QuantumBoost: A lazy, yet fast, quantum algorithm for learning with weak hypotheses

Amira Abbas, **Yanlin Chen**, Tuyen Nguyen, Ronald de Wolf

July, 2026

(arXiv:2510.05089)

PAC Learning & Boosting

PAC Learning & Boosting

Setup:

PAC Learning & Boosting

Setup:

- Input domain $\mathcal{X}$ and target concept $c : \mathcal{X} \to \{-1, 1\}$.

PAC Learning & Boosting

Setup:

- Input domain $\mathcal{X}$ and target concept $c : \mathcal{X} \to \{-1, 1\}$.
- Learner receives training set S of m i.i.d. samples from unknown distribution $\mathcal{D}$ on $\mathcal{X}$.

PAC Learning & Boosting

Setup:

- Input domain $\mathcal{X}$ and target concept $c : \mathcal{X} \to \{-1, 1\}$.
- Learner receives training set S of m i.i.d. samples from unknown distribution $\mathcal{D}$ on $\mathcal{X}$.

PAC Learning and Boosting:

PAC Learning & Boosting

Setup:

- Input domain $\mathcal{X}$ and target concept $c : \mathcal{X} \to \{-1, 1\}$.
- Learner receives training set S of m i.i.d. samples from unknown distribution $\mathcal{D}$ on $\mathcal{X}$.

PAC Learning and Boosting:

- **Strong Learner**: For every $\mathcal{D}'$ and c, with probability $\geq 1 - \delta$ over the choice of S, outputs a hypothesis h with generalization error $\Pr_{x \sim \mathcal{D}'}[h(x) \neq c(x)] \leq \epsilon$.

PAC Learning & Boosting

Setup:

- Input domain $\mathcal{X}$ and target concept $c : \mathcal{X} \to \{-1, 1\}$.
- Learner receives training set S of m i.i.d. samples from unknown distribution $\mathcal{D}$ on $\mathcal{X}$.

PAC Learning and Boosting:

- **Strong Learner**: For every $\mathcal{D}'$ and c, with probability $\geq 1 - \delta$ over the choice of S, outputs a hypothesis h with generalization error $\Pr_{x \sim \mathcal{D}'}[h(x) \neq c(x)] \leq \epsilon$.
- **Weak Learner**: An instantiation of the same framework that only guarantees an error $\leq 1/2 - \gamma$ (slightly better than random guessing) with $\delta = 0$.

PAC Learning & Boosting

Setup:

- Input domain $\mathcal{X}$ and target concept $c : \mathcal{X} \to \{-1, 1\}$.
- Learner receives training set S of m i.i.d. samples from unknown distribution $\mathcal{D}$ on $\mathcal{X}$.

PAC Learning and Boosting:

- **Strong Learner**: For every $\mathcal{D}'$ and c, with probability $\geq 1 - \delta$ over the choice of S, outputs a hypothesis h with generalization error $\Pr_{x \sim \mathcal{D}'}[h(x) \neq c(x)] \leq \epsilon$.
- **Weak Learner**: An instantiation of the same framework that only guarantees an error $\leq 1/2 - \gamma$ (slightly better than random guessing) with $\delta = 0$.
- **Boosting**: Converts a weak learner to a strong one.

Known algorithms

Known algorithms

Classical Algorithms:

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal).

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers.

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).
- **Kale's SmoothBoost** [Kal07, BHK09]: Approximate projection on high-density set Γ_ϵ.

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).
- **Kale's SmoothBoost** [Kal07, BHK09]: Approximate projection on high-density set Γ_ϵ. Uses $T \approx \frac{1}{\gamma^2}$ iterations (roughly same as AdaBoost).

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).
- **Kale's SmoothBoost** [Kal07, BHK09]: Approximate projection on high-density set Γ_ϵ. Uses $T \approx \frac{1}{\gamma^2}$ iterations (roughly same as AdaBoost).

Prior Quantum Speedups:

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).
- **Kale's SmoothBoost** [Kal07, BHK09]: Approximate projection on high-density set Γ_ϵ. Uses $T \approx \frac{1}{\gamma^2}$ iterations (roughly same as AdaBoost).

Prior Quantum Speedups:

- **Q-AdaBoost** [AM20]: Quadratic runtime speedup in m, but severely worsens W- and γ-dependencies.

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).
- **Kale's SmoothBoost** [Kal07, BHK09]: Approximate projection on high-density set Γ_ϵ. Uses $T \approx \frac{1}{\gamma^2}$ iterations (roughly same as AdaBoost).

Prior Quantum Speedups:

- **Q-AdaBoost** [AM20]: Quadratic runtime speedup in m, but severely worsens W- and γ-dependencies.
- **Q-SmoothBoost** [IdW23]: Smoothness enables efficient quantum sample preparation, improving W- and γ-dependencies over [AM20].

Known algorithms

Classical Algorithms:

- **AdaBoost** [FS97]: Uses $T \approx \frac{1}{\gamma^2}$ iterations (near-optimal). $m \approx \frac{d}{\epsilon\gamma^2}$ examples suffice to infer 2ϵ generalization error from ϵ empirical error (d: VC dimension of weak learner hypothesis).
- **SmoothBoost** [Ser03]: Smoothness of intermediate distributions prevents high weights on outliers. Uses more examples ($m \approx \frac{d}{\epsilon^2\gamma^2}$) and iterations ($T \approx \frac{1}{\epsilon\gamma^2}$).
- **Kale's SmoothBoost** [Kal07, BHK09]: Approximate projection on high-density set Γ_ϵ. Uses $T \approx \frac{1}{\gamma^2}$ iterations (roughly same as AdaBoost).

Prior Quantum Speedups:

- **Q-AdaBoost** [AM20]: Quadratic runtime speedup in m, but severely worsens W- and γ-dependencies.
- **Q-SmoothBoost** [IdW23]: Smoothness enables efficient quantum sample preparation, improving W- and γ-dependencies over [AM20].

Goal: A quantum algorithm with better m, γ, and ϵ dependencies

Quantum Setup and Our Results

Quantum Setup and Our Results

Computational model

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines
- $\tilde{\Theta}\left(\frac{d}{\epsilon^2\gamma^2}\right)$ examples stored in quantum-readable read-only memory (QCROM).

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines
- $\tilde{\Theta}\left(\frac{d}{\epsilon^2\gamma^2}\right)$ examples stored in quantum-readable read-only memory (QCROM).
- Time complexity = classical operations + quantum gates + QCROM queries.

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines
- $\tilde{\Theta}\left(\frac{d}{\epsilon^2\gamma^2}\right)$ examples stored in quantum-readable read-only memory (QCROM).
- Time complexity = classical operations + quantum gates + QCROM queries.

Our Result

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines
- $\tilde{\Theta}\left(\frac{d}{\epsilon^2\gamma^2}\right)$ examples stored in quantum-readable read-only memory (QCROM).
- Time complexity = classical operations + quantum gates + QCROM queries.

Our Result

QuantumBoost achieves generalization error $\leq \epsilon$ running in time:

$$\tilde{O}\left(\frac{W}{\sqrt{\epsilon}\gamma^4}\right).$$

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines
- $\tilde{\Theta}\left(\frac{d}{\epsilon^2\gamma^2}\right)$ examples stored in quantum-readable read-only memory (QCROM).
- Time complexity = classical operations + quantum gates + QCROM queries.

Our Result

QuantumBoost achieves generalization error $\leq \epsilon$ running in time:

$$\tilde{O}\left(\frac{W}{\sqrt{\epsilon}\gamma^4}\right).$$

- Matches near-optimal iterations $T = \frac{\log(1/\epsilon)}{\gamma^2}$ of classical AdaBoost.

Quantum Setup and Our Results

Computational model

- **Input**: $\epsilon > 0$; γ-weak learner $\mathcal{W}$ (runtime W, hypothesis class $\mathcal{H}$ with VC dim d).
  Output: A hypothesis H with generalization error $\leq \epsilon$.
- Classical computer calling quantum subroutines
- $\tilde{\Theta}\left(\frac{d}{\epsilon^2\gamma^2}\right)$ examples stored in quantum-readable read-only memory (QCROM).
- Time complexity = classical operations + quantum gates + QCROM queries.

Our Result

QuantumBoost achieves generalization error $\leq \epsilon$ running in time:

$$\tilde{O}\left(\frac{W}{\sqrt{\epsilon}\gamma^4}\right).$$

- Matches near-optimal iterations $T = \frac{\log(1/\epsilon)}{\gamma^2}$ of classical AdaBoost.
- Works with quantum learners requiring quantum examples.

QuantumBoost: A quantized lazy SmoothBoost

QuantumBoost: A quantized lazy SmoothBoost

Algorithm: QuantumBoost

Input : $\epsilon > 0$; $S = \{(x_i, y_i)\}_{i=1}^m$; γ-weak learner $\mathcal{W}$

Initialize $M_i^1 \leftarrow \epsilon \; \forall i \in [m]$; $K = 1/\gamma$;

for $t = 1, \ldots, T$ **do**

  Prepare $|D^t\rangle^{\otimes W}$, $h_t = \mathcal{W}(|D^t\rangle^{\otimes W})$, and (implicitly) compute ℓ^t ;

  if $t \pmod K = 0$ **then**

    $M^{t+1} = \widetilde{\text{Proj}}_{\Gamma_\epsilon}(M^t(1-\gamma)^{\ell^t})$; // Approx projection

  else

    $M^{t+1} = M^t(1-\gamma)^{\ell^t}$; // Lazy update

Output: $H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x))$;

QuantumBoost: A quantized lazy SmoothBoost

Algorithm: QuantumBoost

Input : $\epsilon > 0$; $S = \{(x_i, y_i)\}_{i=1}^m$; γ-weak learner $\mathcal{W}$

Initialize $M_i^1 \leftarrow \epsilon \; \forall i \in [m]$; $K = 1/\gamma$;

for $t = 1, \ldots, T$ **do**

  Prepare $|D^t\rangle^{\otimes W}$, $h_t = \mathcal{W}(|D^t\rangle^{\otimes W})$, and (implicitly) compute ℓ^t ;

  if $t \pmod K = 0$ **then**

    $M^{t+1} = \widetilde{\text{Proj}}_{\Gamma_\epsilon}(M^t(1-\gamma)^{\ell^t})$; // Approx projection

  else

    $M^{t+1} = M^t(1-\gamma)^{\ell^t}$; // Lazy update

Output: $H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x))$;

Technique 1: Implicit Weights

QuantumBoost: A quantized lazy SmoothBoost

Algorithm: QuantumBoost

Input : $\epsilon > 0$; $S = \{(x_i, y_i)\}_{i=1}^m$; γ-weak learner $\mathcal{W}$

Initialize $M_i^1 \leftarrow \epsilon \; \forall i \in [m]$; $K = 1/\gamma$;

for $t = 1, \ldots, T$ **do**

  Prepare $|D^t\rangle^{\otimes W}$, $h_t = \mathcal{W}(|D^t\rangle^{\otimes W})$, and (implicitly) compute ℓ^t ;

  if $t \pmod K = 0$ **then**

    $M^{t+1} = \widetilde{\text{Proj}}_{\Gamma_\epsilon}(M^t(1-\gamma)^{\ell^t})$; // Approx projection

  else

    $M^{t+1} = M^t(1-\gamma)^{\ell^t}$; // Lazy update

Output: $H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x))$;

Technique 1: Implicit Weights

- Avoid updating m weights explicitly. Evaluate entries on the fly using prior hypotheses $h_{<t}$.

QuantumBoost: A quantized lazy SmoothBoost

Algorithm: QuantumBoost

Input : $\epsilon > 0$; $S = \{(x_i, y_i)\}_{i=1}^m$; γ-weak learner $\mathcal{W}$

Initialize $M_i^1 \leftarrow \epsilon \; \forall i \in [m]$; $K = 1/\gamma$;

for $t = 1, \ldots, T$ **do**

  Prepare $|D^t\rangle^{\otimes W}$, $h_t = \mathcal{W}(|D^t\rangle^{\otimes W})$, and (implicitly) compute ℓ^t ;

  if $t \pmod K = 0$ **then**

    $M^{t+1} = \widetilde{\text{Proj}}_{\Gamma_\epsilon}(M^t(1-\gamma)^{\ell^t})$; // Approx projection

  else

    $M^{t+1} = M^t(1-\gamma)^{\ell^t}$; // Lazy update

Output: $H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x))$;

Technique 1: Implicit Weights

- Avoid updating m weights explicitly. Evaluate entries on the fly using prior hypotheses $h_{<t}$.
- Prepare $|D^t\rangle^{\otimes W}$ using $\tilde{O}(W/\sqrt{\epsilon})$ q-queries.

QuantumBoost: A quantized lazy SmoothBoost

Algorithm: QuantumBoost

Input : $\epsilon > 0$; $S = \{(x_i, y_i)\}_{i=1}^m$; γ-weak learner $\mathcal{W}$

Initialize $M_i^1 \leftarrow \epsilon \; \forall i \in [m]$; $K = 1/\gamma$;

for $t = 1, \ldots, T$ **do**

  Prepare $|D^t\rangle^{\otimes W}$, $h_t = \mathcal{W}(|D^t\rangle^{\otimes W})$, and (implicitly) compute ℓ^t ;

  if $t \pmod K = 0$ **then**

    $M^{t+1} = \widetilde{\text{Proj}}_{\Gamma_\epsilon}(M^t(1-\gamma)^{\ell^t})$; // Approx projection

  else

    $M^{t+1} = M^t(1-\gamma)^{\ell^t}$; // Lazy update

Output: $H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x))$;

Technique 1: Implicit Weights

- Avoid updating m weights explicitly. Evaluate entries on the fly using prior hypotheses $h_{<t}$.
- Prepare $|D^t\rangle^{\otimes W}$ using $\tilde{O}(W/\sqrt{\epsilon})$ q-queries.

Technique 2: Lazy projection

QuantumBoost: A quantized lazy SmoothBoost

Algorithm: QuantumBoost

Input : $\epsilon > 0$; $S = \{(x_i, y_i)\}_{i=1}^m$; γ-weak learner $\mathcal{W}$

Initialize $M_i^1 \leftarrow \epsilon \; \forall i \in [m]$; $K = 1/\gamma$;

for $t = 1, \ldots, T$ **do**

  Prepare $|D^t\rangle^{\otimes W}$, $h_t = \mathcal{W}(|D^t\rangle^{\otimes W})$, and (implicitly) compute ℓ^t ;

  if $t \pmod K = 0$ **then**

    $M^{t+1} = \widetilde{\text{Proj}}_{\Gamma_\epsilon}(M^t(1-\gamma)^{\ell^t})$; // Approx projection

  else

    $M^{t+1} = M^t(1-\gamma)^{\ell^t}$; // Lazy update

Output: $H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x))$;

Technique 1: Implicit Weights

- Avoid updating m weights explicitly. Evaluate entries on the fly using prior hypotheses $h_{<t}$.
- Prepare $|D^t\rangle^{\otimes W}$ using $\tilde{O}(W/\sqrt{\epsilon})$ q-queries.

Technique 2: Lazy projection

- Project back to high-density measures every $K \approx 1/\gamma$ iterations to amortize overhead.

Summary and Comparison

Summary and Comparison

QuantumBoost achieves the fastest known total runtime

Summary and Comparison

QuantumBoost achieves the fastest known total runtime

Boosting Algorithm	Total Runtime	Iterations (T)	Ref.
1. AdaBoost	$\frac{W}{\gamma^2} + \frac{d}{\epsilon\gamma^4}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[FS97]
2. Quantum AdaBoost	$\frac{W^{1.5}\sqrt{d}}{\epsilon\gamma^{11}}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[AM20]
3. SmoothBoost	$\frac{W}{\epsilon^2\gamma^2} + \frac{d}{\epsilon^4\gamma^4}$	$\frac{1}{\epsilon\gamma^2}$	[Ser03]
4. Quantum SmoothBoost	$\frac{W}{\epsilon^{2.5}\gamma^4} + \frac{\sqrt{d}}{\epsilon^{3.5}\gamma^5}$	$\frac{1}{\epsilon\gamma^2}$	[IdW23]
5. Kale's SmoothBoost	$\frac{W}{\epsilon\gamma^2} + \frac{d}{\gamma^4} + \frac{1}{\epsilon\gamma^6}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[Kal07, BHK09]
6. **QuantumBoost**	$\mathbf{O}\left(\frac{\mathbf{W}}{\sqrt{\epsilon}\gamma^{\mathbf{4}}}\right)$	$\frac{\log(\mathbf{1}/\epsilon)}{\gamma^{\mathbf{2}}}$	**This work**

Summary and Comparison

QuantumBoost achieves the fastest known total runtime

Boosting Algorithm	Total Runtime	Iterations (T)	Ref.
1. AdaBoost	$\frac{W}{\gamma^2} + \frac{d}{\epsilon\gamma^4}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[FS97]
2. Quantum AdaBoost	$\frac{W^{1.5}\sqrt{d}}{\epsilon\gamma^{11}}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[AM20]
3. SmoothBoost	$\frac{W}{\epsilon^2\gamma^2} + \frac{d}{\epsilon^4\gamma^4}$	$\frac{1}{\epsilon\gamma^2}$	[Ser03]
4. Quantum SmoothBoost	$\frac{W}{\epsilon^{2.5}\gamma^4} + \frac{\sqrt{d}}{\epsilon^{3.5}\gamma^5}$	$\frac{1}{\epsilon\gamma^2}$	[IdW23]
5. Kale's SmoothBoost	$\frac{W}{\epsilon\gamma^2} + \frac{d}{\gamma^4} + \frac{1}{\epsilon\gamma^6}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[Kal07, BHK09]
6. **QuantumBoost**	$\mathbf{O}\left(\frac{\mathbf{W}}{\sqrt{\epsilon}\gamma^{\mathbf{4}}}\right)$	$\frac{\log(\mathbf{1}/\epsilon)}{\gamma^{\mathbf{2}}}$	**This work**

Future work

Summary and Comparison

QuantumBoost achieves the fastest known total runtime

Boosting Algorithm	Total Runtime	Iterations (T)	Ref.
1. AdaBoost	$\frac{W}{\gamma^2} + \frac{d}{\epsilon\gamma^4}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[FS97]
2. Quantum AdaBoost	$\frac{W^{1.5}\sqrt{d}}{\epsilon\gamma^{11}}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[AM20]
3. SmoothBoost	$\frac{W}{\epsilon^2\gamma^2} + \frac{d}{\epsilon^4\gamma^4}$	$\frac{1}{\epsilon\gamma^2}$	[Ser03]
4. Quantum SmoothBoost	$\frac{W}{\epsilon^{2.5}\gamma^4} + \frac{\sqrt{d}}{\epsilon^{3.5}\gamma^5}$	$\frac{1}{\epsilon\gamma^2}$	[IdW23]
5. Kale's SmoothBoost	$\frac{W}{\epsilon\gamma^2} + \frac{d}{\gamma^4} + \frac{1}{\epsilon\gamma^6}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[Kal07, BHK09]
6. **QuantumBoost**	$\mathbf{Q}\left(\frac{\mathbf{W}}{\sqrt{\epsilon}\gamma^{\mathbf{4}}}\right)$	$\frac{\log(\mathbf{1}/\epsilon)}{\gamma^{\mathbf{2}}}$	**This work**

Future work

- Further applications in cryptography and optimization?

Summary and Comparison

QuantumBoost achieves the fastest known total runtime

Boosting Algorithm	Total Runtime	Iterations (T)	Ref.
1. AdaBoost	$\frac{W}{\gamma^2} + \frac{d}{\epsilon\gamma^4}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[FS97]
2. Quantum AdaBoost	$\frac{W^{1.5}\sqrt{d}}{\epsilon\gamma^{11}}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[AM20]
3. SmoothBoost	$\frac{W}{\epsilon^2\gamma^2} + \frac{d}{\epsilon^4\gamma^4}$	$\frac{1}{\epsilon\gamma^2}$	[Ser03]
4. Quantum SmoothBoost	$\frac{W}{\epsilon^{2.5}\gamma^4} + \frac{\sqrt{d}}{\epsilon^{3.5}\gamma^5}$	$\frac{1}{\epsilon\gamma^2}$	[IdW23]
5. Kale's SmoothBoost	$\frac{W}{\epsilon\gamma^2} + \frac{d}{\gamma^4} + \frac{1}{\epsilon\gamma^6}$	$\frac{\log(1/\epsilon)}{\gamma^2}$	[Kal07, BHK09]
6. **QuantumBoost**	$\mathbf{O}\left(\frac{\mathbf{W}}{\sqrt{\epsilon}\gamma^{\mathbf{4}}}\right)$	$\frac{\log(\mathbf{1}/\epsilon)}{\gamma^{\mathbf{2}}}$	**This work**

Future work

- Further applications in cryptography and optimization?
- Improve γ-dependency from $1/\gamma^4$ to $1/\gamma^2$?