Quantum Alphatron: quantum advantage for learning with kernels and noise

Siyi Yang[1], Naixu Guo[1], Miklos Santha[1,2], and Patrick Rebentrost[1]

[1]Centre for Quantum Technologies, National University of Singapore, Singapore 117543
[2]Université de Paris, IRIF, CNRS, F-75013 Paris, France; MajuLab UMI 3654

At the interface of machine learning and quantum computing, an important question is what distributions can be learned provably with optimal sample complexities and with quantum-accelerated time complexities. In the classical case, Klivans and Goel discussed the *Alphatron*, an algorithm to learn distributions related to kernelized regression, which they also applied to the learning of two-layer neural networks. In this work, we provide quantum versions of the Alphatron in the fault-tolerant setting. In a well-defined learning model, this quantum algorithm is able to provide a polynomial speedup for a large range of parameters of the underlying concept class. We discuss two types of speedups, one for evaluating the kernel matrix and one for evaluating the gradient in the stochastic gradient descent procedure. We also discuss the quantum advantage in the context of learning of two-layer neural networks. Our work contributes to the study of quantum learning with kernels and from samples.

1 Introduction

Machine learning is highly successful in a variety of applications using heuristic approaches even though the methods being used are often without strong guarantees on their learning performance. Important questions are why common machine learning algorithms such as stochastic gradient descent and kernel methods [28] work well and what is the best way to interpret the results. Computational learning theory addresses some of the fundamental theoretical questions and provides a systematic framework to discuss provable learning of probability distributions and machine learning architectures (such as neural networks). In a variety of settings and architectures, further assumptions on the underlying distribution can rule out hard instances and lead to provable and fast learning algorithms. Such guarantees have been given for generalized linear models, Ising models, and Markov Random Fields [20], for example. The Alphatron developed by Goel and Klivans [15] is a gradient-descent like algorithm for isotonic regression with the inclusion of kernel functions, which provably learns a kernelized, non-linear concept class of functions with a bounded noise term. As a consequence it can be employed to learn two-layer neural networks, where one layer of activation functions feeds into a single activation function.

Quantum machine learning has received a great deal of attention [6, 10], with the hope of gaining advantages for problems such as regression [17, 33, 25] and neural networks (e.g. [1, 9]).

Siyi Yang: e0016915@u.nus.edu
Naixu Guo: naixug@u.nus.edu
Miklos Santha: cqtms@nus.edu.sg
Patrick Rebentrost: cqtfpr@nus.edu.sg

Quantum gradient computation has been considered in [12]. Many algorithms are envisioned for near-term quantum computers [24, 23, 5]. Kernel methods extend linear learning tasks to non-linear learning tasks and, similarly, quantum kernel methods [29, 18] use a high-dimensional Hilbert space of a quantum system to encode features of data. Some algorithms are similar in spirit to the use of heuristic methods in classical machine leaning. They often cannot obtain provable guarantees for the quality of learning and for the run time. An interesting avenue for quantum algorithms for machine learning is therefore to take provable classical algorithms for learning and study provable quantum speedups which retain the guarantees of the classical algorithms [7, 2, 21].

In this work, we provide quantum speedups for the ALPHATRON and its application to non-linear classes of functions and two-layer neural networks. First, we consider the simple idea of pre-computing the kernel matrix used in the algorithm. Our setting is one where the samples are given via quantum query access. Using this access, we can harness quantum subroutines to estimate the entries of the kernel matrices used in the algorithm. The quantum subroutines we use are adaptations of the amplitude estimation algorithm. We show that the learning guarantees can be preserved despite the erroneous estimation of the kernel matrices. In a subsequent step, we also quantize the ALPHATRON algorithm itself. To this end, we require the storage of intermediate values in a quantum-accessible memory [14, 13, 3]. In particular, we show that there are estimations inside the main loop of the algorithm which can be replaced with quantum subroutines, while keeping the main loop intact. We carefully study the regimes where the algorithms allow for a quantum speedup. We are again able to show that the others parameters of the algorithm remain stable under these estimations. Our main result is that we obtain a quantum algorithm for learning the original concept class, where the a quantum speedup is obtained for a large parameter regime of the concept class. We consider a previously defined class of two-layer neural networks and demonstrate that these networks are outside the regime for quantum advantage. We define a different neural network architecture which can exhibit a quantum advantage for learning.

The paper is organized as follows. Section 2 discusses the mathematical preliminaries, the weak p-concept learning setting, and kernel methods, and introduces the Alphatron algorithm with a run time analysis. Section 3 discusses the kernel matrix estimation in the context of the Alphatron using both classical sampling and quantum estimation. Section 4 discusses the main loop of the Alphatron and the corresponding quantum run time. Section 5 summarizes the results in terms of all relevant parameters and discusses the regime where a quantum speedup is obtained. Finally Section 6 considers the application for learning two-layer neural networks, where we give a neural network architecture that may achieve a quantum speedup.

2 Preliminaries and Alphatron algorithm

2.1 Notations

The vectors are denoted by bold-face $\mathbf{x}$, and their elements by x_j. We leave in plain font the α vector (and all other vectors denoted with Greek symbols). The standard vector space of reals and the unit ball of dimension n are denoted by $\mathbb{R}^n$ and $\mathbb{B}^n$, respectively. The ℓ_p-norm of vectors in $\mathbb{R}^n$ is denoted by $\|\cdot\|_p$. Moreover, the max norm is denoted by $\|\mathbf{x}\|_{\max} = \max_i |\mathbf{x}_i|$. We use $\mathbf{a}\cdot\mathbf{b}$ to denote the standard inner product in $\mathbb{R}^n$. We use the notation $\widetilde{\mathcal{O}}()$ to omit any poly-log factors in the arguments. When we write $g + \mathcal{O}(\ldots)$, we mean $g + f$ with some $f \in \mathcal{O}(\ldots)$. We use $a := b$ to define a in terms of b. Given two random variables X and Y, denote by $\mathbb{E}[Y|X]$ the conditional expectation value.

2.2 Cost of arithmetic operations

We use the following arithmetic model for classical computation. We represent the real numbers with a sufficiently large number of bits. We assume that the number of bits is large enough to make the numerical errors negligible in the correctness and run time proofs of the algorithms under consideration. The implication is that we can ignore numerical errors of arithmetic operations (e.g., addition, subtraction, multiplication, and so on) with respect to truncation or rounding. Hence, we assume all real numbers cost $\mathcal{O}(1)$ space and the basic arithmetic operations between them cost $\mathcal{O}(1)$ time. While the accumulated error can be important, dealing with a proper error analysis would require a substantial deviation from the main purpose of this paper.

For the quantum algorithms, we keep track of the amount of (quantum) bits for storing real numbers. We use a standard fixed-point encoding of real numbers.

Definition 1 (Notation for encoding of real numbers)**.** *Let c_1, c_2 be positive integers and $a \in \{0,1\}^{c_1}$ and $b \in \{0,1\}^{c_2}$ be bit strings. Define the (signed) rational number*

$$\mathcal{Q}(a,b,s) := (-1)^s \left(2^{c_1-1}a_{c_1} + \cdots + 2a_2 + a_1 + \frac{1}{2}b_1 + \cdots + \frac{1}{2^{c_2}}b_{c_2}\right) \in [-R, R], \tag{1}$$

where $R := 2^{c_1} - \frac{1}{2^{c_2}}$. For representing numbers in $[0, 2 - \frac{1}{2^c}]$ with positive integer c bits after the decimal point (the case $[0,1]$ being the most frequently used in this work) we use $c_1 = 1$ and $c_2 = c$, and define the short-hand notation

$$\mathcal{Q}(z) := \mathcal{Q}(a,b,0) = a + \frac{1}{2}b_1 + \cdots + \frac{1}{2^c}b_c, \tag{2}$$

where $z := (a,b) \in \{0,1\}^{c+1}$. Given a vector of bit strings $\mathbf{v} \in (\{0,1\}^{c+1})^n$, the notation $\mathcal{Q}(\mathbf{v})$ means the vector whose j-th component is $\mathcal{Q}(v_j)$.

For any real number $r \in [0, 2^{c_1}]$ there exist $a \in \{0,1\}^{c_1}$ and $b \in \{0,1\}^{c_2}$ such that the difference to $\mathcal{Q}(a,b,0)$ is at most $\frac{1}{2^{c_2+1}}$.

2.3 The learning model

We consider the standard "probabilistic concept" (p-concept) learning model (Ref. [19]) in our paper. Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the output space. A *concept class* $\mathcal{C}$ is a class of functions mapping the input space to the output space, i.e., $\mathcal{C} \subseteq \mathcal{Y}^{\mathcal{X}}$. We define here weak learnability with a fixed lower bound for the error, in contrast to the standard definition of p-concept learnability for all $\epsilon_0 > 0$.

Definition 2 (Weak p-concept learnable)**.** *For $\epsilon_0 > 0$, a concept class $\mathcal{C}$ is "weak p-concept learnable up to ϵ_0" if there exists an algorithm $\mathcal{A}$ such that for every $\delta > 0$, $c \in \mathcal{C}$, and distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ with $\mathbb{E}_y[y|\mathbf{x}] = c(\mathbf{x})$ we have that $\mathcal{A}$, given access to samples drawn from $\mathcal{D}$, outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}$, such that with probability at least $1 - \delta$,*

$$\varepsilon(h) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(h(\mathbf{x}) - c(\mathbf{x}))^2\right] \leq \epsilon_0.$$

The quantity $\varepsilon(h)$ is the generalization error of hypothesis h. Moreover, if we have m samples $(\mathbf{x}_i, y_i)$ drawn from the distribution $\mathcal{D}$, we define the empirical error of h as

$$\hat{\varepsilon}(h) := \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - c(\mathbf{x}_i))^2.$$

For convenience, we also define another similar function as

$$\mathrm{err}(h) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(h(\mathbf{x}) - y)^2].$$

Since $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[y] = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\mathbb{E}_y[y|\mathbf{x}]]$, it is easy to see that

$$\begin{aligned} \mathrm{err}(h) - \mathrm{err}(\mathbb{E}_y[y|\mathbf{x}]) &= \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(h(\mathbf{x}) - y)^2] - \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(\mathbb{E}_y[y|\mathbf{x}] - y)^2] \\ &= \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x})^2 - 2\mathbb{E}_y[y|\mathbf{x}]h(\mathbf{x}) + \mathbb{E}_y[y|\mathbf{x}]^2] \\ &= \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x})^2 - 2c(\mathbf{x})h(\mathbf{x}) + c(\mathbf{x})^2] = \varepsilon(h). \end{aligned} \tag{3}$$

Note that $\mathbb{E}_y[y|\mathbf{x}]$ is independent of the choice of h. Hence, for hypotheses h_1 and h_2, we have $\mathrm{err}(h_1) - \mathrm{err}(h_2) = \varepsilon(h_1) - \varepsilon(h_2)$. Thus, we may use err() instead of $\varepsilon()$ for comparing hypothesis. Moreover, by using the empirical version of the $\mathrm{err}(h)$ function, even without knowing the probability distribution $\mathcal{D}$, we are still able to evaluate the quality of the hypothesis h given m samples $(\mathbf{x}_i, y_i) \sim \mathcal{D}$ as

$$\hat{\mathrm{err}}(h) := \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)^2.$$

By the Chernoff bound, we may bound the generalization error err() in terms of the empirical error $\hat{\mathrm{err}}()$ with high probability.

To learn a good hypothesis, on the one hand, we prefer to assume a relatively simple concept class (e.g., a concept class consisting only of linear functions). Then it is easy to design an algorithm for finding the best hypothesis in that class. On the other hand the real-world data distribution is often complicated and cannot be captured by a hypothesis from a simple concept class. The kernel trick is widely used to turn a simple linear concept class and a given learning algorithm into a non-linear concept class and a corresponding learning algorithm, usually without changing too much the algorithm. In the kernel method, we use a more general function to measure the similarity between two vectors instead of the linear inner product. The kernel function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a (usually non-linear) similarity measure on the input space and it is defined via a feature map. Let $\mathcal{V}$ be an arbitrary metric space with inner product $\langle\cdot,\cdot\rangle$. The feature map $\psi : \mathcal{X} \to \mathcal{V}$ maps any input vector into the metric space (also called feature space). For vectors $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, we define $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \langle\psi(\mathbf{x}), \psi(\mathbf{y})\rangle$.

For our purpose, we use the multinomial kernel function to allow the learning of non-linear concepts. Consider formal polynomials over n variables of total degree d. There are $n_d := \sum_{i=0}^{d} n^i$ monomials, which can be uniquely indexed by the tuples $(k_1, \cdots, k_i) \in [n]^i$ for $i \in \{0, \cdots, d\}$. We consider from now on the input space $\mathcal{X} = \mathbb{B}^n \subseteq \mathbb{R}^n$ and the feature space $\mathcal{V} = \mathbb{R}^{n^d}$. We consider the standard Euclidean metric on the feature space $\mathbb{R}^{n_d}$. We define a normalized feature map $\psi_d : \mathbb{B}^n \to \mathbb{R}^{n_d}$ which maps a vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T \in \mathcal{X}$ to a n_d-dimensional vector of monomials of the variables $x_1, x_2, \cdots, x_n$. For $i \in \{0, \cdots, d\}$ and $(k_1, \cdots, k_i) \in [n]^i$, define

$$[\psi_d(\mathbf{x})]_{(k_1,\cdots,k_i)} := \frac{1}{\sqrt{d+1}}\prod_{j=1}^{i}\mathbf{x}_{k_j}. \tag{4}$$

When $i = 0$, we have the empty tuple and the corresponding component is the constant term 1. Note that for $n \geq 2$, we have both the components x_1x_2 and x_2x_1. This redundancy can be avoided by the use of ordered multisets but will not influence our discussion. For $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, we can compute the inner product as

$$\langle\psi_d(\mathbf{x}), \psi_d(\mathbf{y})\rangle = \frac{1}{d+1}\sum_{i=0}^{d}\sum_{1\leq k_1,\cdots,k_i\leq n}\left[\left(\prod_{j=1}^{i}\mathbf{x}_{k_j}\right)\cdot\left(\prod_{j=1}^{i}\mathbf{y}_{k_j}\right)\right] = \frac{1}{d+1}\sum_{i=0}^{d}(\mathbf{x}\cdot\mathbf{y})^i.$$

Observe that by definition, for all $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$, we have that $\langle \psi_d(\mathbf{x}), \psi_d(\mathbf{y}) \rangle \leq 1$. This can be seen from,

$$\max_{\mathbf{x} \in \mathbb{B}^n} \langle \psi_d(\mathbf{x}), \psi_d(\mathbf{x}) \rangle = \frac{1}{d+1} \max_{\mathbf{x} \in \mathbb{B}^n} \sum_{i=0}^{d} \left(\|\mathbf{x}\|_2^2 \right)^i = 1,$$

where we have used that $\max_{\mathbf{x} \in \mathbb{B}^n} \|\mathbf{x}\|_2^2 = 1$ for the unit ball $\mathbb{B}^n$. Throughout this paper, our definition for the normalized multinomial kernel function with degree d is

$$\mathcal{K}_d(\mathbf{x}, \mathbf{y}) := \langle \psi_d(\mathbf{x}), \psi_d(\mathbf{y}) \rangle = \frac{1}{d+1} \sum_{i=0}^{d} (\mathbf{x} \cdot \mathbf{y})^i.$$

With these definitions, let us consider the following concept class.

Definition 3 (Concept class and distribution)**.** *Let $\mathcal{K}_d$ be the normalized multinomial kernel function corresponding to the feature map $\psi_d : \mathbb{B}^n \to \mathbb{R}^{n_d}$, defined as above. Let $L, B, \zeta, \epsilon > 0$. Let $u : \mathbb{R} \to [0,1]$ be a known L-Lipschitz non-decreasing function. Define the concept class $\mathcal{C} \subseteq [0,1]^{\mathbb{B}^n}$ of functions, where for $c \in \mathcal{C}$ there exists a $\boldsymbol{v} \in \mathbb{R}^{n_d}$ with $\|\boldsymbol{v}\|_2 \leq B$ and a $\xi : \mathbb{R}^n \to [-\zeta, \zeta]$ (noise function) with $\mathbb{E}_{\mathbb{B}^n}\left[\xi(\mathbf{x})^2\right] \leq \epsilon$ such that*

$$c(\mathbf{x}) = u(\langle \boldsymbol{v}, \psi_d(\mathbf{x}) \rangle + \xi(\mathbf{x})). \tag{5}$$

Consider a distribution $\mathcal{D}$ on $\mathbb{B}^n \times [0,1]$ for which $c \in \mathcal{C}$ exists such that the distribution satisfies

$$\mathbb{E}_y[y|\mathbf{x}] = c(\mathbf{x}),$$

where $\mathbb{E}_y[y|\mathbf{x}]$ is the conditional expectation value.

The ϵ in this definition motivates Definition 2, as we will see that we cannot learn the concept class for all $\epsilon_0 > 0$. The intrinsic error ϵ will define a lower bound for the error. The learning guarantee for the ALPHATRON and all algorithms in this work is proven for this concept class.

2.4 Alphatron algorithm

We review the classical ALPHATRON algorithm of [15]. Alphatron is the first provably efficient algorithm for learning neural networks with two nonlinear layers without further assumptions on the structure of the neural network. It can also be used for many problems like Boolean Learning and Multiple Instance Learning. The key idea is to combine isotonic regression with kernel methods, where isotonic regression is to find a non-decreasing function for predicting sequences of observations. More explicitly, isotonic regression is to find

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^m} \sum_{i=1}^{m} (x_i - \beta_i)^2 \text{ subject to } \beta_1 \leq \cdots \leq \beta_m,$$

where $x_i \in \mathbb{R}$ are a sequence of m data. The setting considered for the ALPHATRON algorithm includes a kernel function into the problem, as we can see in the following.

The setting is as follows. We split the data into two parts, training set and validation set. The training data set contains $(\mathbf{x}_i, y_i)_{i=1}^{m_1} \in \mathbb{B}^n \times [0,1]$, where m_1 is the size of the training data. The validation data set contains $(\mathbf{a}_i, b_i)_{i=1}^{m_2} \in \mathbb{B}^n \times [0,1]$ of size m_2. Let $m := m_1 + m_2$ be the total size of the data set. Then, since $m_1, m_2 \in \mathcal{O}(m)$ we can use $\mathcal{O}(m)$ as an upper bound of the size of data. In the ALPHATRON algorithm, we first build several hypotheses from the training set.

Then we use the validation set to evaluate each hypothesis and select the optimal one from them.

Algorithm 1: ALPHATRON

1 **Input** *training data* $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$, *testing data* $(\mathbf{a}_i, b_i)_{i=1}^{m_2}$, *function* $u : \mathbb{R} \to [0,1]$, *number of iterations* T, *degree of the multinomial kernel* d, *learning rate* λ.
2 $\alpha^0 \leftarrow \mathbf{0} \in \mathbb{R}^{m_1}$
3 **for** $t \leftarrow 0$ **to** $T-1$ **do**
4 define $h^t(\mathbf{x}) := u\left(\sum_{i=1}^{m_1} \alpha_i^t \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$
5 **for** $i \leftarrow 1$ **to** m_1 **do**
6 $\alpha_i^{t+1} \leftarrow \alpha_i^t + \frac{\lambda}{m_1}(y_i - h^t(\mathbf{x}_i))$
7 **Output** $\alpha^{t_{\text{out}}}$, *where* $t_{\text{out}} = \arg\min_{t \in [T]} \frac{1}{m_2} \sum_{j=1}^{m_2} (h^t(\mathbf{a}_j) - b_j)^2$

The algorithm has a number of iterations T, which will be related to the other input quantities via the learning guarantees. In each iteration, the algorithm generates a new vector α^{t+1} and a new hypothesis h^{t+1} from the old ones. The output of the algorithm is a vector $\alpha^{t_{\text{out}}} \in \mathbb{R}^{m_1}$ describing the hypothesis $h^{t_{\text{out}}} : \mathbb{B}^n \to [0,1]$, which has some p-concept error according to the input data. First, we are interested in the general run time complexity, before we discuss the guarantees for the weak p-concept learning of the specific concept class.

Theorem 1 (Run time of ALPHATRON)**.** *Algorithm 1 has a run time of* $\mathcal{O}(Tm^2(n + \log d))$.

Proof. For computing the multinomial kernel function $\mathcal{K}_d(\mathbf{x}, \mathbf{y})$, we need to first compute the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ in $\mathcal{O}(n)$ time trivially. For all $r \in \mathbb{R} \setminus \{1\}$, since $\sum_{0 \leq i \leq d} r^i = \frac{r^{d+1}-1}{r-1}$, it costs $\mathcal{O}(\log d)$ time to compute the multinomial kernel function from the inner product r. Thus, by the definition of $h^t(\mathbf{x})$ in line 4 of the algorithm, $h^t(\mathbf{x})$ is computed in $\mathcal{O}(m(n+\log d))$ for a given $\mathbf{x}$. In the first part, the training phase, line 6 is executed for $\mathcal{O}(Tm)$ times. Hence, this part costs $\mathcal{O}(Tm^2(n + \log d))$. Similarly, in the second part, the validation phase (line 7), a number $\mathcal{O}(Tm)$ of calls to the function $h^t(\mathbf{a})$ is used. Hence, the algorithm costs $\mathcal{O}(Tm^2(n + \log d))$ in total. □

For obtaining a learning guarantee in the setting from Definition 3, it is supposed that the following relations between the parameters hold.

Definition 4 (Parameter definitions and relations)**.** *Consider the setting in Definition 3, which defines the distribution* $\mathcal{D}$ *and the parameters* (B, L, ζ, ϵ)*, and consider the Algorithm 1, which uses the parameters* (m_1, m_2, T, λ)*. Define the following additional parameters and fix the following relationships between the parameters.*

1. *Equate the* L*-Lipschitz non-decreasing function from the concept class with the function* u *used in the algorithm.*
2. *Learning rate* $\lambda = 1/L$.
3. *Let the training set* $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$ *be sampled iid from* $\mathcal{D}$.
4. *Let* $C > 0$ *be a large enough constant and set* $T = CBL\sqrt{m_1/\log(1/\delta)}$.
5. *Let* $C' > 0$ *be a large enough constant and set* $m_2 = C' m_1 \log(T/\delta)$*, and let the validation set* $(\mathbf{a}_i, b_i)_{i=1}^{m_2}$ *be sampled iid from* $\mathcal{D}$.
6. *Define* $A_2 := L\sqrt{\epsilon} + L\zeta \sqrt[4]{\frac{\log(1/\delta)}{m_1}} + BL\sqrt{\frac{\log(1/\delta)}{m_1}}$ *and let* $C'' > 0$ *be a large enough constant.*

The following learning guarantee was proven in Ref. [15].

Theorem 2 (Learning guarantee of ALPHATRON, same as Ref. [15]). *Given the learning setting in Definition 3 and the parameters defined in Definition 4, Algorithm 1 outputs $\alpha^{t_{\text{out}}}$ which describes the hypothesis $h^{t_{\text{out}}}(\mathbf{x}) := u\left(\sum_{i=1}^{m_1} \alpha_i^{t_{\text{out}}} \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$ such that with probability $1-\delta$,*

$$\varepsilon\left(h^{t_{\text{out}}}\right) \leq C'' A_2.$$

Next, we discuss a regime where Theorem 2 achieves p-concept learnability. This result was implicit in Ref. [15].

Theorem 3 (P-concept learnability via the ALPHATRON). *Let $m_1' := \frac{16\zeta^4}{\epsilon^2}\log(1/\delta)$ and $m_1'' := \frac{4B^2}{\epsilon}\log(1/\delta)$. If $m_1 \geq \max\{m_1', m_1''\}$, then the concept class in Definition 3 is weak p-concept learnable up to $2C''L\sqrt{\epsilon}$ by the ALPHATRON algorithm.*

Proof. Recall that weak p-concept learnability up to ϵ_0 means that

$$\epsilon(h^{t_{\text{out}}}) \leq \epsilon_0, \tag{6}$$

with probability $1-\delta$. Hence, we desire that $C''A_2 \leq \epsilon_0$. Note the trivial case when $\epsilon_0 < C''L\sqrt{\epsilon}$, which means that the intrinsic error of the concept class is too large, and we fail to achieve learnability. Hence, we can only prove the case when $\epsilon_0 \geq C''L\sqrt{\epsilon}$, and we prove the theorem only for $\epsilon_0 = 2C''L\sqrt{\epsilon}$, where we use a factor 2 to leave room for the other terms in A_2. It follows that we would like to set m_1 such that

$$C''\left(L\zeta\sqrt[4]{\frac{\log(1/\delta)}{m_1}} + BL\sqrt{\frac{\log(1/\delta)}{m_1}}\right) \leq \frac{1}{2}\epsilon_0. \tag{7}$$

Here, Eq. (7) can be achieved by making each term smaller than $\epsilon_0/4$ (alternatively, we can solve a quadratic equation, which leads to more complicated equations). This means that both of following statements have to be true:

$$m_1 \geq \frac{256C''^{,4}L^4\zeta^4}{\epsilon_0^4}\log(1/\delta), \tag{8}$$

$$m_1 \geq \frac{16C''^{,2}B^2L^2}{\epsilon_0^2}\log(1/\delta). \tag{9}$$

Hence, we take m_1 greater than the maximum of the right-hand side expressions. Employing the lower-bound $2C''L\sqrt{\epsilon} \leq \epsilon_0$, we find that $m_1 \geq \max\{m_1', m_1''\}$ leads to weak p-concept learnability up to $2C''L\sqrt{\epsilon}$. □

3 Pre-computation and approximation of the kernel matrix

One bottleneck of the ALPHATRON algorithm is the repeated inner product computation when evaluating the function $h^t(\mathbf{x})$. In Algorithm 1, at every step t out of T steps, we need to evaluate $\mathcal{O}(m^2)$ inner products for the kernel function. This evaluation is redundant because the inner products do not change for different t. A simple pre-computing idea helps to reduce the time complexity to some extent. We improve Algorithm 1 as follows. Given input data $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$ and $(\mathbf{a}_i, b_i)_{i=1}^{m_2}$ and d, we define two matrices

$$K_{ij} := \mathcal{K}_d(\mathbf{x}_i, \mathbf{x}_j),$$

and

$$K'_{ij} := \mathcal{K}_d(\mathbf{a}_i, \mathbf{x}_j).$$

If these two matrices are given by an oracle, then we are able to rewrite Algorithm 1 as below. Algorithm 2 will be used as a subroutine several times in the remainder of this paper.

Algorithm 2: ALPHATRON_WITH_KERNEL

Input *Function* $u : \mathbb{R} \to [0,1]$*, number of iterations* T*, learning rate* λ*, query access to* K_{ij} *and* K'_{ij}
$\alpha^0 \leftarrow \mathbf{0} \in \mathbb{R}^{m_1}$
for $t \leftarrow 0$ **to** $T-1$ **do**
  for $i \leftarrow 1$ **to** m_1 **do**
    $\alpha_i^{t+1} \leftarrow \alpha_i^t + \frac{\lambda}{m_1}\left(y_i - u\left(\sum_{j=1}^{m_1} \alpha_i^t \cdot K_{ij}\right)\right)$
Output $\alpha^{t_{\text{out}}}$*, where* $t_{\text{out}} = \arg\min_{t\in[T]} \frac{1}{m_2}\sum_{i=1}^{m_2}\left(u\left(\sum_{j=1}^{m_1} \alpha_i^t \cdot K'_{ij}\right) - b_i\right)^2$

With equivalent input, Algorithm 2 produces the same output as Algorithm 1, which can be easily checked as follows. Fix the input for Algorithm 1. From these fixed training examples compute the kernel matrices $K_{ij} = \mathcal{K}_d(\mathbf{x}_i, \mathbf{x}_j)$ and $K'_{ij} = \mathcal{K}_d(\mathbf{a}_i, \mathbf{x}_j)$. Use these matrices and the other inputs of Algorithm 1 to fix the input of Algorithm 2. The sequences $(\alpha^t_{\text{Alg1}})_{t\in[T]}$ and $(\alpha^t_{\text{Alg2}})_{t\in[T]}$ of both algorithms are the same and hence for the output it holds that

$$\alpha^{t_{\text{out,Alg1}}}_{\text{Alg1}} = \alpha^{t_{\text{out,Alg2}}}_{\text{Alg2}}.$$

Note that even if we do not explicitly define the hypothesis h^t in Algorithm 2, in the analysis, we still use the same notation h^t for the t-th generated hypothesis as in Algorithm 1.

Theorem 4 (ALPHATRON_WITH_KERNEL)**.** *Algorithm 2 runs in time* $\mathcal{O}(Tm^2)$*.*

Proof. Since each entry of the matrices K and K' is accessible in $\mathcal{O}(1)$ time, the run time of the algorithm is $\mathcal{O}(Tm^2)$. □

We now discuss the pre-computation, i.e., we prepare the matrices K_{ij} and K'_{ij} by evaluating the kernel function for the training and testing data. We present the following algorithm, which performs the pre-computation and then runs Algorithm 2. On the same input, this algorithm produces exactly the same output as Algorithm 1.

Algorithm 3: ALPHATRON_WITH_PRE

Input *training data* $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$*, testing data* $(\mathbf{a}_i, b_i)_{i=1}^{m_2}$*, function* $u : \mathbb{R} \to [0,1]$*, number of iterations* T*, degree of the multinomial kernel* d*, learning rate* λ*.*
for $i \leftarrow 1$ **to** m_1 **do**
  for $j \leftarrow 1$ **to** m_1 **do**
    $K_{ij} \leftarrow \mathcal{K}_d(\mathbf{x}_i, \mathbf{x}_j)$
for $i \leftarrow 1$ **to** m_2 **do**
  for $j \leftarrow 1$ **to** m_1 **do**
    $K'_{ij} \leftarrow \mathcal{K}_d(\mathbf{a}_i, \mathbf{x}_j)$
$\alpha^{t_{out}} \leftarrow$ Run ALPHATRON_WITH_KERNEL (Algorithm 2) with all inputs as above and K_{ij} and K'_{ij}.
Output $\alpha^{t_{out}}$

Theorem 5 (ALPHATRON_WITH_PRE)**.** *Algorithm 3 generates the same output as Algorithm 1, and runs in time* $\mathcal{O}(m^2(n + \log d) + Tm^2)$*.*

Proof. First of all, it is straightforward to see that Algorithm 3 behaves in the same way as Algorithm 1, by using the definition $h^t(\mathbf{x}) = u\left(\sum_{i=1}^{m_1} \alpha_i^t \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$ and noticing that the sequences $\left(\alpha_{\text{Alg1}}^t\right)_{t\in[T]}$ and $\left(\alpha_{\text{Alg3}}^t\right)_{t\in[T]}$ are exactly the same.

For the time complexity, we have $\mathcal{O}\left(m^2\right)$ inner products to be evaluated. For each of them we need time $\mathcal{O}\left(n + \log d\right)$ as we showed in the proof of Theorem 1. Hence it costs $\mathcal{O}(m^2(n + \log d))$ time to pre-compute the results of all $\mathcal{K}_d(\mathbf{x}, \mathbf{y})$. By Theorem 4, the time complexity of ALPHATRON_WITH_KERNEL is $\mathcal{O}(Tm^2)$. In total the algorithm runs in time $\mathcal{O}(m^2(n + \log d) + Tm^2)$. □

By the pre-computation, we evaluate each kernel function only once with the corresponding memory cost of storing the values. Comparing with the $\mathcal{O}\left(Tm^2(n + \log d)\right)$ time used for Algorithm 1, Algorithm 3 achieves a significant speedup.

3.1 Classical inner-product estimation

We can hope to obtain a further speedup by approximating the inner products instead of computing them exactly. Next, we discuss this inner product approximation, where the approximations rely on sampling data structures, which are discussed in Appendix C. These data structures when given a vector allow to sample an index with probability proportional to the components of the vector, as described in Facts 1 and 2. We call them ℓ_1 and ℓ_2 sampling data structures. Here, we use the ℓ_2 case (Fact 2), while the second part of this work uses the ℓ_1 case. Based on these data structures, elementary results can be provided to estimate inner products between two vectors. These are described in Lemmas 9 and 10 in Appendix C, of which we need Lemma 10 here. Our version of the Alphatron algorithm with approximate pre-computation is given in Algorithm 4. We use the inner product estimation of Lemma 10 to improve the run time complexity of Algorithm 3.

Algorithm 4: ALPHATRON_WITH_APPROX_PRE

Input *training data* $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$, *testing data* $(\mathbf{a}_i, b_i)_{i=1}^{m_2}$, *error tolerance parameter* ϵ_K, *failure probability* δ_K, *function* $u : \mathbb{R} \to [0, 1]$, *number of iterations* T, *degree of the multinomial kernel* d, *learning rate* λ

for $i \leftarrow 1$ **to** m_1 **do**
  Prepare sampling data structure for $\mathbf{x}_i$ according to Fact 2.
  for $j \leftarrow 1$ **to** m_1 **do**
    $z_{ij} \leftarrow$ Estimate the inner product $\mathbf{x}_i \cdot \mathbf{x}_j$ to $\epsilon_K/(3d)$ additive error with probability at least $1 - \delta_K/(m_1^2 + m_1 m_2)$ via Lemma 10.
    $\widetilde{K}_{ij} \leftarrow \frac{1}{d+1} \sum_{0\le k\le d} z_{ij}^k$

for $i \leftarrow 1$ **to** m_2 **do**
  Prepare sampling data structure for $\mathbf{a}_i$ according to Fact 2.
  for $j \leftarrow 1$ **to** m_1 **do**
    $z'_{ij} \leftarrow$ Estimate the inner product $\mathbf{a}_i \cdot \mathbf{x}_j$ to $\epsilon_K/(3d)$ additive error with probability at least $1 - \delta_K/(m_1^2 + m_1 m_2)$ via Lemma 10.
    $\widetilde{K}'_{ij} \leftarrow \frac{1}{d+1} \sum_{0\le k\le d} (z'_{ij})^k$

$\alpha^{t_{out}} \leftarrow$ Call ALPHATRON_WITH_KERNEL (Algorithm 2) with all input as above and $\widetilde{K}_{ij}$ and $\widetilde{K}'_{ij}$.

Output $\alpha^{t_{out}}$

Theorem 6 (Run time of ALPHATRON_WITH_APPROX_PRE)**.** *Let $\epsilon_K, \delta_K > 0$. Assume that for all $i \in [m_1]$ and $j \in [m_2]$, $\|\mathbf{x}_i\|_2 = \|\mathbf{a}_j\|_2 = 1$. Lines $2-11$ of Algorithm 4 have a run time of*

$$\widetilde{\mathcal{O}}\left(mn + \frac{m^2 d^2}{\epsilon_K^2} \log \frac{1}{\delta_K}\right),$$

and provide the kernel matrices $\widetilde{K}$ and $\widetilde{K'}$ such that $\max_{ij} \left|\widetilde{K}_{ij} - K_{ij}\right| \leq \epsilon_K$ and $\max_{ij} \left|\widetilde{K'}_{ij} - K'_{ij}\right| \leq \epsilon_K$ with success probability $1 - \delta_K$. Line 12 requires an additional cost of $\mathcal{O}\left(Tm^2\right)$ from the use of Algorithm 2.

Proof. For all vectors $\mathbf{x}_i$ and $\mathbf{a}_j$, the sampling data structure is prepared in total time $\widetilde{\mathcal{O}}\left(mn\right)$. There are $\mathcal{O}\left(m^2\right)$ inner products to be estimated between these vectors. Hence, by Lemma 10, each estimation of inner product with additive accuracy $\epsilon_K/(3d)$ and success probability $1 - \delta_K/(m_1^2 + m_1 m_2)$ costs $\widetilde{\mathcal{O}}\left(\frac{d^2}{\epsilon_K^2} \log \frac{m_1^2 + m_1 m_2}{\delta_K}\right)$ because of $\|\mathbf{x}_i\|_2 = \|\mathbf{a}_j\|_2 = 1$. We ignore the $\log(m_1^2 + m_1 m_2)$ factor under the tilde notation compared to m^2. Again, $\mathcal{O}\left(\log d\right)$ extra time is needed to compute each multinomial kernel function $\mathcal{K}_d$ from the inner product. However, we also ignore the $\log d$ factor under the tilde notation. By Lemma 3 of the Appendix, the Lipschitz constant for $f(z) = \frac{1}{d+1} \sum_{i=0}^{d} z^i$ is bounded from above by $3d$, when $z \in [-1, 1]$. Hence, we obtain $\max_{ij} \left|\tilde{K}_{ij} - K_{ij}\right| \leq (3d) \cdot \epsilon_K/(3d)$ and $\max_{ij} \left|\tilde{K}'_{ij} - K'_{ij}\right| \leq (3d) \cdot \epsilon_K/(3d)$. The last step for calling Algorithm 2 costs $\mathcal{O}\left(Tm^2\right)$ again as the matrices are accessible in $\mathcal{O}\left(1\right)$. □

Since only m sampling state structures are prepared which allow the inner products to be approximated in advance, Algorithm 4 improves the run time complexity of the Algorithm 3. However, as the inner products are approximated, we may lose the correctness of Algorithm 3. In Ref. [15], a theoretical upper bound is proven for the sample complexity of Algorithm 1 in the problem setting of Definition 3. We now show that with approximate pre-computation, under the same problem and parameter settings as in Ref. [15], the p-concept error of the output hypothesis does not increase too much.

Theorem 7 (Correctness of ALPHATRON_WITH_APPROX_PRE)**.** *If Definition 3 and 4 hold, then by setting $\delta_K = \delta$, with probability $1 - 3\delta$, Algorithm 4 outputs $\alpha^{t_{\text{out}}}$ which describes the hypothesis $h^{t_{\text{out}}}(\mathbf{x}) := u\left(\sum_{i=1}^{m_1} \alpha_i^{t_{\text{out}}} \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$ such that,*

$$\varepsilon(h) \in \mathcal{O}\left(A_2 + \epsilon_K^2 T^2 + \epsilon_K T\right),$$

where $A_2 = L\sqrt{\epsilon} + L\zeta \sqrt[4]{\frac{\log(1/\delta)}{m_1}} + BL\sqrt{\frac{\log(1/\delta)}{m_1}}$.

We prove this theorem in Appendix B.

3.2 Quantum Pre-computation

In the previous subsection, we classically estimate the inner products that construct the kernel matrices. Now, given quantum access to the training data, we replace this estimation with a quantum subroutine and obtain a quantum speedup. This section presents the quantum algorithm for pre-computing the kernel matrices used in the Alphatron. We assume quantum access to the training data, which includes classical access and also superposition queries to the data.

Definition 5 (Quantum query access)**.** *Let c and n be two positive integers and $\mathbf{u}$ be a vector of bit strings $\mathbf{u} \in (\{0,1\}^c)^n$. Define element-wise quantum access to $\mathbf{u}$ for $j \in [n]$ by the operation*

$$|j\rangle |0^c\rangle \to |j\rangle |u_j\rangle , \tag{10}$$

on $\mathcal{O}(c + \log n)$ qubits. We denote this access by $\mathbf{QA}(\mathbf{u}, n, c)$.

Data Input 1. *For $k \in [n]$, $i \in [m_1]$, and $j \in [m_2]$, let $\mathbf{x}_i$ and $\mathbf{a}_j$ be the input vectors with $\|\mathbf{x}_i\|_2 = \|\mathbf{a}_j\|_2 = 1$, and let $\mathbf{x}_{ik}$ and $\mathbf{a}_{jk}$ be the entries of the vectors. Assume $c = \mathcal{O}(1)$ bits are sufficient to store $\mathbf{x}_{ik}$ and $\mathbf{a}_{jk}$. Assume that we are given $\mathbf{QA}(\mathbf{x}_i, n, c)$ for each $i \in [m_1]$ and $\mathbf{QA}(\mathbf{a}_j, n, c)$ for each $j \in [m_2]$.*

Our first quantum algorithm is constructed in a straightforward manner. We replace the classical approximation of the kernel matrix inner products with a quantum estimation. For the quantum estimation of inner products refer to Lemma 12 in Appendix D, which requires quantum query access similar to Data Input 1. The run time of Lemma 12 depends on the ℓ_2-norms of the input vectors, which here are 1. The result is Algorithm 5. The run time analysis and the guarantees for the output hypothesis are similar to the classical algorithm. We state them below as a corollary.

Algorithm 5: ALPHATRON_WITH_Q_PRE

1 **Input** *Quantum access to training data $(\mathbf{x}_i, y_i)_{i=1}^{m}$ and testing data $(\mathbf{a}_i, b_i)_{i=1}^{N}$ according to Data Input 1, error tolerance parameter ϵ_K, failure probability δ_K, function $u : \mathbb{R} \to [0,1]$, number of iterations T, degree of the multinomial kernel d, learning rate λ*

2 **for** $i \leftarrow 1$ **to** m_1 **do**

3 **for** $j \leftarrow 1$ **to** m_1 **do**

4 $z_{ij} \leftarrow$ Estimate the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ to $\epsilon_K/(3d)$ additive error with probability at least $1 - \delta_K/(m_1^2 + m_1 m_2)$ via Lemma 12.

5 $\widetilde{K}_{ij} \leftarrow \frac{1}{d+1} \sum_{0 \le k \le d} z_{ij}^k$

6 **for** $i \leftarrow 1$ **to** m_2 **do**

7 **for** $j \leftarrow 1$ **to** m_1 **do**

8 $z'_{ij} \leftarrow$ Estimate the inner product $\langle \mathbf{a}_i, \mathbf{x}_j \rangle$ to $\epsilon_K/(3d)$ additive error with probability at least $1 - \delta_K/(m_1^2 + m_1 m_2)$ via Lemma 12.

9 $\widetilde{K}'_{ij} \leftarrow \frac{1}{d+1} \sum_{0 \le k \le d} (z'_{ij})^k$

10 $\alpha^{t_{out}} \leftarrow$ Call ALPHATRON_WITH_KERNEL (Algorithm 2) with all inputs as above and $\widetilde{K}_{ij}$ and $\widetilde{K}'_{ij}$.

11 **Output** $\alpha^{t_{out}}$

Corollary 1 (Runtime of ALPHATRON_WITH_Q_PRE)**.** *Let $\epsilon_K, \delta_K > 0$. Assume that for all $i \in [m_1]$ and $j \in [m_2]$, we have quantum query access to the vectors $\mathbf{x}_i$ and $\mathbf{a}_j$ via Data Input 1. Lines $2 - 9$ of Algorithm 5 have a run time of*

$$\widetilde{\mathcal{O}}\left(\frac{m^2 d\sqrt{n}}{\epsilon_K} \log \frac{1}{\delta_K}\right)$$

and provide the kernel matrices $\tilde{K}$ and $\tilde{K}'$ such that $\max_{ij} |\tilde{K}_{ij} - K_{ij}| \le \epsilon_K$ and $\max_{ij} |\tilde{K}'_{ij} - K'_{ij}| \le \epsilon_K$ with success probability $1 - \delta_K$.

Proof. For $\epsilon_K \in (0,1)$, the run time of each invocation of Lemma 12 is $\widetilde{\mathcal{O}}\left(\frac{d\sqrt{n}}{\epsilon_K}\log\left(\frac{m}{\delta_K}\right)\right)$, using that the input vectors are in the unit ball. All probabilistic steps in Lines $2-9$ of the algorithm succeed with probability $1-\delta_K$ using a union bound. □

Corollary 2 (Guarantee for ALPHATRON_WITH_Q_PRE)**.** *Let $\delta > 0$. Assume that for all $i \in [m_1]$ and $j \in [m_2]$, we have quantum query access to the vectors $\mathbf{x}_i$ and $\mathbf{a}_j$ via Data Input 1. Let Definitions 3 and 4 hold. If $A_2 \leq 1$, and we set $\epsilon_K = \frac{L\sqrt{\epsilon}}{T}$ and $\delta_K = \delta$, then Algorithm 5 with probability at least $1-3\delta$ outputs $\alpha^{t_{\text{out}}}$ which describes the hypothesis $h^{t_{\text{out}}}(\mathbf{x}) := u\left(\sum_{i=1}^{m_1}\alpha_i^{t_{\text{out}}}\mathcal{K}_d(\mathbf{x},\mathbf{x}_i)\right)$ such that*

$$\varepsilon\left(h^{t_{\text{out}}}\right) \in \mathcal{O}\left(A_2\right),$$

with a run time of

$$\widetilde{\mathcal{O}}\left(\frac{m^2 T d\sqrt{n}}{L\sqrt{\epsilon}}\log\frac{1}{\delta} + Tm^2\right).$$

Proof. The proof is analogous to the proof of Theorem 7, where we use Corollary 1 for the run time of the inner product estimation. □

4 Quantum Alphatron

Up to this point, we have been discussing improvements in the pre-computation step of the Alphatron. We always use the same ALPHATRON_WITH_KERNEL algorithm once we prepare the kernel matrices K and K'. If data dimension n is much larger than the other parameters, the quantum pre-computation costs asymptotically more time than ALPHATRON_WITH_KERNEL. Hence, we do not benefit much from optimizing ALPHATRON_WITH_KERNEL if the cost of preparing the data of size n is taken into account.

However, if we assume that the pre-computation was already done for us, it makes sense to discuss quantum speedups for ALPHATRON_WITH_KERNEL, which is what the remainder of this work is about. In other words, we consider the following scenario.

Data Input 2. *Let there be given two training data sets $(\mathbf{x}_i, y_i)_{i=1}^{m_1} \in \mathbb{B}^n \times [0,1]$ and $(\mathbf{a}_i, b_i)_{i=1}^{m_2} \in \mathbb{B}^n \times [0,1]$, which define the kernel matrices $K_{ij} := \mathcal{K}_d(\mathbf{x}_i, \mathbf{x}_j)$ and $K'_{ij} := \mathcal{K}_d(\mathbf{a}_i, \mathbf{x}_j)$. Let each entry K_{ji} and K'_{ji} be specified by $\mathcal{O}(1)$ bits. We assume that we have query access to each entry in $\mathcal{O}(1)$.*

The bottleneck of the computation in the ALPHATRON_WITH_KERNEL is the cost of about $\mathcal{O}(Tm)$ for the inner product evaluations. By the sampling techniques and quantum estimation, we may speed them up.

4.1 Main loop with approximated inner products

We employ the classical sampling of inner products in the ALPHATRON_WITH_KERNEL algorithm. The result is Algorithm 6. For the kernel matrices K_{ji} and K'_{ji}, define $K_{\max}$ as an upper bound for $|K_{ji}|$ and $|K'_{ji}|$.

Algorithm 6: ALPHATRON_WITH_KERNEL_AND_SAMPLING

Input *training data* $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$, *testing data* $(\mathbf{a}_i, b_i)_{i=1}^{m_2}$, *error parameter* ϵ_I *and failure probability* δ, *function* $u : \mathbb{R} \to [0,1]$, *number of iterations* T, *degree of the multinomial kernel* d, *learning rate* λ, *query access to* K_{ji} *and* K'_{ji}, *the upper bound* $K_{\max}$ *for both* $|K_{ij}|$ *and* $|K'_{ij}|$

$\alpha^0 \leftarrow \mathbf{0} \in \mathbb{R}^{m_1}$

for $t \leftarrow 0$ **to** $T-1$ **do**

  Prepare sampling data structure for α^t via Fact 1

  for $j \leftarrow 1$ **to** m_1 **do**

    Define K_j as the vector $(K_{j1}, K_{j2}, \cdots, K_{jm_1})$

    $r_j^t \leftarrow$ Estimate inner product $\alpha^t \cdot K_j$ to additive accuracy ϵ_I with success probability $1 - \delta/(2Tm_1)$ via Lemma 9

    $\alpha_j^{t+1} \leftarrow \alpha_j^t + \frac{\lambda}{m_1}(y_j - u(r_j^t))$

  for $j \leftarrow 1$ **to** m_2 **do**

    Define K'_j as the vector $(K'_{j1}, K'_{j2}, \cdots, K'_{jm_1})$

    $s_j^t \leftarrow$ Estimate inner product $\alpha^t \cdot K'_j$ to additive accuracy ϵ_I with success probability $1 - \delta/(2Tm_2)$ via Lemma 9

$t_{\text{out}} \leftarrow \arg\min_{t\in[T]} \frac{1}{m_2} \sum_{j=1}^{m_2} (u(s_j^t) - b_j)^2$

Output $\alpha^{t_{\text{out}}}$

Theorem 8. *We assume query access Data Input 2 to the kernel matrices* K *and* K' *with known* $K_{\max}$. *Let* $\epsilon_I, \delta \in (0,1)$. *If the Definitions 3 and 4 hold, the Algorithm 6 outputs* $\alpha^{t_{\text{out}}}$ *which describes the hypothesis* $h^{t_{\text{out}}}(\mathbf{x}) := u\left(\sum_{i=1}^{m_1} \alpha_i^{t_{\text{out}}} \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$ *such that with probability* $1 - 3\delta$,

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}\left(A_2 + L\epsilon_I + L^2\epsilon_I^2\right),$$

where A_2 *is defined in Definition 4. The run time of this algorithm is* $\widetilde{\mathcal{O}}\left(Tm + T^3 m \frac{K_{\max}^2}{L^2\epsilon_I^2} \log\left(\frac{1}{\delta}\right)\right)$. *Moreover, if* $A_2 \leq 1$, *and we set* $\epsilon_I = \sqrt{\epsilon}$, *then we obtain the guarantee*

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}\left(A_2\right),$$

and have a run time of $\widetilde{\mathcal{O}}\left(Tm + T^3 m \frac{K_{\max}^2}{L^2\epsilon} \log\left(\frac{1}{\delta}\right)\right)$.

Proof. By the definition of r_j^t, we have $|r_j^t - \sum_{i=1}^{m_1} \alpha_i^t K_{ji}| \leq \epsilon_I$, and by the definition of s_j^t, we have $|s_j^t - \sum_{i=1}^{m_1} \alpha_i^t K'_{ji}| \leq \epsilon_I$. By Definition 10 and by the Lipschitz condition of u, we obtain that $\left|u(r_j^t) - g(\alpha^t, K, j)\right| \leq L\epsilon_I$, and $\left|u(s_j^t) - g(\alpha^t, K', j)\right| \leq L\epsilon_I$.

Consider the cases in Eqns. (61) and (62) in the proof of Theorem 7 for the sequence of $\tilde{\alpha}^t$ generated by Algorithm 6. Similarily, there exists t^* such that Case 2 holds. Then by Lemma 7 with $\omega = \alpha^{t^*}$, we obtain

$$\|\mathbf{v}(\tilde{\alpha}^{t^*}) - \boldsymbol{v}\|_2^2 - \|\mathbf{v}(\tilde{\alpha}^{t^*+1}) - \boldsymbol{v}\|_2^2 \geq \frac{1}{L^2}\hat{\varepsilon}(h^{t^*}) - A_1 - \epsilon_I^2. \tag{11}$$

Hence it now holds that

$$\frac{B\eta}{L} \geq \frac{1}{L^2}\hat{\varepsilon}(h^{t^*}) - A_1 - \epsilon_I^2, \tag{12}$$

which implies that

$$\hat{\varepsilon}(h^{t^*}) \leq BL\eta + L^2 A_1 + L^2\epsilon_I^2. \tag{13}$$

Using the known bound for η we have

$$\hat{\varepsilon}(h^{t^*}) \in \mathcal{O}\left(A_2 + L^2\epsilon_I^2\right). \tag{14}$$

Again, by the Rademacher analysis in proof of Theorem 7, we obtain

$$\varepsilon(h^{t^*}) \leq \hat{\varepsilon}(h^{t^*}) + \mathcal{O}\left(BL\sqrt{\frac{1}{m_1}} + \sqrt{\frac{\log(1/\delta)}{m_1}}\right) \in \mathcal{O}\left(A_2 + L^2\epsilon_I^2\right). \tag{15}$$

We define $t' := \arg\min_{t\in[T]} \varepsilon(h^t)$. Then $\varepsilon(h^{t'}) \leq \varepsilon(h^{t^*})$. By Lemma 8 with $\Gamma_j^t = u(s_j^t)$, at Line 13 in Algorithm 6, we obtain $h^{\tilde{t}}$ such that

$$|\hat{\text{err}}(h^{\tilde{t}}) - \hat{\text{err}}(h^{t'})| \leq L\epsilon_I. \tag{16}$$

As in the proof of the Theorem 7, by Chernoff bound, setting $m_2 \in \mathcal{O}(m_1 \log(T/\delta))$, with probability $1-\delta$, we have

$$\forall t \in [T], |\text{err}(h^t) - \hat{\text{err}}(h^t)| \in \mathcal{O}(A_2). \tag{17}$$

Using the same idea as in the last part of the proof of the Theorem 7, we relate above inequalities (15), (16), and (17), and obtain that for the output hypothesis $h^{\tilde{t}}$,

$$\varepsilon\left(h^{\tilde{t}}\right) \in \mathcal{O}\left(A_2 + L\epsilon_I + L^2\epsilon_I^2\right). \tag{18}$$

For the run time complexity, the total time of preparing the sampling data structure for α^t is $\widetilde{\mathcal{O}}(Tm)$ because we prepare $\mathcal{O}(T)$ such structures and preparing each of them costs $\widetilde{\mathcal{O}}(m)$. By Lemma 5, we have the upper bound $\max_t \|\alpha^t\|_1 \leq \frac{T}{L}$. Hence, the run time of each estimation r_j^t and s_j^t via Lemma 9 is bounded by $\widetilde{\mathcal{O}}\left(\frac{T^2K_{\max}^2}{L^2\epsilon_I^2}\log 1/\delta\right)$. Then the total run time is

$$\widetilde{\mathcal{O}}\left(Tm + T^3 m\frac{K_{\max}^2}{L^2\epsilon_I^2}\log\left(\frac{1}{\delta}\right)\right). \tag{19}$$

Setting $\epsilon_I = \sqrt{\epsilon}$ obtains $\epsilon(h) \in \mathcal{O}(A_2)$ with the run stated in the theorem. □

Now, replace the classical sampling of the inner product with the quantum estimation of the inner product. With the Lemma 13 in Appendix D, we can remove the explicit dimension dependence of the inner product estimation, at the expense of using a QRAM, see Definition 7 in the next section.

4.2 Quantum speedup for the main loop

For the quantum algorithm, we assume the quantum query access to the kernel matrices K and K. Note the definition of quantum query access in Definition 5 in Appendix D.

Data Input 3. *Assume Data Input 2 for the training data and the kernel matrices. For all $j \in [m_1]$, define K_j as the vector $(K_{j1}, K_{j2}, \cdots, K_{jm_1})$, and for all $j \in [m_2]$, define K'_j as the vector $(K'_{j1}, K'_{j2}, \cdots, K'_{jm_1})$. Assume the availability of the quantum access $\mathbf{QA}(K_j, m_1, \mathcal{O}(1))$, for all $j \in [m_1]$, and the quantum access $\mathbf{QA}(K'_j, m_1, \mathcal{O}(1))$, for all $j \in [m_2]$.*

Based on this input a simple circuit prepares query access to the non-negative versions of the vectors.

Lemma 1. *Assume Data Input 2 and define the non-negative vectors* $(K_j)^+$, $(K_j)^-$, *with* $K_j = (K_j)^+ - (K_j)^-$ *and the non-negative vectors* $(K'_j)^+$, $(K'_j)^-$, *with* $K'_j = (K'_j)^+ - (K'_j)^-$. *Given Data Input 3, then query accesses* $\mathbf{QA}((K_j)^+, m_1, \mathcal{O}(1)), \mathbf{QA}((K_j)^-, m_1, \mathcal{O}(1)), \forall j \in [m_1]$ *and query accesses* $\mathbf{QA}((K'_j)^+, m_1, \mathcal{O}(1)), \mathbf{QA}((K'_j)^-, m_1, \mathcal{O}(1)), \forall j \in [m_2]$ *can be provided with two queries to the respective inputs and a constant depth circuit of quantum gates.*

For our quantum version for the main loop of the ALPHATRON algorithm, we will also require a dynamic quantum data structure for the α vector which allows us to obtain efficient quantum sample access.

Definition 6 (Quantum sample access)**.** *Let* c_1, c_2, *and* n *be positive integers and* $\mathbf{v}' \in (\{0,1\}^{c_1})^n$ *and* $\mathbf{v}'' \in (\{0,1\}^{c_2})^n$ *be vectors of bit strings. Define quantum sample access to a vector* $\mathbf{v}$ *via the operation*

$$|\bar{0}\rangle \to \frac{1}{\sqrt{\|\mathcal{Q}(\mathbf{v}', \mathbf{v}'')\|_1}} \sum_{j=1}^{n} \sqrt{\mathcal{Q}(v'_j, v''_j)}\, |j\rangle\,, \tag{20}$$

on $\mathcal{O}(\log n)$ *qubits. We denote this access by* $\mathbf{QS}(\mathbf{v}, n, c_1, c_2)$. *For the sample access to a vector* $\mathbf{v}$ *which approximates a vector with components in* $[0,1]$, *we use the shorthand notation* $\mathbf{QS}(\mathbf{v}, n, c_2) := \mathbf{QS}(\mathbf{v}, n, 1, c_2)$.

One way to obtain such an access is via quantum random access memory (QRAM) [14, 13, 3]. Such a device stores the data in (classical) memory cells, but allows for superposition queries to the data. If all the partial sums are also stored, then QRAM can provide quantum sample access via the Grover-Rudolph procedure, see Ref. [16]. This costs resources proportional to the length of the vector to set up, but then can provide the the superposition state in a run time logarithmic in the length of the vector.

Definition 7 (Quantum RAM)**.** *Let* c *and* m *be positive integers. Let* $\mathbf{v}$ *be a vector of dimension* m, *where each element of* v *is a bit string of length* c, *i.e.,* $v \in (\{0,1\}^c)^m$. *Quantum RAM is defined such that with a one-time cost of* $\widetilde{\mathcal{O}}(c\ m)$ *we can construct quantum query and sampling access* $\mathbf{QA}(v, m, c)$ *and* $\mathbf{QS}(v, m, c)$, *see Definitions 5 and 6 in Appendix D. Each query costs* $\mathcal{O}(c\ \mathrm{poly}\log m)$.

Based on Data Input 3 and Definition 7, Lemma 13 in Appendix D allows us to estimate the inner products between α^t and K_j more efficiently than the equivalent estimation in Algorithm 6. We have the following algorithm.

Algorithm 7: QUANTUM_ALPHATRON

Input *training data* $(\mathbf{x}_i, y_i)_{i=1}^{m_1}$, *testing data* $(\mathbf{a}_j, b_j)_{j=1}^{m_2}$, *error tolerance parameters* ϵ_I *and* δ_I, *function* $u : \mathbb{R} \to [0,1]$, *number of iterations* T, *degree of the multinomial kernel* d, *learning rate* λ, *quantum query access to* $K_j, \forall j \in [m_1]$ *and* $K'_j, \forall j \in [m_2]$ *via Data Input 3*

$\alpha^0 \leftarrow \mathbf{0} \in \mathbb{R}^{m_1}$

for $j \leftarrow 1$ **to** m_1 **do**

  $p_j^{\max} \leftarrow \max_i |K_{ji}|$ via quantum maximum finding with success probability $1 - \delta_I/(4m_1)$

  Define the non-negative vectors $(K_j)^+$, $(K_j)^-$, with $K_j = (K_j)^+ - (K_j)^-$

  From query access to K_j provide query access to $(K'_j)^+$, $(K'_j)^-$ via Lemma 1

for $j \leftarrow 1$ **to** m_2 **do**

  $q_j^{\max} \leftarrow \max_i |K'_{ji}|$ via quantum maximum finding with success probability $1 - \delta_I/(4m_2)$

  Define the non-negative vectors $(K'_j)^+$, $(K'_j)^-$, with $K_j = (K'_j)^+ - (K'_j)^-$

  From query access to K_j provide query access to $(K'_j)^+$, $(K'_j)^-$ via Lemma 1

for $t \leftarrow 0$ **to** $T - 1$ **do**

  Store in QRAM (see Definition 7) the non-negative vectors $(\alpha^t)^+, (\alpha^t)^-$, where $\alpha^t = (\alpha^t)^+ - (\alpha^t)^-$, where each element of the vector is stored using $\lceil \log(\lambda T/m_1) \rceil + \left\lceil \log\left(\frac{2K_{\max} m_1}{\epsilon_I}\right) \right\rceil$ bits

  $w_t \leftarrow \|\alpha^t\|_1$

  for $j \leftarrow 1$ **to** m_1 **do**

    $r_j^t \leftarrow$ Estimate inner product $\alpha^t \cdot K_j$, by estimating $(\alpha^t)^+ \cdot (K_j)^+$, $(\alpha^t)^+ \cdot (K_j)^-$, $(\alpha^t)^- \cdot (K_j)^+$, and $(\alpha^t)^- \cdot (K_j)^-$ via Statement (iii) of Lemma 13 (using w_t and $p_j^{\max}$), each to additive accuracy $\epsilon_I/8$ with success probability $1 - \delta_I/(16Tm_1)$

    $\alpha_j^{t+1} \leftarrow \alpha_j^t + \frac{\lambda}{m_1}(y_j - u(r_j^t))$

  for $j \leftarrow 1$ **to** m_2 **do**

    $s_j^t \leftarrow$ Estimate inner product $\alpha^t \cdot K'_j$, by estimating $(\alpha^t)^+ \cdot (K'_j)^+$, $(\alpha^t)^+ \cdot (K'_j)^-$, $(\alpha^t)^- \cdot (K'_j)^+$, and $(\alpha^t)^- \cdot (K'_j)^-$ via Statement (iii) of Lemma 13 (using w_t and $q_j^{\max}$), each to additive accuracy $\epsilon_I/8$ with success probability $1 - \delta_I/(16Tm_2)$

$t_{out} \leftarrow \arg\min_{t \in [T]} \frac{1}{m_2} \sum_{j=1}^{m_2} (u(s_j^t) - b_j)^2$

Output $\alpha^{t_{out}}$

Theorem 9 (Quantum Alphatron)**.** *We assume quantum query access to the vectors* K_j *and* K'_j *via Data Input 3. Again, let* $K_{\max}$ *be maximum of all entries in* K *and* K'. *Let* $\delta \in (0,1)$. *Given Definitions 3 and 4 and* $\delta_I = \delta$, *Algorithm 7 outputs* $\alpha^{t_{out}}$ *such that the hypothesis* $h^{t_{\text{out}}} = u\left(\sum_j \alpha_j^{t_{\text{out}}} \psi(\mathbf{x}_j) \cdot \psi(\mathbf{x})\right)$ *satisfies with probability* $1 - 3\delta$,

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}\left(A_2 + L\epsilon_I + L^2\epsilon_I^2\right),$$

where A_2 *is defined in Theorem 7. The run time of this algorithm is*

$$\widetilde{\mathcal{O}}\left(m^{1.5} \log\left(\frac{1}{\delta}\right) + Tm + T^2 m \frac{K_{\max}}{L\epsilon_I} \log\left(\frac{1}{\delta}\right)\right).$$

If $A_2 \leq 1$ *and we set* $\epsilon_I = \sqrt{\epsilon}$ *then we further obtain the guarantee*

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}\left(A_2\right),$$

and the run time is

$$\widetilde{\mathcal{O}}\left(m^{1.5}\log\left(\frac{1}{\delta}\right)+Tm+T^2m\frac{K_{\max}}{L\sqrt{\epsilon}}\log\left(\frac{1}{\delta}\right)\right).$$

Proof. First, consider the numerical error from truncating the α vectors. Recall that in the classical steps of the algorithm we work in the arithmetic model where all the steps occur at infinite precision. Let $\alpha^t \in \mathbb{R}^{m_1}$ be the vector given to infinite precision (arithmetic model) with known $0 < \alpha_{\max} \leq \lambda T/m_1$ (Lemma 5). Set $c_1 \geq \lceil \log(\lambda T/m_1) \rceil$ and $c_2 \geq \left\lceil \log\left(\frac{2K_{\max}m_1}{\epsilon_I}\right)\right\rceil$. Let $\tilde{\alpha} \in \{0,1\}^{c_1+c_2}$ be the element-wise c_1+c_2 bit approximation of α^t (stored in QRAM). Note that

$$|\alpha^t \cdot K_i - \tilde{\alpha}^t \cdot K_i| \leq m_1 K_{\max} \max_{j\in[m_1]} |\alpha_j^t - \tilde{\alpha}_j^t| \leq \frac{m_1 K_{\max}}{2^{c_2+1}} \leq \frac{\epsilon_I}{2}. \tag{21}$$

Aside from the estimation of the inner products, the remaining part of Algorithm 7 is the same as Algorithm 6. Compared to Algorithm 6, we change the accuracy of the inner product estimation to $\epsilon_I/2$, hence we achieve that

$$|r_j^t - \tilde{\alpha}^t \cdot K_j| \leq \frac{\epsilon_I}{2}, \forall t \in [T], \forall j \in [m_1], \tag{22}$$

$$|s_j^t - \tilde{\alpha}^t \cdot K'_j| \leq \frac{\epsilon_I}{2}, \forall t \in [T], \forall j \in [m_2], \tag{23}$$

with the stated success probabilities. Using Eq. (21) for the numerical error, we obtain that

$$|r_j^t - \alpha^t \cdot K_j| \leq \epsilon_I, \forall t \in [T], \forall j \in [m_1], \tag{24}$$

$$|s_j^t - \alpha^t \cdot K'_j| \leq \epsilon_I, \forall t \in [T], \forall j \in [m_2], \tag{25}$$

with the same success probabilities. Hence the same accuracy guarantees holds as in the proof of Theorem 8. For the output hypothesis $h^{t_{\text{out}}}$, we have

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}\left(A_2 + L\epsilon_I + L^2\epsilon_I^2\right). \tag{26}$$

For the run time complexity, there are three terms. From Line 2 to Line 10, we perform $\mathcal{O}(m_1+m_2) = \mathcal{O}(m)$ quantum maximum findings. The run time of a single run of the quantum maximum finding is bounded by $\widetilde{\mathcal{O}}(\sqrt{m}\log(1/\delta))$ [11]. Hence, this part of the algorithm takes $\mathcal{O}(m^{1.5}\log(1/\delta))$ time. In Line 12, the time of storing all α^t in QRAM is $\widetilde{\mathcal{O}}(Tm)$ because we have $\mathcal{O}(T)$ vectors and storing each of them costs $\widetilde{\mathcal{O}}(m)$ time. For each step t, the run time of the estimations r_j^t and s_j^t depends on the norm $\|\alpha^t\|_1 \leq T/L$. Hence, the run time of each estimation r_j^t and s_j^t via (iii) of Lemma 13 is bounded by $\widetilde{\mathcal{O}}\left(\frac{TK_{\max}}{L\epsilon_I}\log\left(\frac{1}{\delta}\right)\right)$, and we need to estimate $\mathcal{O}(Tm)$ inner products. Then the overall run time is

$$\widetilde{\mathcal{O}}\left(m^{1.5}\log\left(\frac{1}{\delta}\right)+Tm+T^2m\frac{K_{\max}}{L\epsilon_I}\log\left(\frac{1}{\delta}\right)\right). \tag{27}$$

If we set $\epsilon_I = \sqrt{\epsilon}$, we obtain $\epsilon(h) \leq O(A_2)$ and the run time is

$$\widetilde{\mathcal{O}}\left(m^{1.5}\log\left(\frac{1}{\delta}\right)+Tm+T^2m\frac{K_{\max}}{L\sqrt{\epsilon}}\log\left(\frac{1}{\delta}\right)\right). \tag{28}$$

□

5 Discussion

In this section, we summarize the results from previous sections and discuss the improvement on the run time complexity by pre-computation and quantum estimation. In Section 3, we have introduced ALPHATRON_WITH_PRE,ALPHATRON_WITH_APPROX_PRE, and ALPHATRON_WITH_Q_PRE, which improve the original ALPHATRON. The scenario is that the dimension of the data n is much larger than the other parameters, a situation that is relevant for many practical applications. Without the pre-computation, we have a run time $\mathcal{O}\left(Tm^2n\right)$ compared to the run time with the pre-computation of $\mathcal{O}\left(m^2n + m^2 \log d + Tm^2\right)$. The factor of the n dependent term loses a factor T which is a small improvement. Moreover, by quantum amplitude estimation, we gain a quadratic speedup in the dimension n. We list the results of Section 3 in Table 1 for comparison.

Name	Pre-computation	Main loop	Proved in
ALPHATRON	not applicable	$\mathcal{O}\left(Tm^2n\right)$	[15], also Theorem 1
ALPHATRON_WITH_PRE	$\mathcal{O}\left(m^2n + m^2 \log d\right)$	$\mathcal{O}\left(Tm^2\right)$	Theorem 5
ALPHATRON_WITH_APPROX_PRE	$\widetilde{\mathcal{O}}\left(mn + \frac{m^2d^2}{\epsilon_K^2} \log \frac{1}{\delta_K}\right)$	$\mathcal{O}\left(Tm^2\right)$	Theorem 6
ALPHATRON_WITH_Q_PRE	$\widetilde{\mathcal{O}}\left(\frac{m^2d\sqrt{n}}{\epsilon_K} \log \frac{1}{\delta_K}\right)$	$\mathcal{O}\left(Tm^2\right)$	Corollary 1

Table 1: Comparison of the first set of algorithms in Section 3. We separate the pre-computation of the multinomial kernel function from the main loop and also estimate the training set inner products instead of computing them exactly, which can improve the time complexity of the computation of the kernel function. For all algorithms, we indicate the general result without using the learning setting in Definition 3. For ALPHATRON_WITH_APPROX_PRE and ALPHATRON_WITH_Q_PRE, the relevant kernel functions are estimated to accuracy ϵ_K with failure probability δ_K. To obtain the weak p-concept learning result of Theorem 3 for all these algorithms, take the concept class defined in Definition 3 and the parameter settings for the algorithms of Definition 4. Also, set $\epsilon_K = L\sqrt{\epsilon}/T$ and $\delta_K = \delta$. We do not further evaluate the formulas (using, e.g., the expressions for T and m_1) as the main focus of this table is on the dependency on n which dominates all other parameters.

Section 4 has introduced ALPHATRON_WITH_KERNEL_AND_SAMPLING and QUANTUM_ALPHATRON. In this scenario, we assume constant time query access to the kernel matrices (i.e., to the result of the pre-computation), while the quantum version requires quantum query access. These algorithms only focus on the main loop part of the ALPHATRON. Hence, these algorithms can be viewed as the improvements for ALPHATRON_WITH_KERNEL. We list the results of Section 4 in Table 2. For comparison, we also list the time complexity of ALPHATRON_WITH_KERNEL.

For Table 2, it is not obvious that the quantum algorithm has a speedup compared to the ALPHATRON_WITH_KERNEL. As mentioned in the preliminary, $\mathcal{K}_d(\mathbf{x}, \mathbf{y}) \leq 1$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$. Hence, we can use $K_{\max} \leq 1$. Recall from Theorem 3 that if $m_1 \geq \max\{m_1', m_1''\}$, with $m_1' = \frac{16\zeta^4}{\epsilon^2} \log(1/\delta)$ and $m_1'' = \frac{4B^2}{\epsilon} \log(1/\delta)$, then the concept class in Definition 3 is weak p-concept learnable up to $2C''L\sqrt{\epsilon}$ by the ALPHATRON algorithm.

Case $m_1' > m_1''$. Consider the first case, which is equivalent to $4\zeta^4 > B^2\epsilon$. Hence, $m_1 \in \mathcal{O}\left(m_1'\right)$ leads to learnability, which we can use to simplify $T = CBL\sqrt{\frac{m_1}{\log(1/\delta)}} \in \mathcal{O}\left(BL\frac{\zeta^2}{\epsilon}\right)$. In addition, $m = m_1 + m_2$, and we have $m_2 = C'm_1 \log(T/\delta)$, hence, $m = (1 + C' \log T +$

Name	Main loop	Theorem
ALPHATRON_WITH_KERNEL	$\mathcal{O}\left(Tm^2\right)$	Theorem 4
ATRON_WITH_KERNEL_AND_SAMPLING	$\widetilde{\mathcal{O}}\left(Tm + T^3 m \frac{K_{\max}^2}{L^2 \epsilon_I^2} \log \frac{1}{\delta}\right)$	Theorem 8
QUANTUM_ALPHATRON	$\widetilde{\mathcal{O}}\left(m^{1.5} \log \frac{1}{\delta} + Tm + T^2 m \frac{K_{\max}}{L\epsilon_I} \log \frac{1}{\delta}\right)$	Theorem 9

Table 2: Comparison of the second set of algorithms ATRON_WITH_KERNEL_AND_SAMPLING and QUANTUM_ALPHATRON, which are discussed in Section 4, to ALPHATRON_WITH_KERNEL. These algorithms change the main loop part by using an inner product estimation. The inner product estimation is performed to accuracy ϵ_I and the total success probability of the algorithm is $1-\delta$. Here, we indicate the general result without the learning setting in Definition 3.

Case	Classical run time	Quantum run time	Advantage
$4\zeta^4 > B^2\epsilon$	$\widetilde{\mathcal{O}}\left(\frac{BL\zeta^{10}}{\epsilon^5} \log^4 \frac{1}{\delta}\right)$	$\widetilde{\mathcal{O}}\left(\frac{B^2 L\zeta^8}{\epsilon^{4.5}} \log^4 \frac{1}{\delta}\right)$	yes
$4\zeta^4 \leq B^2\epsilon$	$\widetilde{\mathcal{O}}\left(\frac{B^6 L}{\epsilon^{2.5}} \log^4 \frac{1}{\delta}\right)$	$\widetilde{\mathcal{O}}\left(\frac{B^6 L}{\epsilon^{2.5}} \log^4 \frac{1}{\delta}\right)$	no

Table 3: Comparison of the algorithms ALPHATRON_WITH_KERNEL and QUANTUM_ALPHATRON for the learning setting in Definition 3. Only in the first case a quantum advantage is obtained.

$C' \log 1/\delta) m_1 \in \widetilde{\mathcal{O}}\left(\frac{\zeta^4}{\epsilon^2} \log^2 \frac{1}{\delta}\right)$. From Theorem 9, we have the run time

$$\begin{aligned} \widetilde{\mathcal{O}}\left(m^{1.5} \log \frac{1}{\delta} + Tm + T^2 m \frac{1}{L\sqrt{\epsilon}} \log \frac{1}{\delta}\right) &= \widetilde{\mathcal{O}}\left(\frac{\zeta^6}{\epsilon^3} \log^4 \frac{1}{\delta} + BL\frac{\zeta^4}{\epsilon^3} \log^2 \frac{1}{\delta} + B^2 L \frac{\zeta^8}{\epsilon^{4.5}} \log^3 \frac{1}{\delta}\right) \\ &= \widetilde{\mathcal{O}}\left(B^2 L \frac{\zeta^8}{\epsilon^{4.5}} \log^4 \frac{1}{\delta}\right). \end{aligned} \tag{29}$$

For the classical run time, we simplify $\mathcal{O}\left(Tm^2\right) \subseteq \widetilde{\mathcal{O}}\left(BL\frac{\zeta^{10}}{\epsilon^5} \log^4 \frac{1}{\delta}\right)$.

Case $m_1'' > m_1'$. Consider the second case, which is equivalent to $4\zeta^4 < B^2\epsilon$. Hence, $m_1 = \mathcal{O}\left(m_1''\right)$ leads to learnability, which we can use to simplify $T = CBL\sqrt{\frac{m_1}{\log(1/\delta)}} \in \mathcal{O}\left(\frac{B^2 L}{\sqrt{\epsilon}}\right)$. In addition, $m \in \widetilde{\mathcal{O}}\left(\frac{B^2}{\epsilon} \log^2 \frac{1}{\delta}\right)$. From Theorem 9, we have the run time $\widetilde{\mathcal{O}}\left(\frac{B^6 L}{\epsilon^{2.5}} \log^4 \frac{1}{\delta}\right)$. For the classical run time we simplify $\mathcal{O}\left(Tm^2\right) \subseteq \widetilde{\mathcal{O}}\left(\frac{B^6 L}{\epsilon^{2.5}} \log^4 \frac{1}{\delta}\right)$. This analysis of the two cases is summarized in Table 3 and allows us to state our final theorem.

Theorem 10 (Quantum p-concept learnability via the QUANTUM_ALPHATRON)**.** *Let the concept class and distribution be defined by Definition 3. For this concept class, let $4\zeta^4 > B^2\epsilon$. In addition, let there be given quantum access to the kernel matrices via Data Input 3. Then, the concept class in Definition 3 is weak p-concept learnable up to $2C''L\sqrt{\epsilon}$ by the QUANTUM_ALPHATRON algorithm with a run time that shows an advantage by a factor $\sim \frac{\zeta^2}{B\sqrt{\epsilon}}$ over the classical algorithm given the same input.*

A note on the condition $4\zeta^4 > B^2\epsilon$ for the speedup. By Definition 3, ζ determines the range of the noise function, while ϵ is an upper bound to the variance of the noise function. For any function the variance will be smaller or equal to the range. Hence, the condition $4\zeta^4 > B^2\epsilon$ is reasonably easy to satisfy and we may obtain a quantum advantage for a broad concept class of functions. In Appendix E, we combine the algorithms for the kernel matrix estimation and the inner loop estimations, both for the classical and quantum cases.

6 Applications for learning two-layer neural networks

In this section, we describe how to use algorithms described above to learn neural networks with two nonlinear layers in the p-concept model. Following previous works [15, 27], first consider an one-layer neural network with k units

$$\mathcal{N}_1 : \mathbf{x} \to \sum_{i=1}^{k} \mathbf{b}_i \sigma(\mathbf{a}_i \cdot \mathbf{x}), \tag{30}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{B}^k$, $\mathbf{a}_i \in \mathbb{B}^n$ for $i \in \{1, \ldots, k\}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function. In the following, we consider the sigmoid function $\sigma = 1/(1 + e^{-x})$ as the activation function. Subsequently, we define a neural work with two nonlinear layers with one unit in the second layer

$$\mathcal{N}_2 : \mathbf{x} \to \sigma' \left(\sum_{i=1}^{k} \mathbf{b}_i \sigma(\mathbf{a}_i \cdot \mathbf{x}) \right), \tag{31}$$

where $\sigma' : \mathbb{R} \to \mathbb{R}$ is an L-Lipschitz non-decreasing function. Ref. [15] proved the following lemma, which states that such kinds of neural networks are efficiently p-concept learnable.

Lemma 2 (Efficient p-concept learning for two-layer neural networks [15])**.** *Consider samples $(\boldsymbol{x}_i, y_i)_{i=1}^m$ drawn i.i.d. from a distribution $\mathcal{D}$ on $\mathbb{B}^n \times [0, 1]$ such that $E[y \mid \boldsymbol{x}] = \mathcal{N}_2(\boldsymbol{x})$ with $\sigma' : \mathbb{R} \to [0, 1]$ is a known L-Lipschitz non-decreasing function and σ is the sigmoid function. There exists an algorithm that outputs a hypothesis h such that with probability $1 - \delta$,*

$$\mathbb{E}_{\boldsymbol{x}, y \sim \mathcal{D}} \left[\left(h(\boldsymbol{x}) - \mathcal{N}_2(\boldsymbol{x}) \right)^2 \right] \leq \epsilon,$$

for $m = \left(\frac{kL}{\epsilon}\right)^{O(1)} \cdot \log^2(1/\delta)$. The algorithm runs in time polynomial with m and n.

The statement of the previous work mentions the training sample complexity m_1 instead of total sample complexity m, where the latter depends on $\log^2(1/\delta)$. Here, we focus on the total sample complexity.

We assume corresponding query access to the kernel matrices, both for the classical and the quantum scenario, respectively. We focus on the comparison of time complexity for Alpha-tron_with_Kernel and Quantum_Alphatron. The sigmoid function can be uniformly approximated by low-degree polynomials. By Lemma 8 and 12 in Ref. [15], we have that there exists a constant C_{sig} and a $\mathbf{v} \in \mathcal{H}_d$ with $\|\mathbf{v}\|_2 \leq (\sqrt{k}/\epsilon_0)^{C_{sig}}$ for $d = O(\log(1/\epsilon_0))$ such that

$$\left| \sum_{i=1}^{k} \mathbf{b}_i \sigma(\mathbf{a}_i \cdot \mathbf{x}) - \langle \mathbf{v}, \psi(\mathbf{x}) \rangle \right| \leq \sqrt{k}\epsilon_0, \tag{32}$$

for every $\mathbf{x} \in \mathbb{B}^n$. Hence, in this case we have $\mathcal{N}_1(\mathbf{x}) = \langle \mathbf{v}, \psi(\mathbf{x}) \rangle + \xi(\mathbf{x})$ for some function $\xi : \mathcal{X} \to [-\sqrt{k}\epsilon_0, \sqrt{k}\epsilon_0]$, and $\mathbb{E}[y \mid \mathbf{x}] = \sigma'(\langle \mathbf{v}, \psi(\mathbf{x}) \rangle + \xi(\mathbf{x}))$ with $\mathbb{E}[\xi(\mathbf{x})^2] \leq k\epsilon_0^2$. From these, we have $\zeta = \sqrt{k}\epsilon_0$, $B = (\sqrt{k}/\epsilon_0)^{C_{\text{sig}}}$ and $\epsilon = k\epsilon_0^2$.

Based on the discussion in Section 5, we can compare the value of $4\zeta^4$ and $B^2\epsilon$ to determine the possibility for a quantum speedup. In this case, we have $4\zeta^4 = 4(\sqrt{k}\epsilon_0)^4$ and $B^2\epsilon = k^2(\sqrt{k}/\epsilon_0)^{2C_{\text{sig}}-2}$. The condition for the quantum speedup is $4\zeta^4 > B^2\epsilon$, which is equivalent to $4 > k^2(\sqrt{k}/\epsilon_0)^{2C_{\text{sig}}-2}$. As $k \geq 1$, $0 < \epsilon_0 < 1$, and $C_{\text{sig}} > 2$ [22], in general the inequality will not hold, and hence we will not achieve a quantum speedup for learning two-layer neural networks with the sigmoid function.

In the following final part, we discuss a type of neural network which is in a regime where there is a quantum advantage according to Section 5. The key idea is that if the neural network has some intrinsic errors, these errors dominate and we may achieve a quantum speedup. We first define the neural-network architecture.

Definition 8 (Erroneous neural networks)**.** *Let $\gamma_1 > 0$ and $\gamma_2 > 0$ and let there be given a function $\xi_1(\mathbf{x}) \in [-\gamma_1, \gamma_1]$ and $\mathbb{E}[\xi_1^2(x)] \leq \gamma_2$. We say that we have an (γ_1, γ_2)-erroneous one-layer neural network with k units if it can be written in the form*

$$\mathcal{N}_1^{\text{err}} : x \to \sum_{i=1}^{k} \mathbf{b}_i \sigma(\mathbf{a}_i \cdot \mathbf{x}) + \xi_1(\mathbf{x}), \tag{33}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{B}^k$, $\mathbf{a}_i \in \mathbb{B}^n$ for $i \in \{1, \ldots, k\}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is the sigmoid function. Based on the one-layer network, we define a two-layer (γ_1, γ_2)-erroneous neural network with

$$\mathcal{N}_2^{\text{err}} : x \to \sigma' \left(\sum_{i=1}^{k} \mathbf{b}_i \sigma(\mathbf{a}_i \cdot \mathbf{x}) + \xi_1(\mathbf{x}) \right), \tag{34}$$

where $\sigma' : \mathbb{R} \to \mathbb{R}$ is an L-Lipschitz non-decreasing function.

This neural network allows for a quantum advantage for learning it.

Theorem 11. *There is a quantum speedup for learning two-layer (γ_1, γ_2)-erroneous neural networks based on the p-concept learning model if $\gamma_1 > \max\{\frac{1}{\sqrt{2}}(\sqrt{k}/\epsilon_0)^{C_{\text{sig}}}, \frac{1}{\sqrt{2}}\sqrt{\gamma_2}\}$.*

Proof. Using Eq. 32, we have that $\forall x \in \mathcal{X}$, $\mathcal{N}_1^{\text{err}} = \langle \mathbf{v}, \psi(\mathbf{x}) \rangle + \xi'(\mathbf{x})$ for some function $\xi' : \mathcal{X} \to [-\zeta_{\text{err}}, \zeta_{\text{err}}]$ with $\zeta_{\text{err}} := \sqrt{k}\epsilon_0 + \gamma_1$ and with $\mathbb{E}[\xi'^2] \leq (\gamma_2 + k\epsilon_0^2 + 2\gamma_1\sqrt{k}\epsilon_0) =: \epsilon_{\text{err}}$. The ℓ_2-norm of the hypothesis vector $\|\boldsymbol{v}\|_2$ is bounded by $B_{\text{err}} := (\sqrt{k}/\epsilon_0)^{C_{\text{sig}}}$. To achieve the quantum speedup, we need to satisfy $4\zeta_{\text{err}}^4 > B_{\text{err}}^2 \epsilon_{\text{err}}$. It suffices to satisfy

$$4(\sqrt{k}\epsilon_0 + \gamma_1)^4 > (\sqrt{k}/\epsilon_0)^{2C_{\text{sig}}}(\gamma_2 + k\epsilon_0^2 + 2\gamma_1\sqrt{k}\epsilon_0). \tag{35}$$

This inequality holds when the following holds

$$\begin{aligned} 2(\sqrt{k}\epsilon_0 + \gamma_1)^2 &> (\sqrt{k}/\epsilon_0)^{2C_{\text{sig}}}, \\ 2(\sqrt{k}\epsilon_0 + \gamma_1)^2 &> (\gamma_2 + k\epsilon_0^2 + 2\gamma_1\sqrt{k}\epsilon_0). \end{aligned}$$

These two inequalities can be achieved if $\gamma_1 > \max\{\frac{1}{\sqrt{2}}(\sqrt{k}/\epsilon_0)^{C_{\text{sig}}}, \frac{1}{\sqrt{2}}\sqrt{\gamma_2}\}$. □

Hence, we have shown that the original two-layer neural network does not admit a quantum speedup with our approach, while a noisy version of the two-layer network does.

7 Acknowledgements

This work was supported by the Singapore National Research Foundation, the Prime Minister's Office, Singapore, the Ministry of Education, Singapore under the Research Centres of Excellence programme under research grant R 710-000-012-135. In addition, this research is supported by the National Research Foundation, Singapore under its CQT Bridging Grant.

A Lipschitz condition for multinomial kernel function

Lemma 3. *Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f(z) = \frac{1}{d+1}\sum_{i=0}^{d} z^i$. Then $f(z)$ is Lipschitz continuous with Lipschitz constant $L \in \mathcal{O}\left(dz_0^d + dz_0 + \frac{1}{d}\right)$, that is*

$$|f(z) - f(z')| \leq L|z - z'|, \tag{36}$$

for all $z, z' \in [-z_0, z_0]$.

Proof. First, $\frac{d}{dz}f(z) = \frac{1}{d+1}\sum_{i=0}^{d-1}(i+1)z^i \leq \frac{1}{d+1}(1 + \sum_{i=1}^{d-1} dz^i)$. When $0 < z < 1$, $z^i \leq z$ for $1 \leq i \leq d-1$. And when $z \geq 1$, $z^i \leq z^d$ for $1 \leq i \leq d-1$. Thus $z^i \leq z + z^d$ for $1 \leq i \leq d-1$. Hence $L \leq \max_{z\in[-z_0,z_0]} |\frac{d}{dz}f(z)| \leq \frac{d}{dz}f(z) \mid_{z=z_0} \in \mathcal{O}\left(dz_0^d + dz_0 + \frac{1}{d}\right)$. □

Note that throughout this paper, it always holds that $z_0 = 1$. In this case, the Lipschitz constant is bounded by $\mathcal{O}(d)$.

B Proof of Theorem 7

We first introduce several definitions and lemmas for proving the theorem. Given the coefficients α_i, we generate a hypothesis vector $\mathbf{v}(\alpha)$ in the feature space by taking the linear combination over vectors $\psi_d(\mathbf{x}_i)$.

Definition 9 (Auxiliary definitions)**.** *In the setting of Definitions 3 and 4, define the generated hypothesis mapping $\mathbf{v} : \mathbb{R}^{m_1} \to \mathbb{R}^{n_d}$ as the linear combination*

$$\mathbf{v}(\alpha) := \sum_{i=1}^{m_1} \alpha_i \psi_d(\mathbf{x}_i). \tag{37}$$

In addition, define $\beta \in \mathbb{R}^{m_1}$ as

$$\beta_i := \frac{1}{m_1}(y_i - u(\langle \boldsymbol{v}, \psi_d(\mathbf{x}_i)\rangle) + \xi(\mathbf{x}_i)), \tag{38}$$

using $\boldsymbol{v}$ from the concept class and $\Delta := \mathbf{v}(\beta)$ with the norm $\eta := \|\Delta\|_2$. Finally, define $\rho := \frac{1}{m_1}\sum_{i=1}^{m_1} \xi(\mathbf{x}_i)^2$ as the average quadratic noise over the input data.

We note the subtle difference of the symbols $\mathbf{v}(\alpha)$ and $\boldsymbol{v}$ but emphasize that $\mathbf{v}(\alpha)$ will always have the parenthesis with the input value, while $\boldsymbol{v}$ is static and fixed by the element of the concept class. To adapt to matrices K_{ij} (with dimension $m_1 \times m_1$) and K'_{ij} (with dimension $m_2 \times m_1$), Definition 10 and Lemma 4 are stated in general terms.

Definition 10 (Hypothesis function)**.** *Let $u : \mathbb{R} \to [0, 1]$ be an L-Lipschitz function. Define the hypothesis function $g : \mathbb{R}^{N_2} \times \mathbb{R}^{N_1 \times N_2} \times [N_1] \to [0, 1]$ as*

$$g(\alpha, M, i) := u\left(\sum_{j=1}^{N_2} \alpha_j M_{ij}\right). \tag{39}$$

In Lemma 4, we show that if we have a good enough estimation $\widetilde{M}$ for the matrix M, then the estimated result $g(\alpha, \widetilde{M}, i)$ is not too far from the exact value $g(\alpha, M, i)$, with a dependence on $\|\alpha\|_1$.

Lemma 4. *Let $M, \tilde{M} \in \mathbb{R}^{N_1 \times N_2}$ be matrices of dimension $N_1 \times N_2$. Let $\epsilon \in (0,1)$. Let $u : \mathbb{R} \to [0,1]$ be L-Lipschitz. If $\max_{i\in[N_1], j\in[N_2]} |M_{ij} - \tilde{M}_{ij}| \leq \epsilon$, then for all $\alpha \in \mathbb{R}^{N_2}$, we have*

$$\max_{i\in[N_1]} \left| g(\alpha, M, i) - g(\alpha, \tilde{M}, i) \right| \leq L\epsilon \|\alpha\|_1.$$

Proof. For all $i \in [N_1]$,

$$\begin{aligned} |g(\alpha, M, i) - g(\alpha, \tilde{M}, i)| &\leq L \left| \sum_{j=1}^{N_2} \alpha_j (M_{ij} - \tilde{M}_{ij}) \right| && (40) \\ &\quad \text{(by the } L\text{-Lipschitz conditon of } u) \\ &\leq L\epsilon \sum_{j=1}^{N_2} |\alpha_j| && (41) \\ &\quad \text{(by assumption)} \\ &= L\epsilon \|\alpha\|_1. && (42) \end{aligned}$$

□

In the following lemma we show that if we update α^t according to the ALPHATRON algorithm, then the max norm of α^t can be bounded in terms of T, λ and m_1.

Lemma 5. *For arbitrary $K \in \mathbb{R}^{m_1 \times m_1}$, $y \in [0,1]^{m_1}$, and $\lambda \in \mathbb{R}_+$, if the initial vector is $\alpha^0 = \mathbf{0}$, then by performing the updates $\alpha_i^{t+1} \leftarrow \alpha_i^t + \frac{\lambda}{m_1}(y_i - g(\alpha^t, K, i))$, for each entry T times, we have $\max_i |\alpha_i^T| \leq \frac{T\lambda}{m_1}$.*

Proof. We prove the statement by induction. The base case is obviously true. Note that y is from $[0,1]$ and the range of g is also $[0,1]$. Hence,

$$\left|\alpha_i^t\right| \leq \left|\alpha_i^{t-1}\right| + \frac{\lambda}{m_1} \left| y_i - g(\alpha^{t-1}, K, i) \right| \leq \frac{(t-1)\lambda}{m_1} + \frac{\lambda}{m_1} = \frac{t\lambda}{m_1}.$$

□

We state the convergence result from the original work Ref. [15] in Lemma 6 before proving our own Lemma 7. Intuitively, these lemmas prove that we indeed make progress towards the target by each iteration. The norm $\|\mathbf{v}(\omega) - \mathbf{v}\|_2^2$ measures the distance between the current vector and the target. If we show that this value decreases as we run the algorithm and if we lower bound this value in term of the empirical error $\hat{\varepsilon}$ of the current vector, then either we made progress, or the quality of the current vector is already good enough.

Lemma 6 (Convergence of Algorithm 1 from [15])**.** *Consider Definition 3 and 4 for the setting and the algorithm parameters, as well as the training labels $y \in [0,1]^{m_1}$, and the kernel matrix $K \in [-1,1]^{m_1 \times m_1}$. For any vector $\omega \in \mathbb{R}^{m_1}$, let $\omega' \in \mathbb{R}^{m_1}$ be the vector defined as*

$$\omega_i' := \omega_i + \frac{\lambda}{m_1}(y_i - g(\omega, K, i)).$$

Let h be the hypothesis function defined as $h(\mathbf{x}) = u(\langle \mathbf{v}(\omega), \psi_d(\mathbf{x}) \rangle)$, and recall from Definition 9 that $\eta = \|\Delta\|_2$. If $\|\mathbf{v}(\omega) - \boldsymbol{v}\|_2 \leq B$, for $B > 1$, and $\eta < 1$, then

$$\|\mathbf{v}(\omega) - \boldsymbol{v}\|_2^2 - \|\mathbf{v}(\omega') - \boldsymbol{v}\|_2^2 \geq \frac{1}{L^2}\hat{\varepsilon}(h) - A_1,$$

where by definition $A_1 := \frac{2}{L}\sqrt{\rho} + \frac{2B\eta}{L} + \frac{\eta^2}{L^2} + \frac{2\eta}{L^2}$.

We claim the following modified convergence result for our Algorithm 4. The difference to the previous lemma is the appearance of a term $-\epsilon_I^2$ in the convergence bound.

Lemma 7. *Consider Definition 3 and 4 for the setting and the algorithm parameters, as well as the training labels $y \in [0,1]^{m_1}$, and the kernel matrix $K \in [-1,1]^{m_1 \times m_1}$. Let $\epsilon_I > 0$. Suppose that Γ_i is an estimation of $g(\alpha^t, K, i)$, for all $i \in [m_1]$, such that*

$$\max_{i \in [m_1]} \left| g(\alpha^t, K, i) - \Gamma_i \right| \leq L\epsilon_I.$$

For any vector $\omega \in \mathbb{R}^{m_1}$, let $\omega' \in \mathbb{R}^{m_1}$ be the vector defined as

$$\omega_i' := \omega_i + \frac{\lambda}{m_1}(y_i - g(\omega, K, i)),$$

and $\tilde{\omega} \in \mathbb{R}^{m_1}$ be the vector defined as

$$\tilde{\omega}_i := \omega_i + \frac{\lambda}{m_1}(y_i - \Gamma_i).$$

Let h be the hypothesis function defined as $h(\mathbf{x}) = u(\langle \mathbf{v}(\omega), \psi_d(\mathbf{x}) \rangle)$. Let A_1 be defined as in Lemma 6. Recall that $\eta = \|\Delta\|_2$. If $\|\mathbf{v}(\omega) - \boldsymbol{v}\|_2 \leq B$, for $B > 1$, and $\eta < 1$, then

$$\|\mathbf{v}(\omega) - \boldsymbol{v}\|_2^2 - \|\mathbf{v}(\tilde{\omega}) - \boldsymbol{v}\|_2^2 \geq \frac{1}{L^2}\hat{\varepsilon}(h) - A_1 - \epsilon_I^2. \tag{43}$$

Proof. Recall from Definition 9 that $\mathbf{v}(\alpha) = \sum_{i=1}^{m_1} \alpha_i \psi(\mathbf{x}_i)$. Then we have

$$\begin{aligned}
\|\mathbf{v}(\tilde{\omega}) - \mathbf{v}(\omega')\|_2^2 &= \frac{\lambda^2}{m_1^2} \left\| \sum_{j=1}^{m_1} (\Gamma_j - g(\omega, K, j)) \psi(\mathbf{x}_j) \right\|_2^2 && (44)\\
&\quad \text{(expand the definition of } \mathbf{v}, \omega', \text{ and } \tilde{\omega}) \\
&\leq \frac{\lambda^2}{m_1^2} \left(\sum_{j=1}^{m_1} |\Gamma_j - g(\omega, K, j)| \, \|\psi(\mathbf{x}_j)\|_2 \right)^2 && (45)\\
&\quad \text{(by triangle's inequality)} \\
&\leq \frac{\lambda^2 L^2 \epsilon_I^2}{m_1^2} m_1^2 && (46)\\
&\quad \text{(by assumptions)} \\
&= \epsilon_I^2. && (47)
\end{aligned}$$

Therefore, using the triangle inequality, we can deduce that

$$\begin{aligned}
\|\mathbf{v}(\tilde{\omega}) - \boldsymbol{v}\|_2^2 &\leq \|\mathbf{v}(\tilde{\omega}) - \mathbf{v}(\omega')\|_2^2 + \|\mathbf{v}(\omega') - \boldsymbol{v}\|_2^2 && (48)\\
&\leq \epsilon_I^2 + \|\mathbf{v}(\omega') - \boldsymbol{v}\|_2^2. && (49)
\end{aligned}$$

Note that except for the vector $\tilde{\omega}$ and the related conditions, Lemma 7 has the same settings as Lemma 6. Thus by the conclusion of Lemma 6, we can lower bound $\|\mathbf{v}(\omega) - \boldsymbol{v}\|_2^2 - \|\mathbf{v}(\omega') - \boldsymbol{v}\|_2^2$ by $\frac{1}{L^2}\hat{\varepsilon}(h) - A_1$. Together with equation (49), we obtain the required bound. □

In the last step of Algorithm 2, we pick the hypothesis with the minimum eêr value. We also lose accuracy here as we use entries from the approximation of the matrix K'. In Lemma 8, we show that these errors are acceptable.

Lemma 8. *Consider the training data samples* $(\mathbf{x}_i, y_i) \in \mathbb{B}^n \times [0,1]$*, for* $i \in [m_1]$*, validation data samples* $(\mathbf{a}_i, b_i) \in \mathbb{B}^n \times [0,1]$*, for* $i \in [m_2]$*, the corresponding kernel matrix* $K' \in [-1,1]^{m_2 \times m_1}$ *where* $K'_{ij} = \mathcal{K}_d(\mathbf{a}_i, \mathbf{x}_j)$*, and the vectors* $\alpha^t \in \mathbb{R}^{m_1}$*, for* $t \in [T]$*. Let* $\epsilon_I > 0$*. Suppose that* Γ_i^t *is an estimation of* $g(\alpha^t, K', i)$*, for all* $t \in [T]$ *and* $i \in [m_2]$*, such that*

$$\max_{i \in [m_2]} \left| g(\alpha^t, K', i) - \Gamma_i^t \right| \leq L\epsilon_I.$$

We define the hypothesis functions h^t *as* $h^t(\mathbf{x}) = u(\langle \mathbf{v}(\alpha^t), \psi_d(\mathbf{x}) \rangle)$*. Let*

$$\tilde{t} = \arg\min_{t \in [T]} \left\{ \frac{1}{m_2} \sum_{i=1}^{m_2} (\Gamma_i^t - b_i)^2 \right\}$$

and let $t' = \arg\min_{t \in [T]} \hat{\text{err}}(h^t)$*. Then*

$$\hat{\text{err}}(h^{\tilde{t}}) - \hat{\text{err}}(h^{t'}) \in \mathcal{O}\left(L\epsilon_I\right). \tag{50}$$

Proof. By Definition 10 and the assumption on the kernel matrix K', we obtain

$$h^t(\mathbf{a}_i) = u(\langle \mathbf{v}(\alpha^t), \psi_d(\mathbf{a}_i) \rangle) = g(\alpha^t, K', i). \tag{51}$$

We have m_2 samples for validation. Recall the definition of the empirical error $\hat{\text{err}}(h) = \frac{1}{m_2} \sum_{i=1}^{m_2} (h(\mathbf{a}_i) - b_i)^2$ in the Preliminary. For fixed t,

$$\begin{aligned}
\left| \hat{\text{err}}(h^t) - \frac{1}{m_2} \sum_{i=1}^{m_2} (\Gamma_i^t - b_i)^2 \right| &= \frac{1}{m_2} \left| \sum_{i=1}^{m_2} (g(\alpha^t, K', i)^2 - (\Gamma_i^t)^2 - 2b_i(g(\alpha^t, K', i) - \Gamma_i^t)) \right| && (52) \\
&\quad \text{(by empirical error with respect to } (\mathbf{a}_i, b_i) \text{ and equation (51))} \\
&\leq \max_{i \in [m_2]} \left| g(\alpha^t, K', i)^2 - (\Gamma_i^t)^2 - 2b_i(g(\alpha^t, K', i) - \Gamma_i^t) \right| && (53) \\
&\quad \text{(the maximum is at least the average)} \\
&\leq \max_{i \in [m_2]} \left| (g(\alpha^t, K', i) + \Gamma_i^t - 2b_i)(g(\alpha^t, K', i) - \Gamma_i^t) \right| && (54) \\
&\leq \max_{i \in [m_2]} \left| g(\alpha^t, K', i) + \Gamma_i^t - 2b_i \right| \cdot |g(\alpha^t, K', i) - \Gamma_i^t| && (55) \\
&\leq \max_{i \in [m_2]} 2 \left| g(\alpha^t, K', i) - \Gamma_i^t \right| && (56) \\
&\quad \left(g(\alpha^t, K', i), \Gamma_i^t, b_i \text{ are in range } [0,1] \right) \\
&\leq 2L\epsilon_I. && (57) \\
&\quad \text{(by assumption)}
\end{aligned}$$

Hence, we have both

$$\hat{\text{err}}(h^{t'}) + 2L\epsilon_I \geq \frac{1}{m_2} \sum_{i=1}^{m_2} (\Gamma_i^{t'} - b_i)^2, \tag{58}$$

when $t = t'$ in (57), and

$$\frac{1}{m_2} \sum_{i=1}^{m_2} (\Gamma_i^{\tilde{t}} - b_i)^2 \geq \hat{\text{err}}(h^{\tilde{t}}) - 2L\epsilon_I, \tag{59}$$

when $t = \tilde{t}$ in (57). And by the minimization of $\tilde{t}$,

$$\frac{1}{m_2}\sum_{i=1}^{m_2}(\Gamma_i^{t'} - b_i)^2 \geq \frac{1}{m_2}\sum_{i=1}^{m_2}(\Gamma_i^{\tilde{t}} - b_i)^2. \tag{60}$$

From the last three equations by transitivity we deduce the required $\hat{\text{err}}(h^{\tilde{t}}) - \hat{\text{err}}(h^{t'}) \in \mathcal{O}(L\epsilon_I)$. □

Now, we are going to prove the main theorem. As the original work, the theorem requires a generalization bound which involves the Rademacher complexity of the function class considered here. The required result is Theorem 12 in the Appendix F.

Proof of Theorem 7. We first consider the success probability of the approximation part. According to Algorithm 4, we estimate each inner product with success probability $1 - \delta_K/(m_1^2 + m_1 m_2)$. A union bound for these $m_1^2 + m_1 m_2$ estimations gives the total success probability to be at least $1 - \delta_K = 1 - \delta$. Then it is sufficient to show that the main body itself succeeds with probability at least $1 - \delta$ and indeed produces a good enough hypothesis.

The remaining proof consists of three parts. In the first part, we show that there exists $t^* \in [T]$ such that the empirical error of h^{t^*} is good enough. In the second part, we show using the Rademacher complexity that for any specific hypothesis in the concept class we introduced, the generalization error is not very far from empirical error. In the third part, we show that by using m_2 additional samples to validate all generated hypotheses, as done in the algorithm, we are able to find a hypothesis with similar error as h^{t^*}.

First, we show that there exists a good enough hypothesis according to the empirical error. Recall the notations Δ and ρ in Definition 9, and let $\eta = \|\Delta\|_2$. Ref. [15] shows that $\eta \leq \frac{1}{\sqrt{m_1}}(1 + \sqrt{2\log(1/\delta)})$, and $\rho \leq \sqrt{\epsilon} + \mathcal{O}\left(\sqrt[4]{\frac{\log(1/\delta)}{m_1}}\right)$ by Hoeffding's inequality. Since η and ρ only depend on the setting in Definition 3, the modification done in Algorithm 4 compared to Algorithm 3 keeps the bounds for η and ρ the same. Hence, we use the same bounds in this proof.

In the Algorithm 2, which is used in Algorithm 4, assume we are presently at the iteration t for computing the vector $\widetilde{\alpha}^{t+1}$ from $\widetilde{\alpha}^t$. In this proof, we use the tilde above the α to emphasize that we indeed construct a different sequence $(\widetilde{\alpha}^t)_{t\in[T]}$ compared to the sequence $(\alpha^t)_{t\in[T]}$ of Algorithm 2 with the exact kernel matrices. One of the following two cases is satisfied,

$$\textbf{Case 1}: \left\|\mathbf{v}\left(\widetilde{\alpha}^t\right) - \boldsymbol{v}\right\|_2^2 - \left\|\mathbf{v}\left(\widetilde{\alpha}^{t+1}\right) - \boldsymbol{v}\right\|_2^2 > \frac{B\eta}{L}, \tag{61}$$

$$\textbf{Case 2}: \left\|\mathbf{v}\left(\widetilde{\alpha}^t\right) - \boldsymbol{v}\right\|_2^2 - \left\|\mathbf{v}\left(\widetilde{\alpha}^{t+1}\right) - \boldsymbol{v}\right\|_2^2 \leq \frac{B\eta}{L}. \tag{62}$$

Let t^* be the first iteration where Case 2 holds. We show that such an iteration exists. Assume the contradictory, that is, Case 2 fails for each iteration. Since $\|\mathbf{v}(\widetilde{\alpha}^0) - \boldsymbol{v}\|_2^2 = \|\mathbf{0} - \boldsymbol{v}\|_2^2 \leq B^2$ by assumption, however,

$$B^2 \geq \left\|\mathbf{v}\left(\widetilde{\alpha}^0\right) - \boldsymbol{v}\right\|_2^2 \geq \left\|\mathbf{v}\left(\widetilde{\alpha}^0\right) - \boldsymbol{v}\right\|_2^2 - \left\|\mathbf{v}\left(\widetilde{\alpha}^k\right) - \boldsymbol{v}\right\|_2^2 \tag{63}$$

$$= \sum_{t=0}^{k-1}\left(\left\|\mathbf{v}\left(\widetilde{\alpha}^t\right) - \boldsymbol{v}\right\|_2^2 - \left\|\mathbf{v}\left(\widetilde{\alpha}^{t+1}\right) - \boldsymbol{v}\right\|_2^2\right) \geq \frac{kB\eta}{L}, \tag{64}$$

for k iterations. Hence, in at most $\frac{BL}{\eta}$ iterations Case 1 will be violated and Case 2 will have to be true. By Assumption 4 and the bound on η, we have that $T \geq \frac{BL}{\eta}$, and then

$t^* \in [T]$ must exist such that Case 2 is true. For all $t \in [T]$, define the hypothesis function h^t as $h^t(\mathbf{x}) = u(\langle \mathbf{v}(\widetilde{\alpha}^t), \psi_d(\mathbf{x})\rangle)$. By Theorem 6, we have that $\max_{ij} \left|K_{ij} - \widetilde{K}_{ij}\right| \leq \epsilon_K$. Define the shorthand $\Gamma_i := g(\widetilde{\alpha}^{t^*}, \widetilde{K}, i)$, with the hypothesis function from Definition 10. Then, with Lemma 4, we obtain $\max_{i\in[m_1]} \left|g(\widetilde{\alpha}^{t^*}, K, i) - \Gamma_i\right| \leq L\epsilon_K \left\|\widetilde{\alpha}^{t^*}\right\|_1$. Then, by Lemma 7 with $\omega = \widetilde{\alpha}^{t^*}$ and $\epsilon_I = \epsilon_K \|\widetilde{\alpha}^{t^*}\|_1$, we obtain

$$\left\|\mathbf{v}\left(\widetilde{\alpha}^{t^*}\right) - \boldsymbol{v}\right\|_2^2 - \left\|\mathbf{v}\left(\widetilde{\alpha}^{t^*+1}\right) - \boldsymbol{v}\right\|_2^2 \geq \frac{1}{L^2}\hat{\varepsilon}(h^{t^*}) - A_1 - \epsilon_K^2 \left\|\widetilde{\alpha}^{t^*}\right\|_1^2. \tag{65}$$

Note that Case 2 holds for the iteration t^*. Together with the upper bound in Eq. (62), it holds by transitivity that

$$\frac{B\eta}{L} \geq \frac{1}{L^2}\hat{\varepsilon}\left(h^{t^*}\right) - A_1 - \epsilon_K^2 \left\|\widetilde{\alpha}^{t^*}\right\|_1^2, \tag{66}$$

which implies that

$$\hat{\varepsilon}\left(h^{t^*}\right) \leq BL\eta + L^2 A_1 + L^2\epsilon_K^2 \left\|\widetilde{\alpha}^{t^*}\right\|_1^2. \tag{67}$$

Recall the definition of $A_2 \triangleq L\sqrt{\epsilon} + L\zeta\sqrt[4]{\frac{\log(1/\delta)}{m_1}} + BL\sqrt{\frac{\log(1/\delta)}{m_1}}$. Using the known bounds for η and ρ, we have

$$\hat{\varepsilon}\left(h^{t^*}\right) \in \mathcal{O}\left(A_2 + L^2\epsilon_K^2 \left\|\widetilde{\alpha}^{t^*}\right\|_1^2\right). \tag{68}$$

The last term can be bounded as

$$L^2\epsilon_K^2 \left\|\widetilde{\alpha}^{t^*}\right\|_1^2 \leq T^2\epsilon_K^2, \tag{69}$$

where we use $\|\widetilde{\alpha}^{t^*}\|_1 \leq T/L$ from Lemma 5.

As a next step, we would like to bound $\varepsilon(h^{t^*})$ in terms of $\hat{\varepsilon}(h^{t^*})$. An argument based on the Rademacher complexity gives us the same bound as in the original work [15]. Define a function class $\mathcal{Z} = \{\mathbf{x} \to u(\langle \mathbf{z}, \psi_d(\mathbf{x})\rangle) - \mathbb{E}_y[y|\mathbf{x}] : \|\mathbf{z}\|_2 \leq 2B\}$. Ref. [15] shows that the Rademacher complexity of $\mathcal{Z}$ is $\mathcal{R}_m(\mathcal{Z}) \in \mathcal{O}\left(BL\sqrt{1/m}\right)$.

Let us show that $\left\|\mathbf{v}\left(\widetilde{\alpha}^{t^*}\right)\right\|_2$ satisfies the norm bound $2B$, same as the $\mathbf{z}$ in class $\mathcal{Z}$. Note that in the first t^*-1 iterations, Case 1 holds. By Eq. (61), we have $\left\|\mathbf{v}\left(\widetilde{\alpha}^t\right) - \boldsymbol{v}\right\|_2^2 - \left\|\mathbf{v}\left(\widetilde{\alpha}^{t+1}\right) - \boldsymbol{v}\right\|_2^2 > \frac{B\eta}{L} \geq 0$, for $t \in [t^* - 1]$. In other words, the distance between $\mathbf{v}\left(\widetilde{\alpha}^t\right)$ and $\boldsymbol{v}$ decreases when t increases in $[t^* - 1]$. Thus, we conclude that

$$\left\|\mathbf{v}\left(\widetilde{\alpha}^{t^*}\right) - \boldsymbol{v}\right\|_2^2 \leq \left\|\mathbf{v}(\widetilde{\alpha}^0) - \boldsymbol{v}\right\|_2^2 = \|\boldsymbol{v}\|_2^2 \leq B^2, \tag{70}$$

and hence by the triangle inequality, $\|\mathbf{v}(\widetilde{\alpha}^{t^*})\|_2 \leq 2B$. Denote $f(\mathbf{x})$ as $h^{t^*}(\mathbf{x}) - \mathbb{E}_y[y|\mathbf{x}]$. Since $h^{t^*}(\mathbf{x}) = u\left(\left\langle \mathbf{v}\left(\widetilde{\alpha}^{t^*}\right), \psi_d(\mathbf{x})\right\rangle\right)$, the function $f(\mathbf{x})$ is an element of $\mathcal{Z}$. Define the loss function $\mathcal{L} : [0,1] \times [0,1] \to [-1,1]$ as $\mathcal{L}(a,a') = a^2$ which ignores the second argument. According to Theorem 12 in the Appendix F, with $\mathcal{J}(f,\mathcal{D}) = \varepsilon(h^{t^*})$ and $\hat{\mathcal{J}}(f,S) = \hat{\varepsilon}(h^{t^*})$ and $b = 1$, and with probability $1-\delta$,

$$\varepsilon\left(h^{t^*}\right) \leq \hat{\varepsilon}\left(h^{t^*}\right) + \mathcal{O}\left(BL\sqrt{\frac{1}{m_1}} + \sqrt{\frac{\log(1/\delta)}{m_1}}\right) \in \mathcal{O}\left(A_2 + T^2\epsilon_K^2\right). \tag{71}$$

The above proof shows the existence of a good hypothesis h^{t^*} for some $t^* \in [T]$. We define the index of the best hypothesis as

$$t' := \arg\min_{t\in[T]} \varepsilon(h^t), \tag{72}$$

which immediately implies that

$$\varepsilon(h^{t'}) \leq \varepsilon(h^{t^*}) \in \mathcal{O}\left(A_2 + T^2\epsilon_K^2\right). \tag{73}$$

In the last part of this proof, we show that at Line 6 of ALPHATRON_WITH_KERNEL, we indeed find and output a good enough hypothesis (though this hypothesis may be different from the hypothesis derived from the output of ALPHATRON_WITH_KERNEL). Our goal is to find a hypothesis h^t which minimizes $\varepsilon(\cdot)$. However, $\varepsilon(\cdot)$ is hard to compute according to the definition. From Eq. (3), we have that for arbitrary hypotheses h_1, h_2,

$$\varepsilon(h_1) - \varepsilon(h_2) = \mathrm{err}(h_1) - \mathrm{err}(h_2). \tag{74}$$

Hence, we may find the best hypothesis by minimizing $\mathrm{err}(\cdot)$ instead of $\varepsilon(\cdot)$. Formally, it holds that

$$t' = \arg\min_{t\in[T]} \mathrm{err}(h^t). \tag{75}$$

As we do not know the distribution $\mathcal{D}$, we are unable to compute $\mathrm{err}(\cdot)$. However, it is possible to compute the empirical version $\hat{\mathrm{err}}(\cdot)$. In Algorithm 2, we use a fresh sample set $(\mathbf{a}_i, b_i)$ of size m_2 as the validation data set, and we compute the empirical error $\hat{\mathrm{err}}(h^t)$, for each h^t, on this data set. Let $\epsilon' = 1/\sqrt{m_1}$. For fixed t, since $\hat{\mathrm{err}}(h^t)$ is in $[0,1]$, by a Chernoff bound on $m_2 \in \mathcal{O}\left(\log(T/\delta)/(\epsilon')^2\right)$ samples, with probability $1 - \delta/T$, we obtain

$$\left|\mathrm{err}\left(h^t\right) - \hat{\mathrm{err}}\left(h^t\right)\right| \leq \epsilon'. \tag{76}$$

Then by the union bound, with probability $1-\delta$, the inequality (76) holds simultaneously for all $t \in [T]$. Since $\epsilon' \in \mathcal{O}\left(BL\sqrt{\frac{\log 1/\delta}{m_1}}\right)$, we obtain the bound

$$\left|\mathrm{err}\left(h^t\right) - \hat{\mathrm{err}}\left(h^t\right)\right| \in \mathcal{O}\left(A_2\right), \tag{77}$$

for all $t \in [T]$. However, in Algorithm 4, to find the hypothesis with the minimum empirical error, we use the estimated kernel matrix $\widetilde{K'}$ instead of the exact inner products. Thus, we have additional errors in computing $\hat{\mathrm{err}}(h^t)$. Let

$$\widetilde{t} = \arg\min_{t\in[T]} \left\{ \frac{1}{m_2}\sum_{i=1}^{m_2}\left(u\left(\sum_{j=1}^{m_1}\alpha_i^t \cdot \widetilde{K'}_{ij}\right) - b_i\right)^2 \right\} = \arg\min_{t\in[T]} \left\{ \frac{1}{m_2}\sum_{i=1}^{m_2}\left(g\left(\alpha^t, \widetilde{K'}, i\right) - b_i\right)^2 \right\} \tag{78}$$

be the index of the hypothesis of the output in Algorithm 4. As before, the exact kernel matrix K' is $K'_{ij} = \mathcal{K}_d(\mathbf{a}_i, \mathbf{x}_j)$. According to Theorem 6, we have $\left|\widetilde{K'}_{ij} - K'_{ij}\right| \leq \epsilon_K$. Via Lemma 4, we obtain $|g(\alpha^t, \widetilde{K'}, i) - g(\alpha^t, K', i)| \leq L\epsilon_K\|\alpha^t\|_1$. By using $\epsilon_I = \epsilon_K\|\alpha^t\|_1$ and $\Gamma_i^t = g(\alpha^t, \widetilde{K'}, i)$, the upper bound on the estimation error $|\hat{\mathrm{err}}(h^{\widetilde{t}}) - \hat{\mathrm{err}}(h^{t'})|$ is shown to be in Lemma 8, as

$$\left|\hat{\mathrm{err}}\left(h^{\widetilde{t}}\right) - \hat{\mathrm{err}}\left(h^{t'}\right)\right| \leq L\epsilon_I. \tag{79}$$

We have $\|\alpha^t\|_1 \leq T/L$. Thus $\epsilon_I = \epsilon_K \|\alpha^t\|_1 \leq T\epsilon_K/L$. From Eq. (77) and Eq. (79), we obtain

$$\left|\mathrm{err}\left(h^{\widetilde{t}}\right) - \mathrm{err}\left(h^{t'}\right)\right| \in \mathcal{O}\left(L\epsilon_I + A_2\right) \subseteq \mathcal{O}\left(T\epsilon_K + A_2\right). \tag{80}$$

With Eq. (74), we obtain $\left|\varepsilon\left(h^{\widetilde{t}}\right) - \varepsilon\left(h^{t'}\right)\right| \in \mathcal{O}\left(T\epsilon_K + A_2\right)$. Hence, we have $\varepsilon\left(h^{\widetilde{t}}\right) \leq \varepsilon\left(h^{t'}\right) + \mathcal{O}\left(T\epsilon_K + A_2\right)$. From Eq. (73), $\varepsilon(h^{t'})$ is bounded by $\mathcal{O}\left(A_2 + T^2\epsilon_K^2\right)$. Thus,

$$\varepsilon(h^{\widetilde{t}}) \in \mathcal{O}\left(A_2 + T^2\epsilon_K^2 + T\epsilon_K\right). \tag{81}$$

The union bound of the probabilistic steps of estimating the full kernel matrix, the Rademacher generalization bound, and the Chernoff bound leads to a total success probability of $1 - 3\delta$. □

Since by definition $\varepsilon(h) \leq 1$, for any hypothesis h, it is reasonable to assume that $A_2 \leq 1$ if we want a useful bound. Then, by setting $\epsilon_K = \frac{A_2}{T}$, we have $\epsilon_K T \leq 1$. Thus, $\mathcal{O}\left(\epsilon_K^2 T^2\right) \subseteq \mathcal{O}\left(\epsilon_K T\right)$, and we can simplify the right hand side of Eq. (81) to $\mathcal{O}\left(A_2 + \epsilon_K T\right)$. From the runtime analysis in Theorem 6 and the accuracy analysis in Theorem 7, we have the following corollary.

Corollary 3. *In the same setting as Theorem 7, if $L\sqrt{\epsilon} \leq 1$, and we set $\epsilon_K = \frac{L\sqrt{\epsilon}}{T}$, then Algorithm 4 with probability at least $1-3\delta$ outputs $\alpha^{t_{\mathrm{out}}}$ which describes the hypothesis $h^{t_{\mathrm{out}}}(\mathbf{x}) := u\left(\sum_{i=1}^{m_1} \alpha_i^{t_{\mathrm{out}}} \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$ such that*

$$\varepsilon\left(h^{t_{\mathrm{out}}}\right) \in \mathcal{O}\left(A_2\right),$$

with a run time of

$$\widetilde{\mathcal{O}}\left(mn + \frac{m^2 d^2 T^2}{L^2 \epsilon} \log\frac{1}{\delta} + Tm^2\right).$$

C Classical sampling

The next facts discuss the construction of a data structure to sample from a vector and the next lemmas discuss the approximation of an inner product of two vectors by sampling. Both ℓ_1 and ℓ_2 cases are required in this work. The **SQ** label can be understood as "sample query". The arithmetic model allows us to assume infinite-precision storage of the real numbers.

Fact 1 (ℓ_1-sampling [31, 32])**.** *Given an n-dimensional vector $\mathbf{u} \in \mathbb{R}^n$, there exists a data structure to sample an index $j \in [n]$ with probability $|u_j|/\|\mathbf{u}\|_1$ which can be constructed in time $\widetilde{\mathcal{O}}(n)$. One sample can be obtained in time $\mathcal{O}(\log n)$. We call this data structure* $\mathbf{SQ1}(\mathbf{u}, n)$.

Fact 2 (ℓ_2-sampling [31, 32])**.** *Given an n-dimensional vector $\mathbf{u} \in \mathbb{R}^n$, there exists a data structure to sample an index $j \in [n]$ with probability $u_j^2/\|\mathbf{u}\|_2^2$ which can be constructed in time $\widetilde{\mathcal{O}}(n)$. One sample can be obtained in time $\mathcal{O}(\log n)$. We call this data structure* $\mathbf{SQ2}(\mathbf{u}, n)$.

Next, we show the estimation of inner products via sampling. The number of samples scales with $1/\epsilon^2$ classically, in contrast to using quantum amplitude estimation which scales with $1/\epsilon$. Lemma 9 is adapted from [30] and Lemma 10 is taken directly from [30].

Lemma 9 (Inner product with ℓ_1-sampling)**.** *Let $\epsilon, \delta \in (0, 1)$. Given query access to $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{SQ1}(\mathbf{u}, n)$ access to $\mathbf{u} \in \mathbb{R}^n$, we can determine $\mathbf{u} \cdot \mathbf{v}$ to additive error ϵ with success probability at least $1 - \delta$ with $\mathcal{O}\left(\frac{\|\mathbf{u}\|_1^2 \|\mathbf{v}\|_{\max}^2}{\epsilon^2} \log\frac{1}{\delta}\right)$ queries and samples, and $\widetilde{\mathcal{O}}\left(\frac{\|\mathbf{u}\|_1^2 \|\mathbf{v}\|_{\max}^2}{\epsilon^2} \log\frac{1}{\delta}\right)$ time complexity.*

Proof. Define a random variable Z with outcome $\mathrm{sgn}(u_j)\|\mathbf{u}\|_1 v_j$ with probability $|u_j|/\|\mathbf{u}\|_1$. Note that $\mathbb{E}[Z] = \sum_j \mathrm{sgn}(u_j)\|\mathbf{u}\|_1 v_j |u_j|/\|\mathbf{u}\|_1 = \mathbf{u}\cdot\mathbf{v}$. Also, $\mathbb{V}[Z] \leq \mathbb{E}[Z^2] = \sum_j \|\mathbf{u}\|_1^2 v_j^2 |u_j|/\|\mathbf{u}\|_1 \leq \|\mathbf{u}\|_1^2 \|\mathbf{v}\|_{\max}^2$. Take the median of $6\log 1/\delta$ evaluations of the mean of $9\|\mathbf{u}\|_1^2\|\mathbf{v}\|_{\max}^2/(2\epsilon^2)$ samples of Z. Then, by using the Chebyshev and Chernoff inequalities, we obtain an ϵ additive error estimation of $\mathbf{u}\cdot\mathbf{v}$ with probability at least $1-\delta$ in $\mathcal{O}\left(\frac{\|\mathbf{u}\|_1^2\|\mathbf{v}\|_{\max}^2}{\epsilon^2}\log\frac{1}{\delta}\right)$ queries. □

Lemma 10 (Inner product with ℓ_2-sampling)**.** *Let $\epsilon, \delta \in (0,1)$. Given query access to $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{SQ2}(\mathbf{u}, n)$ access to $\mathbf{u} \in \mathbb{R}^n$, we can determine $\mathbf{u}\cdot\mathbf{v}$ to additive error ϵ with success probability at least $1-\delta$ with $\mathcal{O}\left(\frac{\|\mathbf{u}\|_2^2\|\mathbf{v}\|_2^2}{\epsilon^2}\log\frac{1}{\delta}\right)$ queries and samples, and $\widetilde{\mathcal{O}}\left(\frac{\|\mathbf{u}\|_2^2\|\mathbf{v}\|_2^2}{\epsilon^2}\log\frac{1}{\delta}\right)$ time complexity.*

Proof. Define a random variable Z with outcome $\|\mathbf{u}\|_2^2 v_j/u_j$ with probability $u_j^2/\|\mathbf{u}\|_2^2$. Note that $\mathbb{E}[Z] = \sum_j \|\mathbf{u}\|_2^2 v_j u_j^2/(u_j\|\mathbf{u}\|_2^2) = \mathbf{u}\cdot\mathbf{v}$. Also, $\mathbb{V}[Z] \leq \mathbb{E}[Z^2] = \sum_j \|\mathbf{u}\|_2^2 v_j^2 = \|\mathbf{u}\|_2^2\|\mathbf{v}\|_2^2$. Take the median of $6\log 1/\delta$ evaluations of the mean of $9\|\mathbf{u}\|_2^2\|\mathbf{v}\|_2^2/(2\epsilon^2)$ samples of Z. Then, by using the Chebyshev and Chernoff inequalities, we obtain an ϵ additive error estimation of $\mathbf{u}\cdot\mathbf{v}$ with probability at least $1-\delta$ in $\mathcal{O}\left(\frac{\|\mathbf{u}\|_2^2\|\mathbf{v}\|_2^2}{\epsilon^2}\log\frac{1}{\delta}\right)$ queries. □

By the above Fact 2 and Lemma 10, given vector $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, the sampling data structure for $\mathbf{u}$ can be constructed in $\widetilde{\mathcal{O}}(n)$ time and an estimation of $\mathbf{u}\cdot\mathbf{v}$ with ϵ additive error can be obtained with probability at least $1-\delta$ at a run time cost of $\widetilde{\mathcal{O}}\left(\frac{\|\mathbf{u}\|_2\|\mathbf{v}\|_2}{\epsilon^2}\log\frac{1}{\delta}\right)$.

D Quantum subroutines

First, we recall the quantum access used for vectors.

Definition 11 (Quantum query access)**.** *Let c and n be two positive integers and $\mathbf{u}$ be a vector of bit strings $\mathbf{u} \in (\{0,1\}^c)^n$. Define element-wise quantum access to $\mathbf{u}$ for $j \in [n]$ by the operation*

$$|j\rangle|0^c\rangle \to |j\rangle|u_j\rangle, \tag{82}$$

on $\mathcal{O}(c+\log n)$ qubits. We denote this access by $\mathbf{QA}(\mathbf{u}, n, c)$.

For the following part of this Appendix, recall Definition 1 regarding the fixed-point encoding of real numbers. In addition, we define the quantum sample access to a normalized semi-positive vector $\mathbf{v}/\|\mathbf{v}\|_1$ which is a fixed-point approximation of a real semi-positive vector. Each component of the vector $\mathbf{v}$ is represented with c_1 bits before the decimal point and with c_2 bits after the decimal point.

Definition 12 (Quantum sample access)**.** *Let c_1, c_2, and n be positive integers and $\mathbf{v}' \in (\{0,1\}^{c_1})^n$ and $\mathbf{v}'' \in (\{0,1\}^{c_2})^n$ be vectors of bit strings. Define quantum sample access to a vector $\mathbf{v}$ via the operation*

$$|\bar{0}\rangle \to \frac{1}{\sqrt{\|\mathcal{Q}(\mathbf{v}', \mathbf{v}'')\|_1}} \sum_{j=1}^{n} \sqrt{\mathcal{Q}(v_j', v_j'')}\,|j\rangle, \tag{83}$$

on $\mathcal{O}(\log n)$ qubits. We denote this access by $\mathbf{QS}(\mathbf{v}, n, c_1, c_2)$. For the sample access to a vector $\mathbf{v}$ which approximates a vector with components in $[0,1]$, we use the shorthand notation $\mathbf{QS}(\mathbf{v}, n, c_2) := \mathbf{QS}(\mathbf{v}, n, 1, c_2)$.

As stated in [26], we have the following lemma for estimating the ℓ_1-norm of a vector with entries in $[0,1]$ and preparing states encoding the square root of the vector elements.

Lemma 11 (Quantum state preparation and norm estimation)**.** *Let c and n be two positive integers and $\mathbf{u} \in (\{0,1\}^{c+1})^n$. Assume quantum access to $\mathbf{u}$ via $\mathbf{QA}(\mathbf{u}, n, c+1)$. Let $\max_j \mathcal{Q}(u_j) = 1$. Then:*

1. *There exists a quantum circuit that prepares the state $\frac{1}{\sqrt{n}} \sum_{j=1}^{n} |j\rangle \left(\sqrt{\mathcal{Q}(u_j)} |0\rangle + \sqrt{1 - \mathcal{Q}(u_j)} |1\rangle \right)$ with two queries to $\mathbf{QA}(\mathbf{u}, n, c)$ and $\mathcal{O}(\log n + c)$ gates.*
2. *Let $\epsilon, \delta \in (0,1)$. There exists a quantum algorithm that provides an estimate $\Gamma_{\mathbf{u}}$ of the ℓ_1-norm $\|\mathcal{Q}(\mathbf{u})\|_1$ such that $|\|\mathcal{Q}(\mathbf{u})\|_1 - \Gamma_{\mathbf{u}}| \leq \epsilon \|\mathcal{Q}(\mathbf{u})\|_1$, with probability at least $1 - \delta$. The quantum circuit of this algorithm makes $\mathcal{O}\left(\frac{1}{\epsilon} \sqrt{\frac{n}{\|\mathcal{Q}(\mathbf{u})\|_1}} \log(1/\delta) \right)$ queries to $\mathbf{QA}(\mathbf{u}, n, c+1)$ and has $\widetilde{\mathcal{O}}\left(\frac{c}{\epsilon} \sqrt{\frac{n}{\|\mathcal{Q}(\mathbf{u})\|_1}} \log(1/\delta) \right)$ gates.*

By using Lemma 11, we estimate the inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$ with additive errors as follows. The vectors can be considered as fixed point approximations to real vectors with elements restricted to $[-1, 1]$.

Lemma 12 (Quantum inner product estimation with additive accuracy)**.** *Let $\epsilon, \delta \in (0,1)$. Let c and n be two positive integers. Let two non-zero vectors of bit strings be $\mathbf{u}, \mathbf{v} \in (\{0,1\}^{c+2})^n$, which leaves one bit for the sign of each component, one bit for the number before the decimal point, and c bits for the number after the decimal point. Let there be given quantum access to $\mathbf{u}$ and $\mathbf{v}$ as $\mathbf{QA}(\mathbf{u}, n, c+2)$ and $\mathbf{QA}(\mathbf{u}, n, c+2)$, respectively. Let the norms $\|\mathcal{Q}(\mathbf{u})\|_2$ and $\|\mathcal{Q}(\mathbf{v})\|_2$ be known. Then, there exists a quantum algorithm which provides an estimate I for the inner product such that $|I - \mathcal{Q}(\mathbf{u}) \cdot \mathcal{Q}(\mathbf{v})| \leq \epsilon$ with success probability $1 - \delta$. This estimate is obtained with $\mathcal{O}\left(\left(\frac{\|\mathcal{Q}(\mathbf{u})\|_2 \|\mathcal{Q}(\mathbf{v})\|_2}{\epsilon} + 1 \right) \sqrt{n} \log\left(\frac{1}{\delta} \right) \right)$ queries and $\widetilde{\mathcal{O}}\left(\left(\frac{\|\mathcal{Q}(\mathbf{u})\|_2 \|\mathcal{Q}(\mathbf{v})\|_2}{\epsilon} + 1 \right) \sqrt{n} \log\left(\frac{1}{\delta} \right) \right)$ quantum gates.*

Proof. Define the vectors $\mathbf{u}^+$ and $\mathbf{u}^-$ as follows

$$u_i^+ := \begin{cases} u_i & \text{if } \operatorname{sign}(u_i) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad u_i^- = \begin{cases} 0 & \text{if } \operatorname{sign}(u_i) = 1 \\ -u_i & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathcal{Q}(\mathbf{u}) = \mathcal{Q}(\mathbf{u}^+) - \mathcal{Q}(\mathbf{u}^-)$. Define the vectors $\mathbf{v}^+$ and $\mathbf{v}^-$ in a similar way. Then,

$$\mathcal{Q}(\mathbf{u}) \cdot \mathcal{Q}(\mathbf{v}) = \mathcal{Q}(\mathbf{u}^+) \cdot \mathcal{Q}(\mathbf{v}^+) + \mathcal{Q}(\mathbf{u}^-) \cdot \mathcal{Q}(\mathbf{v}^-) - \mathcal{Q}(\mathbf{u}^+) \cdot \mathcal{Q}(\mathbf{v}^-) - \mathcal{Q}(\mathbf{u}^-) \cdot \mathcal{Q}(\mathbf{v}^+). \tag{84}$$

Define two more vectors of bit strings $\mathbf{z}^+$ and $\mathbf{z}^-$ from $\mathcal{Q}(z_i^+) = \mathcal{Q}(u_i^+)\mathcal{Q}(v_i^+) + \mathcal{Q}(u_i^-)\mathcal{Q}(v_i^-)$ and $\mathcal{Q}(z_i^-) = \mathcal{Q}(u_i^+)\mathcal{Q}(v_i^-) + \mathcal{Q}(u_i^- v_i^+)$. Then

$$\mathcal{Q}(\mathbf{u}) \cdot \mathcal{Q}(\mathbf{v}) = \|\mathcal{Q}(\mathbf{z}^+)\|_1 - \|\mathcal{Q}(\mathbf{z}^-)\|_1. \tag{85}$$

In the following, we use the standard $\pm$ notation to denote that a statement holds for both the $+$ and the $-$ case. Determine the index of $z_{\max}^{\pm} := \|\mathcal{Q}(\mathbf{z}^{\pm})\|_{\max}$ with the quantum maximum finding algorithm with success probability $1 - \delta/4$, with $\mathcal{O}\left(\sqrt{n} \log\left(\frac{1}{\delta} \right) \right)$ queries and $\widetilde{\mathcal{O}}\left(\sqrt{n} \log\left(\frac{1}{\delta} \right) \right)$ quantum gates [11]. In case that $z_{\max}^{\pm} = 0$, we infer that $\mathbf{z}^{\pm} = \mathbf{0}$, and if both are true we return the estimate 0. Otherwise, for non-zero vector $\mathbf{z}^{\pm}$, we apply Statement 2 of Lemma 11 on the vectors of bit strings corresponding to $\mathcal{Q}(\mathbf{z}^{\pm})/z_{\max}^{\pm}$, respectively. These vectors of bit strings can be computed efficiently from the query access and the result of the maximum finding. We obtain estimates Γ^+ and Γ^- such that

$$\left| \left\| \frac{\mathcal{Q}(\mathbf{z}^{\pm})}{z_{\max}^{\pm}} \right\|_1 - \Gamma^{\pm} \right| \leq \epsilon' \left\| \frac{\mathcal{Q}(\mathbf{z}^{\pm})}{z_{\max}^{\pm}} \right\|_1, \tag{86}$$

with success probability at least $1-\delta/4$ for each of them, with $\mathcal{O}\left(\frac{1}{\epsilon'}\sqrt{\frac{nz^{\pm}_{\max}}{\|\mathcal{Q}(\mathbf{z}^{\pm})\|_1}}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon'}\sqrt{\frac{nz^{\pm}_{\max}}{\|\mathcal{Q}(\mathbf{z}^{\pm})\|_1}}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates. Note that

$$\begin{aligned}\|\mathcal{Q}(\mathbf{z}^+)\|_1 &= \mathcal{Q}(\mathbf{u}^+)\cdot\mathcal{Q}(\mathbf{v}^+)+\mathcal{Q}(\mathbf{u}^-)\cdot\mathcal{Q}(\mathbf{v}^-) &(87)\\ &\leq \|\mathcal{Q}(\mathbf{u}^+)\|_2\|\mathcal{Q}(\mathbf{v}^+)\|_2+\|\mathcal{Q}(\mathbf{u}^-)\|_2\|\mathcal{Q}(\mathbf{v}^-)\|_2\leq 2\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2, &(88)\end{aligned}$$

where the first inequality follows from Cauchy-Schwarz. Similarly, we have that $\|\mathcal{Q}(\mathbf{z}^-)\|_1 \leq 2\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2$.

Hence, we obtain an estimate $I = z^+_{\max}\Gamma^+ - z^-_{\max}\Gamma^-$ such that

$$\begin{aligned}|\mathcal{Q}(\mathbf{u})\cdot\mathcal{Q}(\mathbf{v})-I| &= |\|\mathcal{Q}(\mathbf{z}^+)\|_1-\|\mathcal{Q}(\mathbf{z}^-)\|_1-(z^+_{\max}\Gamma^+-z^-_{\max}\Gamma^-)|\\ &\leq |\|\mathcal{Q}(\mathbf{z}^+)\|_1-z^+_{\max}\Gamma^+|+|\|\mathcal{Q}(\mathbf{z}^-)\|_1-z^-_{\max}\Gamma^-|\\ &\leq 4\epsilon'\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2.\end{aligned}$$

Since $\|\mathcal{Q}(\mathbf{u})\|_2$ and $\|\mathcal{Q}(\mathbf{v})\|_2$ are given, choosing $\epsilon' = \epsilon/(4\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2)$ leads to the result. The run time of the $\pm$ estimation is then, using $\epsilon' = \epsilon/(4\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2)$,

$$\mathcal{O}\left(\frac{\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2\sqrt{n}}{\epsilon}\sqrt{\frac{z^{\pm}_{\max}}{\|\mathcal{Q}(\mathbf{z}^{\pm})\|_1}}\log\left(\frac{1}{\delta}\right)\right).$$

In the absence of more knowledge about the vectors, we take the bound $z^{\pm}_{\max}/\|\mathcal{Q}(\mathbf{z}^{\pm})\|_1 \leq 1$. Then the run time is $\mathcal{O}\left(\frac{\|\mathcal{Q}(\mathbf{u})\|_2\|\mathcal{Q}(\mathbf{v})\|_2\sqrt{n}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$. Combining these resource bounds with the resource bounds for maximum finding leads to the stated result. □

With the following Lemma, we can remove the explicit dimension dependence of the inner product estimation. For this lemma we suppose that one vector is given via quantum query access as before and that the other vector is given via access to a quantum subroutine that prepares an amplitude encoding of the vector. In our work, the quantum sampling access is provided via QRAM in Definition 7. The vectors in this lemma are considered to be fixed-point approximations to real vectors with elements restricted to $[0,1]$.

Lemma 13 (Inner product estimation with quantum sampling and query access)**.** *Let c and n be two positive integers. Let $\mathbf{u} \in (\{0,1\}^{c+1})^n$ be a non-zero vector of bit strings, and let $\mathbf{v} \in (\{0,1\}^{c+1})^n$ be another vector of bit strings. Assume quantum query access to $\mathbf{u}$ via $\mathbf{QA}(\mathbf{u}, n, c+1)$, and quantum sample access $\mathbf{v}$ via $\mathbf{QS}(\mathbf{v}, n, c+1)$. Then:*

(i) If $\max_j \mathcal{Q}(u_j) = 1$, then there exists a quantum circuit that prepares the state

$$\frac{1}{\sqrt{\|\mathcal{Q}(\mathbf{v})\|_1}}\sum_{j=1}^{n}\sqrt{\mathcal{Q}(v_j)}\,|j\rangle\left(\sqrt{\mathcal{Q}(u_j)}\,|0\rangle+\sqrt{1-\mathcal{Q}(u_j)}\,|1\rangle\right)$$

with three queries and $\mathcal{O}(c+\log n)$ additional gates.

(ii) Let $\epsilon, \delta \in (0,1)$. If $\max_j \mathcal{Q}(u_j) = 1$, then there exists a quantum algorithm that provides an estimate Γ of $\frac{\mathcal{Q}(\mathbf{v})\cdot\mathcal{Q}(\mathbf{u})}{\|\mathcal{Q}(\mathbf{v})\|_1}$ such that $\left|\frac{\mathcal{Q}(\mathbf{v})\cdot\mathcal{Q}(\mathbf{u})}{\|\mathcal{Q}(\mathbf{v})\|_1}-\Gamma\right| \leq \epsilon$, with probability at least $1-\delta$. The algorithm requires $\mathcal{O}\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$ gates.

(iii) Let $\epsilon, \delta \in (0,1)$. Let the norm $\|\mathcal{Q}(\mathbf{v})\|_1$ and $j_{\max} := \arg\max_j \mathcal{Q}(u_j)$ be known. There is a quantum algorithm, similar to (ii), which provides an estimate Γ' of $\mathcal{Q}(\mathbf{v})\cdot\mathcal{Q}(\mathbf{u})$ such that $|\mathcal{Q}(\mathbf{v})\cdot\mathcal{Q}(\mathbf{u})-\Gamma'| \leq \epsilon$, with probability at least $1-\delta$. The algorithm requires $\mathcal{O}\left(\frac{\|\mathcal{Q}(\mathbf{v})\|_1\mathcal{Q}(u_{j_{\max}})}{\epsilon}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\|\mathcal{Q}(\mathbf{v})\|_1\mathcal{Q}(u_{j_{\max}})}{\epsilon}\log\frac{1}{\delta}\right)$ gates.

Proof. For (i), with quantum sample access and the quantum query access, perform

$$|\bar{0}\rangle|\bar{0}\rangle|0\rangle \to \frac{1}{\sqrt{\|\mathcal{Q}(\mathbf{v})\|_1}}\sum_{j=1}^{n}\sqrt{\mathcal{Q}(v_j)}|j\rangle|\bar{0}\rangle|0\rangle \to \frac{1}{\sqrt{\|\mathcal{Q}(\mathbf{v})\|_1}}\sum_{j=1}^{n}\sqrt{\mathcal{Q}(v_j)}|j\rangle|u_j\rangle|0\rangle \tag{89}$$
$$\to \frac{1}{\sqrt{\|\mathcal{Q}(\mathbf{v})\|_1}}\sum_{j=1}^{N}\sqrt{\mathcal{Q}(v_j)}|j\rangle|u_j\rangle\left(\sqrt{\mathcal{Q}(u_j)}|0\rangle+\sqrt{1-\mathcal{Q}(u_j)}|1\rangle\right).$$

The first step consists of an oracle query to the vector $\mathbf{v}$ on the first register. The second step consists of an oracle query to the vector $\mathbf{u}$ which puts the vector component in the second register depending on the index in the first register. The last step consists of a controlled rotation. The rotation is well-defined as $\mathcal{Q}(u_j) \leq \max_j \mathcal{Q}(u_j) = 1$ and can be implemented with $\mathcal{O}(c)$ gates. Then we uncompute the data register $|u_j\rangle$ with another oracle query.

For (ii), define a unitary $\mathcal{U} = U_1(\mathbb{I} - 2|\bar{0}\rangle\langle\bar{0}|)U_1^\dagger$, where U_1 is the unitary obtained in (i). Define another unitary by $\mathcal{V} = \mathbb{I} - 2\mathbb{I}\otimes|0\rangle\langle 0|$. Using K applications of $\mathcal{U}$ and $\mathcal{V}$, Amplitude Estimation [8] allows to provide an estimation $\tilde{a}$ of the quantity $a = \frac{\mathcal{Q}(\mathbf{v})\cdot\mathcal{Q}(\mathbf{u})}{\|\mathcal{Q}(\mathbf{v})\|_1}$ to accuracy

$$|\tilde{a}-a| \leq 2\pi\frac{\sqrt{a(1-a)}}{K}+\frac{\pi^2}{K^2}. \tag{90}$$

Note that $0 \leq a = \frac{\mathcal{Q}(\mathbf{v})\cdot\mathcal{Q}(\mathbf{u})}{\|\mathcal{Q}(\mathbf{v})\|_1} \leq \max_j \mathcal{Q}(u_j) = 1$. Set $K > \frac{3\pi}{\epsilon}$. Then we obtain

$$|\tilde{a}-a| \leq \frac{\pi}{K}\left(2\sqrt{a}+\frac{\pi}{K}\right) < \frac{\epsilon}{3}\left(2\sqrt{a}+\frac{\epsilon}{3}\right) \leq \frac{\epsilon}{3}3 \leq \epsilon. \tag{91}$$

Performing a single run of amplitude estimation with K steps requires $\mathcal{O}(K) = \mathcal{O}\left(\frac{1}{\epsilon}\right)$ queries to the oracles and $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ gates and succeeds with probability $8/\pi^2$. The success probability can be boosted to $1-\delta$ with $\mathcal{O}(\log(1/\delta))$ repetitions of amplitude estimation.

For (iii), from the index $j_{\max} = \arg\max_j \mathcal{Q}(u_j)$ we can obtain the bit string $u_{j_{\max}}$ and its corresponding value $\mathcal{Q}(u_{j_{\max}})$. This allows us to prepare the quantum circuit for the transformation

$$|u_j\rangle|0\rangle \to |j\rangle|u_j\rangle\left(\sqrt{\frac{\mathcal{Q}(u_j)}{\mathcal{Q}(u_{j_{\max}})}}|0\rangle+\sqrt{1-\frac{\mathcal{Q}(u_j)}{\mathcal{Q}(u_{j_{\max}})}}|1\rangle\right), \tag{92}$$

from the original query access to $\mathbf{u}$ and basic arithmetic quantum circuits for the division. Then we run the same steps as in (ii) with vector $\mathcal{Q}(\mathbf{u})/\mathcal{Q}(u_{j_{\max}})$, quantum sample access to vector $\mathbf{v}$, and error parameter $\epsilon/(\|\mathcal{Q}(\mathbf{v})\|_1 Q(u_{j_{\max}}))$. We obtain an estimate Γ from (ii) such that

$$\left|\Gamma - \frac{\mathcal{Q}(\mathbf{v})}{\|\mathcal{Q}(\mathbf{v})\|_1}\cdot\frac{\mathcal{Q}(\mathbf{u})}{\mathcal{Q}(u_{j_{\max}})}\right| \leq \frac{\epsilon}{\|\mathcal{Q}(\mathbf{v})\|_1\mathcal{Q}(u_{j_{\max}})}. \tag{93}$$

Then by multiplying both sides of (93) with $\|\mathcal{Q}(\mathbf{v})\|_1\mathcal{Q}(u_{j_{\max}})$, we obtain the required estimate $\Gamma' = \Gamma\|\mathcal{Q}(\mathbf{v})\|_1\mathcal{Q}(u_{j_{\max}})$. □

E Combined algorithm

In this section, we combine the algorithm of the kernel matrix pre-computation and gradient estimation for the learning process. Note that the kernel matrix estimation step introduces an additional error during the process. The classical algorithm is as follows.

The corresponding guarantee and run time is given by the following result.

Algorithm 8: ALPHATRON_COMBINED

Input *Training data* $(\mathbf{x}_i, y_i)_{i=1}^m$ *and testing data* $(\mathbf{a}_i, b_i)_{i=1}^N$*, error tolerance parameter* ϵ_I*, failure probability* δ*, function* $u : \mathbb{R} \to [0,1]$*, number of iterations* T*, degree of the multinomial kernel* d*, learning rate* λ

$\tilde{K}_{ij}, \tilde{K}'_{ij} \leftarrow$ Call ALPHATRON_WITH_APPROX_PRE (Algorithm 4) with all inputs as above.

Prepare query access to $\tilde{K}_{ij}, \tilde{K}'_{ij}$ as Data Input 2.

$\alpha^{t_{out}} \leftarrow$ Call ALPHATRON_WITH_KERNEL_AND_SAMPLING(Algorithm 6) with all inputs as above and $\widetilde{K}_j$ and $\widetilde{K}'_j$ via Data Input 2 as prepared.

Output $\alpha^{t_{out}}$

Corollary 4 (Alphatron combined)**.** *Given training data* $(\mathbf{x}_i, y_i)_{i=1}^m$ *and testing data* $(\mathbf{a}_i, b_i)_{i=1}^N$*. Let* $K_{\max}$ *be maximum of all entries in* K *and* K'*, which are the corresponding kernel matrices. Let* $\epsilon_I, \delta \in (0,1)$*. If the Definitions 3 and 4 hold and* $A_2 \leq 1$ *and with* $\epsilon_I = \sqrt{\epsilon}$*, the Algorithm 8 outputs* $\alpha^{t_{\text{out}}}$ *which describes the hypothesis* $h^{t_{\text{out}}}(\mathbf{x}) := u\left(\sum_{i=1}^{m_1} \alpha_i^{t_{\text{out}}} \mathcal{K}_d(\mathbf{x}, \mathbf{x}_i)\right)$ *such that with probability* $1 - 4\delta$*,*

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}(A_2).$$

The run time for obtaining this output is in

$$\widetilde{\mathcal{O}}\left(mn + \frac{T^2 m^2 d^2}{\epsilon} \log\left(\frac{1}{\delta}\right) + Tm + T^3 m \frac{K_{\max}^2}{L^2 \epsilon} \log\left(\frac{1}{\delta}\right)\right).$$

Proof. The proof is the same as Theorem 8, except for the inequalities as follows. Here, we use ALPHATRON_WITH_APPROX_PRE to estimate the kernel matrices with precision ϵ_K, which are denoted by $\tilde{K}_{ij}$ and $\tilde{K}'_{ij}$ respectively. Therefore, with ALPHATRON_WITH_KERNEL_AND_SAMPLING, we estimate the inner product $\alpha^t \cdot \tilde{K}_j$ instead of $\alpha^t \cdot K_j$, to additive accuracy ϵ'_I. Let the estimated value be r_j^t. To promise

$$\begin{aligned}
\left| r_j^t - \sum_{i=1}^{m_1} \alpha_i^t K_{ji} \right| &= \left| r_j^t - \sum_{i=1}^{m_1} \alpha_i^t \tilde{K}_{ji} + \sum_{i=1}^{m_1} \alpha_i^t \tilde{K}_{ji} - \sum_{i=1}^{m_1} \alpha_i^t K_{ji} \right| && (94)\\
&\leq \left| r_j^t - \sum_{i=1}^{m_1} \alpha_i^t \tilde{K}_{ji} \right| + \left| \sum_{i=1}^{m_1} \alpha_i^t \tilde{K}_{ji} - \sum_{i=1}^{m_1} \alpha_i^t K_{ji} \right| && (95)\\
&\leq \epsilon'_I + 2\lambda T \epsilon_K \leq \epsilon_I, && (96)
\end{aligned}$$

it suffices to take $\epsilon'_I = \epsilon_I/2$ and $\epsilon_K = \epsilon_I/(4\lambda T)$. Recall that by Lemma 5, $0 < \alpha_{\max} \leq \lambda T/m_1$. For the run time, storing $\tilde{K}_{ij}$ and $\tilde{K}'_{ij}$ takes time $\mathcal{O}(m^2)$, which is included in the quoted run time. Set $\epsilon_I = \sqrt{\epsilon}$, we obtain the run time from Theorem 8. □

We continue with the corresponding quantum algorithm. For the result, we slightly change the proof for Theorem 9.

Corollary 5 (Quantum Alphatron combined)**.** *We assume quantum query access to the vectors* $\mathbf{x}_i$ *and* $\mathbf{a}_i$ *via Data Input 3. Let* $K_{\max}$ *be maximum of all entries in* K *and* K'*, which are the corresponding kernel matrices. Let* $\delta \in (0,1)$*. Given Definitions 3 and 4 with* $A_2 \leq 1$ *and with* $\epsilon_I = \sqrt{\epsilon}$*, Algorithm 9 outputs* $\alpha^{t_{out}}$ *such that the hypothesis* $h^{t_{\text{out}}} = u\left(\sum_j \alpha_j^{t_{\text{out}}} \psi(\mathbf{x}_j) \cdot \psi(\mathbf{x})\right)$ *satisfies with probability* $1 - 4\delta$*,*

$$\varepsilon(h^{t_{\text{out}}}) \in \mathcal{O}(A_2).$$

Algorithm 9: QUANTUM_ALPHATRON_COMBINED

Input *Quantum access to training data* $(\mathbf{x}_i, y_i)_{i=1}^m$ *and testing data* $(\mathbf{a}_i, b_i)_{i=1}^N$ *according to Data Input 1, error tolerance parameter* ϵ_I*, failure probability* δ*, function* $u : \mathbb{R} \to [0,1]$*, number of iterations* T*, degree of the multinomial kernel* d*, learning rate* λ

$\tilde{K}_{ij}, \tilde{K}'_{ij} \leftarrow$ Call ALPHATRON_WITH_Q_PRE(Algorithm 5) with all inputs as above.

Store $\tilde{K}_{ij}, \tilde{K}'_{ij}$ via quantum random access memory (QRAM) and prepare the quantum query access to $\tilde{K}_j, \tilde{K}'_j$ as Data Input 3.

$\alpha^{t_{out}} \leftarrow$ Call QUANTUM_ALPHATRON(Algorithm 7) with all inputs as above and $\widetilde{K}_j$ and $\widetilde{K}'_j$ via Data Input 3 as prepared.

Output $\alpha^{t_{out}}$

The run time of this algorithm is

$$\widetilde{\mathcal{O}}\left(\frac{Tm^2d\sqrt{n}}{\sqrt{\epsilon}}\log\left(\frac{1}{\delta}\right) + m^{1.5}\log\left(\frac{1}{\delta}\right) + Tm + T^2m\frac{K_{\max}}{L\sqrt{\epsilon}}\log\left(\frac{1}{\delta}\right)\right).$$

Proof. The proof is the same with Theorem 9, except that instead of Eq. (21), we consider

$$\begin{aligned}
\left|\alpha^t \cdot K_i - \tilde{\alpha}^t \cdot \tilde{K}_i\right| &= \left|\alpha^t \cdot K_i - \tilde{\alpha}^t \cdot K_i + \tilde{\alpha}^t \cdot K_i - \tilde{\alpha}^t \cdot \tilde{K}_i\right| \\
&\leq \left|\alpha^t \cdot K_i - \tilde{\alpha}^t \cdot K_i\right| + \left|\tilde{\alpha}^t \cdot K_i - \tilde{\alpha}^t \cdot \tilde{K}_i\right| \\
&\leq \frac{m_1 K_{\max}}{2^{c_2+1}} + \tilde{\alpha}^t m_1 \epsilon_K \\
&\leq \frac{m_1 K_{\max}}{2^{c_2+1}} + 2\lambda T \epsilon_K \leq \frac{\epsilon_I}{2}, \qquad (97)
\end{aligned}$$

where the last inequality can be achieved by setting $c_2 \geq \left\lceil \log\left(\frac{2K_{\max}m_1}{\epsilon_I}\right) + 1 \right\rceil$, $\epsilon_K \leq \epsilon_I/(8\lambda T)$. Recall that by Lemma 5, $0 < \alpha_{\max} \leq \lambda T/m_1$. The rest follows as in Theorem 9. Storage in QRAM costs $\mathcal{O}(m^2)$. Set $\epsilon_I = \sqrt{\epsilon}$, we obtain the run time from Theorem 9. □

F Rademacher complexity

The following standard generalization bound based on Rademacher complexity is employed in our analysis. For a background on Rademacher complexity, we refer the reader to [4].

Theorem 12 (Generalization bound [4])**.** *Let* $\mathcal{D}$ *be a distribution over* $\mathcal{X} \times \mathcal{Y}$ *and let* $\mathcal{L} : \mathcal{Y}' \times \mathcal{Y} \to [-b, b]$ *(where* $\mathcal{Y} \subseteq \mathcal{Y}' \subseteq \mathbb{R}$*) be a* b*-bounded loss function that is* L*-Lipschitz in its first argument. Let* $\mathcal{F} \subseteq (\mathcal{Y}')^{\mathcal{X}}$ *and for any* $f \in \mathcal{F}$*, let* $\mathcal{J}(f, \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathcal{L}(f(\boldsymbol{x}), y)]$ *and* $\hat{\mathcal{J}}(f, S) = \frac{1}{m}\sum_{i=1}^m \mathcal{L}(f(\boldsymbol{x_i}), y_i)$*, where* $S = ((\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)) \sim \mathcal{D}^m$*. Then for any* $\delta > 0$*, with probability at least* $1-\delta$ *(over the random sample draw for* S*), simultaneously for all* $f \in \mathcal{F}$*, the following is true:*

$$|\mathcal{J}(f, \mathcal{D}) - \hat{\mathcal{J}}(f, S)| \leq 4 \cdot L \cdot \mathcal{R}_{\boldsymbol{m}}(\mathcal{F}) + 2 \cdot b \cdot \sqrt{\frac{\log(1/\delta)}{2m}}$$

where $\mathcal{R}_{\boldsymbol{m}}(\mathcal{F})$ *is the Rademacher complexity of the function class* $\mathcal{F}$*.*

References

[1] Jonathan Allcock et al. "Quantum Algorithms for Feedforward Neural Networks". In: *ACM Transactions on Quantum Computing* 1.1 (2020). ISSN: 2643-6809. DOI: 10.1145/3411466.

[2] J. van Apeldoorn and A. Gilyén. "Quantum algorithms for zero-sum games". In: *arXiv:1904.03180* (2019). DOI: 10.48550/arXiv.1904.03180.

[3] Srinivasan Arunachalam et al. "On the robustness of bucket brigade quantum RAM". In: *New Journal of Physics* 17.12 (2015), p. 123010. DOI: 10.1088/1367-2630/17/12/123010.

[4] Peter L. Bartlett and Shahar Mendelson. "Rademacher and Gaussian Complexities: Risk Bounds and Structural Results". In: *J. Mach. Learn. Res.* 3.null (2003), 463–482. ISSN: 1532-4435. DOI: 10.5555/944919.944944.

[5] Kerstin Beer et al. "Training deep quantum neural networks". In: *Nature Communications* 11.1 (2020), p. 808. DOI: 10.1038/s41467-020-14454-2.

[6] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (2017), pp. 195–202. DOI: 10.1038/nature23474.

[7] Fernando G.S.L. Brandao and Krysta M. Svore. "Quantum Speed-Ups for Solving Semidefinite Programs". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 415–426. DOI: 10.1109/FOCS.2017.45.

[8] Gilles Brassard et al. "Quantum amplitude amplification and estimation". In: *Contemporary Mathematics* 305 (2002), pp. 53–74. DOI: 10.1090/conm/305/05215.

[9] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. "Quantum Neuron: an elementary building block for machine learning on quantum computers". In: *arXiv:1711.11240* (2017). DOI: 10.48550/arXiv.1711.11240.

[10] C. Ciliberto et al. "Quantum machine learning: A classical perspective". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (2018), p. 20170551. DOI: 10.1098/rspa.2017.0551.

[11] C. Dürr and P. Høyer. "A quantum algorithm for finding the minimum". In: *arXiv:9607014* (1996). DOI: 10.48550/arXiv.quant-ph/9607014.

[12] Andrá s Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. "Optimizing quantum optimization algorithms via faster quantum gradient computation". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2019, pp. 1425–1444. DOI: 10.1137/1.9781611975482.87.

[13] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Architectures for a quantum random access memory". In: *Phys. Rev. A* 78 (5 2008), p. 052310. DOI: 10.1103/PhysRevA.78.052310.

[14] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum Random Access Memory". In: *Phys. Rev. Lett.* 100 (16 2008), p. 160501. DOI: 10.1103/PhysRevLett.100.160501.

[15] Surbhi Goel and Adam R. Klivans. "Learning Neural Networks with Two Nonlinear Layers in Polynomial Time". In: *Proceedings of the Thirty-Second Conference on Learning Theory*. Ed. by Alina Beygelzimer and Daniel Hsu. Vol. 99. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1470–1499. DOI: 10.48550/arXiv.1709.06010.

[16] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions". In: *arXiv preprint quant-ph/0208112* (2002). DOI: 10.48550/arXiv.quant-ph/0208112.

[17] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Phys. Rev. Lett.* 103 (15 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502.

[18] Vojtěch Havlíček et al. "Supervised learning with quantum-enhanced feature spaces". In: *Nature* 567.7747 (2019), pp. 209–212. DOI: 10.1038/s41586-019-0980-2.

[19] M.J. Kearns and R.E. Schapire. "Efficient distribution-free learning of probabilistic concepts". In: *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. 1990, 382–391 vol.1. DOI: 10.1109/FSCS.1990.89557.

[20] Adam Klivans and Raghu Meka. "Learning Graphical Models Using Multiplicative Weights". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 2017, pp. 343–354. DOI: 10.1109/FOCS.2017.39.

[21] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. "Sublinear quantum algorithms for training linear and kernel-based classifiers". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3815–3824. DOI: 10.48550/arXiv.1904.02276.

[22] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. "On the Computational Efficiency of Training Neural Networks". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'14. MIT Press, 2014, 855–863. DOI: 10.5555/2968826.2968922.

[23] Jarrod R McClean et al. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* 18.2 (2016), p. 023023. DOI: 10.1088/1367-2630/18/2/023023.

[24] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[25] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum Support Vector Machine for Big Data Classification". In: *Phys. Rev. Lett.* 113 (13 2014), p. 130503. DOI: 10.1103/PhysRevLett.113.130503.

[26] Patrick Rebentrost et al. "Quantum algorithms for hedging and the learning of Ising models". In: *Phys. Rev. A* 103 (1 2021), p. 012418. DOI: 10.1103/PhysRevA.103.012418.

[27] Itay Safran and Ohad Shamir. "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, 2979–2987. DOI: 10.5555/3305890.3305989.

[28] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. DOI: 10.7551/mitpress/4175.001.0001.

[29] Maria Schuld and Nathan Killoran. "Quantum Machine Learning in Feature Hilbert Spaces". In: *Phys. Rev. Lett.* 122 (4 2019), p. 040504. DOI: 10.1103/PhysRevLett.122.040504.

[30] Ewin Tang. "A Quantum-Inspired Classical Algorithm for Recommendation Systems". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Association for Computing Machinery, 2019, 217–228. ISBN: 9781450367059. DOI: 10.1145/3313276.3316310.

[31] M.D. Vose. "A linear algorithm for generating random numbers with a given distribution". In: *IEEE Transactions on Software Engineering* 17.9 (1991), pp. 972–975. DOI: 10.1109/32.92917.

[32] A. J. Walker. “New fast method for generating discrete random numbers with arbitrary frequency distributions”. In: *Electronics Letters* 10.8 (1974), pp. 127–128. DOI: 10.1049/el:19740097.

[33] Nathan Wiebe, Daniel Braun, and Seth Lloyd. “Quantum Algorithm for Data Fitting”. In: *Phys. Rev. Lett.* 109 (5 2012), p. 050505. DOI: 10.1103/PhysRevLett.109.050505.