Quantum Boosting

Srinivasan Arunachalam [1] **Reevu Maity** [2]

Abstract

Boosting is a technique that *boosts* a weak and inaccurate machine learning algorithm into a *strong* accurate learning algorithm. The AdaBoost algorithm by Freund and Schapire (for which they were awarded the Gödel prize in 2003) is one of the widely used boosting algorithms, with many applications in theory and practice. Suppose we have a γ-weak learner for a Boolean concept class $\mathcal{C}$ that takes time $R(\mathcal{C})$, then the time complexity of AdaBoost scales as $\mathsf{VC}(\mathcal{C}) \cdot \mathrm{poly}(R(\mathcal{C}), 1/\gamma)$, where $\mathsf{VC}(\mathcal{C})$ is the VC-dimension of $\mathcal{C}$. In this paper, we show how quantum techniques can improve the time complexity of classical AdaBoost. To this end, suppose we have a γ-weak *quantum* learning algorithm for a Boolean concept class $\mathcal{C}$ that takes time $Q(\mathcal{C})$, we introduce a quantum boosting algorithm whose complexity scales as $\sqrt{\mathsf{VC}(\mathcal{C})} \cdot \mathrm{poly}(Q(\mathcal{C}), 1/\gamma)$; thereby achieving quadratic quantum improvement over classical AdaBoost in terms of $\mathsf{VC}(\mathcal{C})$.

1. Introduction

In the last decade, machine learning (ML) has received tremendous attention due to its success in practice. Given the broad applications of ML, there has been a lot of interest in understanding what are the learning tasks for which *quantum computers* could provide a speedup. In this direction, there has been a flurry of quantum algorithms for practically relevant machine learning tasks that theoretically promise either exponential or polynomial quantum speed-ups over classical computers. In the past, theoretical works on quantum machine learning (QML) have focused on developing efficient quantum algorithms with favourable quantum complexities to solve interesting learning problems. Recently there have been efforts in understanding the interplay between learning algorithms and noisy quantum devices.

The field of QML has given us algorithms for various quantum and classical learning tasks such as (i) quantum improvements to classical algorithms for practically-motivated machine learning tasks such as perceptron learning (Kapoor et al., 2016), support vector machines (Rebentrost et al., 2013), kernel-based classifiers (Havlíček et al., 2019; Li et al., 2019), algorithms to compute gradients (Rebentrost et al., 2019; Gilyén et al., 2019), clustering (Kerenidis et al., 2019; Aïmeur et al., 2007), linear algebra (Prakash, 2014); (ii) learnability of *quantum objects* (Rocchetto, 2018; Yoganathan, 2019; Aaronson, 2007), shadow tomography of quantum states (Aaronson, 2018; Apeldoorn & Gilyén, 2019); (iii) a quantum framework to learn Boolean-valued concept classes (Bernstein & Vazirani, 1993; Bshouty & Jackson, 1999; Atıcı & Servedio, 2005; Arunachalam et al., 2019); (iv) quantum algorithms for optimization (Harrow et al., 2009; Apeldoorn et al., 2020; Chakrabarti et al., 2018); (v) quantum algorithms for machine learning based on generative models (Lloyd & Weedbrook, 2018; Gao et al., 2017).

While these results are promising and establish that quantum computers can indeed provide an improvement for interesting machine learning tasks, there are still several challenges that remain. One important question is whether the assumptions made in some quantum machine learning algorithms are practically feasible? Recently, a couple of works (Chia et al., 2019; Jethwani et al., 2019) demonstrated that under certain assumptions QML algorithms can be dequantized, i.e., they showed the existence of *efficient* classical algorithms for machine learning tasks which were previously believed to provide exponential quantum speedups. In this paper we address another important question:

> Suppose we implement a QML algorithm $\mathcal{A}$ on a quantum device and due to noise in the device, the performance of $\mathcal{A}$ is *weak*, i.e., the output of $\mathcal{A}$ is correct on a *slightly* better-than-half fraction of the inputs. Can we *boost* the performance of $\mathcal{A}$ so that $\mathcal{A}$s output is correct on 99% of the inputs?

Inspired by the classical *Adaptive Boosting* algorithm (also referred to as AdaBoost) due to (Freund & Schapire, 1999), the classical AdaBoost can be used immediately to convert

[1]IBM Research, Yorktown Heights, USA. [2]Clarendon Laboratory, University of Oxford.. Correspondence to: <Srinivasan.Arunachalam@ibm.com>, <reevu.maity@physics.ox.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

a weak quantum learning algorithm to a strong algorithm. In this paper, we provide a *quantum boosting* algorithm that quadratically improves upon the classical AdaBoost algorithm. Using our quantum boosting algorithm, not only can we convert a weak and inaccurate QML algorithm into a strong accurate algorithm, but we can do it in time that is quadratically faster than classical boosting techniques.

2. Classical Boosting

We now briefly describe (Valiant, 1984)s Probably Approximately Correct (PAC) model of learning. Let $n \geq 1$ and $\mathcal{C} \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$ be a concept class. For $\gamma > 0$, we say an algorithm $\mathcal{A}$ *γ-learns* $\mathcal{C}$ in the PAC model if: for every $c \in \mathcal{C}$ and distribution $\mathcal{D} : \{0,1\}^n \to [0,1]$, given examples $(x, c(x))$ where $x \sim \mathcal{D}$, $\mathcal{A}$ outputs $h : \{0,1\}^n \to \{-1,1\}$ such that $\Pr_{x\sim\mathcal{D}}[h(x) = c(x)] \geq 1/2 + \gamma$. In the quantum PAC model, we allow a *quantum learner* to possess a quantum computer and *quantum examples* $\sum_x \sqrt{\mathcal{D}_x}|x, c(x)\rangle$. We call γ the bias of an algorithm, i.e., γ measures the advantage over random guessing. We say $\mathcal{A}$ is a *weak learner* (resp. *strong learner*) if the bias γ scales inverse polynomially with n, i.e., $\gamma = 1/\operatorname{poly}(n)$ (resp. γ is a universal constant independent of n, for simplicity we let $\gamma = 1/6$).

In the early 1990s, (Schapire, 1990; Freund, 1995; Freund & Schapire, 1999) came up with the beautiful boosting algorithm called *AdaBoost* that *efficiently* solves the following problem: suppose we are given a weak learner as a blackbox, can we use this black-box to obtain a strong learner? The AdaBoost algorithm by Freund and Schapire was one of the few theoretical boosting algorithms that were simple enough to be extremely useful in practice, with applications ranging from game theory, statistics, optimization, vision and speech recognition and information geometry. Given the success of AdaBoost in theory and practice, Freund and Schapire won the Gödel prize in 2003.

AdaBoost algorithm. We now give a sketch of the classical AdaBoost algorithm. Let $\mathcal{A}$ be a weak PAC learning algorithm for $\mathcal{C}$ that runs in time $R(\mathcal{C})$ and has bias $\gamma > 0$, i.e., $\mathcal{A}$ does slightly better than random guessing (think of γ as inverse-polynomial in n). The goal of boosting is the following: for every *unknown* distribution $\mathcal{D} : \{0,1\}^n \to [0,1]$ and *unknown* concept $c \in \mathcal{C}$, construct a hypothesis H that satisfies $\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq \frac{2}{3}$. The AdaBoost algorithm by Freund and Schapire produces such an H by invoking $\mathcal{A}$ polynomially many times. The algorithm works as follows: it first obtains M different labelled examples $S = \{(x_i, c(x_i)) : i \in [M]\}$ where $x_i \sim \mathcal{D}$ and then AdaBoost is an iterative algorithm that runs for T steps (for some M, T which we specify later). Let D^1 be the uniform distribution on S. At the tth step, AdaBoost defines a distribution D^t depending on D^{t-1} and invokes $\mathcal{A}$ on the sample S and distribution D^t. Using the output hypothesis h_t of the weak learner $\mathcal{A}$, the AdaBoost algorithm computes the *weighted error* $\varepsilon_t = \Pr_{x\sim D^t}[h_t(x) \neq c(x)]$ which is the probability of h_t misclassifying a randomly selected training example drawn from the distribution D^t. The algorithm then uses ε_t to compute a *weight* $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ and updates the distribution D^t to D^{t+1} as follows

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

where $Z_t = \sum_{x\in S} D_x^t \exp(-c(x)\alpha_t h_t(x))$.[1] After T iterations, the algorithm outputs the hypothesis $H(x) = \operatorname{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$, where α_t is the weight and h_t is the weak hypothesis computed in the tth iteration.

It remains to answer three important questions: (1) What is T, (2) What is M, (3) Why is H a strong hypothesis? The punchline of AdaBoost is the following: by selecting the number of iterations $T = O(\log M)$, the hypothesis H satisfies $H(x) = c(x)$ for every $x \in S$. However, note that this does not imply H is a strong hypothesis. Using a standard Hoeffding bound, Freund and Schapire showed that with high probability (taken over the samples in S), suppose the number of labelled examples M is at least $O(\mathsf{VC}(\mathcal{C}))$ (where $\mathsf{VC}(\mathcal{C})$ is a combinatorial dimension that can be associated with $\mathcal{C}$), then H not only perfectly classifies every $x \in S$, but it also satisfies $\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq 2/3$. Hence H is a strong hypothesis for the target concept c under the unknown distribution $\mathcal{D}$.

Theorem 2.1 *(Schapire & Freund, 2012) Fix $\eta, \gamma > 0$. Let $n \geq 1$ and $\mathcal{C} \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$ be a concept class. Let $\mathcal{D} : \{0,1\}^n \to [0,1]$ be an unknown distribution. Let $\mathcal{A}$ be a weak PAC algorithm that takes time $R(\mathcal{C})$ to learn $\mathcal{C}$ with bias γ. Let M be the smallest integer exceeding $M \geq \frac{\mathsf{VC}(\mathcal{C})}{\gamma^2} \cdot \frac{\log(\mathsf{VC}(\mathcal{C})/\gamma^2)}{\eta^2}$. Suppose we run AdaBoost for $T \geq ((\log M)\cdot\log(1/\delta))/(2\gamma^2)$ rounds, then with probability $\geq 1-\delta$ (over the randomness of the algorithm), we obtain a hypothesis H that has zero training error and small generalization error $\Pr_{x\sim\mathcal{D}}[H(x) \neq c(x)] \leq \eta$. Moreover the time complexity of the classical AdaBoost algorithm is $\widetilde{O}\left(\frac{\mathsf{VC}(\mathcal{C})}{\eta^2} \cdot R(\mathcal{C}) \cdot \frac{n}{\gamma^4} \cdot \log(1/\delta)\right)$.*[2]

3. Our results

The main contribution of this paper is a quantum algorithm that runs in time quadratically faster in $O(\mathsf{VC}(\mathcal{C}))$ to obtain

[1]This distribution update is referred to as the *Multiplicative Weights Update Method* (MMUW). See (Arora et al., 2012) on how one can cast AdaBoost into the standard MMUW framework.

[2]Here, $\widetilde{O}(\cdot)$ hides poly-logarithmic factors in the parenthesis.

a strong learner for the concept class $\mathcal{C}$.

Theorem 3.1 (Informal) *Let* $n \geq 1$ *and* $\mathcal{C} \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$. *Let* $\mathcal{A}$ *be a weak quantum PAC learning algorithm that takes time* $Q(\mathcal{C})$ *and has bias* γ. *Then the quantum complexity of converting* $\mathcal{A}$ *to a strong PAC learning algorithm is* $\widetilde{O}\left(\sqrt{\mathsf{VC}(\mathcal{C})} \cdot Q(\mathcal{C})^{3/2} \cdot \frac{n^2}{\gamma^{11}}\right)$.

We now make a few remarks regarding our main theorem:

1. Comparing our bound with classical AdaBoost complexity, we get a quadratic improvement in terms of $\mathsf{VC}(\mathcal{C})$. Also observe that the time complexity of quantum PAC learning $Q(\mathcal{C})$ could be polynomially or even exponentially smaller than classical PAC learning time complexity $R(\mathcal{C})$.[3]

2. Although our dependence on $1/\gamma$ is worse than the classical complexity, we believe our complexity should be improvable using quantum techniques (and we leave it as an open question to improve the exponent of the factor $1/\gamma$). We remark that although the complexity of our quantum boosting algorithm is weaker than the classical complexity in terms of $1/\gamma = \mathrm{poly}(n)$, observe that many concept classes have VC dimension that scales *exponentially* with n, in which case our quadratic improvement in terms of $\mathsf{VC}(\mathcal{C})$ "beats" the "polynomial loss" (in terms of $1/\gamma$) in the complexity of our quantum boosting algorithm.

3. There have been a few prior works (Neven et al., 2012; Schuld & Petruccione, 2018; Wang et al., 2019) which touch upon AdaBoost but none of them rigorously prove that quantum techniques can improve boosting. As far as we are aware, ours is the *first work* that proves quantum algorithms can quadratically improve the complexity of classical AdaBoost. Given the importance of AdaBoost in classical machine learning, our quadratic quantum improvement could potentially have various applications in QML.[4]

3.1. Why quantum does not "trivially" give a quantum speedup to AdaBoost?

We now give a sketch of our quantum boosting algorithm. The quantum algorithm follows the structure of the classical AdaBoost algorithm. On a very high level, our quantum speedup is obtained by using quantum techniques to estimate the quantity $\varepsilon_t = \sum_{x \in S} D_x^t \cdot [h_t(x) \neq c(x)]$ quadratically faster than classical methods. In order to do so, one could use quantum algorithms for mean estimation, which given a set of numbers $\alpha_1, \ldots, \alpha_M \in [0,1]$, produces an approximation of $\frac{1}{M}\sum_{i \in [M]} \alpha_i$ in time $\Theta(\sqrt{M})$ (Nayak & Wu, 1999; Brassard et al., 2011),[5] whereas classical methods would use time $\Theta(M)$. However, using the mean estimation subroutine to improve the time complexity of classical AdaBoost comes with various issues which we discuss now:

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

$$D_x^{t+1} = \frac{D_x^t}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha_t} & \text{otherwise ,} \end{cases} \tag{2}$$

   where $Z_t = \sum_{x \in S} D_x^t \exp(-c(x)\alpha_t h_t(x))$ and $\alpha_t = \frac{1}{2}\ln((1-\varepsilon_t)/\varepsilon_t)$. Given an additive approximation ε_t' of ε_t, first note that the approximate weights $\alpha_t' = \frac{1}{2}\ln((1-\varepsilon_t')/\varepsilon_t')$ could be very far from α_t. Moreover, it is not clear why the *updated distribution* $\widetilde{D}^{t+1}$ defined as $\widetilde{D}_x^{t+1} = \frac{1}{Z_t} \cdot D_x^t \exp(\alpha_t' c(x) \cdot h_t(x))$ is *even close* to a distribution. Another possible way to update our distribution would be $\widetilde{D}_x^{t+1} = \frac{1}{Z_t'} \cdot D_x^t \exp(-\alpha_t' c(x) \cdot h_t(x))$, where $Z_t' = \sum_{x \in S} D_x^t \exp(-c(x)\alpha_t' h_t(x))$, so by definition $\widetilde{D}^{t+1}$ is a distribution. However, in this case note that a quantum learner cannot compute Z_t' exactly in time $o(M)$ but instead can only *approximate* Z_t' and we face the same issue as mentioned above.[6]

2. **Strong approximation of ε_t:** One possible way to get around this would be to estimate ε_t *very well* so that one could potentially show that $\widetilde{D}^{t+1}$ is *close* to a distribution. However, observe that if $\widetilde{D}^{t+1}$ should be close to a distribution, then we require a $\delta = 1/\sqrt{M}$-approximation of ε_t and such a strong approximation increases the complexity from $O(\sqrt{M})$ to $O(M)$ which removes the entire quantum speedup.

3. **Noisy inputs to a quantum learner:** Let us further assume that we could spend time $O(M)$ as mentioned above to estimate ε_t very well (instead of using classical techniques to compute ε_t). Suppose we obtain $\widetilde{D}^{t+1}$ which is close to a distribution. Recall that the input to a quantum learner should be copies of

[3] In (Arunachalam & Wolf, 2018), the authors prove that the *sample* complexity of classical and quantum PAC learning is the same up to constant factors, but there does exist concept classes demonstrated by (Servedio & Gortler, 2004) for which there could be exponential separations in *time* complexity between quantum and classical learning.

[4] Although our quantum boosting algorithm uses quantum phase estimation as a subroutine, which isn't implementable in near-term quantum computers, we leave it as an open question if one could use variational techniques as proposed by Peruzzo et al. (Peruzzo et al., 2014) in place of phase estimation.

[5] We omit poly-logarithmic factors in the complexity.

[6] Note that computing Z_t' would take time $O(M)$ classically even though we have knowledge of α_t', since Z_t' involves a summation of M terms.

a quantum state $|\psi_t\rangle = \sum_{x\in S}\sqrt{D_x^t}|x, c(x)\rangle$. However, we only have access to a quantum state $|\phi_t\rangle = \sum_{x\in S}\sqrt{\widetilde{D}_x^t}|x, c(x)\rangle + |\chi_t\rangle$, where $|\chi_t\rangle$ is orthogonal to the first part of $|\phi_t\rangle$ (note that $|\phi_t\rangle$ is not a quantum state without the additional register $|\chi_t\rangle$ since $\sum_{x\in S}\widetilde{D}_x^t \leq 1$). Now it is unclear what will be the output of a quantum learner on input $|\phi_t\rangle$ instead of $|\psi_t\rangle$.

4. **Why is the final hypothesis good:** Assume for now that we are able to show that the learner, on input copies of $|\phi_t\rangle$, produces a weak hypothesis h_t for the target concept c. Then, after T steps of the quantum boosting algorithm, the final hypothesis is $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha'_t h_t(x)\right)$. It is not at all clear why H should satisfy $H(x) = c(x)$ for even a constant fraction of the xs in S. Observe that the analysis of classical AdaBoost crucially used that $H(x) = c(x)$ for most $x \in S$ in order to conclude that the generalization error is small, i.e., $\Pr_{x\sim\mathcal{D}}[H(x) \neq c(x)] \leq 1/3$ where $\mathcal{D}$ is an unknown distribution over $\{0,1\}^n$.

3.2. Quantum algorithm for boosting

In this paper, our main contribution is a *quantum boosting algorithm* that overcomes all the issues mentioned above.

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

We now discuss the important modification: the distribution update step. As mentioned before, classically one can compute the quantity $\varepsilon = \Pr_{x\sim D}[h(x) \neq c(x)]$ in time $O(M)$. Quantumly, we describe a subroutine that for a fixed δ, performs the following: outputs 'yes' if $\varepsilon \geq \Omega((1-\delta)/(QT^2))$ and 'no' otherwise. In the 'yes' instance, the algorithm also outputs an approximation ε' that satisfies $|\varepsilon' - \varepsilon| \leq \delta\varepsilon'$ and in the 'no' instance, the algorithm outputs an ε' that satisfies $|\varepsilon' - \varepsilon| \leq 1/(QT^2)$. The essential point here is the subroutine takes time $O(\sqrt{M}/\delta)$.[7] The subroutine crucially uses the fact that in the 'yes' instance, the complexity of the standard quantum mean estimation algorithm (that given $\alpha_1, \ldots, \alpha_M$, outputs a δ-approximation of $\frac{1}{M}\sum_{i\in[M]}\alpha_i$) scales as $O(\sqrt{M}/\delta)$. However, in the 'no' instance, obtaining a good multiplicative approximation of ε using the quantum mean estimation algorithm could potentially take time $O(M)$. In this case, we *do not* need a good approximation of ε and instead we simply set $\varepsilon' = \tau = 1/QT^2$. We will justify this shortly.

Depending on whether we are in the 'yes' instance or 'no' instance of the subroutine, we update the distribution differently. In the 'yes' instance, we make a distribution update that resembles the standard AdaBoost update using the approximation ε'_t instead of ε_t. We let $Z_t = 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)}$, $\alpha'_t = \ln\left(\sqrt{(1-\varepsilon'_t)/\varepsilon'_t}\right)$ and update $\widetilde{D}_x^t$ as follows

$$\widetilde{D}_x^{t+1} = \frac{\widetilde{D}_x^t}{(1+2\delta)\,Z_t} \times \begin{cases} e^{-\alpha'_t} & \text{if } h_t(x) = c(x) \\ e^{\alpha'_t} & \text{otherwise .}\end{cases} \tag{3}$$

However, in the 'no' instance, we cannot hope to get a good multiplicative approximation in time $O(\sqrt{M})$. In this case, we crucially observe that weaker approximations of the weights $\alpha'_t = \ln\left(\sqrt{(1-\varepsilon'_t)/\varepsilon'_t}\right)$ is sufficient in the hypothesis $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha'_t h_t(x)\right)$, i.e., obtaining a worse approximation of ε_t still allows us to show that H has small training error. Hence, in the 'no' instance, we simply let $\varepsilon'_t = \tau$, $Z_t = 2\sqrt{\tau(1-\tau)}$ and $\alpha'_t = \ln\left(\sqrt{(1-\tau)/\tau}\right)$ and update $\widetilde{D}_x^t$ as follows:

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

Note that the distribution update above is *not* the standard boosting distribution update and differs from it by assigning higher weights to the correctly classified training examples and lower weights to the misclassified ones. In both cases of the distribution update, observe that $\widetilde{D}$ need not be a distribution. However we are able to show that $\widetilde{D}$ is very close to a distribution, i.e., with some technical work we can argue that $\sum_{x\in S}\widetilde{D}_x \in [1-30\delta, 1]$. This aspect is very crucial because, in every iteration of the quantum boosting algorithm, we will pass copies of $|\Phi'\rangle = \sum_{x\in S}\sqrt{\widetilde{D}_x}|x, c(x)\rangle + |\chi\rangle$,[8] to the quantum learner instead of the ideal quantum state $|\Phi\rangle = \sum_{x\in S}\sqrt{D_x}|x, c(x)\rangle$ (since our algorithm starts with many copies of $\sum_x \sqrt{\widetilde{D}_x}|x, c(x)\rangle$ our algorithm also has access to these copies of $|\Phi'\rangle$). A priori it is not clear, what will be the output of the weak quantum learner on the input $|\Phi'\rangle$. However, we show that the state $|\Phi'\rangle$ is close to $|\Phi\rangle$, in particular we show that $|\langle\Phi'|\Phi\rangle| \geq 1-\delta$. Suppose a weak quantum learner outputs a weak hypothesis h when given copies of the state $|\Phi\rangle$ (with probability at least $1-1/T$), we show that the same quantum learner will output h when given copies of the state $|\Phi'\rangle$, with probability at least $1-9/T$. By applying a union bound over T rounds of quantum boosting, we can bound the probability of obtaining a good hypothesis. Finally, after T

[7] The 'yes' and 'no' events of this subroutine happen with high probability, we omit this for simplicity in exposition.

[8] Again note that we need $|\chi\rangle$ because $\widetilde{D}$ is not a distribution, and $\sum_x \sqrt{\widetilde{D}_x}|x, c(x)\rangle$ is not a valid quantum state.

rounds, our quantum boosting algorithm outputs the hypothesis $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha'_t h_t(x)\right)$ for all $x \in \{0,1\}^n$.

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

Application to classical AdaBoost. We remark that our main technical contribution, i.e., the *modified distribution update rule* is also applicable to classical Adaboost. In particular, suppose in classical AdaBoost we obtain approximations ε'_t instead of the *exact* weighted errors ε_t in time P. Then our *robust classical Adaboost* algorithm (i.e., AdaBoost with modified distribution update) can still produce a hypothesis H that has zero training error and the complexity of such a robust classical AdaBoost algorithm will be proportional to $O(P)$. Clearly, it is possible that P could be *much smaller* than M (which is the time taken by classical AdaBoost to compute ε_t exactly) in which case the robust classical AdaBoost algorithm can be faster than standard classical AdaBoost. As far as we are aware, ours is the first work that considers *approximating* the weighted errors ε (which could potentially be much faster than exactly computing εs) and shows that changing the distribution update in AdaBoost still allows to produce a strong hypothesis.

4. Preliminaries

Quantum information. In this paper, we assume familiarity with the following quantum information notation. Let $|0\rangle = \binom{1}{0}$ and $|1\rangle = \binom{0}{1}$ be the basis for $\mathbb{C}^2$, the space in which single qubits live. An arbitrary single qubit state is a *superposition* of $|0\rangle, |1\rangle$ and has the form $\alpha|0\rangle + \beta|1\rangle = \binom{\alpha}{\beta}$ where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Multi-qubit quantum states can be simply obtained by taking tensor products of single-qubit quantum states. Overall an arbitrary n-qubit quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ can be written as $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$ where $\alpha_x \in \mathbb{C}$ and $\sum_x |\alpha_x|^2 = 1$. A valid *quantum operation* on quantum states can be expressed as a *unitary matrix* U (which satisfies $UU^* = U^*U = \mathbb{I}$). An application of a unitary U to the state $|\psi\rangle$ results in the quantum state $U|\psi\rangle$.

Quantum oracle access. We say $\mathcal{A}$ is given *query* access to $c : \{0,1\}^n \to \{-1,1\}$ if, $\mathcal{A}$ can *query* c, i.e., $\mathcal{A}$ can obtain $c(x)$ for x of it's choice. Similarly, we say $\mathcal{A}$ has *quantum query* access to c, if $\mathcal{A}$ can query c in a superposition, i.e., $\mathcal{A}$ can perform the map $O_c : |x,b\rangle \to |x, c(x)\cdot b\rangle$ for every $x \in \{0,1\}^n$ and $b \in \{-1,1\}$. The *query complexity* of a quantum algorithm will be in terms of how many *quantum queries* are made throughout the quantum algorithm and the *time complexity* of a quantum algorithm will refer to the total number of gates involved in the quantum algorithm (i.e., the number of gates it takes to implement various unitaries during the quantum algorithm) as well as the number of gates it takes to prepare quantum states.

Quantum subroutines. In this paper we will use two quantum subroutines. The first quantum algorithm by (Brassard et al., 2011) estimates the mean of numbers quadratically faster on a quantum computer than classical algorithms for mean estimation.

Theorem 4.1 (Mean Estimation) *Given a black-box for the function $F : \{1,\dots,N\} \to [0,1]$, there exists a quantum algorithm that with probability at least $2/3$ computes an additive ε-approximation of $\frac{1}{N}\sum_{i=1}^N F(i)$ using $O(1/\varepsilon)$ evaluations of F.*

Observe that classically estimating the mean $\frac{1}{N}\sum_{i=1}^N F(i)$ up to additive precision ε would take $\Theta(1/\varepsilon^2)$ many evaluations of F and Theorem 4.1 gives a quadratic speedup compared to classical algorithm in estimating the mean. The second subroutine which we will use is *amplitude amplification* by (Brassard et al., 2002), a well-known quantum subroutine which performs the following task: suppose we have a (classical) algorithm $\mathcal{A}$ that outputs 1 with probability p and 0 otherwise, then classically we need to repeat $\mathcal{A}$ $\Theta(1/p)$ many times before one of the repetitions of $\mathcal{A}$ outputs a 1. Quantumly, amplitude amplification is a procedure that invokes $\mathcal{A}$ and the inverse of $\mathcal{A}$ (denoted $\mathcal{A}^{-1}$) $O(1/\sqrt{p})$ many times before outputting 1 with high probability, hence providing a quadratic quantum speedup over classical randomized algorithms.

Theorem 4.2 (Amplitude amplification) *Suppose there exists a unitary U on n qubits that satisfies the following $U|0^n\rangle = \sqrt{a}|\psi_0\rangle + \sqrt{1-a}|\psi_1\rangle$ for an unknown $a > 0$ and arbitrary orthogonal quantum states $|\psi_0\rangle, |\psi_1\rangle$. Then there exists a quantum algorithm that outputs $|\psi_0\rangle$ with probability exactly $a' > 0$ using an expected number $\Theta(\sqrt{a'/a})$ of applications of U, U^{-1}.*

PAC learning. The Probably Approximately Correct (PAC) model of learning was introduced by (Valiant, 1984). A *concept class* $\mathcal{C}$ is a collection of Boolean functions $c : \{0,1\}^n \to \{-1,1\}$, which are often referred to as *concepts*. In the PAC model, there is an unknown distribution $\mathcal{D} : \{0,1\}^n \to [0,1]$ under which a learner needs to learn $\mathcal{C}$, i.e., a learner $\mathcal{A}$ is given *labelled examples* $(x, c(x))$ where $x \sim \mathcal{D}$ and $c \in \mathcal{C}$ is an *unknown* target concept (which the learner is trying to learn). The goal of $\mathcal{A}$ is to output a hypothesis $h : \{0,1\}^n \to \{-1,1\}$ and we say that $\mathcal{A}$ is an (η,δ)-*PAC learner* for a concept class $\mathcal{C}$ if it satisfies: for all $c \in \mathcal{C}$ and distributions $\mathcal{D}$, given access to labelled examples $(x, c(x))$, with probability $\geq 1-\delta$, $\mathcal{A}$ outputs a hypothesis h such that $\Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)] \leq \eta$.

The sample complexity and time complexity of a learner is the number of labelled examples and number of bit-wise operations (i.e., time taken) that suffices to learn $\mathcal{C}$ (under the hardest concept $c \in \mathcal{C}$ and distribution $\mathcal{D}$). In the quantum PAC model, a learner is a *quantum algorithm* given access to the quantum examples $\sum_x \sqrt{\mathcal{D}_x}|x, c(x)\rangle$ and a quantum computer. The remaining aspects of the quantum PAC learning algorithm is defined analogous to the classical PAC model. We now define what it means for an algorithm $\mathcal{A}$ to be a strong and weak learner for a concept class $\mathcal{C}$.

Definition 4.3 (Weak and strong learner) *Let* $n \geq 1$ *and* $\mathcal{C} \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$. *We say* $\mathcal{A}$ *is a* weak (resp. strong) learner *for* $\mathcal{C}$ *if it satisfies: for every* $c \in \mathcal{C}$ *and distribution* $\mathcal{D} : \{0,1\}^n \to [0,1]$, *given query access to c,* $\mathcal{A}$ *can output a hypothesis* h *such that* $\Pr_{x\sim\mathcal{D}}[h(x) = c(x)] \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$. *(resp.* $\Pr_{x\sim\mathcal{D}}[h(x) = c(x)] \geq \frac{2}{3}$.*)*

Throughout, we will assume that we have classical or quantum *query* access to the hypothesis h, and will not assume explicit truth table description of h. Similarly, we say h is a *weak-hypothesis* (resp. *strong-hypothesis*) if $\big|\Pr_x[h(x) = c(x)] - 1/2\big| \geq 1/\mathrm{poly}(n)$ (resp. $\geq 1/6$). We now define two misclassification errors. Let $S = \{(x_1, y_1), \ldots, (x_M, y_M)\}$ where $(x_i, y_i) \in \{0,1\}^n \times \{-1,1\}$ is drawn from a joint distribution $\mathcal{D} : \{0,1\}^n \times \{-1,1\} \to [0,1]$. The *training error* of h is defined as the error of h on the *training set* S and given by $\frac{1}{M}\sum_{i=1}^{M}[h(x_i) \neq y_i]$. In order to quantify the *goodness* of the hypothesis h, the *true* error or the *generalization error* of h is defined as $\Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$.

5. Quantum Boosting

In this section, we use *quantum* techniques to improve the complexity of AdaBoost. Like in AdaBoost, we break our quantum boosting algorithm into two stages. Stage (1), reduce training error: produce a hypothesis that does well on the training set and Stage (2), reduce generalization error: we show that for a sufficiently large training set, not only does the hypothesis output in Stage (1) has a small training error, but also has a small generalization error.

5.1. Quantum boosting: reducing training error

The bulk of the technical work in our quantum boosting algorithm lies in reducing the training error. We now state the main theorem for Stage (1) of our quantum algorithm.

Theorem 5.1 *Let* $\gamma > 0$, $n \geq 1$ *and* $\mathcal{C} \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$ *be a concept class and* $\mathcal{D} : \{0,1\}^n \to [0,1]$ *be an unknown distribution. Let* $\mathcal{A}$ *be a quantum algorithm that takes time* $Q(\mathcal{C})$ *to PAC learn* $\mathcal{C}$ *with bias* γ. *Let* M *be sufficiently large*[9] *and* $T = O((\log M)/\gamma^2)$. *Given a training set* $S = \{(x_i, c(x_i))\}_{i\in[M]}$ *where* $x_i \sim \mathcal{D}$ *and* $c \in \mathcal{C}$, *the quantum boosting algorithm takes time* $\widetilde{O}(\sqrt{M} \cdot n^2 \cdot Q(\mathcal{C})^{3/2} T^5)$, *and with probability* $\geq 2/3$, *outputs a hypothesis* H *that satisfies* $H(x_i) = c(x_i)$ *for all* $i \in S$.

We describe the quantum boosting algorithm in this theorem statement now. Since the sample complexity of $\mathcal{A}$ is at most the time complexity, we will assume that it suffices to provide $\mathcal{A}$ with Q quantum examples. Our quantum algorithm is a T-round iterative algorithm similar to classical AdaBoost and in each round, our quantum algorithm produces a distribution $\widetilde{D}$. In the tth round our quantum algorithm follows a three step process:

1. Invoke the weak quantum learner $\mathcal{A}$ to produce a weak hypothesis h_t under an *approximate* distribution $\widetilde{D}^t$ over the training set S.
2. By making quantum queries to h_t, our algorithm computes ε'_t, an approximation to $\widetilde{\varepsilon}_t = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$. We then use ε'_t to update the distribution $\widetilde{D}^t$ to $\widetilde{D}^{t+1}$. In this step, we depart from standard AdaBoost.
3. Using ε'_t compute a *weight* α'_t. After T steps, output a hypothesis $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T}\alpha'_t h_t(x)\right)$.

Before describing our quantum algorithm, we state a lemma which we use in performing step (2) in the procedure above.

Lemma 5.2 *Let* $\delta = 1/(10QT^2)$. *there exists a procedure that outputs* ε' *and satisfies the following: with probability* $\geq 1 - 10\delta/T$, *if the output is* $\{\varepsilon', \mathit{yes}\}$, *then* $|\widetilde{\varepsilon} - \varepsilon'| \leq \delta\varepsilon'$; *and if the output is* $\{\varepsilon' = 1/(QT^2), \mathit{no}\}$, *then* $|\widetilde{\varepsilon} - \varepsilon'| \leq 1/(QT^2)$. *The time complexity of the procedure is* $\widetilde{O}(\sqrt{M}Q^{3/2}T^3n^2)$.

We now describe our quantum boosting algorithm.

[9] We quantify what we mean by *sufficiently large* in the next section, in particular in Theorem 5.5.

Algorithm 1 Quantum boosting algorithm

Input: Quantum weak learner $\mathcal{A}$ with time complexity Q, a training sample $S = \{(x_i, c(x_i))\}_{i\in[M]}$, where $x_i \sim \mathcal{D}$ and $\mathcal{D} : \{0,1\}^n \to [0,1]$ is an unknown distribution.

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

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

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

Phase (1): Obtaining hypothesis h_t

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

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

4: Apply amplitude amplification (in Theorem 4.2) to prepare $|\Phi_6\rangle = \sum_{x\in S}\sqrt{\widetilde{D}^t_x}|x, c(x)\rangle + |\chi_t\rangle$.

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

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

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

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

8: Invoke Lemma 5.2 to estimate $\widetilde{\varepsilon}_t$ with ε'_t.

Phase (3): Updating distributions

9: **If** lemma 5.2 outputs 'yes': let $Z_t = 2\sqrt{\varepsilon'_t(1-\varepsilon'_t)}$, $\alpha'_t = \frac{1}{2}\ln\Big((1-\varepsilon'_t)/\varepsilon'_t\Big)$. Update

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

10: **If** lemma 5.2 outputs 'no': let $Z_t = \Big(2\sqrt{QT^2-1}\Big)/(QT^2)$, $\alpha'_t = \ln\Big(\sqrt{QT^2-1}\Big)$,

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

11: **end for**

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

We do not discuss the steps of our quantum boosting algorithm due to space constraints and give more details in Section B of the supplementary material. Now we make a couple of remarks. First, note that we use the notation $\widetilde{D}^t_x$ in the quantum boosting algorithm because $\{\widetilde{D}^t_x\}_x$ is not a distribution (which is also why we need to use the state $|\chi_t\rangle$ in step (4) because $\sum_{x\in S}\sqrt{\widetilde{D}^t_x}|x, c(x)\rangle$ is not a valid quantum state), instead we show that $\widetilde{D}^t$ is "close" to a distribution in the following sense

Claim 5.3 *Let $t \geq 1$, $\widetilde{D}^t : \{0,1\}^n \to [0,1]$ be defined as in steps $(9), (10)$. Then $\sum_{x\in S}\widetilde{D}^t_x \in [1-30\delta, 1]$.*

We also show that, not only are these distributions are close but the weighted training error of the hypotheses under these "distributions" are close.

Claim 5.4 *Let $t \geq 1$, $\widetilde{\varepsilon}_t = \Pr_{x\sim\widetilde{D}^t}[h_t(x) \neq c(x)]$ be the weighted error corresponding to the approximate distribution $\widetilde{D}^t$ and $\varepsilon_t = \Pr_{x\sim D^t}[h_t(x) \neq c(x)]$ correspond to the true distribution D^t. Then $|\widetilde{\varepsilon}_t - \varepsilon_t| \leq 50\delta$.*

We prove these claims along with a few more facts (which are used in showing that the quantum algorithm produces a strong H) in Section C of the supplementary material.

Working with approximate distributions is an important difference between standard AdaBoost and our quantum boosting algorithm. In AdaBoost, one assumes that the $\widetilde{\varepsilon}$s can be computed exactly by spending time $O(M)$, however quantumly, we can only approximate $\widetilde{\varepsilon}$ with ε' using the quantum mean estimation algorithm (in Theorem 4.1) in time $O(\sqrt{M})$. Hence, using ε' in Phase (3) results in a sub-normalized distribution in the quantum algorithm.

Second, as we mentioned in the introduction we differ from AdaBoost crucially in phase (3). The quantum mean estimation algorithm gives a good approximation in time $O(\sqrt{M})$ *only* when $\widetilde{\varepsilon}_t$ is "large", in which case we use the standard AdaBoost distribution step in Step (9). In case $\widetilde{\varepsilon}_t$ is "small", since we cannot hope to get a good approximation of $\widetilde{\varepsilon}_t$ in time $O(\sqrt{M})$, we *fix* $\varepsilon'_t = 1/QT^2$ and use a different distribution update compared to AdaBoost in Step (10). The intuition as to why fixing $\varepsilon'_t = 1/QT^2$ is sufficient is that, when $\widetilde{\varepsilon}_t$ is "small", we observe that obtaining worse approximations of $\alpha'_t = \frac{1}{2}\ln\big(\frac{1-\widetilde{\varepsilon}_t}{\widetilde{\varepsilon}_t}\big)$ are sufficient in the final output hypothesis $H = \text{sign}\Big(\sum_{t=1}^T \alpha'_t h_t\Big)$. In particular, with weaker approximations of α'_t, we show that using the α'_t obtained by fixing $\varepsilon'_t = 1/QT^2$ (whenever $\widetilde{\varepsilon}_t$ is small) is *sufficient* to show that the final hypothesis H is strong hypothesis. We make this rigorous in the supplementary material in Section D. We remark that Sections C, D of the

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

supplementary material (which prove the correctness of the quantum boosting algorithm) are mathematically technical and are the non-trivial aspects of our quantum algorithm.

5.1.1. Proof of correctness

We first bound the probability of failure of the quantum boosting algorithm in obtaining the strong hypothesis H. The first source of error is due to amplitude amplification in step (4) of the boosting algorithm, which fails with probability $\leq \frac{1}{3T}$. The second error is due to the quantum weak learner failing to output a weak hypothesis in step (5), whose probability is $\leq \frac{1}{3T}$. The third source of error is in estimating $\widetilde{\varepsilon}_t$ in step (8), the probability of failure in estimating $\widetilde{\varepsilon}_t$ is $\leq O(1/(QT^3))$. Applying a union bound over the T rounds and all the failure events, we ensure that the overall probability of not outputting H can be made an arbitrary constant (with a constant overhead in the complexity).

It remains to argue that the training error of H is 0, i.e., $H(x) = c(x)$ for every $(x, c(x)) \in S$. To analyze this we crucially use the structure of the modified distribution update step in Phase (3). Proving that the final H has zero training error departs from standard AdaBoost convergence analysis and due to space constraints we defer it to Section D of the supplementary material.[11]

5.1.2. Complexity of the algorithm

Our quantum algorithm begins with the state $\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle$ given access to $S = \{x_i, c(x_i)\}_i$. Assuming that a quantum RAM can prepare a *uniform* superposition $\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle$ using $O(n\log M)$ gates, the time complexity of preparing the initial state $|\psi_1\rangle \otimes |\Phi_1\rangle^{\otimes Q}$ is $O(nQ\log M)$. We could also assume that a quantum learning algorithm is *given* uniform quantum examples $\frac{1}{\sqrt{M}}\sum_{x\in S}|x, c(x)\rangle$, in which case we do not need to assume a quantum RAM.[12]

[11] For a more coherent exposition of the theorems alongside proofs, we refer the reader to (Arunachalam & Maity, 2020).

[12] Given the QRAM assumption has been controversial and seems strong in quantum machine learning, we make a couple of remarks: (i) our quantum boosting algorithm *only* requires a QRAM to prepare the *uniform* superposition over classical data S at the start of each iteration. Also, our quantum algorithm does not use QRAM as an oracle for Grover-like algorithms, so the negative results of (Arunachalam et al., 2015) do not apply to our algorithm; (ii) we use the QRAM at the start of $T = O(\log M)$ iteration of our algorithm to prepare $|\psi_0\rangle$, so even if the quantum time complexity of preparing $|\psi_0\rangle$ is $O(\sqrt{M})$, then our complexity increases by an *additive* $O(\sqrt{M}\log M)$ term and we still do not lose our quantum speedup; (iii) of course if QRAM is infeasible then we can also assume that a quantum learner has access to uniform quantum examples or has *quantum query access* to the training examples in S (i.e., can perform the map $|x, b\rangle \to |x, b\cdot c(x)\rangle$ for $x \in S$). and in both cases we do not need a QRAM.

In phase (1) of the quantum algorithm, we first update the distribution registers from $\widetilde{D}^1$ to $\widetilde{D}^t$. This step involves using $O(Qt)$ quantum queries to $\{h_1, \ldots, h_{t-1}\}$ which can be performed in time $O(Qt)$, and other arithmetic operations that can be performed in time $O(n^2Qt)$. We then perform amplitude amplification (in Theorem 4.2) to prepare $|\Phi_6\rangle$ which takes time $O(n^2\sqrt{M}Qt)$ (in Section E in the supplementary material we make explicit what are the unitaries for which we are applying amplification.) Finally, we pass Q copies of $|\Phi_6\rangle$ to the weak learner $\mathcal{A}$ which outputs a hypothesis h_t in time Q. Note that we require the quantum learning algorithm to output an oracle for h_t instead of explicitly outputting a circuit for h_t. In phase (2), the algorithm in steps $(6), (7)$ performs a query as well as a quantum gate for phase rotation in order to prepare $|\psi_6\rangle$ using $O(n)$ gates. The next step is the mean estimation step (in Theorem 4.1) to compute ε_t', which takes time $O(\sqrt{M}Q^{3/2}T^3 \cdot tn^2)$ (again in Section (E) we explicitly mention the unitaries to which we apply Theorem 4.1). Finally, Phase (3) of our quantum boosting algorithm involves basic arithmatic operations which takes time $O(n^2)$. The total complexity of the algorithm scales as $\widetilde{O}(\sqrt{M}\cdot \mathrm{poly}(n, Q, T))$.

5.2. Reducing generalization error

In the previous section we showed that our quantum boosting algorithm produces a hypothesis H that *perfectly* classifies the training set $S = \{(x_i, c(x_i))\}_{i\in[M]}$, where $(x_i, c(x_i))$ was sampled according to the unknown $\mathcal{D}$. Recall that the goal of our quantum boosting algorithm is to output a hypothesis $H : \{0,1\}^n \to \{-1,1\}$ that satisfies $\Pr_{x\sim\mathcal{D}}[H(x) = c(x)] \geq 1-\eta$. We saw in Theorem 2.1 that as long as M, i.e., the number of training examples is *large enough*, then not only does H has *zero* training error, but it also ensures small *generalization error*. In particular, in Stage (2) of classical AdaBoost we simply use Theorem 2.1 to argue that: suppose the training error of H is 0, then the *generalization error* of H is at most η as long as $M \geq O(\mathsf{VC}(\mathcal{C})/\eta^2)$. Using Theorem 2.1, we now prove our main theorem:

Theorem 5.5 (Complexity of Quantum Boosting) *Fix $\eta > 0, \gamma > 0$. Let $n \geq 1$ and $\mathcal{C} \subseteq \{c : \{0,1\}^n \to \{-1,1\}\}$ be a concept class and $\mathcal{D} : \{0,1\}^n \to [0,1]$ be an unknown distribution. Let $\mathcal{A}$ be a weak PAC quantum algorithm that has bias γ and takes time $Q(\mathcal{C})$. Suppose M satisfies*

$$M \geq \frac{\mathsf{VC}(\mathcal{C})}{\gamma^2} \cdot \frac{\log(\mathsf{VC}(\mathcal{C})/\gamma^2)}{\eta^2}.$$

Suppose we run Algorithm 1, then with probability $\geq 1-\delta$ (over the randomness of the algorithm), we obtain a hypothesis H that has training error at most $1/10$ and generalization error $\Pr_{x\sim\mathcal{D}}[H(x) \neq c(x)] \leq \eta + 1/10$. Moreover, *the time complexity of the quantum boosting algorithm is*

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

Picking $\eta = 1/10$ we get that H has generalization error at most $1/5$. Recall that the complexity of classical AdaBoost is $T_C = \widetilde{O}\left(\frac{\mathsf{VC}(\mathcal{C})}{\eta^2} \cdot R(\mathcal{C}) \cdot \frac{n}{\gamma^4} \log(1/\delta)\right)$. In comparison, T_Q is quadratically better than T_C in terms of the VC dimension of the concept class $\mathcal{C}$ and $1/\eta$. Additionally, we could potentially have $Q(\mathcal{C}) \ll R(\mathcal{C})$ since the the quantum time complexity of a weak learner can be much lesser than the classical time complexity of learning as shown by (Servedio & Gortler, 2004) (under complexity-theoretic assumptions).

Open questions. We conclude with a few interesting questions: (i) can we improve the polynomial dependence on $1/\gamma$ in the quantum complexity of boosting? (ii) can we use the quantum boosting algorithm to improve the complexities various quantum algorithms that use *classical* AdaBoost on top of a weak quantum algorithms? (iii) are there *practically relevant* concept classes which have large VC dimension for which our quantum boosting algorithm gives a large quantum speedup, (iv) could one replace the quantum phase estimation step in our quantum boosting algorithm by variational techniques developed by (Peruzzo et al., 2014)?

Subsequent work. After our work was submitted to ICML 2020, (Hamoudi et al., 2020) posted a paper on arXiv proposing a quantum speedup for the Hedge algorithm by Freund and Schapire, which can be viewed as a boosting algorithm using multiplicative weights method.

Acknowledgements. RM would like to thank Seth Lloyd for his support and guidance during the course of the project. RM's research was supported by Seth Lloyd in the Research Laboratory of Electronics at MIT. Most of this work was done when SA was a Postdoc at Center for Theoretical Physics, MIT (funded by the MIT-IBM Watson AI Lab under the project *Machine Learning in Hilbert space*) and visiting Henry Yuen in University of Toronto. SA was also supported by the Army Research Laboratory and the Army Research Office under grant number W1975117. We thank Ashley Montanaro and Ronald de Wolf for clarifications regarding multiplicative phase estimation and Ronald de Wolf for several comments that significantly improved the presentation of this paper. We thank Steve Hanneke and Nishant Mehta for various clarifications regarding classical boosting algorithms. We thank the anonymous reviewers of ICML 2020 for helpful suggestions on a prelimnary version of this paper.

References

Aaronson, S. The learnability of quantum states. *Proceedings of the Royal Society of London*, 463(2088), 2007. quant-ph/0608142.

Aaronson, S. Shadow tomography of quantum states. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pp. 325–338, 2018.

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

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

Apeldoorn, J. v., Gilyén, A., Gribling, S., and Wolf, R. d. Convex optimization using quantum oracles. *Quantum*, 4:220, 2020. arXiv:1809.00643.

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

Arunachalam, S. and Maity, R. Quantum boosting. *arXiv:2002.05056*, 2020.

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

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

Arunachalam, S., Chakraborty, S., Lee, T., Paraashar, M., and de Wolf, R. Two new results about quantum exact learning. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, pp. 16:1–16:15, 2019. arXiv:1810.00481.

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

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

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

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

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

Chakrabarti, S., Childs, A. M., Li, T., and Wu, X. Quantum algorithms and lower bounds for convex optimization, 2018. arXiv:1809.01731.

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

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

Freund, Y. and Schapire, R. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14: 771–780, 1999.

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

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

Hamoudi, Y., M.Ray, Rebentrost, P., Santha, M., Wang, X., and Yang, S. Quantum algorithms for hedging and the sparsitron. *arXiv:2002.06003*, 2020.

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

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

Jethwani, D., Gall, F. L., and Singh, S. Quantum-inspired classical algorithms for singular value transformation. 2019. arXiv:1910.05699.

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

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

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

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

Nayak, A. and Wu, F. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pp. 384–393, 1999.

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

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

Prakash, A. Quantum algorithms for linear algebra and machine learning, 2014. PhD Thesis.

Rebentrost, P., Mohseni, M., and Lloyd, S. Quantum support vector machine for big feature and big data classification, 2013. arXiv:1307.0471.

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

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

Schapire, R. and Freund, Y. *Boosting: Foundations and Algorithms*. MIT Press, 2012.

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

Schuld, M. and Petruccione, F. Quantum ensembles of quantum classifiers. *Scientific reports*, 8(1):2772, 2018.

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

Valiant, L. G. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Wang, X., Ma, Y., Hsieh, M., and Yung, M. Quantum speedup in adaptive boosting of binary classification. 2019. arXiv: 1902.00869.

Yoganathan, M. A condition under which classical simulability implies efficient state learnability, 2019. arXiv:1907.08163.