Quantum boosting using domain-partitioning hypotheses

Sagnik Chatterjee1 · Rohan Bhatia2 · Parmeet Singh Chani2 · Debajyoti Bera1

Received: 27 February 2023 / Accepted: 23 June 2023 © The Author(s), under exclusive licence to Springer Nature Switzerland AG 2023

Abstract
Boosting is an ensemble learning method that converts a weak learner into a strong learner in the PAC learning framework. The AdaBoost algorithm is a well-known classical boosting algorithm for weak learners with binary hypotheses. Recently, Arunachalam and Maity presented the first quantum boosting algorithm by quantizing AdaBoost. Their algorithm, which we refer to as QAdaBoost hereafter, is a quantum adaptation of AdaBoost and only works for the binary hypothesis case. QAdaBoost is quadratically faster than AdaBoost in terms of the VC dimension of the hypothesis class of the weak learner. However, QAdaBoost is polynomially worse in the bias of the weak learner. In this work, we address an open question by Izdebski et al. on the existence of boosting algorithms for quantum weak learners with non-binary hypotheses. We answer this question in the affirmative by developing the QRealBoost algorithm motivated by the classical RealBoost algorithm. The main technical challenge was to provide provable guarantees for convergence, generalization bounds, and quantum speedup, given that quantum subroutines are noisy and probabilistic. We prove that QRealBoost retains the quadratic speedup of QAdaBoost over AdaBoost and further achieves a polynomial speedup over QAdaBoost in terms of both the bias of the learner and the time taken by the learner to learn the target concept class. Finally, we perform empirical evaluations on QRealBoost and report encouraging observations on quantum simulators by benchmarking the convergence performance of QRealBoost against QAdaBoost, AdaBoost, RealBoost, and SmoothBoost on a subset of the MNIST dataset and Breast Cancer Wisconsin dataset.

Keywords Quantum computing · Boosting

1 Introduction
The last decade has seen substantial growth in quantum machine learning, giving rise to several quantum machine learning algorithms that promise and provide improvements over their classical counterparts. Several survey papers and books have already been published, and interested readers may consult any of those to obtain an overview of the algorithmic and theoretical advances in quantum machine learning (Arunachalam and de wolf, 2017; Biamonte et al., 2017; Schuld et al., 2015; Wittek, 2014).

Discriminative learning algorithms aim to “learn” an unknown concept in a manner that can help them classify samples. Some learning algorithms. Some learning algorithms are accurate with arbitrarily high accuracy (these are referred to as strong learners), while others perform slightly better than random guessing (referred to as weak learners). Even though very accurate learners are ultimately desired, it might not always be wise to use highly accurate learners for many reasons, such as longer running times, overfitting, and lack of model explainability. On the other hand, many well-known simple learning algorithms are easy to create and are essentially weak learners. These include decision stumps, naïve Bayes over a single variable, clustering algorithms with a fixed number of clusters, etc.
Ensemble learning is a method of converting a “weak” learner to a “strong” learner. In this paper, we focus on a particular type of ensemble learning approach known as **Boosting**, in which the weak learner is trained iteratively over reweighted distributions of a fixed training set. By tweaking the distribution of the training set, we ensure that in each iteration, the learner gives more weight to the misclassified samples in the previous iteration, eventually reducing both bias and variance of the learner (Friedman et al., 2000). Boosting algorithms are now included in standard machine learning libraries, e.g., scikit-learn (Pedregosa et al., 2011).

Arunachalam and Maity (2020) recently quantized the Gödel prize winning AdaBoost algorithm (Freund and Schapire, 1997) with provable speedup guarantees. Following the results of Arunachalam and Maity (2020), Izdebski and de wolf (2020) posed an open question on the existence of quantum boosting algorithms that could boost weak learners with non-binary hypotheses, and simultaneously retain the above provable theoretical speedups. In this work, we answer this question in the affirmative by giving a quantum boosting algorithm that works for hypotheses that output discrete class partitions with similar theoretical guarantees to Arunachalam and Maity’s quantum algorithm. The main contributions of this paper are:

- Designing the first quantum boosting algorithm (QRealBoost) for boosting weak quantum learners with non-binary ranges.
- Analyzing QRealBoost to demonstrate provable guarantees for speedup over existing quantum and classical adaptive boosting algorithms.
- Empirical Verification of the convergence and generalization behavior of QRealBoost and benchmarking its performance against other boosting algorithms.

The rest of this paper is organized as follows: Section 2 summarizes various related works, including classical and state-of-the-art quantum boosting algorithms. Section 3 reviews the theory on various topics such as boosting algorithms, learning theory, and quantum learning theory. The main results and technical contributions of this work are detailed in Section 4. In Section 5, we detail the various parts of the QRealBoost algorithm and give the complexity analysis and the convergence proofs of QRealBoost. We benchmark the performance of QRealBoost against other boosting algorithms and discuss the results in Section 6. Finally, we present some concluding remarks and suggest a few directions for future work in Section 7.

---

1 We define the terms weak and strong formally in Section 3.1.
2 See Section 3.5 for more details.

---

2 Related work

AdaBoost (see Algorithm 1) is one of the first adaptive boosting algorithms proposed by Freund and Schapire (1997). AdaBoost provably converges with zero training error and requires no prior knowledge about the accuracy of the hypotheses generated by the weak learner it is trying to boost. It has also been observed that AdaBoost does not tend to overfit (Bauer and Kohavi, 1999; Drucker and Cortes, 1996) on various problems and performs much better (Dietterich, 2000; Opitz and Machin, 1999) than other ensemble methods in the absence of classification noise. We discuss the original AdaBoost algorithm in Section 3.4.

Early works on quantum boosting algorithms (a) consider quantum algorithms for AdaBoost and its variants in the experimental setting (Neven et al., 2012), (b) use classical AdaBoost as a subroutine (Schuld and Petruccione, 2018), or (c) only consider speedups for a particular aspect of AdaBoost, e.g., speeding up the margin computation step of AdaBoost (Wang et al., 2020). Arunachalam and Maity recently adapted the original AdaBoost algorithm into QAdaBoost (Arunachalam and Maity, 2020) and provided rigorous mathematical guarantees of speed up over the classical version in its query and time complexity in terms of the weak learner’s sample complexity by using approximate counting of quantum states.

Subsequently, Izdebski and de wolf (2020) proposed a quantum variant of Servedio’s classical SmoothBoost algorithm (Servedio, 2003) (which we refer to as QSmoothBoost) which retains the speedup in sample complexity while also achieving a polynomial speedup in the bias of the weak learner as compared to the QAdaBoost algorithm. Despite QSmoothBoost’s impressive complexity speedups, it has a few shortcomings. Similar to its classical counterpart, QSmoothBoost is not adaptive. Additionally, SmoothBoost (hence, QSmoothBoost) takes longer to converge than AdaBoost (hence, QAdaBoost) and does not converge to zero training error. We chose to stick to the AdaBoost framework to circumvent these issues.

The classical RealBoost algorithm (Friedman et al., 2000; Schapire and Singer, 1999) (see Algorithm 3) tackles the problem of boosting weak learners whose hypotheses essentially divide the domain \( \mathcal{X} \) into a small number of mutually exclusive and exhaustive sets of partitions (hence the name “domain-partitioning hypotheses”). We can alternatively characterize domain-partitioning learners as learning algorithms that output real-valued predictions (essentially class probabilities), unlike binary-valued predictions as seen in

---

3 For a detailed discussion on “adaptiveness” see Bshouty and Gavinsky (2002)
4 We describe a generalized framework Algorithm 3.4 for quantum boosting in Section 3.4.
AdaBoost (and QAdaBoost). RealBoost not only retains the zero training error and the generalization behavior of the original AdaBoost family of boosting algorithms but also has been observed in practice to converge much faster than AdaBoost with respect to the empirical error (Ruan et al., 2010; Wang and Wang, 2009; Wu et al., 2004).

In Table 1, we compare the query complexities of our quantum boosting algorithm with the other boosting algorithms discussed above. We also observe via experiments that QRealBoost converges faster than QAdaBoost during training, and has similar generalization performance. See Section 6 for details.

3 Preliminaries

3.1 PAC learning

Let \( \mathcal{X} \) be a domain of \( n \)-bit sized instances. A concept class \( \mathcal{C} \) is a family of concepts and each concept \( c \in \mathcal{C} \) is a set of \( n \)-bit Boolean functions, one for each \( n \). The training set consists of \( M \) labeled examples \( \mathcal{S} = \{x_i, y_i = c(x_i)\}_{i \in [M]} \) where each \( x_i \in \mathcal{X} \) is drawn from an unknown distribution \( \mathcal{D} \) and labeled according to an unknown concept \( c \in \mathcal{C} \). Our objective is to produce a hypothesis \( h \) which is a good approximation of the labeling function \( c \) over all points in \( \mathcal{X} \) sampled according to distribution \( \mathcal{D} \).

We denote a learning algorithm \( A \) as an \( (\eta, \delta) \)-PAC learner for the concept class \( \mathcal{C} \), if it efficiently outputs a hypothesis \( h \) such that with probability at least \( 1 - \delta \) (over its internal randomness) \( \Pr_{x \sim \mathcal{D}} [h(x) \neq c(x)] \leq \eta \). Here efficiently means that the learner runs in time polynomial in \( M, 1/\eta, 1/\delta \), and \( n \).

A \( \gamma \)-weak learner \( A \) is defined as a \((\frac{1}{2} - \gamma, \delta)\)-PAC learner, where \( \gamma = O(1/\eta^k) \) for \( k \geq 1 \) and a \textbf{strong learner} is defined as a \((\frac{1}{2}, \delta)\)-PAC learner, using \( \delta \leq 1/3 \) in both cases. We focus on distribution free PAC learning since our algorithm does not depend on the distribution \( \mathcal{D} \) according to which our examples are sampled.

In order to quantify how accurate our hypothesis is, we define two types of misclassification errors: \textbf{training error} \( \text{err}(h, \mathcal{S}) = \Pr_{(x,y) \sim \mathcal{S}} [h(x) \neq y] \) which is defined with respect to the training set, and \textbf{generalization error} \( \text{err}(h, \mathcal{D}) = \Pr_{(x,y) \sim \mathcal{D}} [h(x) \neq y] \) which is defined with respect to arbitrary samples.

3.2 Quantum PAC learning

Our algorithm is designed in the quantum PAC learning framework, introduced by Bshouty and Jackson (1998). The definition of weak learning and strong learning generalizes straightforwardly to quantum PAC learners, and all classical PAC learnable function classes are learnable in the quantum setting. The sample complexity of quantum and classical PAC-learners are equal up to constant factors (Arunachalam and De Wolf, 2018). A quantum PAC learner \( A \) has access to several copies of the quantum example state defined below. \( A \) performs a POVM measurement at the end to obtain a hypothesis \( h \) belonging to its associated hypothesis class \( \mathcal{H}_A \).

In the classical PAC setting a learner \( A \) with sample complexity \( Q \) can query \( Q \) samples from the training set \( \mathcal{S} = \{x_i, y_i = c(x_i)\}_{i \in [M]} \). However, in the quantum setting (Arunachalam and Maity, 2020; Izdebski and De Wolf, 2020), we assume that the quantum learner is provided with the state \( \frac{1}{\sqrt{M}} \sum_{x \in \mathcal{S}} |x, y_i = c(x_i)\rangle \otimes |\mathcal{Q} \rangle \). We note here that to simulate a classical learner, a quantum learner can measure \( d \), and takes time \( R \) (classical case), or \( Q \) (quantum case) to produce a hypothesis \( h \in \mathcal{H}_A \) for both RealBoost and QRealBoost we assume that the weak PAC learner outputs hypotheses that make a constant number of domain partitions.

| Algorithm                          | Model       | Adaptive | Range   | Query Complexity            |
|-----------------------------------|-------------|----------|---------|----------------------------|
| AdaBoost (Freund and Schapire 1997) | Classical   | Yes      | Binary  | \( O(d \cdot R \cdot \gamma^{-4}) \) |
| RealBoost (Schapire and Singer 1999; Friedman et al. 2000) | Classical   | Yes      | Non-Binary | \( O(d \cdot R \cdot \gamma^{-4}) \) |
| SmoothBoost (Servedio 2003)       | Classical   | No       | Binary  | \( \tilde{O}(d \cdot \gamma^{-4} + R \cdot \gamma^{-2}) \) |
| QAdaBoost (Arunachalam and Maity 2020) | Quantum    | Yes      | Binary  | \( \tilde{O}(\sqrt{d} \cdot Q^{1.5} \cdot \gamma^{-11}) \) |
| QSmoothBoost (Izdebski and de Wolf 2020) | Quantum    | No       | Binary  | \( \tilde{O}(\sqrt{d} \cdot \gamma^{-5} + Q \cdot \gamma^{-4}) \) |
| QRealBoost                        | Quantum     | Yes      | Non-Binary | \( \tilde{O}(\sqrt{d} \cdot Q \cdot \gamma^{-9}) \) |
this state to obtain $Q$ examples chosen uniformly at random (with replacement) from $S$. This state can be efficiently prepared with or without the assumption of a quantum random access memory (aka. QRAM). To use a QRAM to prepare a uniform superposition over the classical samples, we only incur an additive $O(\sqrt{M \log M})$ term in the query complexity, which retains our quantum speedup. We also note that the QRAM (if used) is only for state preparation. For a detailedity, which retains our quantum speedup. We also note that the QRAM, we refer the reader to Izdebski and de wolf (2020), in which the authors only assume quantum query access to the training samples. As in the earlier works on quantum PAC learning (Arunachalam and de wolf, 2017; Bshouty and Jackson, 1998), we also assume the ability to create and query a uniform superposition over the classical samples, we only a "strong" learner that has small empirical and generalization errors with respect to $C$, using just oracleal access to $A$.

In Schapire (1990), Schapire showed us that in the PAC model, the task of producing strong learners from weak learners is not only possible but that the two notions of weak and strong learning are inherently equivalent.

**Lemma 1** (Equivalence of Weak and Strong learning; Schapire 1990) An unknown concept class $C = \bigcup_{n \geq 1} C_n$ is efficiently weakly PAC learnable if and only if $C$ is efficiently strongly PAC learnable.

Subsequently, researchers started proposing boosting algorithms (Freund 1995; Schapire 1990) (among other types of ensemble learning algorithms) to achieve this task. These algorithms are known as boosting algorithms since they somehow "boost" the weak learner to produce a strong learner. These early efforts ultimately culminated in the formation of the adaptive boosting or AdaBoost algorithm (Freund and Schapire, 1997) (described in Algorithm 1). A small point to note here is that the definitions of weak and strong learning generalize to the quantum setting simply by considering that the learner is quantum.

**3.4 AdaBoost and some generalizations**

The AdaBoost algorithm takes two inputs — a classical $\gamma$-weak learner $A$, and $M$ training samples. At the $t$th step, we define a new distribution $D_t$ over the training samples based on $D_{t-1}$. We then use $A$ to produce a new binary-valued hypothesis with respect to $D_t$. We compute the weighted error $\epsilon_t$ and the confidence of the hypothesis $\alpha_t$ for the $t$th step.

**Algorithm 1 The AdaBoost Algorithm**

1. **Input:** Classical weak learner $A$, and $M$ Training Samples $(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)$; $x_i \sim \mathcal{X}, y_i \in \{-1,+1\}$
2. **Initialize:** Set $D_1^t = \frac{1}{M} \forall i \in [M]$.
3. for $t = 1$ to $T$ do
4. Train $A$ using the distribution $D_t$ to obtain the hypothesis $h_t: \mathcal{X} \rightarrow \{-1,+1\}$.
5. Compute the weighted error $\epsilon_t$ and the margin $\alpha_t$ for this iteration as follows
6. $\epsilon_t = \sum_{i \in [M]} D_t^i [h_t(x_i) \neq y_i]$; $\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$ (1)
7. Perform the distribution update $\forall i \in [1, \ldots, M]$ as follows
8. $D_{t+1}^i = \frac{D_i^t \exp (-\alpha_t y_i h_t(x_i))}{Z_t}$ (2)
9. Output: Hypothesis $H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x_i) \right)$ (3)

Using these quantities, we define the distribution update rule Eq. (2) for the next iteration. After at least $T \geq O(\log M / \gamma^2)$ iterations we produce the hypothesis $H$ as given in Eq. (3). Freund and Schapire showed that $H$ has zero training error and very small generalization error given the number of training samples is sufficiently large.

In AdaBoost for binary classification, we had access to $M$ training samples in $S$, which were distributed according to some unknown distribution $D$ over the domain space $\mathcal{X}$. Given $S$ as input, our weak learner $A$ output a hypothesis $h: \mathcal{X} \rightarrow \{-1,+1\}$. Consider the generalized version of the AdaBoost algorithmic framework in which the weak learner outputs real-valued hypotheses $h': \mathcal{X} \rightarrow \mathbb{R}$. Here sign$[h'(x)]$ is our required prediction, and the quantity $|h'(x)|$ gives us the "confidence" for the prediction. Alternatively, the quantity $|h'(x)|$ tells us how confident our learner is while making the prediction sign$[h'(x)]$. This is analogous to the formal notion of margins which is well-known in learning theory. Larger margins on training data directly lead us to better bounds on the generalization error (Schapire and Freund, 2012). In fact, boosting algorithms in the AdaBoost framework tend to keep becoming more confident with their predictions leading to a drop in generalization error even after training error converges to zero.

In Schapire and Singer (1999) it was shown that the generalized model with real-valued hypotheses still adheres to the bound given in Lemma 2 if $h': \mathcal{X} \rightarrow [-1, +1]$. In fact, so far both models of weak learners use Lemma 2 to focus on weak learners such that their hypotheses focus on greedily minimizing the normalization constant $Z_t$ (refer to Algorithm 1).
at each iteration in order to bound the training error. We can therefore consider folding the margin and the hypothesis into one quantity in order to simplify the calculation of $Z_t$ as in the case of the generalized AdaBoost where $-y_i h'_t(x_i)$ can replace the term $-\alpha_t y_i h'_t(x_i)$. In Section 3.5 we explore a different flavor of weak learners introduced in Friedman et al. (2000); Schapire and Singer (1999) that also focuses on this particular simplified criteria. We can immediately see that even though these generalizations retain the general structure of AdaBoost (Algorithm 1), there are a few key steps where they differ. We outline these steps in Algorithm 3.4, and then use this framework to introduce the RealBoost algorithm in Algorithm 3.

Algorithm 2 The AdaBoost framework

1: **Input:** Weak learner $A$ with access to $M$ Training samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$
2: **Initialize:** Set $D^1_i = \frac{1}{M}$ for $i \in [M]$
3: for $t = 1$ to $T$ do
4: Train $A$ using the distribution $D^t$ to obtain the hypothesis $h_t$.
5: Compute the weighted misclassification error $Z_t$.
6: Update the distribution to $D^{t+1}$ using the computed errors and confidences.
7: end for
8: **Output:** Hypothesis $H(x)$ which combines the individual hypotheses $h_t$, $\forall t \geq 1$ according to their computed confidences.

3.5 Boosting using domain partitioning hypotheses

A domain-partitioning hypothesis $h$ partitions the input domain $\mathcal{X}$ into a set of mutually exclusive and exhaustive blocks, such that the hypothesis $h$ predicts the same label for all instances belonging to a given partition $\mathcal{X}_j$. Formally, $h : \mathcal{X} \rightarrow \mathbb{R}$ induces a partitioning of $\mathcal{X}$ into $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_C\}$, where $\bigcup_{j=1}^C \mathcal{X}_j = \mathcal{X}$ and $h(x) = h(x')$, $\forall x, x' \in \mathcal{X}_j$, $j \in \{1, \ldots, C\}$. Since the prediction is constant for all instances assigned to a specific partition, we denote the prediction $h$ for the partition $\mathcal{X}_j$ by the constant $\beta_j \in \mathbb{R}$. Note that $\text{sign}(\beta_j)$ gives us the prediction for the partition $\mathcal{X}_j$ while $|\beta_j|$ gives us the confidence of the prediction. Now, the task at hand reduces to finding good values of $\beta_j$ for each $\mathcal{X}_j$.

We give an example to foster an intuitive way of thinking about the confidence of predictions in this particular context. Suppose $\beta_j$ is calculated by taking the log of the ratio of the weighted fraction of examples with different labels. Consider a partition that contains 100 samples with label $-1$ and 5 samples with label $+1$. Then the weighted prediction for that particular partition will be $-1.3$, which means that we predict all samples in this partition to have a $-1$ label with a confidence rating of 1.3. Another partition which contains 55 samples with label $+1$ and 45 samples with label $-1$, will have a weighted prediction of 0.08. Here we see that because the majority has a $+1$ label, we assign it to the entire partition, but we do so with much lower confidence than in the previous case. This shows us that if there is almost an equal number of samples of both labels in a particular domain, then the confidence for predicting either class will be quite low.

Algorithm 3 The Classical RealBoost Algorithm

1: **Input:** Classical weak learner $A$, and $M$ training samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$; $x_i \sim \mathcal{X}$, $y_i \in \{-1, +1\}$. $\triangleright$Here $A$ outputs domain-partitioning hypotheses unlike AdaBoost.
2: **Initialize:** Set $D^1_i = \frac{1}{M}$ for $i \in [M]$.
3: for $t = 1$ to $T$ do
4: Train $A$ using the distribution $D^t$ to obtain a partitioning $\mathcal{X}' = \{\mathcal{X}'_1, \ldots, \mathcal{X}'_C\}$ of $\mathcal{X}$.
5: for $k = 1$ to $C$ do
6: for $b \in \{-1, +1\}$ do
7: Calculate the partition-label weight $W^t_{b, k}$ as
\begin{equation}
W^t_{b, k} = \sum_{i:x_i \in \mathcal{X}'_k \cap y_i = b} D^t_i
\end{equation}
8: end for
9: end for
10: Set $Z_t = 2 \sum_{j=1}^C \sqrt{W^t_{-1, j} W^t_{+1, j}}$. $\triangleright$A clever reformulation of the normalization constant using partition-label weights.
11: Compute the margins $\beta_{j,t} = \frac{1}{2} \ln \left( \frac{W^t_{+1, j}}{W^t_{-1, j}} \right)$. $\forall j \in \{1, \ldots, C\}$.
12: Perform the distribution update $\forall i \in \{1, \ldots, M\}$ as follows
\begin{equation}
D^{t+1}_i = \frac{D^t_i \exp (-\beta_{j,t} \cdot y_i)}{Z_t}
\end{equation}
13: end for
14: **Output:** Hypothesis $H(\cdot)$ which is defined as
\begin{equation}
H(x) = \text{sign} \left( \sum_{j=1}^T \beta_{j,t} \right) \text{ where } j \text{ indicates the domain partition } \mathcal{X}'_j \in \mathcal{X}' \text{ containing } x.
\end{equation}

We now introduce the RealBoost algorithm (Schapire and Singer, 1999; Friedman et al., 2000) in Algorithm 3. In this setup, the weak learner outputs discrete class partitions (or domain partitions). Here, unlike the binary case, we need to calculate the training error with respect to every domain partition. Since the weak learner simply outputs class partitions and not labels, Schapire and Singer (1999) assumed that the majority label of a given partition is its true label. This allows us to calculate the piecewise training error w.r.t. individual
3.6 Error bounds and sample complexity

In this section, we state some well-known results (without proof) and definitions related to training and generalization error bounds as well as sample complexity of the algorithms that follow the AdaBoost framework, for example, RealBoost (Algorithm 3).

**Lemma 2** (Upper bound on training error; Schapire and Freund 2012) Let $A$ be a $\gamma$-weak learner. Let $\gamma_t = \frac{1}{2} - \epsilon_t$, where $\epsilon_t$ is misclassification error at every iteration of Algorithm 3. Let $D_t$ be an arbitrary initial distribution over the training set. Then the weighted training error of the combined classifier $H$ with respect to $D_1$ is bounded as

$$\text{err}(H) \leq \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)$$

If we look at the first inequality in Lemma 2, we observe that the AdaBoost framework minimizes the training error of the combined hypothesis by greedily minimizing the normalization constant in Algorithm 3 at every step. This produces the following corollary that we shall use later.

**Corollary 1** Let $A$ be a $\gamma$-weak learner and $D_1$ be the uniform distribution over the training set of $M$ examples. Then the training error of the combined classifier $H$ with respect to $D_1$ goes to 0 when $T > \frac{\ln M}{2\gamma^2}$, where $T$ is the number of iterations of our boosting algorithm.

We see that greedily minimizing the training error at every step leads to the algorithm converging exponentially fast in terms of training samples. The next result gives us bounds on the generalization error in the AdaBoost framework.

**Lemma 3** (Generalization error bound; Schapire and Freund 2012) Let us have a $\gamma$-weak learner $A$ for a concept class $C$ which produces classifiers $h$ from a space $\mathcal{H}$ which has finite VC-dimension $d \geq 1$. If we run Algorithm 3 for $T$ rounds on $M$ random samples (sampled from an unknown distribution $D : \{0, 1\}^n \to \{0, 1\}$ which are associated with a concept class $C = \bigcup_{n \geq 1} C_n$ such that $m \geq \max\{d, T\}$, then with high probability (at least $2/3$), the final hypothesis $H : \{0, 1\}^n \to \{-1, +1\}$ satisfies

$$\text{err}(H) \leq \text{err}(H) + \tilde{O}\left(\sqrt{\frac{td}{M}}\right)$$

From Lemma 3, we get the following corollary which helps us lower bound the total number of training samples to obtain a low generalization bound for Algorithm 3.

**Corollary 2** (Sample Complexity for reducing generalization error of AdaBoost) Let us have a $\gamma$-weak learner $A$ which produces classifiers $h$ from a space $\mathcal{H}$ which has finite VC-dimension $d \geq 1$. We sample $M$ random samples from an unknown distribution $D : \{0, 1\}^n \to \{0, 1\}$ which are associated with a concept class $C = \bigcup_{n \geq 1} C_n$ such that $m \geq \max\{d, T\}$. If we run Algorithm 3 for $T$ rounds where $T \geq \tilde{O}(\log M/\gamma^2)$, then with high probability we get a generalization error of at most $\eta > 0$ when

$$M \geq \tilde{O}\left(\frac{d}{\eta^2} \cdot \frac{1}{\gamma^2}\right)$$

3.7 Fidelity and trace distance

**Definition 1** (Fidelity) Fidelity is a measure of the closeness of two quantum states. When we have two pure states $|\psi\rangle$ and $|\phi\rangle$ we define Fidelity between the two states as

$$F(\psi, \phi) = F(\phi, \psi) = |\langle \psi | \phi \rangle|^2$$

Let $\rho$ and $\sigma$ be the density matrices of $\psi$ and $\phi$ respectively. An alternate characterization of Fidelity in terms of density matrices is

$$F(\rho, \sigma) = \|\sqrt{\rho} \sqrt{\sigma}\|_1$$

**Definition 2** (Normalized trace distance) Trace distance is another measure of closeness between two quantum states. If there is a set of POVMs $\{E\}$, then the POVM leading to the largest difference in measurement outcomes between two quantum states is the trace distance.

$$D(\rho, \sigma) = \frac{1}{2} \|\rho \circ \sigma - \sigma\|_1 = \max_{E_i} \sum_i |\text{Tr}[E_i(\rho - \sigma)]|$$

When $\rho$ and $\sigma$ are density matrices of pure states, Trace distance is related to Fidelity as follows:

$$D(\rho, \sigma) = \sqrt{1 - F(\rho, \sigma)}$$

3.8 Quantum subroutines for amplitude amplification and estimation

**Lemma 4** (Amplitude Amplification; Brassard et al. 2002) Let there be a unitary $U$ such that $U|0\rangle = \sqrt{a}|\phi_0\rangle + \sqrt{1-a}|\phi_1\rangle$ for an unknown $a > 0$. If $a > p > 0$, then there exists a quantum amplitude amplification algorithm that outputs the state $|\phi_0\rangle$ with a probability $p'$ $> 0$. The expected
number of calls to $U$ and $U^{-1}$ made by our quantum amplitude amplification algorithm is $\Theta(\sqrt{p}/p)$.

**Lemma 5** (Relative Error Estimation; Ambainis 2008) Given an error parameter $\epsilon$, a constant $k \geq 1$, and a unitary $U$ such that $U|0 \rangle$ outputs $1$ with probability at least $p$. Then there exists a quantum amplitude estimation algorithm that produces an estimate $\hat{a}$ of the success probability $a$ with probability at least $1 - \frac{1}{2k}$ such that $|a - \hat{a}| \leq \epsilon a$ where $a \geq p$. The expected number of calls to $U$ and $U^{-1}$ made by our quantum amplitude estimation algorithm is

$$O \left( \frac{k}{\epsilon \sqrt{p}} \left( 1 + \log \log \frac{1}{p} \right) \right)$$ (13)

**4 Main results**

QRealBoost (as given in Algorithm 4) is a quantum adaptation of the RealBoost algorithm with domain-partitioning learners. In this work, we focus on weak learners that output discrete class partitions rather than class probabilities since this is a more natural model for decision tree algorithms and clustering algorithms. Domain-partitioning hypotheses allow us to calculate the confidences of prediction for each partition\(^5\) which leads to improved estimation of margins, ultimately producing better bounds of generalization error.

QRealBoost maintains the general flavor of RealBoost (Algorithm 3) as discussed in Section 3.5 but implements several steps using quantum algorithms, which lead to a quadratic speedup over RealBoost. The caveat is that the intermediate quantum subroutines involving quantum amplification and estimation are erroneous and therefore require careful analysis to prove that boosting converges, and the convergence is exponentially fast. This is the main technical contribution of this work. QRealBoost is also the first work to our knowledge that performs boosting with non-binary classifiers, which also addresses an open question by Izdebski and de wolf (2020) on whether we can boost tree algorithms with a range other than $\{-1, +1\}$. We now state the main results of our paper.

**Theorem 1** (Complexity of QRealBoost) Let $A$ be a $\gamma$-weak quantum PAC learner with sample complexity $Q$ for a concept class $C$ with VC-dimension $d$. Further, suppose that $A$ produces domain-partitioning hypotheses with a constant number of partitions. There exists a quantum boosting time complexity of boosting $A$ to a strong PAC learner with high probability according to Algorithm 4 is $\tilde{O} \left( \sqrt{d} \cdot Q \cdot \frac{n^2}{\gamma^2} \right)$ and the corresponding query complexity is $\tilde{O} \left( \sqrt{d} \cdot Q \cdot \frac{1}{\gamma^2} \right)$.

**Theorem 2** (Convergence of QRealBoost) Given $m$ training samples, QRealBoost (Algorithm 4) runs for $T \geq \frac{\ln(m)}{2\gamma^2}$ number of iterations, and outputs a hypothesis $H$ that has zero training error and a small generalization error with a high probability.

**4.1 Techniques**

In this section, we go over the main challenges and our techniques to address them. We assume that our weak PAC learner outputs hypotheses that make a constant $C$ number of domain partitions. The central quantity in our algorithm is a partition-label weight, which is the number of samples belonging to a given partition with a particular label. The training error for each partition is defined as a function on these partition-label weights. These training errors allow us to bound the confidence of the hypotheses output by the weak learner. The RealBoost algorithm improves the confidence of prediction in each partition individually from iteration to iteration, and so does our QRealBoost, albeit with slightly inaccurate weights, as we explain below. See Fig. 1 for an overview of QRealBoost as given in Section 5.

We use quantum estimation techniques to calculate the $2C$ partition-label weights $W_{b,j}$ (which are obtained from the distribution weights $\tilde{D}_j$ similar to Eq. (4) for each $j \in \{1, \ldots, C\}$ and $b \in \{-1, +1\}$, and thereby obtain a quadratic speedup compared to classical techniques\(^6\). Earlier quantum boosting algorithms like QAdaBoost (Arunachalam and Maity, 2020) and QSmoothBoost (Izdebski and de wolf, 2020) also employed quantum estimation to obtain speedups over their classical counterparts.

Despite the seemingly straightforward route to speedup, errors resulting from using estimation techniques can and do affect the performance of QRealBoost. It is highly non-trivial to show that boosting occurs despite the resulting errors. We detail several issues which arise from estimating the weights below. Our main technical contribution is showing that QRealBoost deals robust against all these errors.

- **Confidence-rated predictions**: Suppose we naively estimate the partition-label weights $W_{b,j}$\(^{6}\). In that case, there is no way to bound the estimated confidence-rated pre-
Predictions $\beta_j^t$, with respect to the actual confidence-rated predictions $\tilde{\beta}_j^t$.

For that purpose, we bound the latter by carrying out relative error estimation of the partition-label weights see Claim 3. We then perform Laplace correction to deal with corner cases where the values of $W_j^t$ might be extremely small (which may lead to unbounded confidence-rated predictions). Note that the issue of unbounded confidence-rated predictions arises in the underlying classical algorithm itself (Schapire and Freund, 2012; Schapire and Singer, 1999). Laplace correction is well studied in the machine learning literature, especially with respect to decision tree classifiers (Provost and Domingos, 2003), which naturally behave as domain-partitioning learners.

- **Bounding the distribution weights for each iteration:** Errors due to estimation also affect the algorithm when calculating the distribution weights for the next iteration. Unlike the predictions, we can only additively bound the new normalization constant $Z_j^t$ with respect to the ideal constant $Z_j$. Therefore, we cannot immediately guarantee that the updated weights $D_j^{t+1} (for t \geq 1)$ are normalized or even sub-normalized.

In Claim 4, we show that relative error estimation of $W_j^t$ can bound the quantity $Z_j^t$ with relative error. Thus, we define a new distribution update rule Eq. (14). Using the new update rule, we prove Claim 1, which states that the sum of weights in the next iteration is bounded in the range $[1 - 4\varepsilon, 1]$. This is roughly how Algorithm 4 guarantees that the updated weights are close to a distribution.

- **Behavior of intermediate hypotheses:** In an ideal quantum state, the training samples are weighted according to the distribution computed (not estimated) by the classical RealBoost algorithm. We can contrast this with Algorithm 4, where the weights are estimated with relative error. Assume that the quantum weak-learner $A$ outputs a hypothesis $h_t$ with high probability when given the ideal quantum state. We show in Claim 2 that even when a “non-ideal” state is passed by Algorithm 4 to the weak learner, it still outputs the same hypothesis $h_t$ with high probability.

- **Final hypothesis is good:** Even with Claim 2, we still have to prove that our combined classifier $H$ correctly predicts the labels of an arbitrarily high number of training samples since each intermediate classifier is a weak learner. Using Claim 5, we show that our combined classifier has a very small generalization error.

Fig. 1 An overview of QRealBoost
4.2 Empirical validation

Our experiments aim to compare and contrast the convergence and generalization performance of QRealBoost with QAdaBoost, RealBoost, and AdaBoost in practice. Our experiments are carried out on the Breast Cancer Wisconsin and MNIST datasets under varying sample complexity values (4, 6, and 8) and sizes of the training set (16, 32, and 64). For all experiments, the underlying weak learner is the K-means Clustering Algorithm with $k=2$ for QAdaBoost, AdaBoost, and SmoothBoost and $k=3$ for QRealBoost and RealBoost. All experiments are performed on the Qiskit backend provided by Qiskit (Aleksandrowicz et al., 2021). We have made our code available at https://github.com/braqkiku/QRealBoost.

We observe that QRealBoost requires fewer iterations to converge compared to QAdaBoost and AdaBoost in all cases while retaining similar generalization guarantees as QAdaBoost and AdaBoost. For the given experimental parameters, RealBoost generally overfits the data and consequently has the fastest convergence rate and a bad generalization accuracy in most cases, while SmoothBoost is the worst-performing algorithm in terms of convergence and generalization abilities across the board.

5 The QRealBoost algorithm

In this section, we explain the QRealBoost algorithm in detail, which is given in Algorithm 4.

The input to our QRealBoost algorithm consists of the weak learner $A$, and a set of $M$ training samples $S$ as copies of the quantum state $\frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i\rangle$. Since the algorithm is adaptive we can make a worst-case guess for $M$ (number of training samples), $Q$ (sample complexity of $A$), $C$ (number of partitions made by $A$), $γ$ (bias of $A$), and $T$ (number of iterations of Algorithm 4); the algorithm will adapt to the intermediate learners which use more optimistic estimates.

If RealBoost (Algorithm 3) computes the distribution $D^1_i$ in the $i$th iteration, QRealBoost estimates it as $\hat{D}^1_i$ in the $i$th iteration (additional care is taken since the latter may not actually be a probability distribution). Similarly, RealBoost computes the confidence-rated predictions for the $i$th iteration and $j$th partition $\beta^i_{j,t}$, which QRealBoost estimates as $\hat{\beta}^i_{j,t}$. Following the earlier works on quantum boosting algorithms, we too assume that during an iteration in the outermost loop our algorithm, we have quantum query access to the previous hypotheses $h_1, h_2, \ldots, h_{t-1}$ in the form of oracles $O_{h_1}, O_{h_2}, \ldots, O_{h_{t-1}}$, respectively, and the confidence-rated predictions $\beta^i_{j,t}$, for all $j \in \{1, \ldots, C\}$, $t \geq 1$ and $Z^i_t$ (for $t \geq 1$), which are stored in quantum registers.

Algorithm 4 The QRealBoost algorithm.

**Input:** Quantum weak PAC learner $A$ with sample complexity $Q$ which makes at most $C$ partitions in the worst case, and $M$ labeled training samples $\{(x_i, y_i)\}_{i \in [M]}$.

**Initialize:** Worst-case values of $Q$ and $C$, and $\epsilon = O\left(\frac{1}{\sqrt{M^2}}\right), \kappa = \frac{C}{(1-\epsilon)^{\frac{C}{Q}}} \frac{\sqrt{1+\epsilon}}{\sqrt{1-\epsilon}}$. Set $\hat{D}^1_t = \frac{1}{M} \cdot \forall i \in [M]$.

**generate $Q + 2C$ copies of a distribution of training samples**

1: for $t = 1$ to $T$ do 

2: Prepare $|\psi_0\rangle^{\otimes Q} |\psi_2\rangle^{\otimes 2C}$ where $|\psi_0\rangle = |\psi_2\rangle = \frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i\rangle \otimes |D^1_i\rangle$.

3: for $s = 1$ to $t-1$ do

4: On all the $Q + 2C$ copies, apply the transformation $|x_i, y_i\rangle \langle D^s_i | \rightarrow |x_i, y_i\rangle \langle D^{s+1}_i |$ based on the stored values of $\beta^s_{j,t}$ and $Z^s_t$, obtaining the partition $j$ using $O_{h_s}$, and using the distribution update rule

$$\hat{D}^{s+1}_t = \frac{\hat{D}^s_t \cdot \exp \left(-\beta^s_{j,t} y_j\right)}{\kappa Z^s_t}$$

5: end for

6: Perform a conditional rotation on the $|\hat{D}^s_t\rangle$ register to create $|\phi_s\rangle$.

7: Apply amplitude amplification conditioned on the $|1\rangle$ register and then uncompute ancilla to obtain $Q$ copies of $|\phi_s\rangle = \sum_{j \in [C]} \sqrt{D^s_i} |x_i, y_i\rangle + |\phi_s\rangle$.

**obtain the $t$-th hypothesis $h_t$ from the first $Q$ copies**

8: $A(|\phi_s\rangle^{\otimes Q}) \rightarrow$ (followed by a measurement) oracle $O_{h_s}$ corresponding to hypothesis $h_s$.

**obtain confidence-rated predictions using $h_t$ on the last $2C$ copies**

9: On the last $2C$ copies of $|\psi_2\rangle$ use oracles $O_{h_1}, \ldots, O_{h_{t-1}}, O_{h_t}$ to create $2C$ copies of $|\psi_3\rangle = \frac{1}{M} \sum_{i \in [M]} |x_i, y_i, D^1_i, Z^1_t, \ldots, Z^t_t\rangle$.

10: for $k = 1$ to $C$ and $b \in \{-1, +1\}$ do 

11: Take the $(k,b)^{th}$ copy of $|\psi_3\rangle$ and prepare $|\phi_4\rangle = \frac{1}{M} \sum_{i \in [M]} |x_i, y_i, D^{1,\ldots,k}\rangle |D^k_{b,b,t}\rangle$ where $D^k_{b,b,t} = D^k_{j,t}$ if $j_t = k$ and $y_i = b$, and 0 otherwise.

12: Do conditional rotation on the last register to obtain states with amplitudes $\sqrt{W^k_{b,t}}$ where $W^k_{b,t} = \sum_{i \in [M]} D^k_{b,b,t}^2 / M$.

13: Perform amplitude estimation to obtain $\hat{W}^k_{b,t}$ with relative error $\epsilon$ and do Laplace correction on the estimated weights $\hat{W}^k_{b,t}$.

14: end for

15: Compute $Z^t_t = 2 \sum_{j=1}^C \sqrt{W^{j,t}_{b,t}} \hat{W}^{j,t}_{b,t}$, and $\beta^t_{j,t} = \frac{1}{2} \ln \left(\frac{\hat{W}^{j,t}_{b,t}}{\hat{W}^{j',t}_{b,t}}\right) \forall j \in [1, \ldots, C]$.

16: end for

17: **Output:** Hypothesis $H(\cdot)$ defined as

$$H(x) = \sum_{j=1}^C \beta^t_{j,t} x_{j,t}, \text{ where } j_t \in \{1 \ldots C\}$$

is obtained using $|x, 0\rangle \rightarrow O_{h_t} |x, j_t\rangle$ (15)
5.1 Explanation of QRealBoost (Algorithm 4)

5.1.1 Preparing quantum examples for training the Weak Learner

We consider the \(t\)th iteration of the outermost loop. We first initialize \(Q\) copies of \(|\phi_0\rangle\) and 2\(C\) copies of \(|\psi_0\rangle\) both set to

\[
\frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i, \tilde{D}_i^t\rangle
\]

Oracular access to all the previous hypotheses \(\{h_1, \ldots, h_{t-1}\}\) can be expressed as

\[
|x_i, y_i\rangle \rightarrow_{O_{\beta_i}} |x_i, y_i\rangle |j_i^t\rangle
\]

where \(j_i^t = h_t(x_i)\) refers to the domain partition of the \(i\)th sample at the \(t\)th iteration. We query each such oracle in order to produce \(Q + 2C\) copies of

\[
|\phi_1\rangle = |\psi_1\rangle = \frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i, \tilde{D}_i^1|j_i^1, j_i^2, \ldots, j_i^{t-1}\rangle
\]

Next, using the stored class weights \(\beta_{j_i^t, z_i^t}\), and the oracles to the hypotheses, we construct a unitary mapping \(U_D\) for updating the weight register using \(t-1\) applications of Eq. (14) as follows:

\[
|\phi_2\rangle \rightarrow_{U_D} \sum_{i \in [M]} |x_i, y_i, \tilde{D}_i^t|j_i^1, j_i^2, \ldots, j_i^{t-1}\rangle
\]

We perform this update on all \(Q\) copies of \(|\phi_1\rangle\) and 2\(C\) copies of \(|\psi_1\rangle\).

5.1.2 Training \(A\) to obtain a new hypothesis

For all \(Q\) copies of \(|\phi_2\rangle\), we perform a conditional rotation on the register \(|\tilde{D}_i^t\rangle\) to obtain the state

\[
|\phi_3\rangle = \frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i, \tilde{D}_i^t|j_i^1, j_i^2, \ldots, j_i^{t-1}\rangle
\]

\[
\left(\sqrt{\tilde{D}_i^t} |1\rangle + \sqrt{1 - \tilde{D}_i^t} |0\rangle\right)
\]

Let \(U_{0 \rightarrow 3}\) be the unitary that performs \(|0\rangle \rightarrow |\phi_3\rangle\). We perform Amplitude Amplification as stated in Lemma 4 on \(|\phi_3\rangle\) to obtain the state \(|\phi_4\rangle\) (using \(O(\sqrt{M} \log T)\) applications of \(U_{0 \rightarrow 3}\) and \(U_{0 \rightarrow 3}^{-1}\)) with probability at least \(O(1 - 1/T)\). The amplified state \(|\phi_4\rangle\) is as follows

\[
|\phi_4\rangle = \frac{1}{\sqrt{M}} \sum_{i \in [M]} \sqrt{\tilde{D}_i^t} |x_i, y_i, \tilde{D}_i^t|j_i^1, j_i^2, \ldots, j_i^{t-1}\rangle + |\xi_i\rangle
\]

The state \(|\xi_i\rangle\) is present since \(\sum_{i \in [M]} \tilde{D}_i^t \leq 1\) (i.e., the weights are sub-normalized). We state a claim now that shows that the sum of the weights is very close to 1, and hence, very little interference is expected from \(|\xi_i\rangle\).

**Claim 1** For \(\tilde{D}_i^t\) updated as given in Eq. (14) and \(t \in \{1, \ldots, T\}\), we can bound the sum of the sub-normalized weights as

\[
\sum_{i \in [M]} \tilde{D}_i^t \in \left[1 - \frac{4\epsilon}{1 + 2T}, 1\right].
\]

**Proof** We restate the new distribution update rule from Algorithm 4

\[
\tilde{D}_i^{t+1} = \frac{\tilde{D}_i^t \exp \left(-\beta_{j_i^t, y_i}\right)}{\kappa Z_i^t}
\]

Here, we make two observations.

\[
\sum_{i \in [M]} \tilde{D}_i^t \leq 1; \quad \forall t \geq 1
\]

The trivial upperbound in Eq. (19) follows from observing that our weights are sub-normalized as given in \(|\phi_4\rangle\). We know that

\[
W_{b'}^i = \sum_{i : x_i \in X} \tilde{D}_i^t
\]

The observation in Eq. (20) follows immediately.

\[
\frac{\sum_{i \in [M]} \tilde{D}_i^t \exp \left(-\beta_{j_i^t, y_i}\right)}{\sum_{j = 1}^C (W_{+}^{j, t} \exp \left(-\beta_{j_i^t, y_i}\right) + W_{-}^{j, t} \exp \left(\beta_{j_i^t, y_i}\right))} = 1.
\]

Let us start by obtaining a preliminary bound on the quantity \(\sum_i \tilde{D}_i^{t+1}\). We expand the R.H.S. in Eq. (14) as

\[
\sum_{i \in [M]} \tilde{D}_i^t \exp \left(-\beta_{j_i^t, y_i}\right) \sum_{j = 1}^C (W_{+}^{j, t} \exp \left(-\beta_{j_i^t, y_i}\right) + W_{-}^{j, t} \exp \left(\beta_{j_i^t, y_i}\right))
\]

\[
\sum_{j = 1}^C (W_{+}^{j, t} \exp \left(-\beta_{j_i^t, y_i}\right) + W_{-}^{j, t} \exp \left(\beta_{j_i^t, y_i}\right))
\]

\[
\kappa Z_i^t
\]

From Eq. (20), we can set the first term to 1. This gives us the simplified equation

\[
\tilde{D}_i^{t+1} = \frac{\sum_{j = 1}^C (W_{+}^{j, t} \exp \left(-\beta_{j_i^t, y_i}\right) + W_{-}^{j, t} \exp \left(\beta_{j_i^t, y_i}\right))}{\sum_{j = 1}^C \sqrt{W_{+}^{j, t} \cdot W_{-}^{j, t}}}
\]
In the second equality we use the value of $Z_t$ given in Algorithm 4. Now we upperbound and lowerbound the quantity $\sum_{i \in [M]} \tilde{D}_t^{j+1}$ by plugging in (40) as

$$\sum_{i \in [M]} \tilde{D}_t^{j+1} \leq \frac{1}{2\kappa (1-\epsilon)} \sum_{j=1}^{C} W_t^j \exp \left( -\beta_{j,t}^j \right) + W_t^j \exp \left( \beta_{j,t}^j \right)$$

(22)

$$\sum_{i \in [M]} \tilde{D}_t^{j+1} \geq \frac{1}{2\kappa (1+\epsilon)} \sum_{j=1}^{C} W_t^j \exp \left( -\beta_{j,t}^j \right) + W_t^j \exp \left( \beta_{j,t}^j \right)$$

(23)

Substituting $\kappa = \frac{C}{(1-\epsilon)} \sqrt{\frac{1+\epsilon}{1-\epsilon}}$ in Eq. (22) we have

$$\sum_{i \in [M]} \tilde{D}_t^{j+1} \leq \frac{1}{2\kappa (1-\epsilon)} \sum_{j=1}^{C} \frac{W_t^j \exp \left( -\beta_{j,t}^j \right) + W_t^j \exp \left( \beta_{j,t}^j \right)}{\sqrt{W_t^j + W_t^j}}$$

$$= \frac{1}{2\kappa (1-\epsilon)} \sum_{j=1}^{C} \sqrt{\frac{W_t^j + W_t^j}{W_t^j + W_t^j}} - \frac{W_t^j + W_t^j}{W_t^j + W_t^j}$$

$$\leq \frac{1}{2\kappa (1-\epsilon)} \sum_{j=1}^{C} \frac{2(1+\epsilon)}{1-\epsilon}$$

$$= \frac{C}{\kappa (1-\epsilon)} \sqrt{1 + \epsilon} = 1$$

(24)

Similarly, substituting $\kappa = \frac{C}{(1+\epsilon)} \sqrt{\frac{1+\epsilon}{1-\epsilon}}$ in Eq. (23) we have

$$\sum_{i \in [M]} \tilde{D}_t^{j+1} \geq \frac{1}{2\kappa (1+\epsilon)} \sum_{j=1}^{C} \frac{W_t^j \exp \left( -\beta_{j,t}^j \right) + W_t^j \exp \left( \beta_{j,t}^j \right)}{\sqrt{W_t^j + W_t^j}}$$

$$= \frac{1}{2\kappa (1+\epsilon)} \sum_{j=1}^{C} \sqrt{\frac{W_t^j + W_t^j}{W_t^j + W_t^j}} - \frac{W_t^j + W_t^j}{W_t^j + W_t^j}$$

$$\geq \frac{C}{\kappa (1+\epsilon)} \sqrt{1 - \epsilon} = (1 - \frac{2\epsilon}{1+\epsilon})^2 = (1 - \frac{2\epsilon}{1+\epsilon})^2$$

$$\geq 1 - \frac{4\epsilon}{1+\epsilon}$$

(25)

Combining Eq. (24) and Eq. (25) we have for any $t = 1, 2, \ldots, T$

$$\sum_{i \in [M]} \tilde{D}_t^{j} \in \left[ 1 - \frac{4\epsilon}{1+\epsilon}, 1 \right]$$

(26)

Now, apply $U_{D}^{-1}$ and $O_{h_i}^{-1}, \ldots, O_{h_{i-1}}^{-1}$ to $|\phi_4\rangle$ to obtain the state $|\phi_5\rangle = \sum_{i \in [M]} \sqrt{D_t^j} |x_t, y_t \rangle + |\zeta_t^j\rangle$. We pass $Q$ copies of $|\phi_5\rangle$ to the weak learner $A$. In turn, the weak learner produces a hypothesis $h_t$ to which we assume oracular access. The following claim shows that the learned hypothesis is a good hypothesis.

Claim 2 If at the $t$th iteration, the $\gamma$-weak learner $A$ produces a hypothesis $h_t$ on being fed $Q$ copies of the ideal state $|\phi_5^j\rangle = \sum_{i \in [M]} \sqrt{D_t^j} |x_t, y_t \rangle$, then $A$ produces the same hypothesis $h_t$ with high probability when given $Q$ copies of $|\phi_5\rangle$.

Proof Let $p$ be the probability that $A$ outputs the hypothesis $h_t$ on being fed $Q$ copies of the ideal state $|\phi_5^j\rangle$. Let $q$ be the probability that $A$ outputs the hypothesis $h_t$ on being fed $Q$ copies of the state $|\phi_5\rangle$. We want to bound the quantity $|p - q|$ and show that this is a small quantity. Let the density matrices corresponding to $|\phi_5\rangle$ and $|\phi_5^j\rangle$ be $\rho$ and $\sigma$ respectively. We denote the class of P.O.V.M.’s on the hypothesis space $\mathcal{H}$ as $\{E_h\}_{h \in \mathcal{H}}$ such that $\sum_{h \in \mathcal{H}} E_h = I$. Then by Definition 1 and Definition 2 we have

$$|p - q| \leq \max_{\{E_h\}} |\text{Tr}(E_h (\rho - \sigma))| \leq \sum_{h \in \mathcal{H}} \left| \text{Tr}(E_h (\rho - \sigma)) \right|$$

$$= D(\rho - \sigma) = \sqrt{1 - p^2} (\rho, \sigma) = \left( 1 - \left| \langle \phi_5 | \phi_5^j \rangle \right|^2 \right)^{1/2}$$

(27)

$$\leq \left( 1 - \left| \langle \phi_5 | \phi_5^j \rangle \right|^2 \right)^{1/2}$$

Now we bound the quantity $\langle \phi_5 | \phi_5^j \rangle$.

$$\langle \phi_5 | \phi_5^j \rangle = \left| \sum_{i \in [M]} \sqrt{D_t^j} \cdot D_t^j + \langle \zeta_t | \phi_5^j \rangle \right| \geq \left| \sum_{i \in [M]} \sqrt{D_t^j} \cdot D_t^j - \langle \zeta_t | \phi_5^j \rangle \right|$$

(28)

Let us bound the term $\tilde{D}_t^{j+1} \cdot \tilde{D}_t^{j+1}$ first as follows

$$\tilde{D}_t^{j+1} \cdot \tilde{D}_t^{j+1} = \sum_{i \in [M]} \frac{D_t^j \cdot e^{-\beta_{j,t}^j}}{k} \cdot Z_t^j \cdot \frac{D_t^j \cdot e^{-\beta_{j,t}^j}}{k}$$

$$= \frac{Z_t}{kZ_t} \sum_{i \in [M]} \sqrt{\frac{W_t^j \cdot e^{-\beta_{j,t}^j}}{W_t^j + W_t^j} + \frac{W_t^j \cdot e^{\beta_{j,t}^j}}{W_t^j + W_t^j}}$$

$$= \frac{1}{\sqrt{2\kappa}} \sum_{i=1}^{C} \frac{W_t^j \cdot e^{-\beta_{j,t}^j}}{\sqrt{W_t^j + W_t^j}} + \frac{W_t^j \cdot e^{\beta_{j,t}^j}}{\sqrt{W_t^j + W_t^j}}$$

(29)

We also know that $|\langle \zeta_t | \phi_5^j \rangle| \leq \| \zeta_t \| \leq 1 - \left( 1 - \frac{4\epsilon}{1+\epsilon} \right) = \frac{4\epsilon}{1+\epsilon}. \therefore$ Therefore we have

$$\langle \phi_5 | \phi_5^j \rangle \geq \left( \frac{C}{\kappa (1+\epsilon)} \right)^{1/2} - \frac{4\epsilon}{1+\epsilon}$$

(30)
Substituting $\kappa = \frac{C}{1 - \varepsilon} \sqrt{\frac{1 + \varepsilon}{1 + \varepsilon}}$, in the above equation, we have

$$|\langle \phi_5 | \phi_5^\prime \rangle| \geq \left( C \cdot (1 - \varepsilon) \frac{1 - e}{1 + e} \right)^{\frac{1}{2}} - \frac{4\varepsilon}{1 + e} = \left(1 - \frac{e}{1 + e} \right)^{\frac{1}{2}} - \frac{4\varepsilon}{1 + e} = \left(1 - \frac{2\varepsilon}{1 + e} \right)^{\frac{1}{2}} - \frac{4\varepsilon}{1 + e} \quad (31)$$

Since $(1 - x)^t \geq 1 - tx, \forall x \leq 1, t > 0$, we have

$$|\langle \phi_5 | \phi_5^\prime \rangle| \geq 1 - \frac{e}{1 + e} - \frac{4\varepsilon}{1 + e} = 1 - \frac{5\varepsilon}{(1 + e)} \quad (32)$$

Plugging this back into (27), we get

$$|p - q| \leq 2\sqrt{1 - \left(1 - \frac{5\varepsilon}{(1 + e)}\right)^\frac{3}{2}} \leq 2\sqrt{\frac{10Q\varepsilon}{1 + e}} < 8\sqrt{Q\varepsilon} \quad (33)$$

We now set $\varepsilon = \frac{1}{Q^2}$ which gives us $q = O(1 - \frac{1}{t})$ if $p = O(1 - \frac{1}{t})$. □

5.1.3 Obtaining confidence-rated predictions on sample points

Applying $U_D$ to $|\psi_1\rangle$, we have oracle access to the hypothesis $h_j$ and 2C copies of the state

$$|\psi_2\rangle = \frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i, D_{j}^{i}||j_i^1, j_i^2, \ldots, j_i^{t-1}\rangle \quad (34)$$

We perform the following unitary transformation.

$$[[\psi_2]|0\rangle^{\otimes 2C} \xrightarrow{Q_m} [[[\psi_3]|0\rangle^{\otimes 2C} \quad (35)$$

Consider the $(k, b)^{th}$ copy of $|\langle \psi_3\rangle|^{\otimes 2C}$ for $k \in \{1, 2, \ldots, C\}$ and $b \in \{-1, +1\}$. Perform the update $\psi_3 \rightarrow \psi_4$ as

$$|\psi_3(k, b) \rangle (0)^{\otimes 2C} \rightarrow |\psi_4(k, b) \rangle$$

$$= \frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i| y_i \rangle |D_{j}^{i}||j_i^1, j_i^2, \ldots, j_i^{t-1}\rangle |[i]_1 \rangle |[y_i]_2 \rangle \quad (36)$$

Note here that $[i]_1$ and $[i]_2$ are binary valued states. Using $[i]_1$ and $[i]_2$ as controls, we obtain the state

$$|\psi_5(k, b) \rangle = \frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i| y_i \rangle |D_{j}^{i}||j(i, t)\rangle |[i]_1 \rangle |[i]_2 \rangle |\psi_j|_1 \rangle |\psi_j|_2 \rangle \quad (37)$$

Now we perform a conditional rotation on the $|\tilde{D}_j^{k, b}\rangle$ register to obtain

$$|\psi_6(k, b) \rangle = \sqrt{W_{b}^{k, t} |\chi_1|^{\frac{1}{2}}_{(k, b)} |1 \rangle + \sqrt{1 - W_{b}^{k, t}} |\chi^0_{(k, b)} |0 \rangle}$$

In Section 5.1.3, we have $W_{b}^{k, t} = \sum_{i \in [M]} |\tilde{D}_j^{k, b}\rangle / M$, and

$$|\chi_1|^{\frac{1}{2}}_{(k, b)} = \frac{1}{\sqrt{M}} \sum_{i \in [M]} \sqrt{|\tilde{D}_j^{k, b}|} |x_i| |y_i\rangle |\tilde{D}_j^{i}||j (i, t)\rangle |[i]_1 \rangle |[i]_2 \rangle$$

$$|\chi^0_{(k, b)} = \frac{1}{\sqrt{M}} \sum_{i \in [M]} \sqrt{|1 - \tilde{D}_j^{k, b}|} |x_i| |y_i\rangle |\tilde{D}_j^{i}||j (i, t)\rangle |[i]_1 \rangle |[i]_2 \rangle$$

Let $V_{0, 6}^{(k, b)}$ be the unitary that performs $|0\rangle \rightarrow |\psi_6(k, b)\rangle$. We perform relative-error amplitude estimation as stated in Lemma 5, with an expected $O(\sqrt{MQT^2})$ queries to $V_{0, 6}^{(k, b)}$ and $V_{0, 6}^{-1}$ to obtain the quantity $\tilde{W}_{b}^{k, t}$ that is an estimate of $W_{b}^{k, t}$ with high probability. We carry out relative error amplitude estimation as stated in Lemma 5 to estimate the quantity $W_{b}^{k, t}$.

Hence, we obtain all $2C$ values of $\tilde{W}_{b}^{j, t}$ for all $j \in \{1, 2, \ldots, C\}$, $b \in \{-1, +1\}$. Note that it is possible for the value of $W_{b}^{j, t}$ to be very small (even zero) for some $j$. This would result in the quantities $\beta_j$, becoming very large or unbounded, thus increasing the tendency of the learner to overfit. We use a general smoothing technique known as Laplace correction (Clark and Niblett, 1989) to overcome this issue and use the smoothed values to calculate the margins as $\beta_j = \frac{1}{2} \ln \left( \frac{\tilde{W}_{j, t}}{\tilde{W}_{b, j, t}} \right)$ for all $j \in \{1, \ldots, C\}$ and the normalization constant as

$$Z_j = 2 \sqrt{\frac{C}{2C}} \tilde{W}_{j, t}$$

We give a brief overview of Laplace correction here. Let $V_{b}^{k, t} = \tilde{W}_{b}^{k, t} \cdot M$. We update the values of $\tilde{W}_{b}^{k, t}$ to $\frac{V_{b}^{k, t}}{M + 2C}$. We also note that

$$\beta_j = \frac{1}{2} \ln \left( \frac{\tilde{W}_{j, t}}{\tilde{W}_{b, j, t}} \right) \quad \forall j \in \{1, \ldots, C\} \quad (39)$$

Let us look at the corner cases now. If there exists a partition where $W_{b}^{k, t} = 0$ or very small, then we now have $\tilde{W}_{b}^{k, t} \sim 1/\sqrt{M + 2C} \sim 1/M$, essentially resetting the weight. On the other hand, consider a partition where $V_{b}^{k, t} \sim M$. This implies that $V_{b}^{k, t} \sim 0$. By Eq.(39), this would give us
unbounded margins $\beta_{j,t} = \frac{1}{2} \ln \left( \frac{V_{+j,t}}{V_{-j,t}} \right) \sim \infty$. Now, due to the smoothing, the confidence for this domain partition will still be large but bounded above by $O(\log M)$.

**Claim 3** Let the weights be relatively estimated using the error parameter $\varepsilon$ by Algorithm 4; i.e., $\left| W_{b}^{j,t} - \bar{W}_{b}^{j,t} \right| \leq \varepsilon \cdot W_{b}^{j,t}$. Then the difference between the actual margins $\beta_{j,t}$ and the estimated margins $\beta_{j,t}'$, is bounded as $\left| \beta_{j,t}' - \beta_{j,t} \right| \leq \frac{1}{2} \ln \left( \frac{1+\varepsilon}{1-\varepsilon} \right)$; $j \in \{1, 2, \ldots, C\}$.

**Proof** We know that
\[
\left| W_{b}^{j,t} - \bar{W}_{b}^{j,t} \right| \leq \varepsilon \cdot W_{b}^{j,t} \tag{40}
\]

Also, recall that the actual margin given in Algorithm 3 is $\beta_{j,t} = \frac{1}{2} \ln \left( \frac{W_{+j,t}}{W_{-j,t}} \right)$ and the estimated margin in Algorithm 4 is $\beta_{j,t}' = \frac{1}{2} \ln \left( \frac{\bar{W}_{+j,t}}{\bar{W}_{-j,t}} \right)$. We upper bound the difference in margins as follows:
\[
\beta_{j,t}' - \beta_{j,t} = \frac{1}{2} \left[ \ln \frac{\bar{W}_{+j,t}}{W_{+j,t}} - \ln \frac{\bar{W}_{-j,t}}{W_{-j,t}} \right] = \frac{1}{2} \left[ \ln \frac{\bar{W}_{+j,t}}{W_{+j,t}} - \ln \frac{\bar{W}_{-j,t}}{W_{-j,t}} \right] \\
\leq \frac{1}{2} \ln \left( (1+\varepsilon) - (1-\varepsilon) \right) \\
= \frac{1}{2} \ln \left( \frac{1+\varepsilon}{1-\varepsilon} \right) \tag{41}
\]

Similarly, we obtain the lower bound as
\[
\beta_{j,t}' - \beta_{j,t} \geq \frac{1}{2} \ln \left( \frac{1-\varepsilon}{1+\varepsilon} \right) \tag{42}
\]

Combining Eq. (41) and Eq. (42) we get
\[
\left| \beta_{j,t}' - \beta_{j,t} \right| \leq \frac{1}{2} \ln \left( \frac{1+\varepsilon}{1-\varepsilon} \right) \tag{43}
\]

From Claim 3 we see that the difference between the actual margin and the estimated margins is very small. In fact, a very simple calculation shows us that $\left| \beta_{j,t}' - \beta_{j,t} \right| \leq 0.1$ for $\varepsilon \leq 0.1$. We note that the error parameter $\varepsilon$ is far smaller than a constant fraction, which means our estimated margins are quite close to the ideal margin values.

**Claim 4** In the same setting as in Claim 3, the deviation in the normalization constant at every iteration is bounded as $\left| Z_{t}' - Z_{t} \right| \leq \varepsilon \cdot Z_{t}$.

**Proof** The normalization constant in the classical RealBoost algorithm (Algorithm 3) is calculated as
\[
Z_{t} = 2 \sum_{j=1}^{C} \sqrt{W_{+j,t} \cdot W_{-j,t}}
\]

In Algorithm 4, we substitute the calculated weights with estimated weights to obtain the quantity
\[
Z_{t}' = 2 \sum_{j=1}^{C} \sqrt{\bar{W}_{+j,t} \cdot \bar{W}_{-j,t}}
\]

Using (40), we upper bound the difference between the quantities as
\[
Z_{t}' \leq 2 \sum_{j=1}^{C} \sqrt{\bar{W}_{+j,t} \cdot \bar{W}_{-j,t} \cdot (1+\varepsilon)} \tag{44}
\]

Similarly, the lower bound is obtained as
\[
Z_{t}' \geq 2 \sum_{j=1}^{C} \sqrt{W_{+j,t} \cdot W_{-j,t} \cdot (1-\varepsilon)} \tag{45}
\]

Combining Eq. (44) and Eq. (45) we obtain
\[
\left| Z_{t}' - Z_{t} \right| \leq \varepsilon \cdot Z_{t} \tag{46}
\]

Claim 4 shows that when we minimize the normalization constant at every step using the estimated values $\bar{W}_{+j,t}$, these quantities are themselves relatively bounded by the actual normalization constant. This implies that the training error of the combined classifier is greedily minimized when the normalization constant is minimized at every step. Hence, our training error at every step does not blow up due to estimation of the partition weights.

Now, we plug in the values of $\kappa$ (as initialized in Algorithm 4), $\beta_{j,t}'$, and $Z_{t}'$ in to Eq. (14) to perform the update from $\bar{D}_{i}$ to $\bar{D}_{i+1}$ for all $i \in [M]$. The output of the algorithm is the
final hypothesis \( H(x) = \text{sign} \left( \sum_{j=1}^{T} \beta_{j,t} \right) \) where our weak learner assigns any training example \( x \sim D \) the \( j \)th partition at the \( t \)th iteration, and \( \beta_{j,t} \) is the weighted prediction of the \( j \)th partition at the \( t \)th iteration.

5.2 Proof of correctness

The probability of failure of Algorithm 4 stems primarily from the steps 7, 8, and 13, where each step fails with a probability at most \( O(1/T) \). When we take a union bound over all \( T \) iterations for all three steps, the overall failure probability dips to an arbitrary constant which is at most 1/3.

There is an extra log factor incurred due to error reduction which can be absorbed in the \( \tilde{O}(.) \) notation.

**Claim 5** For a sufficiently large number of iterations \( T \geq \ln M / 2\gamma^2 \), our combined classifier \( H \) has zero training error with respect to the uniform superposition \( \tilde{D}^1 \) with high probability.

**Proof** To prove the convergence of training error we follow the classical analysis as presented in Schapire and Freund (2012). From Eq. (14) we have

\[
\tilde{D}^1_{t+1} = \frac{\tilde{D}^1_t}{\prod_{i=1}^{T} \kappa \cdot Z_i} \cdot \exp \left( -y \sum_{i=1}^{T} \beta_{j,t} \right)
\]

Let \( x \sim D \) and \( |x, 0 \rangle \xrightarrow{O_{H_i}} |x', j' \rangle \) for all \( t \in \{1, \ldots, T\} \) and \( F(x) = \sum_{i=1}^{T} \beta_{j,t} \). Here, note that \( H(x) = \text{sign}(F(x)) \) according to Eq. (15). Since error means the hypothesis gives a different output than the label, we have,

\[ H(x) \neq y \implies y \cdot F(x) \leq 0 \]
\[ \implies \exp(-y F(x)) \geq 1 \]
\[ \implies \exp(-y F(x)) \geq \mathbb{I}[H(x) \neq y] \]

The last inequality follows from the fact that \( \mathbb{I}[H(x) \neq c(x)] \in [0, 1] \). Now if we try to upper bound the training error, we have

\[
\Pr_{x \sim \tilde{D}^1} \left[ H(x) \neq y \right] = \sum_{i=1}^{M} \tilde{D}^1_{t_i} \cdot \mathbb{I}[H(x) \neq y] \leq \sum_{i=1}^{M} \tilde{D}^1_{t_i} \cdot \exp(-y F(x_i)) \]
\[ = \sum_{i=1}^{M} \tilde{D}^1_{t_i} \cdot \exp \left( -y \sum_{j=1}^{T} \beta_{j,t_i} \right) \]
\[ = \sum_{i=1}^{T} \tilde{D}^1_{t_i} \cdot \prod_{j=1}^{T} \kappa \cdot Z_i \]
\[ \leq \prod_{i=1}^{T} \kappa \cdot Z_i \]

The last inequality is due to the fact that \( \sum_{i=1}^{M} \tilde{D}_t^1 \in \left[ 1 - \frac{4\kappa}{1+\gamma}, 1 \right] \) as given in Claim 1. Now from Claim 4, we know that \( Z_i \leq Z_t(1 + \epsilon) \). This means

\[
\Pr_{x \sim \tilde{D}^1} \left[ H(x) \neq y \right] \leq \prod_{i=1}^{T} \kappa \cdot Z_t(1 + \epsilon) = \kappa^T (1 + \epsilon)^T \prod_{i=1}^{T} Z_i \tag{50}
\]

Substituting \( \kappa = \frac{C}{(1-\gamma)\sqrt{1+\epsilon}} \), and the fact that \( \epsilon = O(1/QT^2) \) we have

\[
\Pr_{x \sim \tilde{D}^1} \left[ H(x) \neq y \right] \leq CT \left( 1 + \frac{1 + 1/2T^2}{1 - 1/2T^2} \right) \prod_{i=1}^{T} Z_i \]
\[ \leq CT \left( \frac{T^2 + 1}{T^2 - 1} \right) \prod_{i=1}^{T} Z_i \]
\[ \leq CT \left( 1 + \frac{2}{T^2 - 1} \right) \prod_{i=1}^{T} Z_i \]
\[ \leq CT \exp \left( \frac{2T}{T^2 - 1} \right) \prod_{i=1}^{T} Z_i \tag{51}
\]

For sufficiently large \( T \), we have

\[
\Pr_{x \sim \tilde{D}^1} \left[ H(x) \neq y \right] \leq CT e^{2T} \prod_{i=1}^{T} Z_i \]
\[ \leq CT e^{2T} \prod_{i=1}^{T} \sqrt{1 - 4\gamma^2} \]
\[ \leq CT \exp \left( \frac{2}{T} - 2 \sum_{i=1}^{T} \gamma_i^2 \right) \]
\[ \leq CT \exp \left( \frac{2}{T} - 2\gamma^2 T \right) \]
\[ \leq CT \exp \left( -2\gamma^2 T + \frac{2}{T} \right) \tag{52}
\]

The second inequality follows from Lemma 2. We note here that the term \( e^{2T} \) goes down fast depending on the constant in \( T = O(\log M / \gamma^2) \), which can be as small as \( O(\log C) \). This leaves us with the term \( e^{2T} \) to take care of. Substituting \( T = O(\log M / \gamma^2) \) gives implies

\[
\Pr_{x \sim \tilde{D}^1} \left[ H(x) \neq y \right] < \frac{1}{M} \tag{53}
\]

We recall the fact that \( \tilde{D}^1 \) is the uniform distribution, which implies that we have zero training error.

From Corollary 2 and Claim 5, we also conclude that if we run Algorithm 4 for a sufficiently large number of iterations \( T \), then with a high probability we output a hypothesis \( H \) according to Eq. (15) that has zero training error and a small generalization error.
5.3 Complexity analysis

In this section we state the query complexity and the time complexity of our algorithm.

**Theorem 3 (Query Complexity)** Suppose we boost a $\gamma$-weak learner $A$ with sample complexity $Q$, and an associated hypothesis class $\mathcal{H}$ having VC dimension $d$ using Algorithm 4. If the weak learner $A$ produces at most $C$ partitions at every iteration, then the query complexity of Algorithm 4 is $O\left(\frac{\sqrt{d} C Q}{\gamma^2}\right)$.

**Proof** The query complexity, as in previous works (Arunachalam and Maiti, 2020; Izdebski and de Wolf, 2020), considers the number of queries to the hypothesis oracles $\{O_{h_1}, \ldots, O_{h_{T-1}}\}$ made by Algorithm 4. We now start calculating the query complexity by considering the queries made in the $r$th iteration.

We require $r-1$ queries to the oracles $O_{h_1}, O_{h_2}, \ldots, O_{h_{r-1}}$ for each copy of $|\psi\rangle_0$ and $|\phi_0\rangle$. Using Lemma 4, we see that our amplitude amplification algorithm uses an expected $\Theta(p' \log T / \rho)$ calls to the unitaries $U_{0 \rightarrow 3}$ and $U_{0^{-1} \rightarrow 3}$, to obtain $|\psi_4\rangle$ with a high probability as discussed in Footnote 7. We observe from $|\psi_3\rangle$ and $|\phi_4\rangle$ that

$$p = \sum_{i \in [M]} \sqrt{D_i^4} / M; \quad p' = \sum_{i \in [M]} \sqrt{D_i^4}$$

Hence, the Amplitude Amplification step to obtain $|\phi_4\rangle$ requires $O(\sqrt{M} \log T (t - 1))$ queries to the oracles for each copy of $|\psi_3\rangle$. The uncompute step to obtain $|\phi_5\rangle$ requires a further $t - 1$ queries to the oracles $O_{h_1}, O_{h_2}, \ldots, O_{h_{t-1}}$ for each copy of $|\phi_4\rangle$.

For estimating the partition weights with high probability as discussed in Footnote 7 we make an expected $\tilde{O}\left(\sqrt{M} QT^2 \log T \cdot t\right)$ queries to $O_{h_1}, O_{h_2}, \ldots, O_{h_t}$. We obtain this by plugging in $p = O(1/M), \epsilon = O(\frac{1}{QT^2})$, and $k = log T$ in Lemma 5. Hence, the total query complexity is

$$\sum_{t=1}^{T} \left( O(\sqrt{M} Q \log T \cdot (t - 1)) + O((Q + C) \log T \cdot (t - 1)) \right)$$

$$+ \tilde{O}\left(\sqrt{MCQT^2 \log T \cdot t}\right)$$

$$= O(\sqrt{MCQT^2 \log T}) + O((Q + C)T^2 \log T)$$

$$+ \tilde{O}\left(\sqrt{MCQT^4 \log T}\right)$$

$$= \tilde{O}(\sqrt{MCQT^4}) = O\left(\frac{\sqrt{MCQ}}{\gamma^8}\right)$$

The last equality follows from Lemma 2 by setting $T = O(\log M / \gamma^2)$. From Corollary 2, and by setting the parameter $\eta = 0.1$, we get the query complexity as $O\left(\frac{\sqrt{d} C Q}{\gamma^2}\right)$.

**Theorem 4 (Time Complexity)** Suppose we boost a $\gamma$-weak learner $A$ with sample complexity $Q$, and an associated hypothesis class $\mathcal{H}$ having VC dimension $d$ using Algorithm 4. The size of the class $C$ is assumed to be $n$. If the weak learner $A$ produces at most $C$ partitions at every iteration, then the time complexity of Algorithm 4 is $O\left(\frac{n^2 \sqrt{MCQ}}{\gamma^8}\right)$.

**Proof** As discussed in Section 3.2, we can assume a QRAM to prepare the uniform superposition $\frac{1}{\sqrt{M}} \sum_{i \in [M]} |x_i, y_i, D_i\rangle$ using $O(n \log M)$ gates. Hence the time complexity for preparing the state $|\phi_0\rangle \otimes \psi_0 \otimes |\psi_0\rangle \otimes 2C$ is $O(n(Q + C))$. The step from $|\phi_0\rangle$ to $|\phi_1\rangle$ and $|\psi_0\rangle$ to $|\psi_1\rangle$ requires $t - 1$ queries each, which can be performed in time $O((Q + C)(t - 1))$. Next we perform the distribution update which is an arithmetic operation, using the unitary $U_Q$ with the $|j_1, \ldots, j_{t-1}\rangle$ register as control. This step requires time $O(n^2(Q + C)(t - 1))$. We perform amplitude amplification to obtain the state $|\psi_4\rangle$. This requires $O(\sqrt{M}(t - 1) \log T)$ applications of $U_{0 \rightarrow 3}$ and $U_{0^{-1} \rightarrow 3}$ as discussed in the previous section. The total time taken is therefore $O(n^2 \sqrt{M}(t - 1) \log T)$. The time taken by our weak learner to output $O_{h_t}$ is $O(n^2 Q)$. The arithmetic operations to update state $|\psi_3\rangle_{(k,b)}$ to $|\psi_5\rangle_{(k,b)}$ and perform controlled rotation use $O(n)$ gates. Finally we make $\tilde{O}\left(\sqrt{MCQT^2 \log T}\right)$ queries for the amplitude estimation part, and each query requires time $O(n^2t)$. Therefore our final time complexity is

$$\sum_{t=1}^{T} \left( O(n^2 \sqrt{M}(t - 1) \log T) + \tilde{O}\left(\frac{n^2 \sqrt{MCQT^2 \log T \cdot t}}{\gamma^8}\right)\right)$$

$$+ O(n^2(Q + C)(t - 1))$$

$$= \tilde{O}\left(n^2 \sqrt{MCQT^4 \log T}\right) = O\left(\frac{n^2 \sqrt{MCQ}}{\gamma^8}\right)$$

\[ \square \]

6 Experiments

We evaluate the generalization ability and convergence of our proposed QRealBoost algorithm on two datasets: Breast Cancer Wisconsin (Wolberg et al., 1995), and MNIST (Deng, 2012). We compare the performance of QRealBoost against four alternatives: The classical AdaBoost algorithm (Freund and Schapire, 1997), the classical RealBoost algorithm (Schapire and Singer, 1999), the classical
SmoothBoost algorithm (Servedio, 2003), and the QAdaBoost algorithm (Arunachalam and Maity, 2020). We discuss our design decisions in Section 6.1 and focus on our methodology in Section 6.2. We discuss our findings on the Breast Cancer Wisconsin Dataset and MNIST datasets in Section 6.3 and Section 6.4 respectively. Our code is freely available at https://github.com/braqiiit/QRealBoost.

6.1 Implementation details

Since there are no quantum simulators or quantum backends large enough to test QRealBoost, we had to make some design and implementation choices detailed below.

We focus on the qualitative analysis behavior (training and test convergence) of the algorithms in these experiments rather than their efficiency due to the lack of efficient quantum simulators and quantum backends supporting a sufficient number of qubits. Thus, instead of computing the distribution weights from scratch, we store the updated weights after every iteration. This is done because in the former approach, the number of qubits needed to store the weights up to a reasonable degree of precision blows up with the number of iterations, taking even experiments with few training samples out of our reach. Even though this choice sacrifices the quantum speedup, it has no bearing on the convergence behavior of QRealBoost. On an actual quantum backend with a sufficient number of qubits, the implementation could have remained true to Algorithm 4, which would have saved on both space needed to store the weights, as well as the time needed to create the $Q + 2C$ superposition states $|ϕ_2⟩$ and $|ψ_2⟩$. Moreover, we could have performed experiments with larger training set sizes, which would have given us more insight into the actual empirical performance of QRealBoost.

We used a weak classical learner ($k$-means) since off-the-shelf quantum weak learners (Prakash, 2014; Rebentrost et al., 2014) are not readily available right now, and implementing one was out of the scope of this work. The implementation is modular and can be easily modified for any learner implemented as a quantum circuit. We measure the $|ϕ_5⟩$ state and pass the top $Q$ training samples to the $k$-means algorithm. In Section 3.2 we pointed out that this is exactly how quantum learners could simulate classical learners. Suppose we could use efficient implementations of weak quantum learners (with speedups over their classical counterparts). In that case, we could have demonstrated faster convergence times for QRealBoost instead of simply comparing convergence with respect to training and test accuracy per iteration.

We use the Iterative Quantum Amplitude Estimation (IQAE) algorithm (Grinko et al., 2021) due to its ready availability on QISKIT as a library function. IQAE replaces quantum phase estimation with clever use of Grover iterations, essentially performing additive estimation. Our choice was motivated by the availability and performance of the algorithm, which helped us decrease the number of qubits needed for the implementation.

An important note is that even though the experiments were conducted with additive estimation instead of relative estimation, we still managed to boost the weak learner. This is interesting since simply using additive estimation does not allow us to theoretically bound the intermediate quantities as explained in Section 4.1. We hypothesize that storing the distribution weights explicitly in every iteration offsets a lot of errors arising from the probabilistic subroutines. This gives us an insight into efficient implementations of quantum-inspired classical algorithms.

6.2 Methodology

We carry out two sets of experiments on both datasets. First, we fix the size of the training set to $M = 32$ samples and evaluate the generalization ability and convergence behavior for different sample complexities $Q = 4, 6, 8$ for both QAdaBoost and QRealBoost. We note here that in our implementation (which uses a weak classical learner), the sample complexity constraint is enforced by having the weak learner sample from the quantum state $|ϕ_5⟩^⊗Q$ produced by the quantum boosting algorithms.

For the classical algorithms, we do not enforce any sample complexity restriction on the weak learner and therefore do not enforce sampling inside the weak learner for the classical boosting algorithms. Hence, $K$-means samples $Q$ examples from the $M$ samples (with replacement) in every iteration for the quantum examples, while $K$-means uses all $M$ samples for clustering in the case of the classical algorithms. Therefore, the worst-case guess for $Q$ is $M$ for the classical algorithms. If the ratio $Q/M$ is too low, the underlying learner can access very few labeled samples and underfit the training set. Thus, the weak learner would have a high bias $γ$, which would, in turn, drive the training error up.

In the next set of experiments, we fix the sample complexity $Q$ of the quantum boosting algorithms to 8 and vary the size of the training set $M = 16, 32, 64$. Then we increase the size of the training set that increases the variance of the underlying learner, which further increases the generalization error. We investigate the generalization performance of our boosting algorithm through these experiments.

In Figs. 2, 3, 4 and 5, the lines for QAdaBoost and QRealBoost represent a mean accuracy over 5 independent experiments, and the hue bands represent the standard deviation across all experiments. Due to quantum resource limitations, the QAdaBoost and QRealBoost algorithms are tested on quantum simulators (instead of actual quantum backends).
6.3 Breast cancer

In Fig. 2, we see that the training performance of QRealBoost matches or exceeds QAdaBoost in all 3 cases. Both the quantum algorithms converge fastest for $Q = 8$. As we increase the sample complexity from $Q = 4$ to $Q = 8$, QRealBoost increases its generalization accuracy, unlike QAdaBoost.

In Fig. 3, we see that for $M = 16$, QRealBoost has higher training accuracy than QAdaBoost and similar generalization performance (QAdaBoost outperforms QRealBoost by...
a hair). For the case of \( M = 16 \), the classical AdaBoost algorithm does not converge, but generally has a good test accuracy. The classical RealBoost algorithm overfits during training, and has poor test accuracy. The classical SmoothBoost algorithm does not converge and has extremely poor training and test accuracy for \( M = 16 \). For \( M = 32 \), QRealBoost has faster convergence and higher training accuracy than QAdaBoost but suffers a drop in generalization performance. AdaBoost and RealBoost converge faster than the quantum algorithms and perform better generalization. The classical Smoothboost algorithm converges faster than the quantum algorithms but has the worst generalization performance out of all the algorithms. Finally, for \( M = 64 \), both QRealBoost and QAdaBoost converge fast to a high

**Fig. 4** Performance of QRealBoost, QAdaBoost, RealBoost, AdaBoost, and SmoothBoost on the MNIST Dataset with different sample complexities

**Fig. 5** Performance of QRealBoost, QAdaBoost, RealBoost, AdaBoost, and SmoothBoost on the MNIST Dataset with different training set sizes
training accuracy and with good generalization performance. AdaBoost does not converge for \( M = 64 \) but has good generalization performance, while RealBoost has poor training and test performance. SmoothBoost does not converge and has the worst generalization performance out of all five boosting algorithms.

6.4 MNIST

On the MNIST dataset, we see (refer to Fig. 4) that for all three cases of sample complexity \( Q = \{4, 6, 8\} \), quantum boosting algorithms outperform classical boosting algorithms in generalization accuracy. As expected, for the cases of \( Q = \{4, 6\} \), the convergence is slow for both QRealBoost and QAdaBoost, compared to the case where \( Q = 8 \). We found that for the MNIST, QAdaBoost and QRealBoost behaved similarly in every sample complexity variation.

The above trend continues even under variations on the training set sizes refer Fig. 5. Quantum algorithms converge faster, have higher training accuracy, and have better generalization performance than classical boosting algorithms for all cases. Once again, QAdaBoost and QRealBoost have similar performances.

7 Conclusions

In this work, we designed the QRealBoost algorithm, which tackles an open question posed by Izdebski and de Wolf (2020) to boost weak quantum PAC learners that output non-binary hypotheses. QRealBoost retains the performance of RealBoost, which has superior theoretical properties (supported by empirical evidence, too) compared to the AdaBoost algorithm. We also establish that both theoretically and empirically QRealBoost outperforms QAdaBoost, the only other known adaptive quantum boosting algorithm.

An issue with QRealBoost is the dependence of the query and time complexity on \( \gamma \), arising from recomputing \( \tilde{D}^\gamma \) over the training samples at every iteration from scratch. We believe this computation can be avoided by maintaining a “distribution oracle” that only needs to be updated in each iteration. If it turns out that the lower bound on \( \gamma \) is worse for quantum boosting algorithms compared to classical boosting algorithms in the general case, the next question would be finding (or even determining the existence of) relevant hypotheses classes in which quantum boosting provides us with an advantage.

We also observe that the constant factor \( C \) in the numerator of the time complexity may be exponentially reduced by simultaneously estimating the individual domain partition weights using amplitude estimation techniques as shown in van Apeldoorn (2021), and this is a possible direction of future work.

A logical continuation of this work is quantizing other variants of AdaBoost which depend on domain partitioning hypotheses such as GentleBoost (Friedman et al., 2000), ModestBoost (Vezhnevets and Vezhnevets, 2005), Parameterized AdaBoost (Wu and Nagahashi, 2014), and Penalized AdaBoost (Wu and Nagahashi, 2015). Each variant has different generalization abilities, which make them useful in different contexts. The algorithmic framework followed in this work for estimating the partition weights may be useful to model quantum versions of these variants.

Author contribution S.C., D.B. formulated the research problem and designed the solution; S.C., P.S.C., R.B. designed the experiments and analyzed the results; P.S.C., R.B. implemented the algorithms and ran the experiments; S.C., P.S.C., R.B., D.B. wrote the paper, D.B. supervised the entire work.

Availability of data and materials The code used in this paper is freely available at https://github.com/braqiiit/QRealBoost. The Breast Cancer Wisconsin (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html#sklearn.datasets.load_breast_cancer) and MNIST (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits) datasets analyzed in the current paper are publicly available as part of the scikit-learn datasets repository (Pedregosa et al., 2011).

Declarations

Competing interests The authors declare no competing interests.

References

Aaronson S, Rall P (2022) Quantum Approximate Counting, Simplified. pp. 24–32. https://doi.org/10.1137/1.9781611976014.5. https://epubs.siam.org/doi/abs/10.1137/1.9781611976014.5

Aleksandrowicz G, et al (2021) Qiskit: An open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505

Ambainis A (2008) STACS ed by P.W. Susanne Albers (IBFI Schloss Dagstuhl, Bordeaux, France, 2008), pp. 49–60 2008

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

Arunachalam S, De Wolf R (2018) Optimal quantum sample complexity of learning algorithms. J Mach Learn Res 19(1):2879–2878

Arunachalam S, Maity R (2020) 37th Int. Conf. Mach. Learn. ICML 2020., vol. Part F1684 (PMLR, 2020), pp. 354–364

Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: Bagging, boosting, and variants 36(1):105–139

Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. Nature 549(7671):195–202

Brassard G, Høyer P, Mosca M, Tapp A (2002) Quantum Comput. Inf. (Washington, DC, 2000), Contemp Math, vol. 305 (Amer. Math.
Bshouty NH, Gavinsky D (2002) On boosting with polynomially bounded distributions. J Mach Learn Res 3(Nov):483–506
Bshouty NH, Jackson JC (1998) Learning dnf over the uniform distribution using a quantum example oracle. SIAM J Comput 28(3):1136–1153
Clark P, Niblett T (1989) The cn2 induction algorithm. Mach learn 3(4):261–283
Deng L (2012) The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine 29(6):141–142
Dietterich TG (2000) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine learning 40(2):139–157
Drucker H, Cortes C (1995) Boosting a weak learning algorithm by majority. Inf Comput 121(2):256–285
Opitz D, Maclin R (1999) Popular ensemble methods: An empirical study. J artif intell res 11:169–198
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al (2011) Scikit-learn: Machine learning in python. the Journal of machine Learning research 12:2825–2830
Prakash A (2014) Quantum algorithms for linear algebra and machine learning. Ph.D. thesis, University of California, Berkeley 2014
Provost F, Domingos P (2003) Tree induction for probability-based ranking. Mach learn 52(3):199–215
Rebentrost P, Mohseni M, Lloyd S (2014) Quantum support vector machine for big data classification. Phys Rev Lett 113(13):503. https://doi.org/10.1103/PhysRevLett.113.130503
Ruan C, Ruan Q, Li X (2010) IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS , pp. 1402–1405. https://doi.org/10.1109/ICOSP.2010.5656917
Schapire RE, Freund Y (2012) Boosting: Foundations and Algorithms (The MIT Press, 2012)
Schapire RE (1990) The strength of weak learnability. Mach Learn 5(2):197–227. https://doi.org/10.1007/BF00116037
Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336. https://doi.org/10.1023/A:1007614523901
Schuld M, Petruccione F (2018) Quantum ensembles of quantum classifiers. Sci Rep 8(1):277. https://doi.org/10.1038/s41598-018-20403-3
Schuld M, Sinayskiy I, Petruccione F (2015) An introduction to quantum machine learning. Contemporary Physics 56(2):172–185
Servedio RA (2003) Smooth boosting and learning with malicious noise. J Mach Learn Res 4:633–648
van Apeldoorn J (2021) 16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021) (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021)
Vezhnevets A (2005) Vezhnevets V. Graphicon 12:987–997
Wang X, Ma Y, Hsieh MH, Yung MH (2020) Quantum speedup in adaptive boosting of binary classification. Sci China Physics Mech Astron 64(2):1–10. https://doi.org/10.1007/S11433-020-1638-5
Wang S, Wang H (2009) 2009 7th International Conference on Information, Communications and Signal Processing (ICICS) (IEEE, 2009), pp. 1–5
Wittek P (2014) Quantum machine learning: what quantum computing means to data mining. (Academic Press, 2014)
Wolberg DWH, Street WN, Mangasarian OL (1995) UCI machine learning repository. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
Wu S, Nagahashi H (2014) Parameterized adaboost: introducing a parameter to speed up the training of real adaboost. IEEE Signal Process Lett 21(6):687–691
Wu S, Nagahashi H (2015) Penalized adaboost: improving the generalization error of gentle adaboost through a margin distribution. IEICE Trans Inf Syst 98(11):1906–1915
Wu B, Ai H, Huang C, Lao S (2004) Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings , pp. 79–84. https://doi.org/10.1109/AFGR.2004.1301512

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.