No Quantum Speedup over Gradient Descent

Lower Bounds for Convex Optimization

Ankit Garg [1] Robin Kothari [2] Praneeth Netrapalli [1] Suhail Sherif [1,3]

[1] Microsoft Research India

[2] Microsoft Quantum and Microsoft Research

[3] Tata Institute of Fundamental Research

Gradient Descent

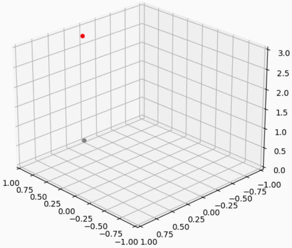
Gradient Descent

The Gradient Descent method [Cauchy ‘47]

Gradient Descent

The Gradient Descent method [Cauchy '47]

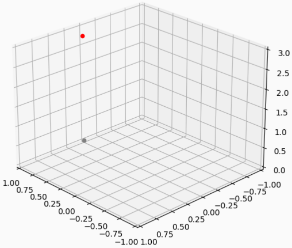
- Compute the function at a point.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

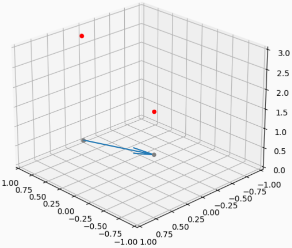
- Compute the function at a point.
- Find the gradient at that point.



Gradient Descent

The Gradient Descent method [Cauchy '47]

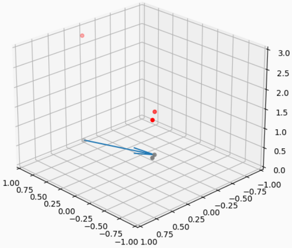
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.



Gradient Descent

The Gradient Descent method [Cauchy '47]

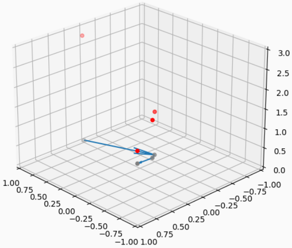
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

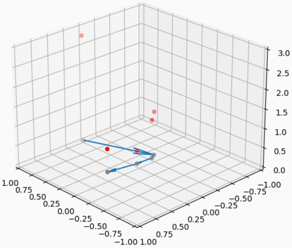
The Gradient Descent method [Cauchy '47]

- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



The Gradient Descent method [Cauchy '47]

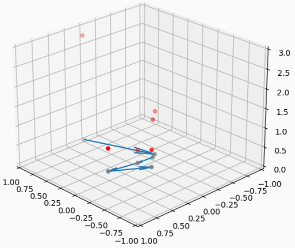
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy '47]

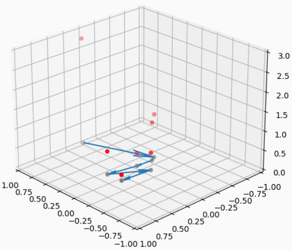
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy '47]

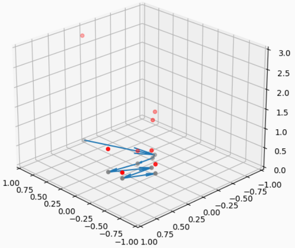
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy '47]

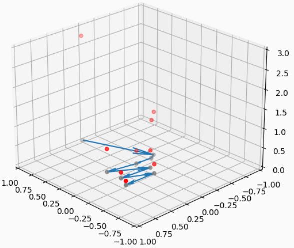
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy '47]

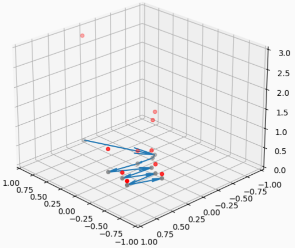
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

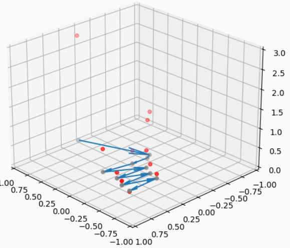
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy '47]

- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.

- Used in practice.

Gradient Descent

The Gradient Descent method [Cauchy '47]

- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.

- Used in practice.
    - Cheap Gradient Principle [GW '08]
      Compute $f(x), \nabla f(x)$ in time linear in computing $f(x)$.

Gradient Descent

The Gradient Descent method [Cauchy '47]

- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.

- Used in practice.
  - Cheap Gradient Principle [GW '08]
    Compute $f(x), \nabla f(x)$ in time linear in computing $f(x)$.
  - Finding parameters that decrease the loss function.

Gradient Descent

The Gradient Descent method [Cauchy ‘47]

- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.

- Used in practice.
  - Cheap Gradient Principle [GW ‘08]
    Compute $f(x), \nabla f(x)$ in time linear in computing $f(x)$.
  - Finding parameters that decrease the loss function.
- Abstracting out Gradient Descent:
  - Function value oracle
  - Function gradient oracle

Can quantum "do gradient descent" faster?

- Can find gradient with 1 query to function oracle. [Jordan '05]

Can quantum "do gradient descent" faster?

- Can find gradient with 1 query to function oracle. [Jordan '05]
- Hope: Use less than k queries to get the result of k gradient descent steps?

Can quantum "do gradient descent" faster?

- Can find gradient with 1 query to function oracle. [Jordan '05]
- Hope: Use less than k queries to get the result of k gradient descent steps?
  - Would join the ranks of unstructured search, period finding, vector reconstruction etc.

Can quantum "do gradient descent" faster?

- Can find gradient with 1 query to function oracle. [Jordan '05]
- Hope: Use less than k queries to get the result of k gradient descent steps?
  - Would join the ranks of unstructured search, period finding, vector reconstruction etc.

Can quantum "do gradient descent" faster?

- Can find gradient with 1 query to function oracle. [Jordan '05]
- Hope: Use less than k queries to get the result of k gradient descent steps?
  - Would join the ranks of unstructured search, period finding, vector reconstruction etc.

For negative result, need to reformulate our question.

Can quantum "do gradient descent" faster?

- Can find gradient with 1 query to function oracle. [Jordan '05]
- Hope: Use less than k queries to get the result of k gradient descent steps?
  - Would join the ranks of unstructured search, period finding, vector reconstruction etc.

For negative result, need to reformulate our question.
Use First-Order Convex Optimization as a proxy for Gradient Descent.

First-Order Convex Optimization

The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

On input x, oracle O_f returns $f(x), \nabla f(x)$.

The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

Find $x^* = \arg\min_{x \in B} f(x)$.

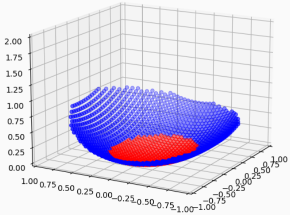
The Task

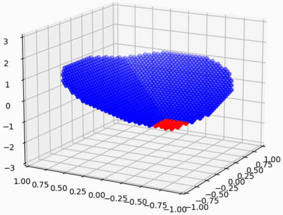
Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

Find $x' \in B$ s.t. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

Find $x' \in B$ s.t. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.

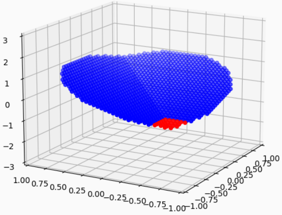


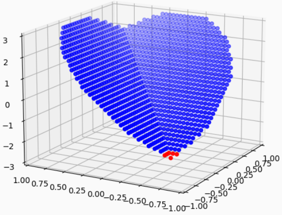


The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

$$\text{Find } x' \in B \text{ s.t. } f(x') \leq \min_{x \in B} f(x) + \epsilon.$$





The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

Find $x' \in B$ s.t. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.

ϵ-optimal for G-Lipschitz function in ball of radius R

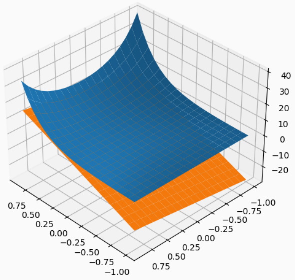
$\iff$

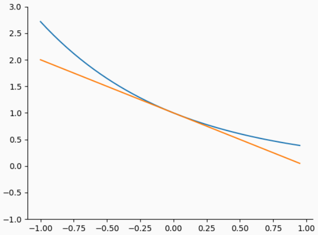
ϵ/GR-optimal for 1-Lipschitz function in ball of radius 1

The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

$$\text{Find } x' \in B \text{ s.t. } f(x') \leq \min_{x \in B} f(x) + \epsilon.$$

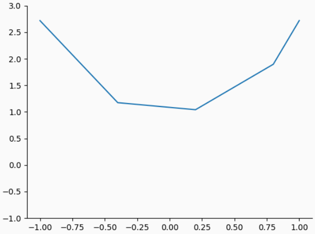




The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

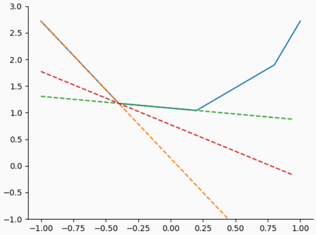
Find $x' \in B$ s.t. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.



The Task

Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

Find $x' \in B$ s.t. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.



$g \in \nabla f(x) \Leftrightarrow f(x+v) \geq f(x) + \langle v, g \rangle$ for all v

Known Algorithms

Center of Gravity Method

Known Algorithms

Center of Gravity Method

$n \log(1/\epsilon)$ steps.

Known Algorithms

Center of Gravity Method

$n \log(1/\epsilon)$ steps.

Projected Subgradient Descent

Known Algorithms

Center of Gravity Method

$n \log(1/\epsilon)$ steps.

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Known Algorithms

Center of Gravity Method

$n \log(1/\epsilon)$ steps.

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$

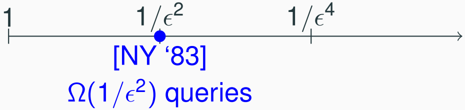
Classical: Deterministic

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$



Classical: Deterministic

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$

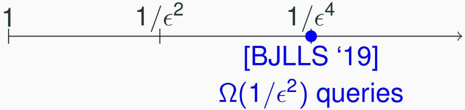
Classical: Randomized

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$



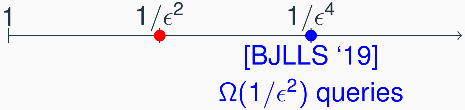
Classical: Randomized

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$



Classical: Randomized

Theorem (Garg-Kothari-Netrapalli-S. ‘20)

For any $\epsilon > 0$, there is a family of 1-Lipschitz functions $\{f : \mathbb{R}^n \rightarrow \mathbb{R}\}$ with $n = \Theta(1/\epsilon^2)$ such that any randomized algorithm solving first-order convex optimization on these requires $\Omega(1/\epsilon^2)$ queries.

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$

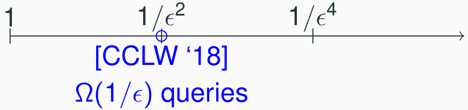
Quantum

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$



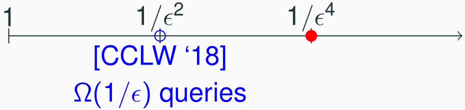
Quantum

Known Algorithms

Projected Subgradient Descent: $1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \rightarrow$



Quantum

Theorem (Garg-Kothari-Netrapalli-S. '20)

For any $\epsilon > 0$, there is a family of 1-Lipschitz functions $\{f : \mathbb{R}^n \rightarrow \mathbb{R}\}$ with $n = \tilde{\Theta}(1/\epsilon^4)$ such that any quantum algorithm solving first-order convex optimization on these requires $\Omega(1/\epsilon^2)$ queries.

Known Algorithms

Theorem (Garg-Kothari-Netrapalli-S. '20)

For any $\epsilon > 0$, there is a family of 1-Lipschitz functions $\{f : \mathbb{R}^n \to \mathbb{R}\}$ with $n = \Theta(1/\epsilon^2)$ such that any randomized algorithm solving first-order convex optimization on these requires $\Omega(1/\epsilon^2)$ queries.

Theorem (Garg-Kothari-Netrapalli-S. '20)

For any $\epsilon > 0$, there is a family of 1-Lipschitz functions $\{f : \mathbb{R}^n \to \mathbb{R}\}$ with $n = \tilde{\Theta}(1/\epsilon^4)$ such that any quantum algorithm solving first-order convex optimization on these requires $\Omega(1/\epsilon^2)$ queries.

Lower Bounds

Theme for Lower Bounds

Theme for Lower Bounds

- Find functions that encode information.

Theme for Lower Bounds

- Find functions that encode information.
- ϵ-minimizing $f \implies$ Learning the encoded information.

Theme for Lower Bounds

- Find functions that encode information.
- ϵ-minimizing $f \implies$ Learning the encoded information.
- Each query should access the information in a controlled manner.

Lower Bounds

Randomized Lower Bound

The Base Function

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x) = \max\{x_1, x_2, \ldots, x_n\}.$$

The Base Function

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x) = \max\{x_1, x_2, \ldots, x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$,

The Base Function

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x) = \max\{x_1, x_2, \ldots, x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{1}{\sqrt{n}}, \ldots, -\frac{1}{\sqrt{n}}\right)$.

The Base Function

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x) = \max\{x_1, x_2, \ldots, x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{1}{\sqrt{n}}, \ldots, -\frac{1}{\sqrt{n}}\right)$.

If x_i is a maximum, then e_i is a subgradient.

Function Class

$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Function Class

$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$,

Function Class

$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right)$.

Function Class

$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right)$.

Set $\epsilon = \frac{.9}{\sqrt{n}}$.

Function Class

$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right)$.

Set $\epsilon = \frac{.9}{\sqrt{n}}$.

The behaviour of f

z_1 z_2 z_3 z_4 z_n

Function Class

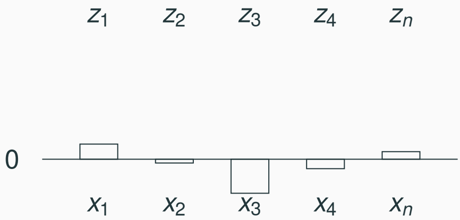
$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

$$\text{Minimum} = -\frac{1}{\sqrt{n}}, \text{ at } x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right).$$

$$\text{Set } \epsilon = \frac{.9}{\sqrt{n}}.$$

The behaviour of f



Function Class

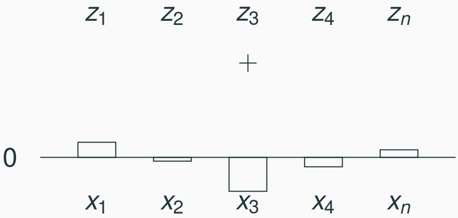
$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

$$\text{Minimum} = -\frac{1}{\sqrt{n}}, \text{ at } x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right).$$

$$\text{Set } \epsilon = \frac{.9}{\sqrt{n}}.$$

The behaviour of f



Function Class

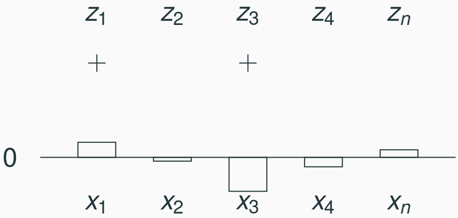
$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

$$\text{Minimum} = -\frac{1}{\sqrt{n}}, \text{ at } x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right).$$

$$\text{Set } \epsilon = \frac{.9}{\sqrt{n}}.$$

The behaviour of f



Function Class

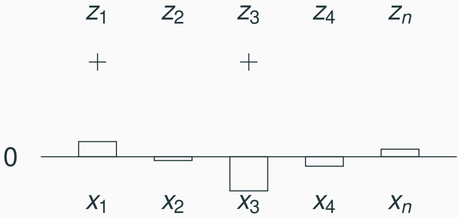
$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right)$.

Set $\epsilon = \frac{.9}{\sqrt{n}}$.

The behaviour of f



$\approx$ 2 bits of z revealed per query.

Function Class

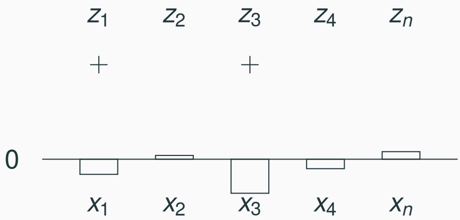
$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

$$\text{Minimum} = -\frac{1}{\sqrt{n}}, \text{ at } x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right).$$

$$\text{Set } \epsilon = \frac{.9}{\sqrt{n}}.$$

The behaviour of f



Function Class

$$z \in \{+1, -1\}^n$$

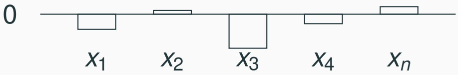
$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right)$.

Set $\epsilon = \frac{.9}{\sqrt{n}}$.

The behaviour of f

z_1	z_2	z_3	z_4	z_n
$+$	$-$	$+$	$+$	$-$



Function Class

$$z \in \{+1, -1\}^n$$

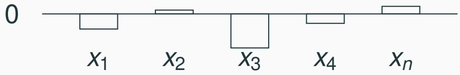
$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

$$\text{Minimum} = -\frac{1}{\sqrt{n}}, \text{ at } x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right).$$

$$\text{Set } \epsilon = \frac{.9}{\sqrt{n}}.$$

The behaviour of f

z_1	z_2	z_3	z_4	z_n
$+$	$-$	$+$	$+$	$-$



Finding ϵ-optimal point $\implies$ learning z.

Function Class

$$z \in \{+1, -1\}^n$$

$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

Minimum = $-\frac{1}{\sqrt{n}}$, at $x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right)$.

Set $\epsilon = \frac{.9}{\sqrt{n}}$.

Requires $\Omega(n) = \Omega(1/\epsilon^2)$ queries.

Quantum Speedup

A quantum algorithm can ϵ-optimize the previous function class in $O(\sqrt{n})$ queries.

Quantum Speedup

A quantum algorithm can ϵ-optimize the previous function class in $O(\sqrt{n})$ queries.

For any $S \subseteq [n]$, can find x such that

$$f_z(x) = 1 \text{ iff } \bigvee_{i \in S} z_i = +$$

Quantum Speedup

A quantum algorithm can ϵ-optimize the previous function class in $O(\sqrt{n})$ queries.

For any $S \subseteq [n]$, can find x such that

$$f_z(x) = 1 \text{ iff } \bigvee_{i \in S} z_i = +$$

Can then use Belovs' algorithm to learn z from such OR queries.

Lower Bounds

Quantum Lower Bound

The Query Model

We are provided with an oracle O_f to access the unknown function f.

The Query Model

We are provided with an oracle O_f to access the unknown function f.

- Input register *INPUT* with orthogonal states $\{|x\rangle\}_{x \in \mathbb{R}^n}$

The Query Model

We are provided with an oracle O_f to access the unknown function f.

- Input register *INPUT* with orthogonal states $\{|x\rangle\}_{x \in \mathbb{R}^n}$ (rather for a discretization of $\mathbb{R}^n$)

The Query Model

We are provided with an oracle O_f to access the unknown function f.

- Input register *INPUT* with orthogonal states $\{|x\rangle\}_{x \in \mathbb{R}^n}$ (rather for a discretization of $\mathbb{R}^n$)
- Oracle O_f ‘answers’ function value and subgradient queries.

The Query Model

We are provided with an oracle O_f to access the unknown function f.

- Input register *INPUT* with orthogonal states $\{|x\rangle\}_{x\in\mathbb{R}^n}$ (rather for a discretization of $\mathbb{R}^n$)
- Oracle O_f 'answers' function value and subgradient queries. Usually

$$O_f|x\rangle_{INPUT}|b\rangle_{OUTPUT} = |x\rangle_{INPUT}|b \oplus \text{“}f(x), \nabla f(x)\text{”}\rangle_{OUTPUT}$$

The Query Model

We are provided with an oracle O_f to access the unknown function f.

- Input register *INPUT* with orthogonal states $\{|x\rangle\}_{x \in \mathbb{R}^n}$ (rather for a discretization of $\mathbb{R}^n$)
- Oracle O_f ‘answers’ function value and subgradient queries. Usually

$$O_f |x\rangle_{INPUT} |b\rangle_{OUTPUT} = |x\rangle_{INPUT} |b \oplus \text{“}f(x), \nabla f(x)\text{”}\rangle_{OUTPUT}$$

- For f, f' s.t. $f(x) = f'(x)$ and $\nabla f(x) = \nabla f'(x)$:

$$O_f |x\rangle_{INPUT} |\phi\rangle_{REST} = O_{f'} |x\rangle_{INPUT} |\phi\rangle_{REST}$$

The Base Function

"Complexity of Highly Parallel Non-Smooth Convex Optimization"
- Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, Aaron Sidford

The Base Function

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x) = \max\{x_1, x_2 - \gamma, x_3 - 2\gamma, \ldots, x_k - (k-1)\gamma\}.$$

γ is small.

The Base Function

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(x) = \max\{x_1, x_2 - \gamma, x_3 - 2\gamma, \ldots, x_k - (k-1)\gamma\}.$$

γ is small.

Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx \left(-\frac{1}{\sqrt{k}}, \ldots, -\frac{1}{\sqrt{k}}, 0, 0, \cdots\right)$.

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

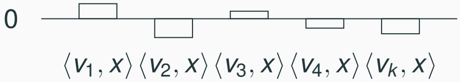
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

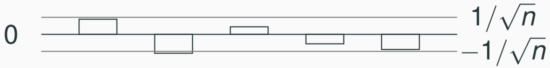
$$\text{Minimum} \approx -\frac{1}{\sqrt{k}}, \text{ at } x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}.$$

$$\text{Set } \epsilon = \frac{.9}{\sqrt{k}}.$$

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



The Function Class

$V = (v_1, v_2, \dots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \dots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

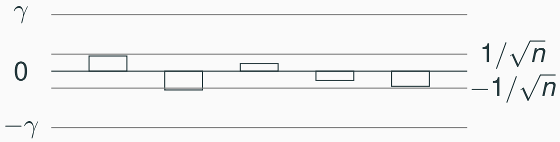
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \dots, v_k$ be orthonormal vectors sampled uniformly at random.



The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

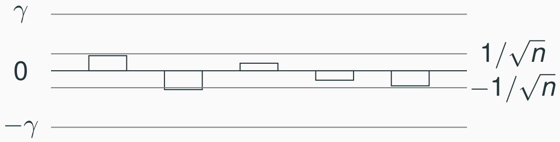
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



Whp, first query reveals v_1.

v_2 through v_k nearly random from $n-1$ dimensional space.

The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

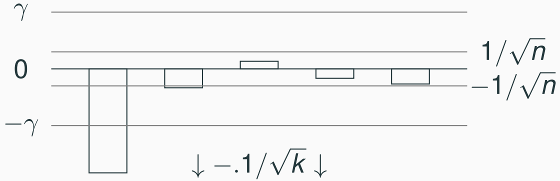
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

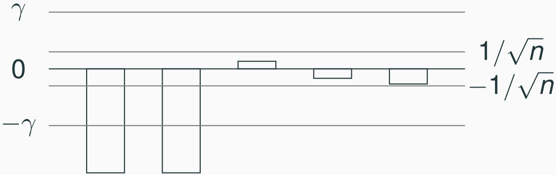
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

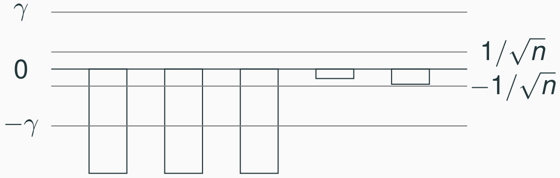
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



The Function Class

$V = (v_1, v_2, \ldots, v_k)$ is a set of k orthonormal vectors in $\mathbb{R}^n$.

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

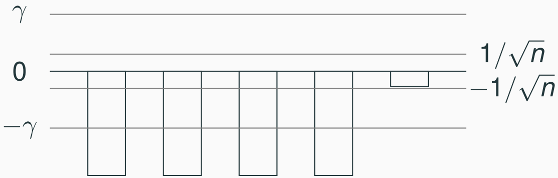
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

Set $\epsilon = \frac{.9}{\sqrt{k}}$.

The behaviour of f

Let $x \in \mathbb{R}^n$ with $\|x\| = 1$.

Let $v_1, \ldots, v_k$ be orthonormal vectors sampled uniformly at random.



v_k still nearly at random from $n - k$ dimensional space.

Can't output ϵ-optimal point.

The Hybrid Argument

First
query

$$
\begin{aligned}
&|x_1\rangle|\phi_1\rangle \\
+&|x_2\rangle|\phi_2\rangle \\
+&|x_3\rangle|\phi_3\rangle \\
+&|x_4\rangle|\phi_4\rangle \\
+&|x_5\rangle|\phi_5\rangle \\
+&\cdots
\end{aligned}
$$

The Hybrid Argument

First query

$$|x_1\rangle|\phi_1\rangle$$
$$+|x_2\rangle|\phi_2\rangle$$
$$+|x_3\rangle|\phi_3\rangle$$
$$+|x_4\rangle|\phi_4\rangle$$
$$+|x_5\rangle|\phi_5\rangle$$
$$+\cdots$$

$$\xrightarrow{\text{pass through oracle for } f_V(x)}$$

The Hybrid Argument

First query

$$|x_1\rangle|\phi_1\rangle + |x_2\rangle|\phi_2\rangle + |x_3\rangle|\phi_3\rangle + |x_4\rangle|\phi_4\rangle + |x_5\rangle|\phi_5\rangle + \cdots$$

$$\xrightarrow{\text{pass through oracle for } f_V(x)}$$

First answer

$$|x_1\rangle|\psi_1\rangle + |x_2\rangle|\psi_2\rangle + |x_3\rangle|\psi_3\rangle + |x_4\rangle|\psi_4\rangle + |x_5\rangle|\psi_5\rangle + \cdots$$

The Hybrid Argument

First query

$$|x_1\rangle|\phi_1\rangle + |x_2\rangle|\phi_2\rangle + |x_3\rangle|\phi_3\rangle + |x_4\rangle|\phi_4\rangle + |x_5\rangle|\phi_5\rangle + \cdots$$

$$\xrightarrow{\text{pass through oracle for } f_{(v_1)}(x)=\langle v_1, x\rangle}$$

Corrupted answer

$$|x_1\rangle|\psi_1\rangle + |x_2\rangle|\psi_2\rangle + |x_3\rangle|\psi_3\rangle + |x_4\rangle|\psi_4'\rangle + |x_5\rangle|\psi_5\rangle + \cdots$$

The Hybrid Argument

First query		Corrupted answer
$\lvert x_1\rangle\lvert\phi_1\rangle$		$\lvert x_1\rangle\lvert\psi_1\rangle$
$+\lvert x_2\rangle\lvert\phi_2\rangle$		$+\lvert x_2\rangle\lvert\psi_2\rangle$
$+\lvert x_3\rangle\lvert\phi_3\rangle$	$\xrightarrow{\text{pass through oracle for } f_{(v_1)}(x)=\langle v_1,x\rangle}$	$+\lvert x_3\rangle\lvert\psi_3\rangle$
$+\lvert x_4\rangle\lvert\phi_4\rangle$		$+\lvert x_4\rangle\lvert\psi'_4\rangle$
$+\lvert x_5\rangle\lvert\phi_5\rangle$		$+\lvert x_5\rangle\lvert\psi_5\rangle$
$+\cdots$		$+\cdots$

- Changing oracle #1 barely changes the resulting state after 1 query. (with high probability)

The Hybrid Argument

First query		Corrupted answer
$\|x_1\rangle\|\phi_1\rangle$		$\|x_1\rangle\|\psi_1\rangle$
$+\|x_2\rangle\|\phi_2\rangle$		$+\|x_2\rangle\|\psi_2\rangle$
$+\|x_3\rangle\|\phi_3\rangle$	$\xrightarrow{\text{pass through oracle for } f_{(v_1)}(x)=\langle v_1, x\rangle}$	$+\|x_3\rangle\|\psi_3\rangle$
$+\|x_4\rangle\|\phi_4\rangle$		$+\|x_4\rangle\|\psi_4'\rangle$
$+\|x_5\rangle\|\phi_5\rangle$		$+\|x_5\rangle\|\psi_5\rangle$
$+\cdots$		$+\cdots$

- Changing oracle #1 barely changes the resulting state after 1 query. (with high probability)
- The actual, corrupted states at the end are also close.

The Hybrid Argument

First query		Corrupted answer
$\vert x_1\rangle\vert\phi_1\rangle$		$\vert x_1\rangle\vert\psi_1\rangle$
$+\vert x_2\rangle\vert\phi_2\rangle$		$+\vert x_2\rangle\vert\psi_2\rangle$
$+\vert x_3\rangle\vert\phi_3\rangle$	$\xrightarrow{\text{pass through oracle for } f_{(v_1)}(x)=\langle v_1,x\rangle}$	$+\vert x_3\rangle\vert\psi_3\rangle$
$+\vert x_4\rangle\vert\phi_4\rangle$		$+\vert x_4\rangle\vert\psi'_4\rangle$
$+\vert x_5\rangle\vert\phi_5\rangle$		$+\vert x_5\rangle\vert\psi_5\rangle$
$+\cdots$		$+\cdots$

- Changing oracle #1 barely changes the resulting state after 1 query. (with high probability)
- The actual, corrupted states at the end are also close.
- Actual, corrupted algorithm nearly the same.

The Hybrid Argument: Second query

Second query (corrupted)

$$
\begin{aligned}
&|x_1\rangle|\tau_1\rangle \\
+&|x_2\rangle|\tau_2\rangle \\
+&|x_3\rangle|\tau_3\rangle \\
+&|x_4\rangle|\tau_4\rangle \\
+&|x_5\rangle|\tau_5\rangle \\
+&\cdots
\end{aligned}
$$

The Hybrid Argument: Second query

Second query (corrupted)

$$
\begin{aligned}
&|x_1\rangle|\tau_1\rangle \\
+&|x_2\rangle|\tau_2\rangle \\
+&|x_3\rangle|\tau_3\rangle \\
+&|x_4\rangle|\tau_4\rangle \\
+&|x_5\rangle|\tau_5\rangle \\
+&\cdots
\end{aligned}
\qquad \xrightarrow{\text{pass through oracle for } f_V(x)}
$$

The Hybrid Argument: Second query

Second query (corrupted)

$$|x_1\rangle|\tau_1\rangle + |x_2\rangle|\tau_2\rangle + |x_3\rangle|\tau_3\rangle + |x_4\rangle|\tau_4\rangle + |x_5\rangle|\tau_5\rangle + \cdots$$

$$\xrightarrow{\text{pass through oracle for } f_V(x)}$$

Second answer (corrupted)

$$|x_1\rangle|\chi_1\rangle + |x_2\rangle|\chi_2\rangle + |x_3\rangle|\chi_3\rangle + |x_4\rangle|\chi_4\rangle + |x_5\rangle|\chi_5\rangle + \cdots$$

The Hybrid Argument: Second query

Second query (corrupted)

$$|x_1\rangle|\tau_1\rangle + |x_2\rangle|\tau_2\rangle + |x_3\rangle|\tau_3\rangle + |x_4\rangle|\tau_4\rangle + |x_5\rangle|\tau_5\rangle + \cdots$$

$$\xrightarrow{\text{pass through oracle for } f_{(v_1, v_2)}(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma\}}$$

Second answer (twice corrupted)

$$|x_1\rangle|\chi_1\rangle + |x_2\rangle|\chi_2'\rangle + |x_3\rangle|\chi_3\rangle + |x_4\rangle|\chi_4\rangle + |x_5\rangle|\chi_5\rangle + \cdots$$

The Hybrid Argument: Second query

Second query (corrupted)

$$|x_1\rangle|\tau_1\rangle + |x_2\rangle|\tau_2\rangle + |x_3\rangle|\tau_3\rangle + |x_4\rangle|\tau_4\rangle + |x_5\rangle|\tau_5\rangle + \cdots$$

pass through oracle for $f_{(v_1,v_2)}(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma\}$ $\longrightarrow$

Second answer (twice corrupted)

$$|x_1\rangle|\chi_1\rangle + |x_2\rangle|\chi_2'\rangle + |x_3\rangle|\chi_3\rangle + |x_4\rangle|\chi_4\rangle + |x_5\rangle|\chi_5\rangle + \cdots$$

- Changing oracle #2 barely changes the resulting state after 2 queries. (with high probability)

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.
- The $k-1$-times corrupted algorithm fails.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.
- The $k-1$-times corrupted algorithm fails.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.
- The $k-1$-times corrupted algorithm fails.

Success probability of the actual algorithm is also small.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.
- The $k-1$-times corrupted algorithm fails.

Success probability of the actual algorithm is also small.

Actual function used is slightly modified to account for queries outside B.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.
- The $k-1$-times corrupted algorithm fails.

Success probability of the actual algorithm is also small.

Actual function used is slightly modified to account for queries outside B.

n can be as small as $1/\epsilon^6$ for the above argument.

The Hybrid Argument: After $k-1$ queries

- Actual and $k-1$-times corrupted algorithms are nearly the same.
- The $k-1$-times corrupted state is independent of v_k given $v_1, \ldots, v_{k-1}$.
- The $k-1$-times corrupted algorithm fails.

Success probability of the actual algorithm is also small.

Actual function used is slightly modified to account for queries outside B.

n can be as small as $1/\epsilon^6$ for the above argument.

Modifications taken from Bubeck et al. can bring n down to $1/\epsilon^4$.

Accelerated Gradient Descent

When the function is guaranteed to not have a rapid change of slope, the optimal algorithm is Accelerated Gradient Descent.

Accelerated Gradient Descent

When the function is guaranteed to not have a rapid change of slope, the optimal algorithm is Accelerated Gradient Descent.

Accelerated Gradient Descent is also dimension-independent.

Accelerated Gradient Descent

When the function is guaranteed to not have a rapid change of slope, the optimal algorithm is Accelerated Gradient Descent.

Accelerated Gradient Descent is also dimension-independent.

Quantum can't do better here either.

Accelerated Gradient Descent

When the function is guaranteed to not have a rapid change of slope, the optimal algorithm is Accelerated Gradient Descent.

Accelerated Gradient Descent is also dimension-independent.

Quantum can't do better here either.

Similar proof to the one shown, but the function requires smoothing.

Open Problems

Quantum computers can't speed up gradient descent in general.
Yet...

- What is the quantum complexity of convex optimization in small dimensions?

Quantum computers can’t speed up gradient descent in general.
Yet...

- What is the quantum complexity of convex optimization in small dimensions?
- What other classes of convex optimization problems get quantum speedups?

Quantum computers can't speed up gradient descent in general.
Yet...

- What is the quantum complexity of convex optimization in small dimensions?
- What other classes of convex optimization problems get quantum speedups?
- What is the quantum complexity of optimizing the function class

$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle, \ldots, \langle v_k, x\rangle\}?$$