Quantum algorithms for hedging and the learning of Ising models

Patrick Rebentrost,* Yassine Hamoudi,† Maharshi Ray,‡ Xin Wang,§ Siyi Yang,‡ and Miklos Santha¶

(Dated: November 30, 2020)

A paradigmatic algorithm for online learning is the Hedge algorithm by Freund and Schapire. An allocation into different strategies is chosen for multiple rounds and each round incurs corresponding losses for each strategy. The algorithm obtains a favorable guarantee for the total losses even in an adversarial situation. This work presents quantum algorithms for such online learning in an oracular setting. For T time steps and N strategies, we exhibit run times of about $\mathcal{O}\left(\mathrm{poly}(T)\sqrt{N}\right)$ for estimating the losses and for betting on individual strategies by sampling. In addition, we discuss a quantum analogue of the Sparsitron, a machine learning algorithm based on the Hedge algorithm. The quantum algorithm inherits the provable learning guarantees from the classical algorithm and exhibits polynomial speedups. The speedups may find relevance in finance, for example for hedging risks, and machine learning, for example for learning generalized linear models or Ising models.

I. INTRODUCTION

Optimization is a cornerstone of machine learning and artificial intelligence. Examples are the training of support vector machines and neural networks for tasks such as the analysis of audio, texts, and images. A great deal of quantum algorithm developments have been concerned with quantum speedups for optimization problems, and in particular for convex optimization problems. Generic convex optimization in the quantum oracle model was discussed in [1, 2]. Special cases of convex programming are linear programs (LP) and semidefinite programs (SDP), which involve optimizing a linear function of a vector or matrix subject to linear contraints, respectively. Several quantum algorithms have been discussed for these two convex programs using amplitude amplification and estimation and Gibbs sampling [3–6]. Linear programs can also be mapped to zero-sum games, for which a linear time classical solver was developed by Grigoriadis and Khachiyan [7]. For zero-sum games, quantum algorithms were obtained in [8, 9], which in turn also apply to LPs. Beyond LPs and SDPs, sublinear algorithms for quadratic constraints are discussed by Clarkson, Hazan and Woodruff [10], which includes for example the classification of data points with a margin, a kernel-based classification, the minimum enclosing ball problem, and ℓ_2-margin support vector machines. Reference [9] provides corresponding quantum algorithms based on the amplitude amplification and estimation subroutines.

These quantum optimization algorithms assume a quantum oracle that can be queried in superposition. Such a setting dates back to early quantum algorithms such as Grover's or Deutsch-Jozsa's. In machine learning, data are usually generated from an external source, such as users providing ratings to movies or products. In this case, quantum random access memory (QRAM) is discussed as a way to make such data available to a quantum algorithm [11, 12]. The oracle framework used in this work encompasses such a QRAM data access and also the case of access to a computable function. For a quantum algorithm's output, a possible way is to encode the output in a quantum state. A famous example is the work of Harrow, Hassidim and Lloyd (HHL) [13] for solving linear system of equations. The solution to a linear system $Ax = b$ is provided by a quantum state $|x\rangle$ upon which measurements can be performed to obtain classically relevant information. Under some well-discussed conditions [13, 14], the HHL algorithm can achieve an exponential quantum speedup. In contrast, in the optimization works mentioned above and in the present work, the output of the algorithm is inherently classical. These algorithms are hybrid, that is partly of classical and partly of quantum nature, and designed in a modular way so that the quantum part of the algorithm can be treated as a separate building block. The algorithms make quantum improvements on parts of the (best) available classical algorithm while keeping its overall structure intact. In contrast to the HHL algorithm for example, here the quantum versus classical speedup is usually at most polynomial, in most cases at most quadratic in the domain size of the input function. The quantum algorithm can deliver some speedup in the dimension of the problem, while in other relevant parameters it might not necessarily achieve any speedup, sometimes it can even be worse than the best classical algorithm.

Consider a game with T rounds, where we have the chance to play a mixture of N different strategies at each round and observe the results of our choice in the next round. The setting is "online" in the sense that the results are unknown ahead of time, which also can be seen

*Centre for Quantum Technologies, National University of Singapore, Singapore 117543; Electronic address: cqtfpr@nus.edu.sg
†Université de Paris, IRIF, CNRS, F-75013 Paris, France.; Electronic address: yassine.hamoudi@irif.fr
‡Centre for Quantum Technologies, National University of Singapore, Singapore 117543.
§Institute for Quantum Computing, Baidu Research, Beijing 100193, China.
¶Université de Paris, IRIF, CNRS, F-75013 Paris, France; and Centre for Quantum Technologies and MajuLab UMI 3654, National University of Singapore, Singapore 117543.; Electronic address: cqtms@nus.edu.sg

as an idealized version of sports betting or stock market trading. The Hedge algorithm by Freund and Schapire adaptively changes the mixture of strategies (a probability vector) via multiplicative weight updates [15]. This strategy allows for losses after T rounds that are not much worse than the minimum achievable "offline" loss. Here, "offline" means that the strategy is picked in advance without any adaptation. This difference of online and minimum achievable offline loss is often called "regret", a slight misnomer as in the online setting it is impossible to compute the minimum offline loss in advance. Precisely, it can be shown that the regret of the Hedge algorithm is not worse than $\sqrt{2T\log N}+\log N$. The classical complexity for this algorithm is $\mathcal{O}(TN)$: at each round we have to perform the multiplicative update, an effort that is proportional to N. While T and N are arbitrary, in most applications when applied in learning theory for example, T is much smaller than N, e.g., $T=\mathcal{O}(\log N)$.

In this work, we provide quantum algorithms in the online learning/hedging scenario. Assuming appropriate oracles for the online loss information, we first exhibit quantum speedups for two settings, which can be considered the passive and active setting. In the passive setting, we are interested in estimating the total loss after T rounds without ever writing down the full probability vector and without making active decisions. We obtain an ϵ-accurate estimate of the total loss with high probability and with a query complexity of $\mathcal{O}\left(\frac{T^2\sqrt{N}}{\epsilon}\right)$ and a gate complexity of $\widetilde{\mathcal{O}}\left(\frac{T^2\sqrt{N}}{\epsilon}\right)$ via amplitude estimation. We use the notation $\widetilde{\mathcal{O}}()$ to hide poly-logarithmic factors in any of the variables. The improvement in N and worsening in T is acceptable in the common situation when T is much smaller than N. Furthermore, we consider the active setting where at each round the strategy is executed and incurs a transaction cost. For this active setting we prepare the relevant quantum state with amplitude amplification, sample from it, and bet on the outcome. In the case when every allocation into a particular strategy comes at a transaction cost, the sampling setting has the advantage of reducing such costs. We again obtain a quantum speedup in N while the loss of such a strategy remains close to the minimum loss with high probability. We note work on quantum speedups for the Hedge algorithm and the related adaptive boosting technique in [16, 17].

The main motivation for discussing the Hedge algorithm is its application in optimization and machine learning. In the second part of this work, we provide a quantum algorithm for the Sparsitron [18]. The Sparsitron is a supervised learning algorithm based on the multiplicative weights algorithm by Freund and Schapire. The algorithm can be used to learn a Generalized Linear Model (GLM) and Ising models from training examples. For each N-dimensional training example, a loss is computed which takes into account a non-linear, potentially *non-convex*, activation function and an inner product between the training example and a weight vector. For guaranteed learning from the data with a certain accuracy ϵ (to be defined), the run time is about $\widetilde{\mathcal{O}}\left(\frac{N}{\epsilon^4}\right)$. We present a classical approximate Sparsitron algorithm, which estimates the inner products instead of computing them exactly (Theorem 6). The runtime of this algorithm is $\widetilde{\mathcal{O}}\left(\frac{N}{\epsilon^2}+\frac{1}{\epsilon^6}\right)$, improving on the original algorithm if, say, $N>\frac{1}{\epsilon^2}$. Subsequently, we present a quantum algorithm called the Quantum Sparsitron (Theorem 7). The quantum algorithm uses quantum inner product estimation and achieves a run time of about $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon^7}\right)$, a polynomial quantum speedup compared to the classical algorithm with respect to the dimension of the data. As a corollary, we derive a polynomial quantum speedup for the learning of Ising models (Corollary 1).

Regarding notation, we use $[N]$ to denote the set $\{1,\ldots,N\}$, where $N\in\mathbb{Z}_+$. We write a vector plainly as x without any special furnishing, however we use $\vec{0}$ and $\vec{1}$ to denote the all 0s and all 1s vector, respectively. The ℓ_1-norm of a vector $x\in\mathbb{R}^N$ is given by $\|x\|_1:=\sum_{j=1}^N|x_j|$. The maximum element of a vector $x\in\mathbb{R}^N$ is given by $\|x\|_{\max}:=\max_{j\in[N]}|x_j|$, also sometimes denoted by $x_{\max}$. Equivalently, the maximum absolute element of a matrix A is denoted by $\|A\|_{\max}$. For $x,y\in\mathbb{R}^N$, we write the inner product as $x\cdot y$ and the element-wise vector multiplication as $x\odot y\in\mathbb{R}^N$, where $(x\odot y)_j=x_jy_j$. For $c\in\mathbb{R}$ and $x\in\mathbb{R}^N$, $c^x\in\mathbb{R}^N$ is understood element-wise as $(c^x)_j=c^{x_j}$. We use $|\bar{0}\rangle$ to denote the multi-qubit state $|0\rangle\otimes\cdots\otimes|0\rangle$, where the number of qubits is clear from the context. As mentioned, we use the notation $\widetilde{\mathcal{O}}()$ to hide poly-logarithmic factors in any of the variables.

II. CLASSICAL HEDGE ALGORITHMS

A. Original algorithm

We follow Ref. [15] for the discussion of the classical Hedge algorithm. We are given N strategies for a game that takes T rounds. Before each time $t\in[T]$, we choose an assignment (portfolio) of the N strategies. This assignment shall be given by the weights $w^{(t)}=\left(w_1^{(t)},\ldots,w_N^{(t)}\right)^\dagger\in[0,1]^N$, which form the probability vector

$$p^{(t)}=\left(p_1^{(t)},\ldots,p_N^{(t)}\right)^\dagger=\frac{1}{\|w^{(t)}\|_1}\left(w_1^{(t)},\ldots,w_N^{(t)}\right)^\dagger. \quad (1)$$

The initial allocation is taken to be uniform, i.e., $w^{(1)}=(1/N,\ldots,1/N)^\dagger$ and $p^{(1)}=(1/N,\ldots,1/N)^\dagger$. The algorithm considers an online learning setting, where information arrives over time and the weights are updated accordingly. Specifically, at each time $t\in[T]$, we observe the loss vector

$$l^{(t)}=\left(l_1^{(t)},\ldots,l_N^{(t)}\right)^\dagger\in[0,1]^N. \quad (2)$$

Algorithmically, we describe this as "Receive loss vector $l^{(t)}$", which means we obtain access to the loss vector. To avoid further complexities, we assume that each loss $l_j^{(t)}$ takes a constant number of bits to specify. The loss at time t is given by

$$L^{(t)} := \sum_{i=1}^{N} p_i^{(t)} l_i^{(t)} \equiv p^{(t)} \cdot l^{(t)} \in [0,1]. \tag{3}$$

Algorithmically, this loss is taken into account with the statement "Suffer loss $L^{(t)}$", which means we add $L^{(t)}$ to the overall loss amount. A strategy to minimize losses was shown in Ref. [15]. Take $\beta \in (0,1)$. The strategy is based on multiplicative updates to the weights given the incoming loss information as $w_j^{(t)} = \beta^{l_j^{(t-1)}} w_j^{(t-1)}$, which for the full path up to t is $w_j^{(t)} = \beta^{\sum_{t'=1}^{t-1} l_j^{(t')}} w_j^{(1)}$. We also write $w^{(t)} \odot \beta^{l^{(t)}}$, using the notation for the element-wise vector multiplication, and $\beta^{l^{(t)}}$ is understood element-wise.

The original algorithm is given in Algorithm 1, setting the flag to "null". The accumulated loss of this algorithm (denoted by $\mathcal{H}$ for hedge) over T rounds is

$$L_{\mathcal{H}} := \sum_{t=1}^{T} L^{(t)}. \tag{4}$$

On the other hand, consider the "offline loss", $L_{\min}^{(T)} = \min_{j\in[N]} \sum_{t=1}^{T} l_j^{(t)}$, which gives the minimum loss achievable when choosing the same single strategy for all rounds of the game. Ref. [15] shows a "regret" bound for the losses of the multiplicative update strategy.

Theorem 1 (Regret bound [15])**.** *With the definitions above and $\beta = 1/(1+\sqrt{2\log N/T})$, Algorithm 1 with flag "null" achieves*

$$L_{\mathcal{H}} - L_{\min}^{(T)} \leq \sqrt{2T\log N} + \log N. \tag{5}$$

This bound is better than the naive bound $L_{\mathcal{H}} - L_{\min}^{(T)} \leq T$. The run time of the classical algorithm is given by $\mathcal{O}(TN)$. At every step, the algorithm updates N probabilities and there are T steps overall.

A quantum version of this algorithm with the appropriate oracles for the loss vectors will be able to provide an estimate of the total loss $L_{\mathcal{H}}$ with polynomial speedup in N, see Algorithm 2 below. The "deterministic" and "sampled" flags will be explained in the next subsection.

B. Active betting by sampling

In a simple extension to the original algorithm, we model the cost of allocating individual bets in the portfolio. This allocation/transaction cost serves to illustrate the benefits of a sampling strategy over the deterministic strategy. The allocation costs are taken to be as follows. Each allocation based on the elements of the vector $p^{(t)}$, however small, will come with a cost C_0. For example, buying even a single stock on the stock market comes with the cost of contacting a broker and a broker fee. This cost is counted separately for the algorithm in a total transaction cost C. We keep these costs separate from the losses $L_{\mathcal{H}}$, and it is straightforward to combine them. Algorithm 1 formalizes this additional feature. With "Allocate portfolio" we denote the step of concretely betting on respective strategies which incurs a cost of C_0 per strategy allocated. The flag "deterministic" leads to an allocation into all the strategies with corresponding transaction cost. To minimize the transaction cost, the flag "sampled" can be used, for which at each time t an index $j^{(t)}$ is sampled according to $p^{(t)}$ and an investment is made on that single outcome. This sampling requires preparing a sampling data structure, as explained in Fact 2 in Appendix A. In the quantum case, see Algorithm 3, a quantum state $|p^{(t)}\rangle$ is prepared from which samples are obtained.

The run time of the algorithm is the same for all three flags. A simple statement about the total transaction cost is as follows.

Fact 1. *The transaction cost $C := \sum_{t=1}^{T} C^{(t)}$ of Algo-*

Algorithm 1 Hedge algorithm by Freund and Schapire [15] with transaction cost and sampling

Input: Number of strategies N, number of rounds T, parameter $\beta \in (0,1)$, flag $\in$ {null, deterministic, sampled}, fixed transaction cost per strategy C_0.

$w^{(1)} \leftarrow \vec{1}/N$.
for $t \leftarrow 1$ to T **do**
  $p^{(t)} \leftarrow w^{(t)}/\|w^{(t)}\|_1$. ▷ Update probabilities
  if flag = null **then**
    $q^{(t)} \leftarrow p^{(t)}$.
    $C^{(t)} \leftarrow 0$. ▷ No transaction cost
  else if flag = deterministic **then**
    $q^{(t)} \leftarrow p^{(t)}$.
    Allocate portfolio according to $q^{(t)}$.
    $C^{(t)} \leftarrow N \times C_0$.
  else if flag = sampled **then**
    Prepare sampling data structure for $p^{(t)}$ via Fact 2.
    $j \leftarrow$ Sample from $p^{(t)}$.
    $q^{(t)} \leftarrow \vec{0}$, $q_j^{(t)} \leftarrow 1$.
    Allocate portfolio according to $q^{(t)}$.
    $C^{(t)} \leftarrow C_0$.
  end if
  Receive loss vector $l^{(t)}$.
  Suffer loss $L^{(t)} \leftarrow q^{(t)} \cdot l^{(t)}$.
  $w^{(t+1)} \leftarrow w^{(t)} \odot \beta^{l^{(t)}}$. ▷ Multiplicative weight update
end for

Output: $\sum_{t=1}^{T} L^{(t)}$, $\sum_{t=1}^{T} C^{(t)}$.

rithm 1 with the different flags is

$$\text{null} \ : \ C = 0. \tag{6}$$
$$\text{deterministic} \ : \ C = N \times T \times C_0, \tag{7}$$
$$\text{sampled} \ : \ C = T \times C_0. \tag{8}$$

Proof. For "deterministic", we have T iterations, at each of which the investment cost is $N \times C_0$. For "sampled", at each step we prepare the data structure according to Fact 2, sample, and invest in the single sampled strategy. The transaction cost per step is C_0, independent of N. □

Furthermore, the output of the sampled algorithm is an unbiased estimator of $L_{\mathcal{H}}$ and satisfies a regret bound.

Theorem 2. *Algorithm 1 with flag "sampled" and* $\beta = 1/(1+\sqrt{2\log N/T})$ *outputs* $L_{\text{samp}} := \sum_{t=1}^{T} l_{j^{(t)}}^{(t)}$ *such that* $\mathbb{E}[L_{\text{samp}}] = L_{\mathcal{H}}$. *For* $\delta \in (0,1)$, *the regret bound is* $L_{\text{samp}} - \min_{j\in[N]} L_j \leq 3\sqrt{T\log(N/\delta)} + \log N$ *with probability at least* $1-\delta$.

Proof. The expectation value of L_{samp} is

$$\mathbb{E}\left[L_{\text{samp}}\right] \ = \ \sum_{t=1}^{T} \mathbb{E}\left[l_{j^{(t)}}^{(t)}\right] = \sum_{t=1}^{T} p^{(t)} \cdot l^{(t)} \equiv L_{\mathcal{H}}. \tag{9}$$

Hoeffding's inequality is $P\left[\frac{L_{\text{samp}}}{T} - \frac{L_{\mathcal{H}}}{T} \geq r\right] \leq e^{-2Tr^2}$ for $r \in \mathbb{R}_+$ and hence

$$P\left[L_{\text{samp}} - L_{\mathcal{H}} \geq s\sqrt{T}\right] \leq e^{-2s^2}, \tag{10}$$

using $r = s/\sqrt{T}$. Now, bound the difference to the minimum loss using Theorem 1 as

$$\begin{aligned} L_{\text{samp}} - \min_{j\in[N]} L_j \ &= \ L_{\text{samp}} - L_{\mathcal{H}} + L_{\mathcal{H}} - \min_{j\in[N]} L_j \\ &\leq \ L_{\text{samp}} - L_{\mathcal{H}} + \sqrt{2T\log N} + \log N \\ &\leq \ s\sqrt{T} + \sqrt{2T\log N} + \log N, \end{aligned} \tag{11}$$

with probability at least $1 - e^{-2s^2}$. Setting $s = \sqrt{\ln(1/\delta)/2} \leq \sqrt{\log(N/\delta)}$ leads to

$$L_{\text{samp}} - \min_{j\in[N]} L_j \leq 3\sqrt{T\log(N/\delta)} + \log N \tag{12}$$

with probability at least $1-\delta$. □

Hence the sampling Hedge algorithm achieves the correct loss in expectation and the regret bound up to a scaling factor with high probability.

III. QUANTUM HEDGE ALGORITHMS

We now turn to quantum algorithms in the Hedge setting. We provide two simple algorithms, one for estimating the losses, one for the active betting scenario, before discussing the Sparsitron. The algorithms are based on quantum minimum finding, see Lemma 4, amplitude amplification and estimation, see Lemma 5, and quantum inner product estimation, see Lemma 6, all in Appendix B. We first discuss the rescaling of relevant quantities. We then discuss the quantum data input model and state preparation subroutines. In a passive setting, we use amplitude estimation to estimate the total loss of the Hedge algorithm given the data input, see Algorithm 2. In an active setting, we discuss the allocation of a portfolio via amplitude amplification and sampling, see Algorithm 3.

One of the quantities estimated via a quantum algorithm is the ℓ_1-norm $\left\|w^{(t)}\right\|_1$ for all t. Recall that the algorithm starts with a uniform initial weight vector $w^{(1)} = \vec{1}/N$. As we are decreasing each weight by at most β per step, the minimum weight achievable after T steps is β^T/N. In a situation where also the maximum weight $w_{\max}^{(T)} \equiv \left\|w^{(T)}\right\|_{\max}$ is $\mathcal{O}\left(\beta^T/N\right)$, the ℓ_1-norm is small, i.e., $\left\|w^{(T)}\right\|_1 \sim \beta^T \sim 1/2^T$. To overcome this inconvenient worst case, we rescale the weights in the estimation. For each $1 \leq t \leq T$, define similar to before the minimum offline loss up to t, $L_{\min}^{(t)} := \min_{j\in[N]} \sum_{t'=1}^{t} l_j^{(t')} \leq t$. Note that the maximum element of $w^{(t)}$ is

$$w_{\max}^{(t)} = \beta^{L_{\min}^{(t-1)}}/N. \tag{13}$$

We consider the rescaled weights $\frac{w_j^{(t)}}{w_{\max}^{(t)}} \leq 1$, which keep the expected loss $L^{(t)} \equiv \frac{w^{(t)}\cdot l^{(t)}}{\left\|w^{(t)}\right\|_1}$ the same because the $w_{\max}^{(t)}$ factor cancels out. However, we now have the lower bound for the ℓ_1-norm $\left\|\frac{w^{(t)}}{w_{\max}^{(t)}}\right\|_1 \geq 1$. In the quantum context, we find $L_{\min}^{(t-1)}$ via the minimum finding algorithm [19] in run time $\widetilde{\mathcal{O}}\left(\sqrt{N}\right)$ with high success probability. See Lemma 4 in Appendix B for the statement of the minimum finding algorithm. With Eq. (13) we can then compute the maximum weight.

A. Quantum data input model

We translate the online learning setting into the quantum domain. The input data for the quantum algorithms are the losses experienced at every step t. First, we assume T different oracles, where the sequential access to these oracles embodies the online setting.

Data Input 1 (Loss oracles)**.** *Assume that* $\mathcal{O}(1)$ *bits are sufficient to specify the losses* $l_j^{(t)}$. *For* $t \in [T]$ *and* $j \in [N]$, *assume unitaries* $U_{l^{(t)}}$ *such that* $U_{l^{(t)}} |j\rangle |\bar{0}\rangle = |j\rangle \left|l_j^{(t)}\right\rangle$, *operating on* $\mathcal{O}(\log N)$ *quantum bits.*

To simplify the notation, we have used the $|\bar{0}\rangle$ initial state in the second register. In fact, the precise assumption here is $U_{l^{(t)}} |j\rangle |c\rangle = |j\rangle \left|c \oplus l_j^{(t)}\right\rangle$, where c is any bit string allowed in the second register and $\oplus$ is the bitwise addition modulo 2. With this precise assumption,

Algorithm 2 Quantum estimation of the total loss of the Hedge algorithm

Input: Number of strategies N, number of rounds T, parameter $\beta \in (0,1)$, error $\epsilon \in (0,1]$, success probability $1-\delta \in (0,1)$.

1: **for** $t = 1$ to T **do**
2: Construct unitary for $w^{(t)} = w^{(1)}\beta^{l^{(1)}} \cdots \beta^{l^{(t-1)}}$ using oracles $\{U_{l^{(t')}} : t' \in [t-1]\}$ and Lemma 1.
3: Receive loss oracle $U_{l^{(t)}}$.
4: $L^{(t)} \leftarrow$ Quantum estimate $\frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot l^{(t)}$ to relative accuracy ϵ with success probability $1 - \frac{\delta}{T}$ via Lemma 6.
5: **end for**

Output: $\sum_{t=1}^{T} L^{(t)}$.

we have $(U_{l^{(t)}})^2 |j\rangle |c\rangle = |j\rangle |c\rangle$, hence we can uncompute the result with another query. This assumption shall hold also for the other data inputs below. The oracles allow us to perform the following computations.

Lemma 1. *Let $t \in [T]$ and $\beta \in (0,1)$. Let the set of unitaries $U_{l^{(t')}}$ for $t' \in [t-1]$ be given as in Data Input 1. There exists unitaries performing the computations $|j\rangle|\bar{0}\rangle \to |j\rangle \left|\sum_{t'=1}^{t-1} l_j^{(t')}\right\rangle$, $|j\rangle|\bar{0}\rangle \to |j\rangle \left|w_j^{(t)}\right\rangle$, and, if knowledge of $w_{\max}^{(t)}$ is given, $|j\rangle|\bar{0}\rangle \to |j\rangle \left|\frac{w_j^{(t)}}{w_{\max}^{(t)}}\right\rangle$ to sufficient accuracy $\mathcal{O}(1/N)$. These computations take $\mathcal{O}(T)$ queries to the data input and $\mathcal{O}(T + \log N)$ qubits and quantum gates.*

Hence, given the loss unitaries, we can compute the weights with overhead about $\mathcal{O}(T)$. This computation allows the estimation of norms and inner products and preparation of weight quantum states.

B. Quantum algorithm to obtain the total loss of the Hedge algorithm

Our first quantum algorithm is simple. At each time step, we receive a loss oracle according to Data Input 1. From this input, we estimate the total loss of the multiplicative weight update method. We never fully exhibit the full weight vector but rather only the total loss at each step given the weight vectors. The quantum algorithm is given in Algorithm 2.

We use the Lemma 6 for estimating the inner product with relative accuracy and hence obtain a relative accuracy for the total loss. We obtain a statement on the accuracy of the loss, the run time, and the success probability of the quantum algorithm.

Theorem 3. *Let $\epsilon \in (0,1]$, $\delta \in (0,1)$, and $\beta = (0,1)$. Algorithm 2 provides an estimate $\sum_{t=1}^{T} L^{(t)}$ of the total loss $L_{\mathcal{H}}$ such that $\left|\sum_{t=1}^{T} L^{(t)} - L_{\mathcal{H}}\right| \leq \epsilon L_{\mathcal{H}}$ with probability at least $1-\delta$. This quantum algorithm requires $\mathcal{O}\left(\frac{T^2\sqrt{N}}{\epsilon}\log\left(\frac{T}{\delta}\right)\right)$ queries to the oracles and $\widetilde{\mathcal{O}}\left(\frac{T^2\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ gates.*

Proof. Regarding the correctness, assuming all subroutines succeed, we have for all $t \in [T]$ that $\left|L^{(t)} - \frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot l^{(t)}\right| \leq \epsilon \frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot l^{(t)}$, and thus $\left|\sum_{t=1}^{T} L^{(t)} - L_{\mathcal{H}}\right| \leq \epsilon L_{\mathcal{H}}$.

The run times for computing the entries w_j are given via Lemma 1, amounting to $\mathcal{O}(T)$ queries to the Data Input 1 and $\mathcal{O}(T + \log(TN/\epsilon))$ quantum gates. Lemma 6 at each step requires $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{T}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{T}{\delta}\right)\right)$ quantum gates. These complexities are multiplied by T as we are proceeding for T steps. From Lemma 1, the complexities are multiplied by another factor $\mathcal{O}(T)$ for the query complexity and a factor $\mathcal{O}(T + \log(TN/\epsilon))$ for the gate complexity. Each single-step, single-estimate success probability is $1 - \frac{\delta}{T}$. Hence the overall success probability of the algorithm is $\left(1 - \frac{\delta}{T}\right)^T \geq 1 - \delta$. □

C. Active betting with quantum sampling

Next, we provide a quantum version of Algorithm 1 with the "sampled" flag. Instead of obtaining the sample classically, we prepare a corresponding quantum state and measure it. We approximately prepare the quantum states of square-root probabilities $\left|p^{(t)}\right\rangle = \sum_{j=1}^{N} \sqrt{p_j^{(t)}} |j\rangle$, for every time $t = 1, \ldots, T$, with the probabilities given in Eq. (1). The quantum algorithm is given in Algorithm 3. We obtain a biased estimate of the loss where the bias is set to $\sqrt{T \log N}$ to obtain the usual regret bound up to constant factors.

Theorem 4. *Let $\delta \in (0,1)$. Algorithm 3 outputs $L_{\text{samp}}^{Q} := \sum_{t=1}^{T} l_{j^{(t)}}^{(t)}$ with a bias $\left|\mathbb{E}[L_{\text{samp}}^{Q}] - L_{\mathcal{H}}\right| \leq \sqrt{T \log N}$ with success probability at least $1-\delta$. With $\beta = 1/(1 + \sqrt{2\log N/T})$, the regret bound is $L_{\text{samp}}^{Q} - \min_{j\in[N]} L_j \leq 4\sqrt{T\log(N/\delta)} + \log N$ with success probability at least $1-2\delta$. This quantum algorithm requires $\mathcal{O}\left(T^2\sqrt{N}\log\left(\frac{T}{\delta}\right)\right)$ queries to the oracles and $\widetilde{\mathcal{O}}\left(T^2\sqrt{N}\log\left(\frac{1}{\delta}\right)\right)$ gates. The transaction cost is $T \times C_0$.*

Proof. For every step $t \in [T]$, amplitude amplification produces an erroneous output state, see Lemma 5, with probabilities denoted by $\tilde{p}_j$. Let $\xi \in (0,1]$ as in Lemma 5 (iii), where by construction we have $\left\|p^{(t)} - \tilde{p}^{(t)}\right\|_1 \leq \xi$. The expectation value using these probabilities is

$$\tilde{L}_{\mathcal{H}} := \mathbb{E}_{\tilde{p}}\left[L_{\text{samp}}^{Q}\right] = \sum_{t=1}^{T} \tilde{p}^{(t)} \cdot l^{(t)}. \tag{14}$$

Algorithm 3 Active Hedge algorithm with transaction cost and quantum sampling

Input: Number of strategies N, number of rounds T, parameter $\beta \in (0,1)$, transaction cost C_0, success probability $1-\delta \in (0,1)$.

1: **for** $t = 1$ to T **do**
2: $L_{\min}^{(t-1)} \leftarrow$ Find $\min_{j\in[N]} \sum_{t'=1}^{t-1} l_j^{(t')}$ using oracles $\{U_{l^{(t')}} : t' \in [t-1]\}$ with success probability $1-\frac{\delta}{2T}$ using Lemma 1 and Lemma 4.
3: $w_{\max}^{(t)} \leftarrow \beta^{L_{\min}^{(t-1)}}/N$.
4: Prepare an approximation $\left|\tilde{p}^{(t)}\right\rangle$ of the quantum state $\left|p^{(t)}\right\rangle$ with $p^{(t)} = w^{(t)}/\|w^{(t)}\|_1$ using $w_{\max}^{(t)}$, Lemma 1, and Lemma 5 (iii) with accuracy $\xi = \sqrt{\log(N)/T}$ and success probability $1-\frac{\delta}{2T}$.
5: $j^{(t)} \leftarrow$ Measure $\left|\tilde{p}^{(t)}\right\rangle$ in the computational basis.
6: Allocate portfolio in $j^{(t)}$-th strategy at cost C_0.
7: Receive loss oracle $U_{l^{(t)}}$.
8: $l_{j^{(t)}}^{(t)} \leftarrow$ Query $U_{l^{(t)}}$ with $\left|j^{(t)}\right\rangle|0\rangle$.
9: Suffer loss $l_{j^{(t)}}^{(t)}$.
10: **end for**

Output: $L_{\text{samp}}^{Q} := \sum_{t=1}^{T} l_{j^{(t)}}^{(t)}$.

Note that $\left|l^{(t)} \cdot \left(p^{(t)} - \tilde{p}^{(t)}\right)\right| \leq \left\|p^{(t)} - \tilde{p}^{(t)}\right\|_1 \left\|l^{(t)}\right\|_{\max} \leq \xi$. Thus, we have the bias

$$\left|\tilde{L}_{\mathcal{H}} - L_{\mathcal{H}}\right| \leq T\xi. \tag{15}$$

The success probability for obtaining this bias is $\left(1-\frac{\delta}{2T}\right)^{2T} \geq (1-\delta)$.

For the difference to the minimum loss strategy, we find using Theorem 1 that

$$\begin{aligned} L_{\text{samp}}^{Q} - \min_{j\in[N]} L_j \leq & \\ \leq\ & L_{\text{samp}}^{Q} - \tilde{L}_{\mathcal{H}} + \left|\tilde{L}_{\mathcal{H}} - L_{\mathcal{H}}\right| + L_{\mathcal{H}} - \min_{j\in[N]} L_j \\ \leq\ & L_{\text{samp}}^{Q} - \tilde{L}_{\mathcal{H}} + T\xi + \sqrt{2T\log N} + \log N \\ \leq\ & T\xi + 3\sqrt{T\log(N/\delta)} + \log N. \end{aligned} \tag{16}$$

Here, we used that $L_{\text{samp}}^{Q} - \tilde{L}_{\mathcal{H}} \leq \sqrt{T\log(N/\delta)}$ with probability $1-\delta$ from Hoeffding's inequality as in Theorem 2. We have a total success probability for this regret bound of $(1-\delta)^2 \geq 1-2\delta$. Finally, set $\xi = \sqrt{\log(N)/T}$ to obtain the stated regret bound.

The minimum findings take a total run time of $\mathcal{O}\left(T^2\sqrt{N}\log\left(\frac{T}{\delta}\right)\right)$ via Lemma 4. Lemma 5 with $u_j = w_j^{(t)}/w_{\max}^{(t)}$ takes $\mathcal{O}\left(T^2\sqrt{N}\log\left(\frac{T}{\delta}\right)\right)$ queries to the oracles and $\widetilde{\mathcal{O}}\left(T^2\sqrt{N}\log\left(\frac{1}{\delta}\right)\right)$ gates. □

We complete this section with comparison of the run time of Theorem 4 with the run time of the classical algorithm from Theorem 2. Choose $\delta = \Theta(1)$. Then Theorem 4 achieves a regret bound of $\mathcal{O}\left(\sqrt{T\log N}\right)$ with a run time of $\widetilde{\mathcal{O}}\left(T^2\sqrt{N}\right)$ and high success probability. The classical algorithm discussed in Theorem 2 achieves a similar regret bound $\mathcal{O}\left(\sqrt{T\log N}\right)$ with a run time of $\widetilde{\mathcal{O}}(TN)$ and high success probability.

IV. SPARSITRON

At a high level, statistical classification in machine learning is performed with two main approaches. Let X denote the observable variables (features) and Y the target variables (labels). The first approach is using a *discriminative model*. In this approach, one models the conditional distribution $P[Y|X=x]$, i.e., the probability of the labels given an input example x. On the other hand, a *generative model* constructs a joint distribution $P[X,Y]$ of variables and labels. In this more general approach, inferences in both directions features → label and label → features can be made.

Undirected graphical models, or Markov random fields, are a powerful, modern statistical tool for modeling high-dimensional probability distributions [18, 20]. The joint probability distribution of such a model depends on an underlying graph, where the presence of an edge gives conditional dependence and the absence of an edge gives conditional independence. The Ising model is a special type of Markov random field which uses binary variables and pairwise interactions. Consider here N binary variables $Z_j \in \{-1,1\}$ for $j \in [N]$. Associated with these variables is an undirected dependency graph. The graph enters a probability distribution

$$P[Z=z] \propto \exp\left(\sum_{i,j:i\neq j} A_{ij} z_i z_j + \sum_i \theta_i z_i\right), \tag{17}$$

where $A \in \mathbb{R}^{N\times N}$ is the graph adjacency matrix and $\theta \in \mathbb{R}^N$ describes bias terms. The width of an Ising model is defined as $\lambda(A,\theta) = \max_i \left(\sum_j |A_{ij}| + |\theta_i|\right)$. An important task in a machine learning context is the following *unsupervised* learning task: Given samples from the distribution Eq. (17) on $\{-1,1\}^N$, learn the matrix A. This problem is also known as the inverse Ising problem, in contrast to modelling a physical system (e.g., a spin system) and studying its behavior and effects such as phase transitions.

We provide a short overview of the hardness of the Ising model learning and the complexities of known algorithms. We refer the reader to [18, 20] as entry points into the literature, from which also the following overview derives. An unconditional lower bound on the sample complexity of learning Ising models with N vertices was given by Santhanam and Wainwright [21]. Even if the weights of the underlying graph are known and greater than $\eta > 0$, any algorithm for learning the graph struc-

ture must use $\Omega\left(\frac{2^{\lambda(A,\theta)/4}\log N}{\eta 2^{3\eta}}\right)$ samples. A famous early example of an efficient algorithm is due to Chow and Liu from 1968 [22] for when the underlying graph is a tree. For learning Ising models on general graphs with N nodes of degree at most d, the best known method is based on exhaustive search costing N^d. Subsequent works addressed the inverse Ising model problem for restricted classes of graphs or restricted nature of interactions between the variables. Some of these include learning Ising models when the underlying graph structure is a polytree [23], hypertree [24], tree mixture [25], or when the underlying graph has the correlation decaying property (which, informally means that two variables are asymptotically independent as the graph distance between them increases) [26]. With modest assumptions, Bresler gave a simple greedy algorithm [20] which reconstructs arbitrary Ising models on N nodes of maximum degree d in time $\widetilde{\mathcal{O}}\left(N^2\right)$ but has a sample complexity that depends doubly exponentially on d, whereas a singly exponential dependence on d is necessary. He also gave evidence that this run-time is optimal by showing that beating the $\widetilde{\mathcal{O}}\left(N^2\right)$ run-time bound would imply an improved run-time algorithm for the well-studied *light-bulb* problem [27]. Subsequently, the work of Bresler was subsumed by Klivans and Meka [18], who gave a $\mathcal{O}\left(N^2\right)$ algorithm with optimal sample complexity. Their main tool is a multiplicative weight update method algorithm (called *Sparsitron*) for learning an hypothesis class called the *generalized linear models* (discussed below). As an application of the Sparsitron algorithm, they show the Ising model can be learned efficiently in both time and sample complexity sense. We discuss their algorithm in more details below and eventually turn it into a quantum algorithm with the same sample complexity but with an improved run-time. In terms of the nature of algorithms used, besides exhaustive searching and greedy algorithms, several works have used convex optimization techniques such as in [28, 29]. A very recent work [30] gives a time and sample efficient algorithm to learn Ising models where the underlying graph is a tree, without making any further assumptions on the underlying distribution (such as bounds on the strengths of the model's edges/hyper-edges). In the quantum setting, to the best of our knowledge our work provides the first quantum algorithm for the inverse Ising model problem.

We continue with a more detailed discussion, which provides the connection to the following algorithm, the Sparsitron [18]. Focusing on a particular variable Z_j in Eq. (17), and setting the other variables to $x \in \{-1,1\}^{[N]\setminus\{j\}}$, we have for the conditional probability of the Ising model

$$\begin{aligned} P[Z_j &= -1|Z_{\neq j}=x] = \frac{P[Z_j=-1, Z_{\neq j}=x]}{P[Z_{\neq j}=x]} \\ &= \frac{P[Z_j=-1, Z_{\neq j}=x]}{P[Z_j=-1, Z_{\neq j}=x]+P[Z_j=1, Z_{\neq j}=x]} \\ &= \frac{1}{1+\exp(4\sum_{k\neq j}A_{jk}x_k+2\theta_j)} \\ &= \sigma(w\cdot x-2\theta_j). \end{aligned} \tag{18}$$

Here, we have used the sigmoid function $\sigma(z)=1/(1+e^{-z})$ and the vector $w\in\mathbb{R}^{[N]\setminus\{j\}}$ with $w_k=-4A_{jk}$. This single-variable conditional probability suggests a way to turn the original unsupervised learning problem into a *supervised* learning problem. Setting $X=(Z_k, k\neq j)$ and $Y=(1-Z_j)/2$ turns all variables except the j-th one into features and the j-th variable into a label. Note that $\mathbb{E}[Y|X=x]=P[Z_j=-1|Z_{\neq j}=x]=\sigma(w\cdot x-2\theta_j)$. In addition, we can redefine the vectors to absorb the bias term [18]. Increase the dimension of w by 1 and change $x\leftarrow[x_1,\cdots,x_{N-1},1]^T$. This change straightforwardly allows the learning of the bias $-2\theta_j$. We have the following problem statement for learning such a generalized linear model (GLM). Given samples (X,Y) from a distribution $\mathcal{D}$ with the conditional mean function $\mathbb{E}[Y|X=x]=\sigma(w\cdot x)$, learn w.

Reference [18] developed the Sparsitron, an efficient classical method to learn GLMs based on the Hedge algorithm by Freund and Schapire. One assumption is that a true w with $\|w\|_1\leq\lambda$ exists where $\lambda\geq 0$ is known. Without loss of generality one can take $w\geq 0$ and that $\|w\|_1=\lambda$, see [18]. A square loss function, or "risk", for any vector $v\in[-1,1]^N$ is given by

$$\varepsilon(v):=\mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\left(\sigma\left(v\cdot X\right)-\sigma\left(w\cdot X\right)\right)^2\right]. \tag{19}$$

The learning task here is defined in terms of finding a v such that this risk is ϵ small with high probability. The classical algorithm assumes access to two training sets, one with T samples which will be used for constructing predictions ($\lambda p^{(t)}$ in the algorithm) and the other with M samples for finding the best prediction among them. The second training set is used to evaluate the risk in an empirical manner, as we do not have access directly to the distribution $\mathcal{D}$. The "empirical risk" for any vector $v\in[-1,1]^N$ and given samples $(a^{(m)},b^{(m)})\in[-1,1]^N\times[0,1]$ for $m\in[M]$ from $\mathcal{D}$ is given by

$$\widehat{\varepsilon}(v):=\frac{1}{M}\sum_{m=1}^{M}\left(\sigma\left(v\cdot a^{(m)}\right)-b^{(m)}\right)^2. \tag{20}$$

We first show the original classical algorithm, then an approximate classical algorithm, then the quantum algorithm. The classical algorithm for the Sparsitron is given in Algorithm 4. The algorithm consists of one main loop that goes over the first set of training examples and is equivalent to the Hedge algorithm. For each training

Algorithm 4 Sparsitron [18] with risk approximation

Input: Parameter $\beta \in (0,1)$, norm $\lambda \geq 0$, training set $(x^{(t)}, y^{(t)}) \in [-1,1]^N \times [0,1]$ for $t \in [T]$, training set $(a^{(m)}, b^{(m)}) \in [-1,1]^N \times [0,1]$ for $m \in [M]$, flag $\in$ {original, approximate}.

$w^{(1)} \leftarrow \vec{1}/N$.
for $t = 1$ to T **do**
  $p^{(t)} \leftarrow \frac{w^{(t)}}{\|w^{(t)}\|_1}$.
  $l^{(t)} \leftarrow \frac{1}{2}\left(\vec{1} + \left(\sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right) - y^{(t)}\right) x^{(t)}\right)$.
  $w^{(t+1)} \leftarrow w^{(t)} \odot \beta^{l^{(t)}}$.
  if flag = original **then**
    for $m = 1$ to M **do**
      $z^{(t,m)} \leftarrow p^{(t)} \cdot a^{(m)}$.
    end for
  else if flag = approximate **then**
    Prepare sampling data structure for $p^{(t)}$ via Fact 2.
    for $m = 1$ to M **do**
      $z^{(t,m)} \leftarrow$ Estimate $p^{(t)} \cdot a^{(m)}$ to accuracy $\frac{\epsilon}{16\lambda}$ with success probability $1 - \frac{\delta}{MT}$.
    end for
  end if
  $\widehat{\varepsilon}^{(t)} \leftarrow \frac{1}{M}\sum_{m=1}^{M}\left(\sigma\left(\lambda z^{(t,m)}\right) - b^{(m)}\right)^2$.
end for

Output: $v = \lambda p^{(t')}$ for $t' = \arg\min_{t\in[T]} \widehat{\varepsilon}^{(t)}$.

example $x^{(t)}$, a prediction vector $\lambda p^{(t)}$ is constructed. From this vector one can compute the predicted label for the training example by using the activation function as $\sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right)$. A measure of the quality of the prediction is given by $\sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right) - y^{(t)}$ using the given label $y^{(t)}$ from the training set. In the loop, a loss vector $l^{(t)} = \frac{1}{2}\left(\vec{1} + \left(\sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right) - y^{(t)}\right) x^{(t)}\right) \in [0,1]^N$ is computed, which has entries close to $1/2$ in the case of correct prediction. In the last step of the loop, the second training set is used to compute the empirical risk $\widehat{\varepsilon}\left(\lambda p^{(t)}\right)$ to determine the quality of each prediction vector $p^{(t)}$. The algorithm finally returns the vector $v = \lambda p^{(t')}$ which performs best on this set. The provable learning guarantee and the run time is summarized in the following theorem. Plugging in T and M, the run time can be expressed as $\widetilde{\mathcal{O}}\left(N\frac{\lambda^2}{\epsilon^4}\log^2\frac{1}{\delta}\right)$.

Theorem 5 (Sparsitron [18])**.** *Let $\mathcal{D}$ be a distribution on $[-1,1]^N \times \{0,1\}$ where for $(X,Y) \sim \mathcal{D}$, $\mathbb{E}[Y|X=x] = \sigma(w \cdot x)$ for a non-decreasing 1-Lipschitz function $\sigma : \mathbb{R} \to [0,1]$. Suppose that $\|w\|_1 \leq \lambda$ for a known $\lambda \geq 0$. Let $\epsilon, \delta \in (0,1)$. Given $T + M = \mathcal{O}\left(\lambda^2 \log(N/\delta\epsilon)/\epsilon^2\right)$ independent samples from $\mathcal{D}$ and $\beta = 1 - \sqrt{\log N/T}$, Algorithm 4 with flag "original" produces a vector $v \in \mathbb{R}^N$ such that with probability at least $1-\delta$,*

$$\varepsilon(v) \leq \epsilon. \tag{21}$$

The run time of the algorithm is $\mathcal{O}(N \times T \times M)$, where $T = \mathcal{O}\left(\lambda^2 \log(N/\delta\epsilon)/\epsilon^2\right)$ and $M = \mathcal{O}\left(\log(T/\delta)/\epsilon^2\right)$. Moreover, the algorithm can be run in an online manner.

While the sample complexity of this algorithm is near-optimal [21], it is interesting to investigate classical run time improvements. To this end, we now discuss our approximate Sparsitron algorithm, i.e., Algorithm 4 with the "approximate" flag. The difference to the original algorithm is that most inner products are estimated instead of computed exactly.

The provable learning guarantee for the approximate algorithm follows from the original work but relies on a few additional ideas. First, Fact 2 discusses the construction of a data structure to sample from a probability vector p. Given this data structure, inner products $p \cdot x$ for $x \in [-1,1]^N$ can be determined efficiently to additive accuracy ϵ and success probability $1-\delta$. The corresponding result is Lemma 3 in Appendix A. Since here $\|x\|_{\max} \leq 1$, the run time for a single inner product estimation is $\mathcal{O}\left(\frac{\|x\|_{\max}^2}{\epsilon^2}\log\frac{1}{\delta}\right) = \mathcal{O}\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$. A scaling with $1/\epsilon^2$ is obtained, in contrast to quantum estimation which scales with $1/\epsilon$.

Second, as mentioned before the empirical risk is an approximation to the true risk as only M training examples are used. The inner product estimation leads to an approximate empirical risk. For any vector $v \in [-1,1]^N$ and given samples $(a^{(m)}, b^{(m)}) \in [-1,1]^N \times [0,1]$ for $m \in [M]$ from $\mathcal{D}$, let $z^{(m)}$ be the estimates of the inner product $v \cdot a^{(m)}$. The approximate empirical risk is defined as

$$\widetilde{\varepsilon}(v) := \frac{1}{M}\sum_{m=1}^{M}\left(\sigma\left(z^{(m)}\right) - b^{(m)}\right)^2. \tag{22}$$

We can use this approximate risk to bound the true risk via the triangle inequality. Third, each inner product estimation is probabilistic, hence we bound the overall success probability of the algorithm with a union bound together with the probabilistic steps of the original algorithm.

As we now show, the run time of the approximate classical algorithm is about $\widetilde{\mathcal{O}}\left(T(N + \frac{\lambda^4 M}{\epsilon^2}\log\frac{1}{\delta})\right)$. The multiplicative updates and maintaining of a sampling data structure still cost $\widetilde{\mathcal{O}}(N)$. Plugging in T and M, the run time can be expressed as $\widetilde{\mathcal{O}}\left(N\frac{\lambda^2}{\epsilon^2}\log\frac{1}{\delta} + \frac{\lambda^6}{\epsilon^6}\log^3\frac{1}{\delta}\right)$. In some ranges of parameters, this run time can be considered an improvement over the original Sparsitron which has a run time of $\widetilde{\mathcal{O}}\left(N\frac{\lambda^2}{\epsilon^4}\log^2\frac{1}{\delta}\right)$.

Theorem 6 (Approximate Sparsitron)**.** *With the same assumptions on $\mathcal{D}$, σ, ϵ, δ, λ, T, M, and β as in Theorem 5, Algorithm 4 with flag "approximate" produces a vector $v \in \mathbb{R}^N$ such that with probability at least $1-\delta$,*

$$\varepsilon(v) \leq \epsilon. \tag{23}$$

The run time of the algorithm is

$\widetilde{\mathcal{O}}\left(T\left(N+\frac{M\lambda^4}{\epsilon^2}\log\frac{1}{\delta}\right)\right)$. *Again, the algorithm can be run in an online manner.*

Proof. In the original work [18], due to the guarantees of the Hedge algorithm, it was derived that for the true risk and all computed $p^{(t)}$ it holds that

$$\min_{t\in[T]}\varepsilon\left(\lambda p^{(t)}\right)\leq\epsilon. \tag{24}$$

We can easily let $\epsilon\to\epsilon/2$. Our algorithm selects $v=\lambda p^{(t')}$ for $t'=\arg\min_{t\in[T]}\widetilde{\varepsilon}(\lambda p^{(t)})$, where $\widetilde{\varepsilon}$ is the empirical risk estimated from the imprecise inner products. We need to show that for v we have a similar guarantee.

The closeness of the true and the empirical risk is achieved as in the original work by choosing $M=C\log(T/\delta)/\epsilon^2$, with a constant C, such that for all $t\in[T]$ with probability $1-\delta$

$$\left|\varepsilon\left(\lambda p^{(t)}\right)-\widehat{\varepsilon}\left(\lambda p^{(t)}\right)\right|\leq\frac{\epsilon}{8}. \tag{25}$$

In addition, the error of inner products used for computing the empirical risk is set to $\left|z^{(t,m)}-p^{(t)}\cdot a^{(m)}\right|\leq\frac{\epsilon}{16\lambda}$, hence the induced error in the empirical risk is bounded as

$$\begin{aligned}\left|\widetilde{\varepsilon}\left(\lambda p^{(t)}\right)-\widehat{\varepsilon}\left(\lambda p^{(t)}\right)\right| &\leq \frac{2\lambda}{M}\sum_{m=1}^{M}\left|z^{(t,m)}-p^{(t)}\cdot a^{(m)}\right|\\ &\leq \frac{\epsilon}{8},\end{aligned} \tag{26}$$

since $b^{(m)}\in[0,1]$ and the Lipschitz constant of x^2 on $[-1,1]$ is 2. Hence we also have that

$$\left|\widetilde{\varepsilon}\left(\lambda p^{(t)}\right)-\varepsilon\left(\lambda p^{(t)}\right)\right|\leq\frac{\epsilon}{4}. \tag{27}$$

Therefore, the true risk of v can be bounded as

$$\begin{aligned}\varepsilon(v) &\leq \frac{\epsilon}{4}+\widetilde{\varepsilon}(v)=\frac{\epsilon}{4}+\min_{t\in[T]}\widetilde{\varepsilon}\left(\lambda p^{(t)}\right)\\ &= \frac{\epsilon}{4}+\min_{t\in[T]}\left(\widetilde{\varepsilon}\left(\lambda p^{(t)}\right)-\varepsilon\left(\lambda p^{(t)}\right)+\varepsilon\left(\lambda p^{(t)}\right)\right)\\ &\leq \frac{\epsilon}{2}+\min_{t\in[T]}\varepsilon\left(\lambda p^{(t)}\right)\leq\epsilon.\end{aligned} \tag{28}$$

Next, we discuss the run time. Computing the loss vector and maintaining the $p^{(t)}$ sampling data structure costs $\widetilde{\mathcal{O}}(N)$. Computing the multiplicative update costs $\mathcal{O}(N)$. Estimating the M inner products for risk estimation at every step to accuracy $\epsilon/(16\lambda)$ with success probability $1-\frac{\delta}{MT}$ costs $\widetilde{\mathcal{O}}\left(M\frac{\lambda^2}{\epsilon^2}\log\frac{MT}{\delta}\right)$. Hence the total run time is $\widetilde{\mathcal{O}}\left(T\left(N+M\frac{\lambda^2}{\epsilon^2}\log\frac{MT}{\delta}\right)\right)$ which can be simplified to $\widetilde{\mathcal{O}}\left(T\left(N+\frac{M\lambda^4}{\epsilon^2}\log\frac{1}{\delta}\right)\right)$. The success probability of the inner loop is $\left(1-\frac{\delta}{MT}\right)^M\geq 1-\frac{\delta}{T}$. The success probability over all steps is $\left(1-\frac{\delta}{T}\right)^T\geq 1-\delta$. Together with the success probability of the martingale estimation and the risk estimation of the original algorithm this gives a total success probability of the algorithm of at least $1-3\delta$. For the theorem statement, let $\delta\to\delta/3$. □

To achieve this run time, we were able to compute the first inner product $p^{(t)}\cdot x^{(t)}$ exactly and only estimate the M inner products $p^{(t)}\cdot a^{(m)}$. In the quantum case, we will also estimate the first inner product, which increases the amount of error analysis, as will be shown now.

V. QUANTUM SPARSITRON

In this section, we construct a quantum algorithm for the Sparsitron. The algorithm is again based on quantum minimum finding, see Lemma 4, amplitude amplification and estimation, see Lemma 5, and inner product estimation, see Lemma 7, all in Appendix B. Similar to the quantum Hedge algorithms above, the core idea is to never explicitly store the weight vector $w^{(t)}$. Rather, norms and inner products are estimated and stored. In the iteration, access to these quantities and the new training datum allow to prepare a new loss oracle and a new weight quantum state. This state preparation can then in turn be used to compute the new norm and inner products. We can expect to obtain a quantum speedup in the dimension N. On the other hand, we do not expect a quantum speedup in the number of samples T as the provable learning guarantees are classical and invoke the Hedge algorithm. In fact, we obtain again a worsening of the performance in T. If λ and $1/\epsilon$ are $\operatorname{poly}\log N$, this worsening is however tolerable as T is then also $\operatorname{poly}\log N$.

We first specify the input model. Here, we assume quantum access to the training data. The access model can be turned into an online setting by providing sequential access to the unitaries.

Data Input 2 (Training sets)**.** *Let $j\in[N]$, $t\in[T]$, and $m\in[M]$. Assume $\mathcal{O}(1)$ bits are sufficient to store $x_j^{(t)}$ and $a_j^{(m)}$. Assume to be given access to T unitaries $U_{\text{train1}}^{(t)}$ and M unitaries $U_{\text{train2}}^{(m)}$ on $\mathcal{O}(\log N)$ qubits that perform the operations $|j\rangle|\bar{0}\rangle\to|j\rangle\left|x_j^{(t)}\right\rangle$ and $|j\rangle|\bar{0}\rangle\to|j\rangle\left|a_j^{(m)}\right\rangle$, respectively.*

The same discussion regarding the $|\bar{0}\rangle$ state as in Data Input 1 applies. The data access allows to arithmetically compute the desired losses in quantum superposition.

Lemma 2 (Loss quantum circuits)**.** *Given Input 2 and classical access to the numbers $\lambda\geq 0$, $h\in[-1,1]$ and $y\in[0,1]$. For $t\in[T]$, the quantum operation $|j\rangle|\bar{0}\rangle\to|j\rangle\left|\frac{1}{2}\left(1+(\sigma(\lambda h)-y)\,x_j^{(t)}\right)\right\rangle$ for $j\in[N]$ can be constructed on $\mathcal{O}(\log N)$ qubits, where the result is encoded to constant additive accuracy. The run time is $\mathcal{O}(1)$. We denote these quantum circuits by $U_{\tilde{l}^{(t)}}$.*

Proof. Compute $z := \sigma(\lambda h) - y$ classically. Use the quantum access to $|j\rangle \left|x_j^{(t)}\right\rangle$ in superposition. Then we compute $|j\rangle \left|x_j^{(t)}\right\rangle \left|\frac{1}{2}\left(1 + z x_j^{(t)}\right)\right\rangle$, using the well-known quantum circuits for basic arithmetic operations. Uncompute the second register via another query to obtain $|j\rangle \left|\frac{1}{2}\left(1 + z x_j^{(t)}\right)\right\rangle$. □

Hence, we are able to construct the loss unitaries, which in the previous quantum Hedge algorithm were assumed to be given in Input 1. With these loss unitaries, we can compute the weights via Lemma 1 and perform minimum finding and ℓ_1-norm estimation, as before. Computing the inner products is the core step in the Sparsitron. Consider the inner product $h^{(t)} := \frac{w^{(t)}}{\|w^{(t)}\|} \cdot x^{(t)}$. Instead of this inner product, we estimate a shifted inner product because the $x_j^{(t)}$ are $[-1, 1]$ and there can be cancelation effects which make the inner product zero or very close to zero. Lemma 7 discusses the quantum estimation of the inner product. As a byproduct this lemma also provides $w_{\max}^{(t)}$ and an estimate of $\left\|\frac{w^{(t)}}{w_{\max}^{(t)}}\right\|_1$, hence we do not describe these steps separately.

Note that it is not important that the weights $w^{(t)}$ follow exactly the original Sparsitron weights. It is only important that we have a final guarantee from the Hedge algorithm. If the inner product estimations are accurate enough we only obtain a small additional error to the Hedge error bound. We note related work which analyzes the regret bounds with noisy estimates of the gradients in the bandit setting [31].

We proceed with the algorithm for the Quantum Sparsitron, see Algorithm 5. The output of the algorithm are inner product estimates and a norm estimate. The output is not a full classical vector, which would take $\mathcal{O}(N)$ time and space to write down. The inner products estimates allow the preparation of a quantum state proportional to the desired vector q from which one can take samples or compute inner products with other quantum states. Using T and M, the run time is given by $\widetilde{\mathcal{O}}\left(\frac{\lambda^6\sqrt{N}}{\epsilon^7}\log^4\left(\frac{1}{\delta}\right)\right)$, compared to the run time of $\widetilde{\mathcal{O}}\left(N\frac{\lambda^2}{\epsilon^2}\log\frac{1}{\delta} + \frac{\lambda^6}{\epsilon^6}\log^3\frac{1}{\delta}\right)$ of the approximate Sparsitron and the run time of $\widetilde{\mathcal{O}}\left(N\frac{\lambda^2}{\epsilon^4}\log^2\frac{1}{\delta}\right)$ of the original Sparsitron. The statement is as follows.

Theorem 7 (Quantum Sparsitron)**.** *Let the same assumptions on $\mathcal{D}$, σ, ϵ, δ, λ, and β hold as in Theorem 5. Given $T + M = \mathcal{O}\left(\lambda^2\log(N/\delta\epsilon)/\epsilon^2\right)$ independent samples from $\mathcal{D}$ accessed via Data Input 2, Algorithm 5 returns $\left(h^{(1)}, \cdots, h^{(t')}, \Gamma^{(t')}, w_{\max}^{(t')}\right)$, i.e, inner product estimates, a norm estimate, and a maximum weight for some $t' \in [T]$. The run time of the algorithm to obtain this output is $\widetilde{\mathcal{O}}\left(\frac{\lambda^2T^2M\sqrt{N}}{\epsilon}\log(1/\delta)\right)$, where $M = \mathcal{O}\left(\log(T/\delta)/\epsilon^2\right)$. Again, the algorithm can be run in an online manner. Given this output of Algorithm 5, there exists a vector $q \in \mathbb{R}^N$ such that its coordinates q_j can be constructed separately in time $\widetilde{\mathcal{O}}(T)$ and q satisfies with probability at least $1 - \delta$ that*

$$\varepsilon(q) \leq \epsilon. \tag{29}$$

In addition, an approximation to the quantum state $|q\rangle = \sum_{j=1}^N \sqrt{q_j/\|q\|_1}\,|j\rangle$ can be prepared in time $\widetilde{\mathcal{O}}\left(T\sqrt{N}\log(1/\delta)\log(1/\xi)\right)$ with success probability at least $1 - \delta$, where the accuracy is $\xi \in (0, 1)$.

Proof. Correctness. We generalize the analysis of [18] to the case where the inner products needed in the Sparsitron algorithm are only estimated to some accuracy (instead of being computed exactly). Recall that the loss vector at every step is given by

$$\tilde{l}^{(t)} = \frac{1}{2}\left(\vec{1} + \left(\sigma\left(\lambda h^{(t)}\right) - y^{(t)}\right)x^{(t)}\right). \tag{30}$$

Define the random variable

$$Q^{(t)} := p^{(t)} \cdot \tilde{l}^{(t)} - w \cdot \tilde{l}^{(t)}/\lambda. \tag{31}$$

For this random variable, we can establish three useful facts, generalizing the original work with respect to the estimated inner products. The first fact is that since the weights are updated with the loss vector $\tilde{l}^{(t)}$, the resulting probability vector $p^{(t)}$ satisfies a regret bound. From Theorem 1 it follows that

$$\sum_{t=1}^T p^{(t)} \cdot \tilde{l}^{(t)} \leq \min_{j\in[N]} \tilde{L}_j + \sqrt{2T\log N} + \log N, \tag{32}$$

where $\tilde{L}_j = \sum_{t=1}^T \tilde{l}_j^{(t)}$. The second fact is that the sequence $Q^{(t)}$ is not too far from its expectation value. As in the original work, for the bounded martingale difference sequence $Q^{(t)} - \mathbb{E}_{(x^{(t)},y^{(t)})}\left[Q^{(t)}\middle|(x^{(1)}, y^{(1)}), \cdots, (x^{(t-1)}, y^{(t-1)})\right]$, the Azuma-Hoeffding inequality implies with probability at least $1 - \delta$ that

$$\sum_{t=1}^T \mathbb{E}_{(x^{(t)},y^{(t)})}\left[Q^{(t)}\middle|(x^{(1)}, y^{(1)}), \cdots, (x^{(t-1)}, y^{(t-1)})\right] \leq \sum_{t=1}^T Q^{(t)} + \mathcal{O}\left(\sqrt{T\log(1/\delta)}\right). \tag{33}$$

The third fact involves lower-bounding the expectation value of $Q^{(t)}$, taking into account that the inner products are estimated. Consider first

$$\zeta^{(t)} := p^{(t)} \cdot x^{(t)} - w \cdot x^{(t)}/\lambda, \tag{34}$$

for which

$$\left|\zeta^{(t)}\right| \leq \left\|p^{(t)} - w/\lambda\right\|_1 \leq 2, \tag{35}$$

Algorithm 5 Quantum Sparsitron

Input: Error $\epsilon \in (0,1)$, probability $\delta \in (0,1)$, parameter $\beta \in (0,1)$, norm $\lambda \geq 0$, quantum access to training set $(x^{(t)}, y^{(t)}) \in [-1,1]^N \times [0,1]$ for $t \in [T]$ and training set $(a^{(m)}, b^{(m)}) \in [-1,1]^N \times [0,1]$ for $m \in [M]$.

1: **for** $t = 1$ to T **do**
2: Construct unitary for $w^{(t)} = w^{(1)} \beta^{\tilde{l}^{(1)}} \cdots \beta^{\tilde{l}^{(t-1)}}$ via Lemma 1 using unitaries $\{U_{\tilde{l}^{(t')}} : t' \in [t-1]\}$ from Lemma 2.
3: $h^{(t)} \leftarrow$ Estimate $\frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot x^{(t)}$ to additive accuracy $\frac{\epsilon}{8\lambda^2}$ with success probability $1 - \frac{\delta}{2T}$ via Lemma 7. From this subroutine, also store $w^{(t)}_{\max}$ and the estimate of $\left\| \frac{w^{(t)}}{w^{(t)}_{\max}} \right\|_1$.
4: **for** $m = 1$ to M **do**
5: $z^{(t,m)} \leftarrow$ Estimate $\frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot a^{(m)}$ to additive accuracy $\frac{\epsilon}{16\lambda}$ with success probability $1 - \frac{\delta}{2MT}$ via Lemma 7.
6: **end for**
7: $\widetilde{\varepsilon}^{(t)} \leftarrow \frac{1}{M} \sum_{m=1}^{M} \left(\sigma\left(\lambda z^{(t,m)}\right) - b^{(m)} \right)^2$.
8: Construct unitary $U_{l^{(t)}}$ that prepares $|j\rangle \left|\tilde{l}^{(t)}_j\right\rangle$ with $\tilde{l}^{(t)} = \frac{1}{2}\left(\vec{1} + \left(\sigma\left(\lambda h^{(t)}\right) - y^{(t)}\right) x^{(t)}\right)$ for the next step.
9: **end for**
10: $t' = \arg\min_{t \in [T]} \widetilde{\varepsilon}^{(t)}$.

Output: $\left(h^{(1)}, \cdots, h^{(t')}, \Gamma^{(t')}, w^{(t')}_{\max}\right)$.

since $\left\|x^{(t)}\right\|_{\max} \leq 1$. For the expectation value of $Q^{(t)}$ we obtain

$$\mathbb{E}_{\left(x^{(t)}, y^{(t)}\right)} \left[Q^{(t)} \middle| \left(x^{(1)}, y^{(1)}\right), \cdots, \left(x^{(t-1)}, y^{(t-1)}\right) \right] = \\ = \frac{1}{2} \mathbb{E}_{(x^{(t)}, y^{(t)})} \left[\zeta^{(t)} \left(\sigma\left(\lambda h^{(t)}\right) - \sigma\left(w \cdot x^{(t)}\right) \right) \right], \quad (36)$$

using that $p^{(t)} \cdot \vec{1} = 1$ and $w \cdot \vec{1}/\lambda = 1$. Let $h^{(t)}$ be an estimate of $p^{(t)} \cdot x^{(t)}$ with error $\left|h^{(t)} - p^{(t)} \cdot x^{(t)}\right| \leq \epsilon_{px}$. From the 1-Lipschitz property, it is easy to see that $\left|\sigma\left(\lambda h^{(t)}\right) - \sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right)\right| \leq \lambda \epsilon_{px}$. Using this fact and Eq. (35), we obtain

$$\zeta^{(t)} \sigma\left(\lambda h^{(t)}\right) \geq \zeta^{(t)} \sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right) - 2\lambda \epsilon_{px}. \quad (37)$$

Using Eq. (37) in Eq. (36), we obtain the lower bound

$$\mathbb{E}_{\left(x^{(t)}, y^{(t)}\right)} \left[Q^{(t)} \middle| \left(x^{(1)}, y^{(1)}\right), \cdots \right] \\ \geq \frac{1}{2\lambda} \mathbb{E}_{(x^{(t)}, y^{(t)})} \left[\left(\sigma\left(\lambda p^{(t)} \cdot x^{(t)}\right) - \sigma\left(w \cdot x^{(t)}\right) \right)^2 \right] \\ - \lambda \epsilon_{px}. \quad (38)$$

Here we have used that for all $a, b \in \mathbb{R}$ that $(a-b)(\sigma(a) - \sigma(b)) \geq (\sigma(a) - \sigma(b))^2$.

These three facts related to $Q^{(t)}$ are now combined to derive the guarantee of the algorithm. The first term in Eq. (38) is by definition the risk $\frac{1}{2\lambda} \varepsilon\left(\lambda p^{(t)}\right)$. Hence, for the risk we have

$$\frac{\varepsilon\left(\lambda p^{(t)}\right)}{2\lambda} \leq \mathbb{E}_{(x^{(t)}, y^{(t)})} \left[Q^{(t)} \middle| (x^{(1)}, y^{(1)}), \cdots \right] \\ + \lambda \epsilon_{px}. \quad (39)$$

Using the result derived from the Azuma-Hoeffding inequality, Eq. (33), we obtain with probability at least $1 - \delta$ that

$$\frac{1}{2\lambda} \sum_{t=1}^{T} \varepsilon\left(\lambda p^{(t)}\right) \leq \sum_{t=1}^{T} Q^{(t)} + \mathcal{O}\left(\sqrt{T \log(1/\delta)}\right) \\ + \lambda \epsilon_{px} T. \quad (40)$$

Using the Hedge regret bound Eq. (32), we can bound

$$\sum_{t=1}^{T} Q^{(t)} \leq \min_{j \in [N]} \tilde{L}_j + \sqrt{2T \log N} + \log N - \sum_{t=1}^{T} \frac{w \cdot \tilde{l}^{(t)}}{\lambda} \\ \leq \sqrt{2T \log N} + \log N. \quad (41)$$

The second inequality follows from $\min_{j \in [N]} \tilde{L}_j - \sum_{t=1}^{T} w \cdot \tilde{l}^{(t)}/\lambda \leq 0$ since $\lambda = \|w\|_1$. From the upper bound for the sum, we obtain an upper bound for the minimum element since $\min_{t \in [T]} \varepsilon\left(\lambda p^{(t)}\right) \leq \frac{1}{T} \sum_{t=1}^{T} \varepsilon\left(\lambda p^{(t)}\right)$. Similar to the original work, setting $T > C' \lambda^2 \log(N/\delta)/\epsilon^2$ with a constant C' and $\epsilon_{px} = \frac{\epsilon}{8\lambda^2}$ we obtain

$$\min_{t \in [T]} \varepsilon\left(\lambda p^{(t)}\right) \leq \\ \leq \mathcal{O}(\lambda) \frac{\sqrt{2T \log N} + \log N + \sqrt{T \log 1/\delta}}{T} + 2\lambda^2 \epsilon_{px} \\ \leq \frac{\epsilon}{4} + \frac{\epsilon}{4} = \frac{\epsilon}{2}. \quad (42)$$

Note that the algorithm selects $v = \lambda p^{(t')}$ for $t' = \arg\min_{t \in [T]} \widetilde{\varepsilon}\left(\lambda p^{(t)}\right)$, where $\widetilde{\varepsilon}$ is the empirical risk estimated from the imprecise inner products. The final risk bound is achieved as in the original work and the approximate Algorithm 4. As in Eq. (28), the true risk of v is bounded as

$$\varepsilon(v) \leq \epsilon. \quad (43)$$

Run time. We discuss the inner product estimation and the total run time. Fix $t \in [T]$. Assume we are given the Input 2, the Sparsitron inner product estimates up to time $t-1$, $h^{(1)}, \cdots, h^{(t-1)}$, and the corresponding unitaries $U_{\tilde{l}^{(1)}}, \cdots, U_{\tilde{l}^{(t-1)}}$ from Lemma 2. Together with the weight computation Lemma 1 and quantum access to the training data, construct the operation that computes

$|j\rangle \left|w_j^{(t)}\right\rangle$ by basic arithmetic operations and uncomputing unnecessary registers.

Using Lemma 7, we obtain an estimate $h^{(t)}$ of the inner product $\frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot x^{(t)}$. The additive accuracy is $\frac{\epsilon}{8\lambda^2}$ and the success probability is $1 - \frac{\delta}{2T}$. This step uses in total $\widetilde{\mathcal{O}}\left(\frac{\lambda^2 T\sqrt{N}}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$ quantum gates.

For the M inner products for the risk estimation at every step, we again use Lemma 7 to obtain an estimate $z^{(t,m)}$ of the inner product $\frac{w^{(t)}}{\|w^{(t)}\|_1} \cdot a^{(m)}$. The additive accuracy is $\frac{\epsilon}{16\lambda}$ and the success probability is $1 - \frac{\delta}{2MT}$. This step uses in total $\widetilde{\mathcal{O}}\left(\frac{\lambda T\sqrt{N}}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$ quantum gates.

The total cost of the algorithm is thus $\widetilde{\mathcal{O}}\left(\frac{\lambda^2 T^2\sqrt{N}}{\epsilon} \log\left(\frac{1}{\delta}\right) + \frac{\lambda T^2 M\sqrt{N}}{\epsilon} \log(\frac{1}{\delta})\right)$ which can be simplified to $\widetilde{\mathcal{O}}\left(\frac{\lambda^2 T^2 M\sqrt{N}}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$. The success probability of the inner loop is $(1 - \frac{\delta}{2MT})^M \geq 1 - \frac{\delta}{2T}$. The success probability of all the probabilistic steps in the algorithm is $\left(1 - \frac{\delta}{2T}\right)^{2T} \geq 1 - \delta$. This leads to a total success probability of at least $1 - 3\delta$. Finally, let $\delta \to \delta/3$.

Output. From the output of the algorithm, the construction of a single element q_j of a classical vector takes time $\widetilde{\mathcal{O}}(T)$. That is because

$$q_j = \lambda \frac{\beta^{\sum_{t=1}^{t'-1} \frac{1}{2}\left(1+\left(\sigma\left(h^{(t)}\right)-y^{(t)}\right)x_j^{(t)}\right)}}{w_{\max}^{(t')}\Gamma^{(t')}}, \tag{44}$$

where the computation of the sum takes time $\mathcal{O}(T)$. For the quantum state preparation, use Lemma 5 with $u_j = q_j/q_{\max}$, where $q_{\max} = \max_j q_j$, and the efficient computability of q_j in $\mathcal{O}(T)$. The maximum finding can be done as before. By Lemma 5 (iii), preparing $|q\rangle = \sum_{j=1}^N \sqrt{q/\|q\|_1}\,|j\rangle$ takes a run time of $\widetilde{\mathcal{O}}\left(T\sqrt{N}\log\left(\frac{1}{\delta}\right)\log\left(\frac{1}{\xi}\right)\right)$. □

We continue with remarks on the practicality of the output of the quantum algorithm. The main scenario is the classification of a new sample. Let the new sample be $x_{\text{new}} \in [-1,1]^N$ and quantum access to the sample be given similar to Data Input 2. With the same amplitude estimation methods as above, estimate the inner product $q \cdot x_{\text{new}}$ of the new sample and the vector q. Then apply the Lipschitz function to the estimate as $\sigma(q \cdot x_{\text{new}})$ to obtain an estimate of the label $y_{\text{new}} \in [0,1]$. These steps only add a proportional overhead to the run time of the Sparsitron and provide classification information on new samples with correctness guarantees. One can envision other scenarios. First, we may not need classical knowledge of all the elements q_j but rather only a small known subset of them. Hence we compute the q_j only on that subset, which can be done efficiently. Second, we may use the quantum state $|q\rangle$ to obtain knowledge about the large elements of the vector. If $|q\rangle$ is sparse, then after a small number of measurements in the computational basis we obtain the positions j of the large elements and use this knowledge to compute them. Finally, we may have measurement operators O_i providing information about global properties of w. We can measure these operators on the state $|q\rangle$ to provide information about the true w more efficiently than a classical algorithm. In summary, the algorithm may find practical use in these and other scenarios. Similar to many quantum algorithms for machine learning, one main obstacle for practical use is quantum data access. Functioning large-scale quantum data input devices remain yet to be developed, hence our algorithm may show practical use only in the longer term once such devices are available.

Finally, we show the application of the Quantum Sparsitron to Ising models as a corollary. Please refer to the beginning of Section IV for a brief introduction to the problem. Recall that the width of an Ising model is defined as $\lambda(A,\theta) = \max_i \left(\sum_j |A_{ij}| + |\theta_i|\right)$, see Eq. (17) for the definition of A and θ. The algorithm is a combination of the algorithm in [18] with the Quantum Sparsitron discussed above.

Corollary 1 (Quantum Learning of Ising models)**.** *Given an N-variable Ising model with width $\leq \lambda$ for $\lambda \geq 0$. Given quantum query access to the entries of the samples from the Ising model. Given $\epsilon, \delta \in (0,1)$, and $T = \mathcal{O}\left(\lambda \exp(\mathcal{O}(\lambda))/\epsilon^4) \log(N/\delta\epsilon)\right)$ independent samples from the Ising distribution, there exists a quantum algorithm that produces classical inner product estimates and norms such that every element of a matrix A^* can be computed in time $\widetilde{\mathcal{O}}(T)$. For the matrix A^* it holds that $\|A - A^*\|_{\max} \leq \epsilon$ with probability at least $1 - \delta$. The run time of the algorithm is $\widetilde{\mathcal{O}}\left(\frac{\lambda^2 T^2 M N^{3/2}}{\epsilon} \log\left(\frac{1}{\delta}\right)\right)$, where $M = \mathcal{O}\left(\log(T/\delta)/\epsilon^2\right)$. Quantum states of the columns/rows of the matrix A^* with ϵ distance can be prepared in time $\widetilde{\mathcal{O}}\left(\frac{T\sqrt{N}}{\epsilon}\right)$. Again, the algorithm can be run in an online manner.*

Proof. Apply the Quantum Sparsitron $\mathcal{O}(N)$ times for each node of the Ising model, hence the run time is $\widetilde{\mathcal{O}}\left(\frac{\lambda^2 T^2 M N^{3/2}}{\epsilon} \log\left(\frac{T}{\delta}\right)\right)$. At this cost, the Quantum Sparsitron has the same guarantee as the classical Sparsitron. It was shown in [18] that the obtained guarantees for the GLMs can be translated to the guarantee $\|A - A^*\|_{\max} \leq \epsilon$ for the matrix of the Ising model. Hence we obtain the guarantee for the Ising model learning. The rows/columns of A^* can be reconstructed as in Theorem 7, either classically element by element or as quantum states. □

The classical algorithm by Klivans and Meka [18] has a run time of $\mathcal{O}\left(N^2 T\right)$ with the same near-optimal sample complexity. Corollary 1 can hence be considered an improvement in the N dependency, which comes at the price of a worsening of the run time in T and ϵ.

VI. DISCUSSION AND CONCLUSION

In the main part of this work, we have presented a quantum machine learning algorithm with both provable learning guarantee and provable quantum speedup over the best known classical algorithm. The starting point is a classical algorithm called the Sparsitron, which is a dimensionally sample-optimal algorithm for generalized linear models under modest assumptions. Generalized linear models have a large range of applications, including logistic and Poisson regression, and they appear also in a variety of problems such as the learning of Ising models and Markov Random Fields (MRFs).

The run time of our quantum algorithm, the Quantum Sparsitron, shows a speedup polynomial in the dimension of the problem and a slowdown in the error dependency, while the sample complexity remains the same as for the classical algorithm. The setting here is the standard quantum gate model, i.e., many logical quantum bits with the physical errors being kept under control via error correction. In addition, we assume the availability of unitaries (oracles) which provide access to the training examples. The training examples can be given via efficient quantum circuits or via classical data collected from sampling the true distribution and quantum RAM access to these data. The main quantum subroutines are the well-known amplitude amplification and estimation algorithms, which are here applied in a way that the provable learning guarantee and success probability of the classical algorithm are preserved. Due to the use of amplitude amplification and oracles, the algorithm can be considered more far-term in nature, requiring significantly more resources than the presently available 50-100 noisy qubits.

The optimization problem here is in principle non-convex. The Lipschitz function defining the generalized linear model can be a non-convex function (such as the *quasi*-convex sigmoid function), which leads to hardness results for learning even a single neuron [18, 32]. In the sense that the classical Sparsitron solves this non-convex problem, the quantum algorithm solves the same non-convex problem. While many of the recent quantum algorithms are for convex problems, such as LPs and SDPs [5, 6], this work can be seen as an extension of the same underlying quantum techniques to non-convex problems.

On the classical side, we have shown that the Sparsitron run time (but not the sample complexity) can be sped up via inner product estimation techniques. Randomized linear algebra is a well-studied area which has recently also found application in the discussion of dimension-efficient classical algorithms for various problems considered for quantum machine learning [33–38]. Our work relates to these results in the sense that we have started at a near-optimal classical machine learning algorithm. It may be interesting to take near-optimal versions of [33–38] as starting points and exhibit quantum speedups for the various problems.

VII. 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, and Baidu-NUS Research Project Nr. 2019-03-07. This work was also partially funded by QuantERA ERA-NET Cofund project QuantAlgo and the ANR project ANR-18-CE47-0010 QUDATA.

Appendix A: Classical sampling

We mention a result for the efficient sampling from a probability vector.

Fact 2 (ℓ_1-sampling [39, 40])**.** *Given an N-dimensional probability vector p. There exists a data structure to sample an index $j \in [N]$ with probability p_j which can be constructed in time $\widetilde{\mathcal{O}}(N)$. One sample can be obtained in time $\widetilde{\mathcal{O}}(1)$.*

Regarding this result, Ref. [39] shows an algorithm with preparation in $\mathcal{O}(N)$ and sampling in $\mathcal{O}(1)$ assuming constant time operations for addition, comparison, and random number generation, among others. However, storing and processing pointers $j \in [N]$ takes $\mathcal{O}(\log N)$ bits and operations, hence we take the slightly worse $\widetilde{\mathcal{O}}(N)$ and $\widetilde{\mathcal{O}}(1)$, respectively.

We discuss the sampling of inner products. The proof is standard and adapted from [33] which shows the ℓ_2-sampling case.

Lemma 3 (Inner product estimation)**.** *Let $\epsilon, \delta \in (0,1)$. Given query access to $x \in [-1,1]^N$ and ℓ_1-sampling access to an N-dimensional probability vector p. We can determine $p \cdot x$ to additive error ϵ with success probability at least $1-\delta$ with $\mathcal{O}\left(\frac{\|x\|_{\max}^2}{\epsilon^2}\log\frac{1}{\delta}\right)$ queries and samples, and $\widetilde{\mathcal{O}}\left(\frac{\|x\|_{\max}^2}{\epsilon^2}\log\frac{1}{\delta}\right)$ time complexity.*

Proof. Define a random variable Z with outcome x_j with probability p_j. Note that $\mathbb{E}[Z] = \sum_j p_j x_j = p \cdot x$. Also, $\mathbb{V}[Z] \leq \sum_j x_j^2 p_j \leq \|x\|_{\max}^2$. Take the median of $6\log 1/\delta$ evaluations of the mean of $9/(2\epsilon^2)$ samples of Z to be within $\epsilon\sqrt{\mathbb{V}[Z]} \leq \epsilon\|x\|_{\max}$ of $p \cdot x$ with probability at least $1-\delta$ in $\mathcal{O}\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$ queries. □

Appendix B: Quantum subroutines

Proof of Lemma 1. With a computational register involving $\mathcal{O}(T)$ ancilla qubits for the losses, perform $|j\rangle|\bar{0}\rangle \rightarrow |j\rangle\left|l_j^{(1)}\right\rangle\ldots\left|l_j^{(t-1)}\right\rangle|\bar{0}\rangle \rightarrow$

$|j\rangle \left|l_j^{(1)}\right\rangle \ldots \left|l_j^{(t-1)}\right\rangle \left|\sum_{t'=1}^{t-1} l_j^{(t')}\right\rangle$ to sufficient accuracy, using the oracles. Uncomputing the loss registers via additional queries leads to the result. With the quantum circuits for basic arithmetic operations, the operations for $|j\rangle \left|w_j^{(t)}\right\rangle$ and $|j\rangle \left|w_j^{(t)}/w_{\max}^{(t)}\right\rangle$ can also be prepared. □

In the following, for the data access to the vector u, consider the discussion regarding the $|\bar{0}\rangle$ state given after Data Input 1.

Lemma 4 (Quantum minimum finding [19])**.** *Given quantum access to a vector $u \in [0,1]^N$ via the operation $|j\rangle|\bar{0}\rangle \to |j\rangle|u_j\rangle$ on $\mathcal{O}(\log N)$ qubits, where u_j is encoded to additive accuracy $\mathcal{O}(1/N)$. Then, we can find the minimum $u_{\min} = \min_{j\in[N]} u_j$ with success probability $1-\delta$ 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.*

The minimum finding can be turned straightforwardly into a maximum finding algorithm. Next, we state the results for estimating the ℓ_1-norm of a vector and preparing states encoding the square root of the vector elements.

Lemma 5 (Quantum state preparation and norm estimation)**.** *Let $\eta > 0$. Given a non-zero vector $u \in [0,1]^N$, with $\max_j u_j = 1$. Given quantum access to u via the operation $|j\rangle|\bar{0}\rangle \to |j\rangle|u_j\rangle$ on $\mathcal{O}(\log N + \log 1/\eta)$ qubits, where u_j is encoded to additive accuracy η. Then:*

(i) There exists a unitary operator that prepares the state $\frac{1}{\sqrt{N}}\sum_{j=1}^N |j\rangle\left(\sqrt{u_j}|0\rangle + \sqrt{1-u_j}|1\rangle\right)$ with two queries and number of gates $\mathcal{O}(\log N + \log 1/\eta)$. Denote this unitary by U_χ.

(ii) Let $\epsilon > 0$ such that $\eta \le \epsilon/(2N)$ and $\delta \in (0,1)$. There exists a quantum algorithm that provides an estimate Γ_u of the ℓ_1-norm $\|u\|_1$ such that $|\|u\|_1 - \Gamma_u| \le \epsilon\|u\|_1$, with probability at least $1-\delta$. The algorithm requires $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log(1/\delta)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log(1/\delta)\right)$ gates.

(iii) Let $\xi \in (0,1]$ such that $\eta \le \xi/4N$ and $\delta \in (0,1)$. An approximation $|\tilde{p}\rangle = \sum_{j=1}^N \sqrt{\tilde{p}_j}|j\rangle$ to the state $|u\rangle := \sum_{j=1}^N \sqrt{\frac{u_j}{\|u\|_1}}|j\rangle$ can be prepared with probability $1-\delta$, using $\mathcal{O}\left(\sqrt{N}\log(1/\delta)\right)$ calls to the unitary of (i) and $\widetilde{\mathcal{O}}\left(\sqrt{N}\log(1/\xi)\log(1/\delta)\right)$ gates. The approximation in ℓ_1-norm of the probabilities is $\left\|\tilde{p} - \frac{u}{\|u\|_1}\right\|_1 \le \xi$.

Proof. For (i), prepare a uniform superposition of all $|j\rangle$ with $\mathcal{O}(\log N)$ Hadamard gates. With the quantum query access, perform

$$\frac{1}{\sqrt{N}}\sum_{j=1}^N |j\rangle\ |\bar{0}\rangle \to \frac{1}{\sqrt{N}}\sum_{j=1}^N |j\rangle|u_j\rangle|0\rangle \tag{B1}$$
$$\to \frac{1}{\sqrt{N}}\sum_{j=1}^N |j\rangle|u_j\rangle\left(\sqrt{u_j}|0\rangle + \sqrt{1-u_j}|1\rangle\right).$$

The steps consist of an oracle query and a controlled rotation. The rotation is well-defined as $u_j \le 1$ and costs $\mathcal{O}(\log 1/\eta)$ gates. Then uncompute the data register $|u_j\rangle$ with another oracle query.

For (ii), define a unitary $\mathcal{U} = U_\chi(\mathbb{1} - 2|\bar{0}\rangle\langle\bar{0}|)(U_\chi)^\dagger$, with U_χ from (i). Define another unitary by $\mathcal{V} = \mathbb{1} - \mathbb{1}\otimes|0\rangle\langle 0|$. Using K applications of $\mathcal{U}$ and $\mathcal{V}$, Amplitude Estimation [3] allows to provide an estimate $\tilde{a}$ of the quantity $a = \frac{\|u\|_1}{N}$ to accuracy $|\tilde{a}-a| \le 2\pi\frac{\sqrt{a(1-a)}}{K} + \frac{\pi^2}{K^2}$. Following [41], take $K > \frac{6\pi}{\epsilon}\sqrt{N}$, which obtains

$$|\tilde{a}-a| \le \frac{\pi}{K}\left(2\sqrt{a}+\frac{\pi}{K}\right) < \frac{\epsilon_1}{6}\sqrt{\frac{1}{N}}\left(2\sqrt{a}+\frac{\epsilon}{12}\sqrt{\frac{1}{N}}\right)$$
$$\le \frac{\epsilon}{6}\sqrt{\frac{1}{N}}\left(3\sqrt{a}\right) = \frac{\epsilon\sqrt{\|u\|_1}}{2N}. \tag{B2}$$

Since $\|u\|_1 \ge 1$ by assumption, we have $|\tilde{a}-a| \le \frac{\epsilon\|u\|_1}{2N}$. Also, there is an inaccuracy arising from the additive error η of each u_j. As it was assumed that $\eta \le \epsilon/(2N)$, the overall multiplicative error ϵ is obtained for the estimation. For performing a single run of amplitude estimation with K steps, we require $\mathcal{O}(K) = \mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\right)$ queries to the oracles and $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}(\log N + \log(N/\epsilon))\right)$ gates.

For (iii), rewrite the state from (i) as

$$\sqrt{\frac{\|u\|_1}{N}}\ \sum_{j=1}^N\ \sqrt{\frac{u_j}{\|u\|_1}}|j\rangle|0\rangle \tag{B3}$$
$$+\quad \sqrt{1-\frac{\|u\|_1}{N}}\sum_{j=1}^N\sqrt{\frac{1-u_j}{N-\|u\|_1}}|j\rangle|1\rangle.$$

Now amplify the $|0\rangle$ part using Amplitude Amplification [3] via the exponential search technique without knowledge of the normalization, to prepare $\sum_{j=1}^N |j\rangle\sqrt{\frac{u_j}{\|u\|_1}}$ with success probability $1-\delta$. The amplification requires $\mathcal{O}\left(\sqrt{\frac{N}{\|u\|_1}}\log(1/\delta)\right) = \mathcal{O}\left(\sqrt{N}\log(1/\delta)\right)$ calls to the unitary of (i), as $\|u\|_1 \ge 1$. The gate complexity derives from the gate complexity of (i). Denote the η-additive approximation to u_j by $\tilde{u}_j$, and evaluate the ℓ_1-distance of the probabilities. First, $|\|u\|_1 - \|\tilde{u}\|_1| \le N\eta$. One obtains $\left\|\tilde{p} - \frac{u}{\|u\|_1}\right\|_1 = \left\|\frac{\tilde{u}}{\|\tilde{u}\|_1} - \frac{u}{\|u\|_1}\right\|_1 \le \sum_j\left|\frac{\tilde{u}_j}{\|\tilde{u}\|_1} - \frac{u_j}{\|\tilde{u}\|_1}\right| + \sum_j\left|\frac{u_j}{\|\tilde{u}\|_1} - \frac{u_j}{\|u\|_1}\right| \le \frac{N\eta}{\|\tilde{u}\|_1} + \frac{N\eta}{\|\tilde{u}\|_1}$. We also obtain $\frac{1}{\|\tilde{u}\|_1} \le \frac{1}{\|u\|_1 - N\eta} \le \frac{2}{\|u\|_1}$ for $\eta \le \|u\|_1/2N$. Since $\eta \le \|u\|_1\xi/(4N)$, the distance is $\left\|\tilde{p} - \frac{u}{\|u\|_1}\right\|_1 \le \xi$ as desired. □

Lemma 6 (Quantum inner product estimation with relative accuracy)**.** *Let $\epsilon, \delta \in (0,1)$. Given quantum access to two vectors $u, v \in [0,1]^N$, where u_j and v_j are encoded to additive accuracy $\eta = \mathcal{O}(1/N)$. Then, an estimate I for the inner product can be provided such that $|I - u \cdot v/\|u\|_1| \leq \epsilon\ u \cdot v/\|u\|_1$ with success probability $1-\delta$. This estimate is obtained with $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates.*

Proof. Via Lemma 4, determine $u_{\max}$ with success probability $1-\delta$ with $\mathcal{O}\left(\sqrt{N}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\sqrt{N}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates. Apply Lemma 5 with the vector $\frac{u}{u_{\max}}$ to obtain an estimate Γ_u of the norm $\left\|\frac{u}{u_{\max}}\right\|_1$ to relative accuracy $\epsilon_u = \epsilon/2$ with success probability $1-\delta$. This estimation takes $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates.

Define the vector z with $z_j = u_j v_j$. Via Lemma 4, determine $z_{\max}$ with success probability $1-\delta$ with $\mathcal{O}\left(\sqrt{N}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\sqrt{N}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates. If $z_{\max} = 0$ up to numerical accuracy, the estimate is $I = 0$ and we are done. Otherwise, apply Lemma 5 with the vector $\frac{z}{z_{\max}}$ to obtain an estimate Γ_z of the norm $\left\|\frac{z}{z_{\max}}\right\|_1$ to relative accuracy $\epsilon_u = \epsilon/2$ with success probability $1-\delta$. This estimation takes $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates.

With Lemma 8, we have

$$\left|\frac{\Gamma_z}{\Gamma_u} - \frac{u_{\max}}{z_{\max}}\frac{u\cdot v}{\|u\|_1}\right| \leq \frac{u_{\max}}{z_{\max}}\frac{u\cdot v}{\|u\|_1}\frac{\epsilon_z+\epsilon_u}{(1-\epsilon_u)} \tag{B4}$$

$$\leq 2\epsilon\frac{u_{\max}}{z_{\max}}\frac{u\cdot v}{\|u\|_1}, \tag{B5}$$

since $\epsilon_u < 1/2$. Set

$$I = \frac{z_{\max}}{u_{\max}}\frac{\Gamma_z}{\Gamma_u}, \tag{B6}$$

and we have $|I - u\cdot v/\|u\|_1| \leq 2\epsilon\ u\cdot v/\|u\|_1$. The total success probability of the four probabilistic steps is at least $1-4\delta$ via a union bound. Choosing $\epsilon \to \epsilon/2$ and $\delta \to \delta/4$ leads to the result. □

Lemma 7 (Quantum inner product estimation with additive accuracy)**.** *Let $\epsilon, \delta \in (0,1)$. Given quantum access to a non-zero vector $u \in [0,1]^N$ and another vector $v \in [-1,1]^N$, where u_j and v_j are encoded to additive accuracy $\eta = \mathcal{O}(1/N)$. Then, an estimate I for the inner product can be provided such that $|I - u\cdot v/\|u\|_1| \leq \epsilon$ with success probability $1-\delta$. This estimate is obtained with $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates.*

Note that as a byproduct, the value $u_{\max}$ and an estimate of $\|u/u_{\max}\|_1$ with relative accuracy ϵ can be provided with probability at least $1-\delta$.

Proof. Via Lemma 4, determine $\|u\|_{\max}$ with success probability $1-\delta$ with $\mathcal{O}\left(\sqrt{N}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\sqrt{N}\log\left(\frac{1}{\eta}\right)\log\left(\frac{1}{\delta}\right)\right)$ quantum gates. Apply Lemma 5 with the vector $\frac{u}{u_{\max}}$ to obtain an estimate Γ_u of the norm $\left\|\frac{u}{u_{\max}}\right\|_1$ to relative accuracy $\epsilon_u = \epsilon/2$ with success probability $1-\delta$. This estimation takes $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates.

Similarily, consider the vector z with elements $z_j := u_j(v_j+3) \in [0,4]$. Determine $\|z\|_{\max}$ with success probability $1-\delta$ with $\mathcal{O}\left(\sqrt{N}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\sqrt{N}\log\left(\frac{1}{\delta}\right)\right)$ quantum gates. Apply Lemma 5 with the vector $z/z_{\max}$ to obtain an estimate Γ_z of the norm $\|z/z_{\max}\|_1$ to relative accuracy $\epsilon_z = \epsilon/2$ with success probability $1-\delta$. This estimation takes $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$.

The exact quantities are related via

$$\frac{u\cdot v}{\|u\|_1} = \frac{z_{\max}}{u_{\max}}\frac{\|\frac{z}{z_{\max}}\|_1}{\|\frac{u}{u_{\max}}\|_1} - 3. \tag{B7}$$

Considering the estimator $I = \frac{z_{\max}}{u_{\max}}\frac{\Gamma_z}{\Gamma_u} - 3$, from Lemma 8, we have

$$\left|I - \frac{u\cdot v}{\|u\|_1}\right| = \frac{z_{\max}}{u_{\max}}\left|\frac{\Gamma_z}{\Gamma_u} - \frac{\|\frac{z}{z_{\max}}\|_1}{\|\frac{u}{u_{\max}}\|_1}\right| \tag{B8}$$

$$\leq \frac{\epsilon_u+\epsilon_z}{1-\epsilon_u}\frac{\|z\|_1}{\|u\|_1} \leq 8\epsilon.$$

In the last steps we have used that

$$\frac{\|z\|_1}{\|u\|_1} \equiv \frac{\sum_j u_j(v_j+3)}{\sum_j u_j} \leq \frac{4\sum_j u_j}{\sum_j u_j} = 4, \tag{B9}$$

and $\epsilon_u < 1/2$.

All steps together take $\mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\log\frac{1}{\delta}\right)$ queries and $\widetilde{\mathcal{O}}\left(\frac{\sqrt{N}}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ gates. The total success probability of all the probabilistic steps is at least $1-4\delta$ via a union bound. Choosing $\epsilon \to \epsilon/8$ and $\delta \to \delta/4$ leads to the result.

□

Lemma 8. *Let $\tilde{a}$ be an estimate of $a > 0$ such that $|\tilde{a} - a| \leq \epsilon_a a$. with $\epsilon_a \in (0,1)$. Similarly, let $\tilde{b}$ be an estimate of $b > 0$ and $\epsilon_b \in (0,1)$ such that $|\tilde{b} - b| \leq \epsilon_b b$. Then the ratio a/b is estimated to relative error $\left|\frac{\tilde{a}}{\tilde{b}} - \frac{a}{b}\right| \leq \left(\frac{\epsilon_a+\epsilon_b}{1-\epsilon_b}\right)\frac{a}{b}$.*

Proof. Note that $b - \tilde{b} \leq |\tilde{b} - b| \leq \epsilon_b b$, from which we deduce $\frac{1}{\tilde{b}} \leq \frac{1}{b(1-\epsilon_b)}$. In addition, $\left|\frac{\tilde{a}}{\tilde{b}} - \frac{a}{b}\right| = \left|\frac{\tilde{a}b - a\tilde{b}}{\tilde{b}b}\right| = \left|\frac{\tilde{a}b - ab + ab - a\tilde{b}}{\tilde{b}b}\right| = \left|\frac{\tilde{a}-a}{\tilde{b}} + \frac{a}{\tilde{b}}\frac{b-\tilde{b}}{b}\right| \leq \left|\frac{\tilde{a}-a}{\tilde{b}}\right| + \frac{a}{\tilde{b}}\left|\frac{b-\tilde{b}}{b}\right| \leq \frac{\epsilon_a a + \epsilon_b a}{\tilde{b}} \leq \frac{a}{b}\frac{\epsilon_a + \epsilon_b}{(1-\epsilon_b)}$. □

[1] S. Chakrabarti, A. M. Childs, T. Li, and X. Wu, arXiv:1809.01731 (2018).

[2] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, arXiv:1809.00643 (2018).

[3] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, Contemporary Mathematics **305**, 53 (2002).

[4] S. Arora and S. Kale, J. ACM **63**, 12:1 (2016).

[5] F. G. S. L. Brandão and K. M. Svore, in *Proceedings of the 58th Symposium on Foundations of Computer Science* (2017), FOCS '17, pp. 415–426.

[6] J. van Apeldoorn and A. Gilyén, in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming* (2017), vol. 132 of *ICALP '19*, pp. 99:1–99:15.

[7] M. Grigoriadis and L. Khachiyan, Operations Research Letters **18**, 53 (1995).

[8] J. van Apeldoorn and A. Gilyén, arXiv:1904.03180 (2019).

[9] T. Li, S. Chakrabarti, and X. Wu, arXiv:1904.02276 (2019).

[10] K. Clarkson, E. Hazan, and D. Woodruff, Journal of the ACM **59**, 23 (2012).

[11] V. Giovannetti, S. Lloyd, and L. Maccone, Physical Review Letters **100**, 160501 (2008).

[12] S. Lloyd, M. Mohseni, and P. Rebentrost, arXiv:1307.0411 (2013).

[13] A. Harrow, A. Hassidim, and S. Lloyd, Physical Review Letters **103** (2009).

[14] S. Aaronson, Nature Physics **11**, 291 (2015).

[15] Y. Freund and R. E. Schapire, J. Comput. Syst. Sci. **55**, 119 (1997).

[16] X. Wang, Y. Ma, M.-H. Hsieh, and M. Yung, arXiv:1902.00869 (2019).

[17] S. Arunachalam and R. Maity, arXiv:2002.05056 (2020).

[18] A. Klivans and R. Meka, in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)* (2017), pp. 343–354.

[19] C. Dürr and P. Høyer, arXiv:quant-ph/9607014 (1996).

[20] G. Bresler, in *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing* (2015), STOC '15, pp. 771–782.

[21] N. P. Santhanam and M. J. Wainwright, IEEE Transactions on Information Theory **58**, 4117 (2012).

[22] C. Chow and C. Liu, IEEE Transactions on Information Theory **14**, 462 (1968).

[23] S. Dasgupta, in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999), UAI'99, p. 134–141, ISBN 1558606149.

[24] N. Srebro, Artif. Intell. **143**, 123–138 (2003).

[25] A. Anandkumar, D. Hsu, F. Huang, and S. M. Kakade, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Curran Associates Inc., Red Hook, NY, USA, 2012), NIPS'12, p. 1052–1060.

[26] J. Bento and A. Montanari, in *Proceedings of the 22nd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2009), NIPS'09, p. 1303–1311, ISBN 9781615679119.

[27] L. G. Valiant, in *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence* (AAAI Press, 1988), AAAI'88, p. 629–634.

[28] P. Ravikumar, M. J. Wainwright, and J. D. Lafferty, Ann. Statist. **38**, 1287 (2010).

[29] S.-I. Lee, V. Ganapathi, and D. Koller, in *Proceedings of the 19th International Conference on Neural Information Processing Systems* (MIT Press, Cambridge, MA, USA, 2006), NIPS'06, p. 817–824.

[30] C. Daskalakis and Q. Pan, arXiv preprint arXiv:2010.14864 (2020).

[31] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, USA, 2005), SODA '05, p. 385–394.

[32] P. Auer, M. Herbster, and M. K. K. Warmuth, in *Advances in Neural Information Processing Systems 8*, edited by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (MIT Press, 1996), pp. 316–322.

[33] E. Tang, Electronic Colloquium on Computational Complexity **128** (2018).

[34] E. Tang, arXiv:1811.00414 (2018).

[35] N.-H. Chia, H.-H. Lin, and C. Wang, arXiv:1811.04852 (2018).

[36] A. Gilyén, S. Lloyd, and E. Tang, arXiv:1811.04909 (2018).

[37] N.-H. Chia, T. Li, H.-H. Lin, and C. Wang, arXiv:1901.03254 (2019).

[38] N.-H. Chia, A. Gilyén, T. Li, H.-H. Lin, E. Tang, and C. Wang, arXiv:1910.06151 (2019).

[39] M. D. Vose, IEEE Transactions on Software Engineering **17**, 972 (1991).

[40] A. J. Walker, Electronics Letters **10**, 127 (1974).

[41] J. van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, arXiv:1705.01843 (2017).