s.t. \quad & Gx \leq g \\ & Dx = d, \end{aligned} \tag{45}$$

where K is positive definite with smallest eigenvalue $\lambda > 0$. Let K' be a positive definite matrix such that $\|K' - K\|_F \leq \varepsilon < \lambda$. Let x_0' be the solution to (45) with K replaced by K'. Then

$$\|x_0' - x_0\|_2 \leq \frac{\varepsilon}{\lambda - \varepsilon}\|x_0\|_2. \tag{46}$$

Before going into the analysis of robustness against noise, notice that the bound in Lemma 14 is multiplicative. Therefore, it is useful to establish an upper bound on $\|\alpha\|_2$, the norm of the solution to the noiseless quadratic program (43). Recall from the KKT conditions that $\alpha_i = \lambda \xi_i$, where the dual variables are directly related to the slack variables in the primal program (42). The following lemma establishes a useful property for ξ_i^* for the ground truth hyperplane.

Lemma 15. *For the ground truth hyperplane w^* as defined in Definition 13, the corresponding slack variables ξ_i^* in the primal program (42) satisfy*

$$\mathbb{E}\left[\|\xi^*\|_2^2\right] \leq \mathcal{O}(m\Delta), \tag{47}$$

where the expectation is taken over the training set.

Proof. We can write the slack variables as

$$\xi_i^* = \max\{1 - y_i \langle x_i, w^* \rangle_E, 0\}. \tag{48}$$

By the properties given in Definition 13, we have

$$\begin{aligned} &\Pr\left[\xi_i^* = 0\right] = 1 - \Delta, \\ &\xi_i^* \leq 2. \end{aligned} \tag{49}$$

Therefore,

$$\mathbb{E}\left[\|\xi^*\|_2^2\right] = m\,\mathbb{E}\left[\xi_1^{*2}\right] \leq 4m\Delta. \tag{50}$$

□

Now we are ready to bound the norm of the dual variables α_i. Intuitively, we can do so because $\|\xi\|_2^2$ is part of the training loss in the primal program (42). The loss of the solution returned by the program can only be smaller than the loss of the ground truth hyperplane, which is guaranteed to be small.

Lemma 16. *Let α_0 be the solution returned by the dual program (43). We have*

$$\mathbb{E}\left[\|\alpha_0\|_2^2\right] = \mathcal{O}\left(\frac{1}{\Delta^2} + m\Delta\right), \tag{51}$$

where the expectation is over the training set.

Proof. Let $w_0 = \sum_i \alpha_{0i} y_i x_i$ be the hyperplane which corresponds to α_0, and let ξ_0 be the corresponding slack variable. Then

$$\|\alpha_0\|_2^2 = \lambda^2 \|\xi_0\|_2^2 \leq \lambda\left(\|w_0\|_2^2 + \lambda\|\xi_0\|_2^2\right) \leq \lambda\left(\|w^*\|_2^2 + \lambda\|\xi^*\|_2^2\right). \tag{52}$$

Here, the first line follows from the KKT condition $\alpha_i = \lambda\xi_i$, and the third line is because w_0 is the optimal solution to (42). Therefore by Lemma 15, $\mathbb{E}\left[\|\alpha_0\|_2^2\right] = \mathcal{O}\left(\frac{1}{\Delta^2} + m\Delta\right)$. □

Remark 2. *Recall that we have the freedom to choose $\Delta = \mathcal{O}(n^{-t})$ for any constant t. Suppose we have polynomially many training samples $m \approx n^c$, and let $t = c/3$. Then the above bound gives $\mathbb{E}\left[\|\alpha_0\|_2^2\right] = \mathcal{O}\left(m^{2/3}\right)$.*

Having established the above lemmas, now we are ready to prove our key result for noise robustness (Lemma 17). Let Q' be the noisy kernel measured by a quantum computer, and let $\lambda \in (0, 1)$ be a constant. Here we briefly recall the steps in Algorithm 1 and 2. The classifier is constructed in two steps. First, use a classical computer to run the dual program (44) with Q replaced by the experimental estimate Q', and let α' be the solution returned by the program. Second, given a new data sample x, use a quantum computer to obtain noisy estimates $K'(x, x_i)$ for all i, and output

$$y_{\text{pred}} = \text{sign}\left(\sum_i \alpha_i' y_i K'(x, x_i)\right). \tag{53}$$

Let $h(x) = \sum_i \alpha_i y_i K(x, x_i)$ and $h'(x) = \sum_i \alpha_i' y_i K'(x, x_i)$, which corresponds to the value of the noiseless/noisy classifier before taking the sign. We will prove the following result which establishes the noise robustness of h.

Lemma 17 (Noise robustness)**.** *Suppose we take $R = \mathcal{O}(m^4)$ measurement shots for each quantum kernel estimation circuit. Then, with probability at least 0.99 (over the choice of random training samples and measurement noise), for every $x \in \mathcal{X}$ we have*

$$\left|h(x) - h'(x)\right| \leq 0.01. \tag{54}$$

Proof. Consider the (noisy) quadratic program (43). The Frobenius norm is given by

$$\|Q' - Q\|_F^2 = 2\sum_{i<j} e_{ij}^2, \tag{55}$$

where e_{ij} are independent random variables satisfying $\mathbb{E}[e_{ij}] = 0$ and $\mathbb{E}\left[e_{ij}^2\right] \leq \frac{1}{R}$. Therefore $\mathbb{E}\left[\|Q' - Q\|_F^2\right] \leq \mathcal{O}\left(\frac{m^2}{R}\right)$. Now we invoke Lemma 14 and Lemma 16 (see Remark 2). Using

Markov's inequality: with probability at least 0.999 (over the choice of training samples and measurement noise), we have that

$$\|Q' - Q\|_F^2 \leq \mathcal{O}\left(\frac{m^2}{R}\right), \text{ and } \quad \|\alpha\|_2^2 = \mathcal{O}\left(m^{2/3}\right). \tag{56}$$

Let $\delta_i = \alpha_i' - \alpha_i$. Since $\lambda_{\min}\left(Q + \frac{1}{\lambda}\mathbb{I}\right) \geq \frac{1}{\lambda}$ is lower bounded by a constant, Lemma 14 gives

$$\|\delta\|_2 \leq \mathcal{O}\left(\frac{m^{4/3}}{\sqrt{R}}\right). \tag{57}$$

Then, let $\nu_i = K'(x, x_i) - K(x, x_i)$ for $i = 1, \ldots, m$. Similarly, ν_i are independent random variables satisfying $\mathbb{E}[\nu_i] = 0$ and $\mathbb{E}\left[\nu_i^2\right] \leq \frac{1}{R}$. By Markov's inequality, we have $\|\nu\|_2 \leq \mathcal{O}\left(\frac{\sqrt{m}}{\sqrt{R}}\right)$ with probability at least 0.999. Overall for any $x \in \mathcal{X}$, the error bound gives

$$\begin{aligned}
\left|h(x) - h'(x)\right| &= \left|\sum_i (\alpha_i + \delta_i) y_i (K(x, x_i) + \nu_i) - \sum_i \alpha_i y_i K(x, x_i)\right| \\
&\leq \sum_i \left|\alpha_i \nu_i + \delta_i K(x, x_i) + \delta_i \nu_i\right| \\
&\leq \|\alpha\|_2 \cdot \|\nu\|_2 + \sqrt{m}\|\delta\|_2 + \|\delta\|_2 \cdot \|\nu\|_2 \\
&\leq \mathcal{O}\left(\frac{m^{11/6}}{\sqrt{R}}\right),
\end{aligned} \tag{58}$$

where the third line uses Cauchy–Schwarz inequality. Therefore, $R = \mathcal{O}(m^4)$ measurement shots is sufficient for achieving $|h(x) - h'(x)| \leq 0.01$, and by a simple union bound this holds with probability at least 0.99. □

Having established noise robustness, it remains to prove a generalization error bound for the noisy classifier: if the classifier has small training error/loss, it should also have small test error, which is referred to as *generalization error* in learning theory. The main idea is a two-step argument:

1. The noiseless classifier $y = \operatorname{sign}(h(x))$ (we have defined $h(x) = \sum_i \alpha_i y_i K(x, x_i) = \langle w, x\rangle_E$) has small generalization error, which follows from standard generalization bounds for soft margin classifiers.

2. We have established that the noisy classifier is close to the noiseless classifier. Therefore, the noisy classifier should also have small generalization error.

For the first step, we refer to standard results on the generalization of soft margin classifiers (see, for example [14–19]). Recall that a hyperplane w correctly classifies a data point (x, y) if and only if $y\langle w, x\rangle_E > 0$. Therefore for a specific concept $f \in \mathcal{C}$, the test accuracy of $f^*(x) = \operatorname{sign}(\langle w, x\rangle_E)$ is given by

$$\operatorname{acc}_f(f^*) = \Pr_{x\sim\mathcal{X}}\left[f^*(x) = f(x)\right] = 1 - \Pr_{x\sim\mathcal{X}}\left[y\langle w, x\rangle_E < 0\right], \tag{59}$$

where we have used $y = f(x)$. Our results will be given in the form of an upper bound on $\Pr_{x\sim\mathcal{X}}\left[y\langle w, x\rangle_E < 0\right]$. The following result gives a generalization bound that coincides with our $L2$ training loss up to polylog factors, as indicated by the $\tilde{\mathcal{O}}$ notation, and therefore is directly applicable to the noiseless classifier.

Lemma 18 ([18, Theorem VII.11])**.** *For any hyperplane w satisfying the constraints of the primal program (42), with probability $1-\delta$ over randomly drawn training set S of size m, the generalization error is bounded by*

$$\Pr_{x\sim\mathcal{X}}[y\langle w,x\rangle_E<0]\leq\frac{1}{m}\tilde{\mathcal{O}}\left(\|w\|_2^2+\|\xi\|_2^2+\log\frac{1}{\delta}\right). \tag{60}$$

However, although this result establishes step 1, it cannot be directly applied to step 2: our noise robustness result, which states that $h'(x)$ is close to $h(x)$, does not guarantee that $\mathrm{sign}(h'(x))$ agrees well with $\mathrm{sign}(h(x))$. The above lemma implies that $h(x)$ is on the correct side of the origin with high probability, but it could still be very close to the origin, which may lead to a bad noisy classifier $\mathrm{sign}(h'(x))$.

A simple solution to this problem is to show a stronger generalization bound, which in addition to $h(x)$ being correct, also shows that $h(x)$ is bounded away from the origin. We indeed prove such a result, by combining ideas from the aforementioned references. Notice that the only difference between the following lemma and the previous lemma is that we replaced 0 with 0.1.

Lemma 19. *For any hyperplane w satisfying the constraints of the primal program (42), with probability $1-\delta$ over randomly drawn training set S of size m, the generalization error is bounded by*

$$\Pr_{x\sim\mathcal{X}}[y\langle w,x\rangle_E<0.1]\leq\frac{1}{m}\tilde{\mathcal{O}}\left(\|w\|_2^2+\|\xi\|_2^2+\log\frac{1}{\delta}\right). \tag{61}$$

The proof is presented in Section V. Combining Lemma 17 with Lemma 19, we arrive at our main theorem for the learnability of $\mathcal{C}$ with our quantum algorithm.

Theorem 20. *For any concept $f_s\in\mathcal{C}$, the* SVM-QKE *algorithm returns a classifier with test accuracy at least 0.99 in polynomial time, with probability at least $2/3$ over the choice of random training samples and over noise.*

Proof. Below we do not specify the concept, as the proof works equivalently for every concept $f_s\in\mathcal{C}$. Let w^* be the ground truth hyperplane as in Definition 13. Note that $\|w^*\|_2=\mathcal{O}(\Delta^{-1})$. Using Lemma 15, we have that with probability at least 0.99 over the choice of training samples, the $L2$ training loss of w^* satisfies

$$\mathrm{Loss}(w^*):=\frac{1}{2}\|w^*\|_2^2+\frac{\lambda}{2}\|\xi^*\|_2^2\leq\mathcal{O}\left(\frac{1}{\Delta^2}+m\Delta\right). \tag{62}$$

Let w_0 be the optimal solution of the primal program (42), and let $h(x)=\langle w_0,x\rangle_E$. Let h' be the noisy classifier obtained by the dual program (43). By Lemma 17, for any $x\in\mathcal{X}$ we have

$$|yh'(x)-yh(x)|\leq 0.01, \tag{63}$$

with probability at least 0.99 over the choice of training samples and noise. Therefore, by a simple union bound, with probability at least $2/3$, the test error of the noisy classifier is upper bounded by

$$\begin{aligned}\Pr_{x\sim\mathcal{X}}[yh'(x)<0]&\leq\Pr_{x\sim\mathcal{X}}[yh'(x)<0.09]\\&\leq\Pr_{x\sim\mathcal{X}}[y\langle w_0,x\rangle_E<0.1]\\&\leq\frac{1}{m}\tilde{\mathcal{O}}(\mathrm{Loss}(w_0))\\&\leq\frac{1}{m}\tilde{\mathcal{O}}(\mathrm{Loss}(w^*))\leq\tilde{\mathcal{O}}\left(\frac{1}{m\Delta^2}+\Delta\right),\end{aligned} \tag{64}$$

where in the third line we use Lemma 19, and the fourth line is because w_0 is the optimal solution to (42). Finally, notice that the above bound holds for arbitrary $\Delta = \mathcal{O}\left(n^{-t}\right)$ for constant t. In order to optimize this bound, we can choose $t = c/3$ for $m = n^c$ (also see Remark 2). This gives the final bound

$$\Pr_{x\sim\mathcal{X}}[yh'(x) < 0] \leq \tilde{\mathcal{O}}\left(m^{-1/3}\right). \tag{65}$$

Therefore, polynomially many training samples are sufficient for learning the concept class $\mathcal{C}$ with high accuracy. □

V. GENERALIZATION BOUND FOR SOFT MARGIN SVM

In this section we prove Lemma 19, a generalization bound for the L2 soft margin SVM in Eq. (42) (restated below for convenience).

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\|w\|_2^2 + \frac{\lambda}{2}\sum_i \xi_i^2 \\ \text{s.t.} \quad & y_i\langle x_i, w\rangle_E \geq 1 - \xi_i \\ & \xi_i \geq 0. \end{aligned} \tag{66}$$

Let w be an unnormalized feasible solution to Eq. (66). The first step is to use a trick developed by [18], that converts the soft margin problem to a hard margin problem by mapping to a larger space. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be the training set. Consider the mapping

$$\begin{aligned} x &\mapsto \tilde{x} = (x, \delta_x) \\ w &\mapsto \tilde{w} = \left(w, \sum_{(x,y)\in S} y\delta_x \cdot \xi(x, y, w)\right) \end{aligned} \tag{67}$$

where $\delta_x : \mathcal{X} \to \{0, 1\}$ is a function defined as $\delta_x(x') = 1$ if and only if $x' = x$ and $\xi(x, y, w) = \max\{0, 1 - y\langle w, x\rangle_E\}$ are the slack variables used in Eq. (66). Denote the enlarged space by L_X and $\|\cdot\|$ its induced norm. The following useful properties hold for this transform:

1. If $(x, y) \in S$, $y\langle\tilde{w}, \tilde{x}\rangle_E \geq 1$.
2. If $(x, y) \notin S$, $\langle\tilde{w}, \tilde{x}\rangle_E = \langle w, x\rangle_E$.
3. $\|\tilde{w}\|^2 = \|w\|_2^2 + \|\xi\|_2^2$.
4. $\|\tilde{x}\|^2 = 2$.

In the following, we can assume that the training data does not appear when testing the classifier, which is the case with high probability. By property 2, to bound the generalization performance of the hyperplane w, we only need to bound the generalization performance of $\tilde{w}$ in the enlarged space, which corresponds to a hard margin problem.

For the generalization error of hard margin classifiers, it is well-known that the generalization error bound is captured by the VC-dimension which characterizes the complexity of the classifier family. Intuitively, a hard margin classifier corresponds to a "thick" hyperplane in the data space, which reduces its complexity compared with margin-less hyperplanes. The relevant complexity measure in our results is the so-called *fat-shattering dimension* which we define now.

Definition 21 ([18, Definition III.4]). *Let $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathbb{R}\}$ be a set of real-valued functions. We say that a set of points $\hat{X} \subseteq X$ is γ-shattered by $\mathcal{F}$, if there are real numbers r_x indexed by $x \in \hat{X}$ such that for all binary vectors b_x indexed by $x \in \hat{X}$, there is a function $f_b \in \mathcal{F}$ satisfying*

$$f_b(x) \begin{cases} \geq r_x + \gamma, & \text{if } b_x = 1, \\ \leq r_x - \gamma, & \text{otherwise.} \end{cases} \tag{68}$$

The γ-fat-shattering dimension of $\mathcal{F}$, denoted $\mathsf{fat}_{\mathcal{F}}(\gamma)$, is the size of the largest set $\hat{X}$ that is γ-shattered by $\mathcal{F}$, if this is finite or infinity otherwise.

Let $\mathcal{H}$ be a set of linear functions that map from L_X to $\mathbb{R}$, such that their norm equals to $\|\tilde{w}\|$. Since the data vectors $\tilde{x}$ have bounded norm, the fat-shattering dimension of $\mathcal{H}$ was shown [18] to be bounded by

$$\mathsf{fat}_{\mathcal{H}}(\gamma) \leq \mathcal{O}\left(\frac{\|\tilde{w}\|^2}{\gamma^2}\right). \tag{69}$$

Next we invoke the following lemma from Ref. [14] (also see [15]) which uses fat-shattering dimension to understand generalization bounds for learning real-valued concept classes.

Lemma 22 ([14, Corollary 3.3]). *Let $\mathcal{C}, \mathcal{H}$ be sets of functions that map from a set $\mathcal{X}$ to $[0,1]$. Then for all $\eta, \gamma, \delta \in (0,1)$, for every $f \in \mathcal{C}$ and for every probability measure $\mathcal{D}$ on $\mathcal{X}$, with probability at least $1-\delta$ (over the choice of $S = \{x_1, \ldots, x_m\}$ where $x_i \sim \mathcal{D}$), if $h \in \mathcal{H}$ and $|h(x_i) - f(x_i)| \leq \eta$ for $1 \leq i \leq m$, then*

$$\Pr_{x\sim\mathcal{D}}\left[|h(x) - f(x)| \geq \eta + \gamma\right] \leq \frac{1}{m} \cdot \tilde{\mathcal{O}}\left(\mathsf{fat}_{\mathcal{H}}\left(\frac{\gamma}{8}\right) + \log\frac{1}{\delta}\right). \tag{70}$$

To apply this lemma, consider the function $h(\tilde{x}) = \frac{1}{\|\tilde{w}\|}\langle \tilde{w}, \tilde{x}\rangle_E$. Let $\gamma_0 = \frac{1}{\|\tilde{w}\|}$ and $\gamma = 0.9\gamma_0$. Let $y = f(\tilde{x}) \in \{-1, 1\}$ be any labeling rule, and $S = \{(\tilde{x}_1, y_1), \ldots, (\tilde{x}_m, y_m)\}$ be a training set. By the properties of the mapping, for all $\tilde{x} \in S$ we have $yh(\tilde{x}) \geq \gamma_0$, which means that

$$|h(\tilde{x}) - f(\tilde{x})| = |yh(\tilde{x}) - 1| \leq 1 - \gamma_0. \tag{71}$$

Applying Lemma 22, we have with probability at least $1-\delta$,

$$\Pr\left[|h(\tilde{x}) - f(\tilde{x})| \geq 1 - 0.1\gamma_0\right] \leq \frac{1}{m}\tilde{\mathcal{O}}\left(\|\tilde{w}\|^2 + \log\frac{1}{\delta}\right). \tag{72}$$

Finally, note that $yh(\tilde{x}) \leq 0.1\gamma_0$ implies that $|h(\tilde{x}) - f(\tilde{x})| \geq 1 - 0.1\gamma_0$, therefore

$$\Pr\left[yh(\tilde{x}) \leq 0.1\gamma_0\right] \leq \frac{1}{m}\tilde{\mathcal{O}}\left(\|\tilde{w}\|^2 + \log\frac{1}{\delta}\right). \tag{73}$$

This concludes the proof of Lemma 19, as

$$\begin{aligned} \Pr_{x\sim\mathcal{X}}[y\langle w, x\rangle_E < 0.1] &= \Pr_{x\sim\mathcal{X}}[y\langle \tilde{w}, \tilde{x}\rangle_E < 0.1] \\ &= \Pr_{x\sim\mathcal{X}}[yh(\tilde{x}) < 0.1\gamma_0] \\ &\leq \frac{1}{m}\tilde{\mathcal{O}}\left(\|\tilde{w}\|^2 + \log\frac{1}{\delta}\right) = \frac{1}{m}\tilde{\mathcal{O}}\left(\|w\|_2^2 + \|\xi\|_2^2 + \log\frac{1}{\delta}\right). \end{aligned} \tag{74}$$

[1] M. J. Kearns, *The computational complexity of machine learning* (MIT press, 1990).

[2] M. J. Kearns and U. V. Vazirani, *An introduction to computational learning theory* (MIT press, 1994).
[3] M. Blum and S. Micali, SIAM J. Comput. **13**, 850–864 (1984).
[4] P. W. Shor, SIAM Journal on Computing **26**, 1484 (1997).
[5] M. Mosca and C. Zalka, International Journal of Quantum Information **02**, 91 (2004).
[6] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019).
[7] M. Schuld and N. Killoran, Phys. Rev. Lett. **122**, 040504 (2019).
[8] C. J. Burges, Data Mining and Knowledge Discovery **2**, 121 (1998).
[9] A. J. Smola and B. Schölkopf, Statistics and Computing **14**, 199 (2004).
[10] D. Aharonov and A. Ta-Shma, SIAM Journal on Computing **37**, 47 (2007).
[11] C. Gidney and M. Ekerå, How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits (2019), arXiv:1905.09749 [quant-ph].
[12] Note that here the halfspace learning problem is defined in feature space; our original concept class is not a halfspace learning problem. Also, in our case the noise is defined in terms of additive error in the kernel, which is different from the well-studied question of *learning halfspaces with noise* in computational learning theory.
[13] J. W. Daniel, Mathematical Programming **5**, 41 (1973).
[14] M. Anthony and P. L. Bartlett, Combinatorics, Probability and Computing **9**, 213–225 (2000).
[15] P. L. Bartlett and P. M. Long, Journal of Computer and System Sciences **56**, 174 (1998).
[16] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony, IEEE Transactions on Information Theory **44**, 1926 (1998).
[17] P. Bartlett and J. Shawe-Taylor, Generalization performance of support vector machines and other pattern classifiers, in *Advances in Kernel Methods: Support Vector Learning* (MIT Press, Cambridge, MA, USA, 1999) p. 43–54.
[18] J. Shawe-Taylor and N. Cristianini, IEEE Transactions on Information Theory **48**, 2721 (2002).
[19] A. Grønlund, L. Kamma, and K. G. Larsen, Proceedings of International Conference on Machine Learning, ICML (2020).