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] Vector Institute, Toronto

Gradient Descent

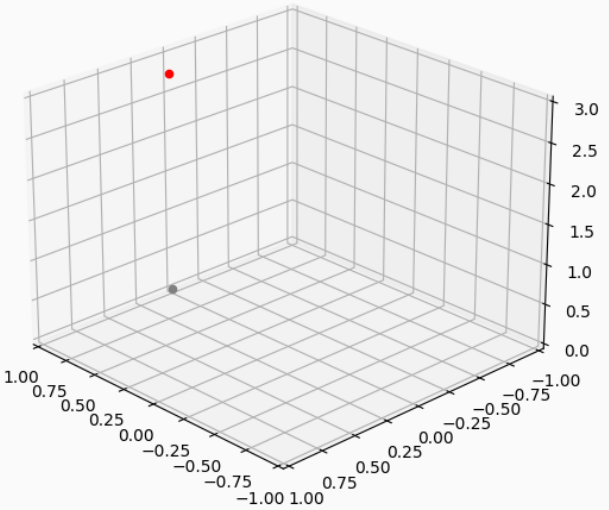
Gradient Descent

The Gradient Descent method [Cauchy ‘47]

Gradient Descent

The Gradient Descent method [Cauchy ‘47]

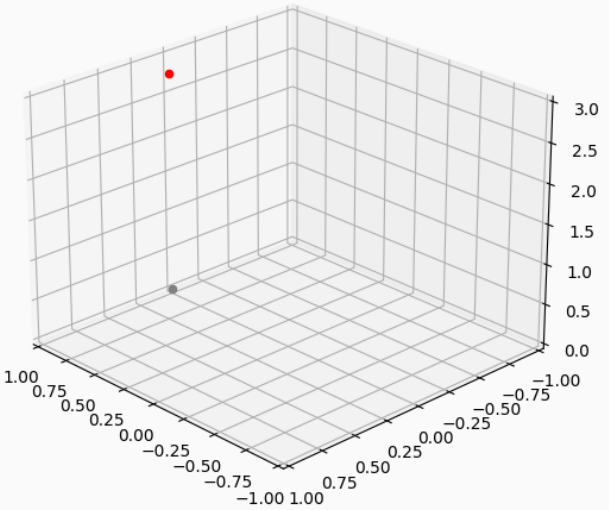
- Compute the function at a point.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

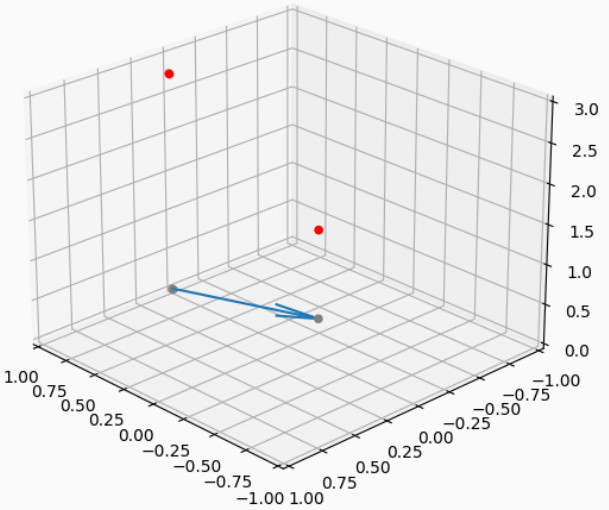
- Compute the function at a point.
- Find the gradient at that point.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

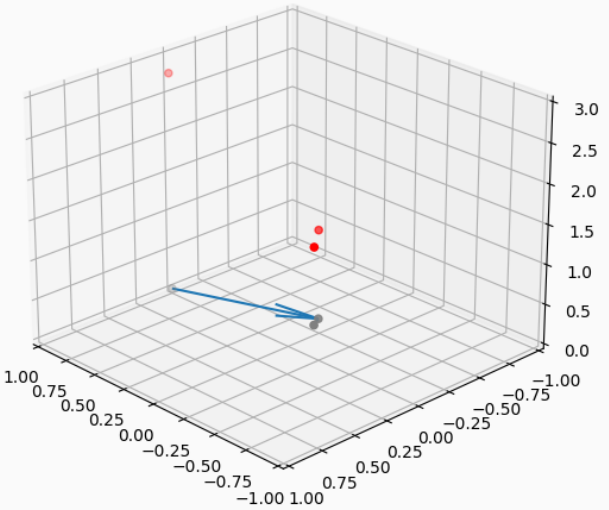
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

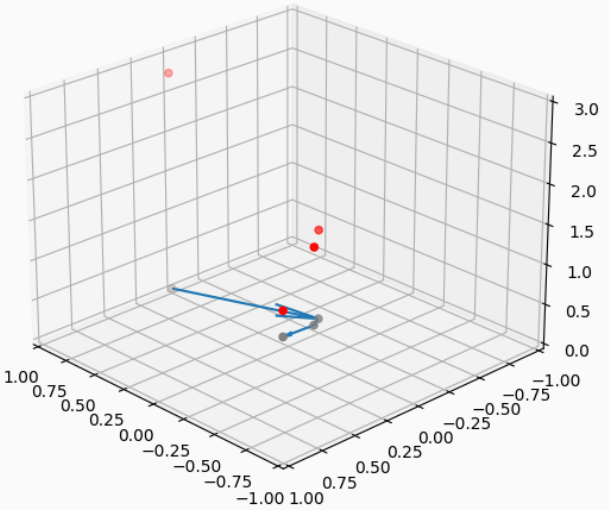
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

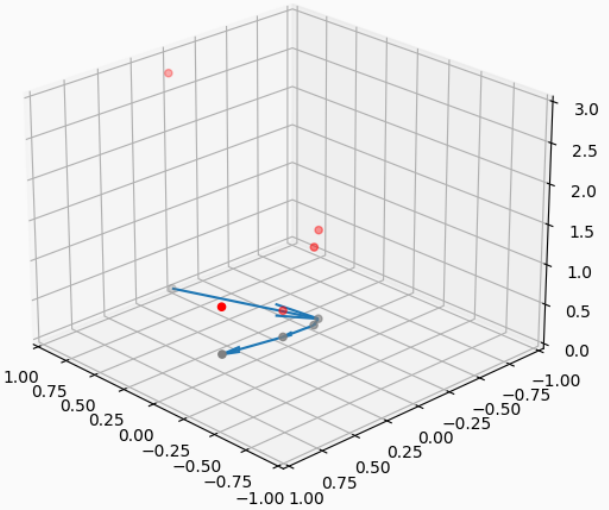
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

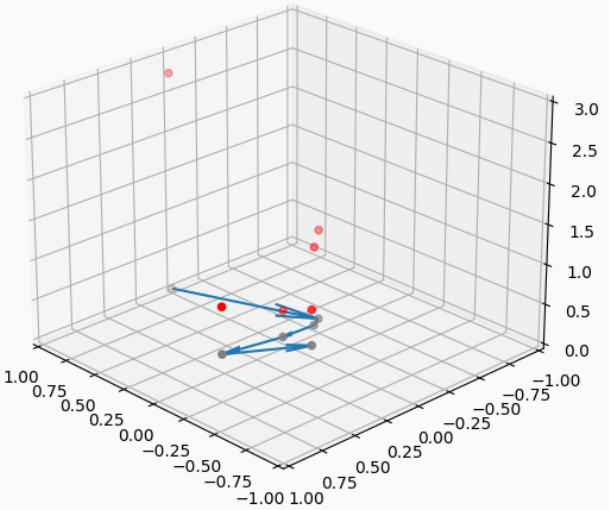
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

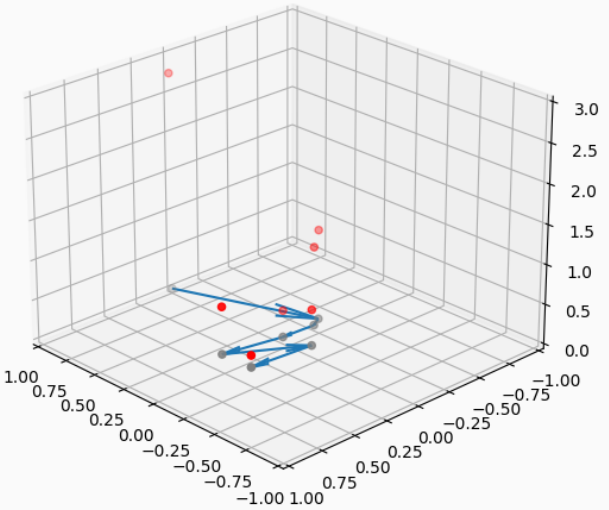
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

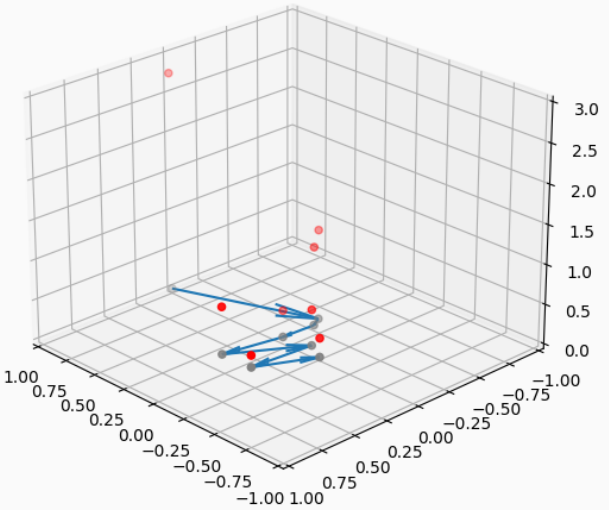
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

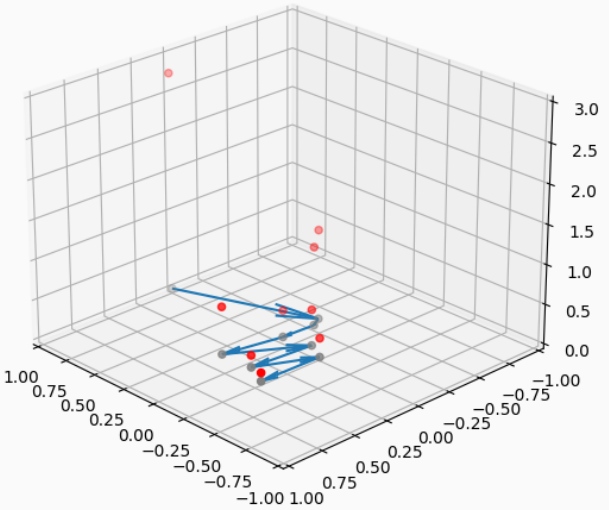
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

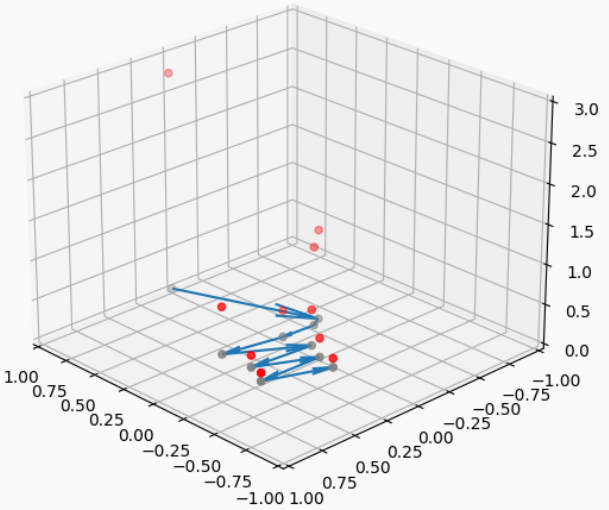
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



Gradient Descent

The Gradient Descent method [Cauchy ‘47]

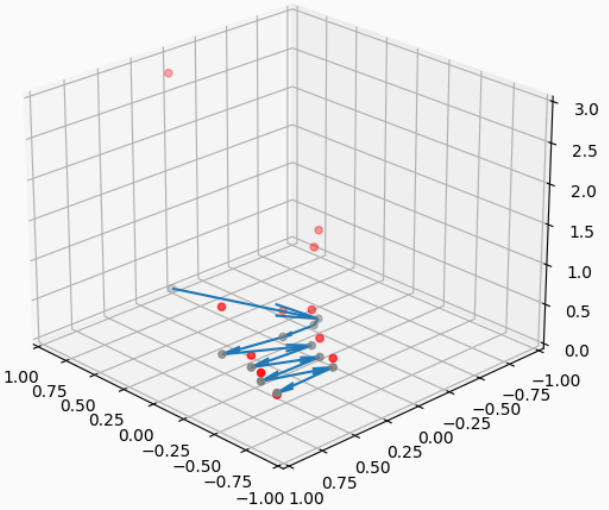
- Compute the function at a point.
- Find the gradient at that point.
- Take a step in the opposite direction.
- Repeat.



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.

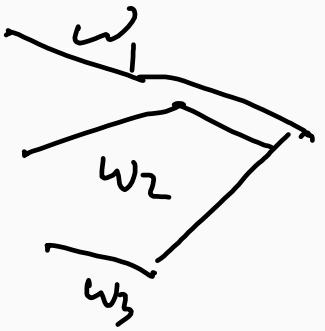


Gradient Descent

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.

- Magical performance in high dimensions.



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.

- Magical performance in high dimensions.
- Very useful for finding optimal parameters.

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.

- Magical performance in high dimensions.
- Very useful for finding optimal parameters.
- Finding gradients:

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.

- Magical performance in high dimensions.
- Very useful for finding optimal parameters.
- Finding gradients:
  - Hard: Requires $\sqrt{\text{dimension}}$ many function evaluations.

$f(x) = \langle x, v \rangle$ Gradient $= v$, $\|v\| = 1$

Query x_1 $\Rightarrow$ Get $\langle x_1, v \rangle$ w.h.p. $\frac{1}{\sqrt{n}}$

(n dim)

Need $\sqrt{n}$ queries to learn v.

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.

- Magical performance in high dimensions.
- Very useful for finding optimal parameters.
- Finding gradients:
  - Hard: Requires $\sqrt{\text{dimension}}$ many function evaluations.
  - Easy: 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.

- Magical performance in high dimensions.
- Very useful for finding optimal parameters.
- Finding gradients:
  - Hard: Requires $\sqrt{\text{dimension}}$ many function evaluations.
  - Easy: Cheap Gradient Principle [GW '08]
    Compute $f(x), \nabla f(x)$ in time linear in computing $f(x)$.
- Guarantees?

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

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

The Task

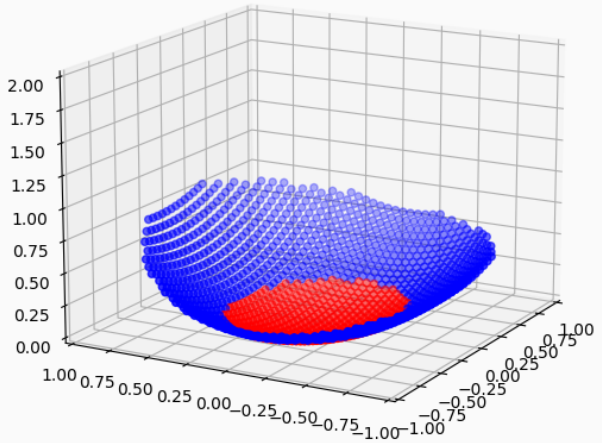
Given: a convex region B, first-order oracle access to a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

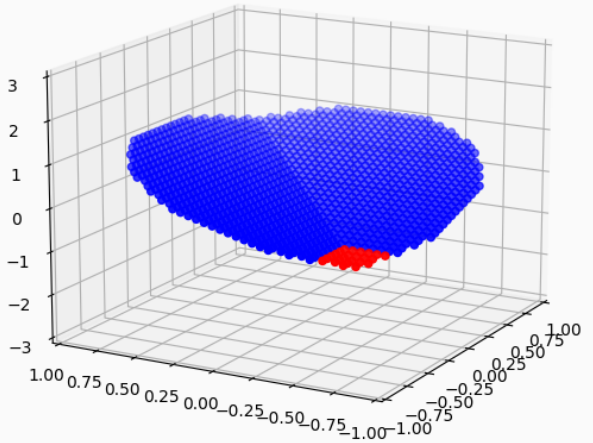
Find $x' \in B$ s.t. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.

The Task

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

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

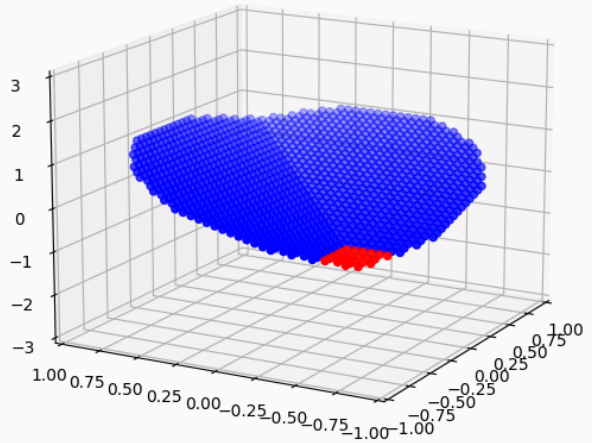




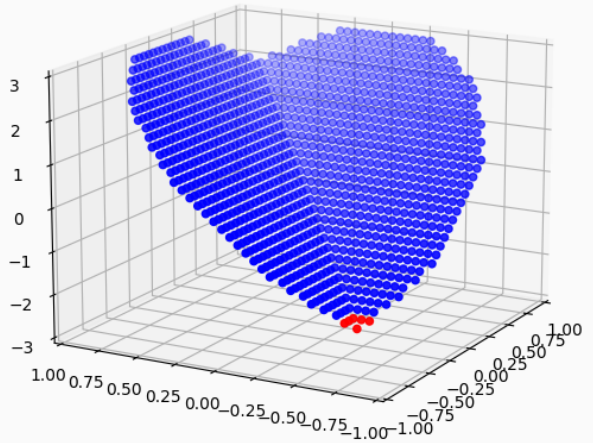
The Task

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

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



ϵ-opt



3ϵ-opt

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

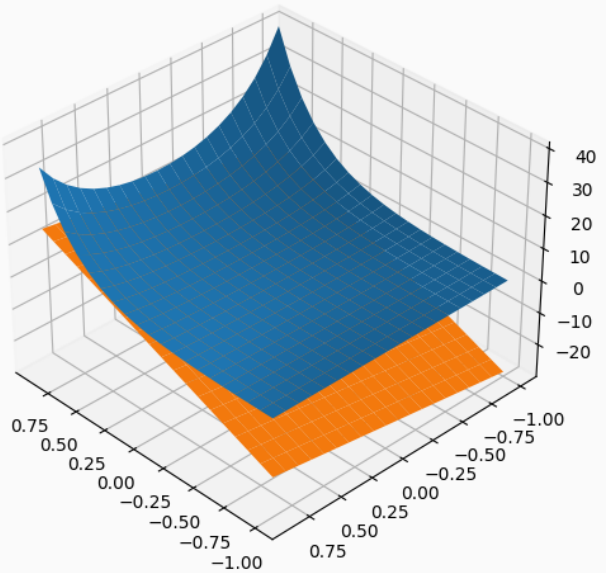
$\Longleftrightarrow$

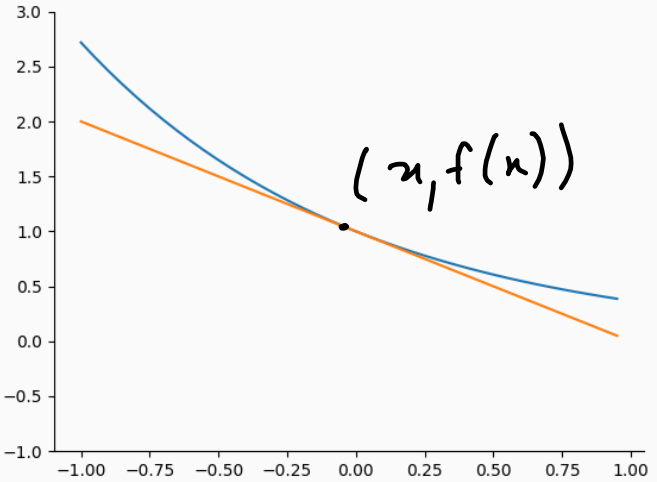
ϵ/GR-optimal for 1-Lipschitz function in ball of radius 1

The Task

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

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

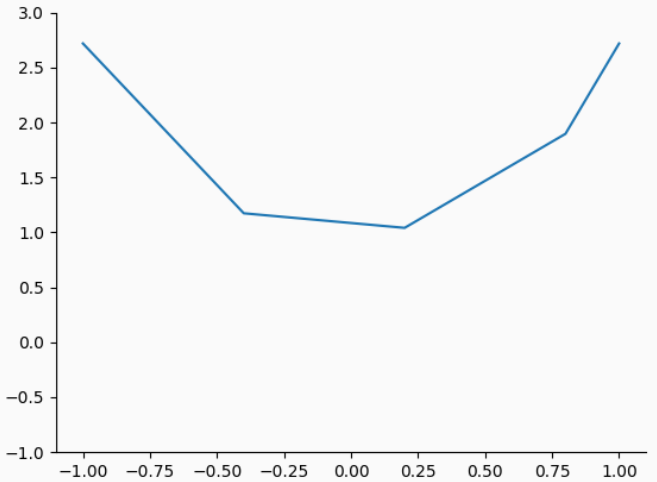




The Task

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

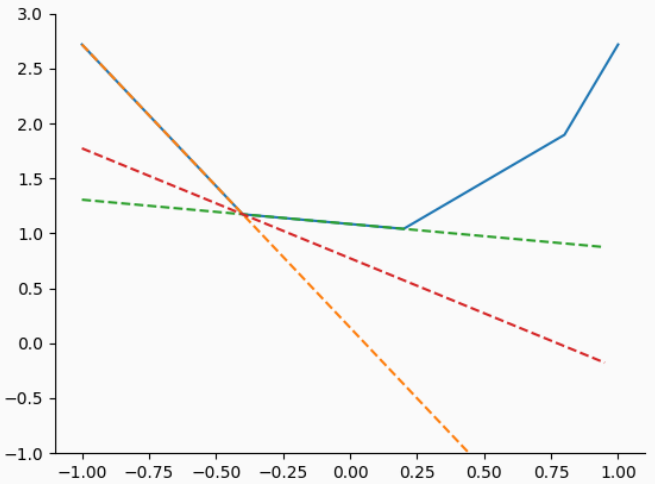
$$\text{Find } x' \in B \text{ s.t. } f(x') \leq \min_{x \in B} f(x) + \epsilon.$$



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. $f(x') \leq \min_{x \in B} f(x) + \epsilon$.



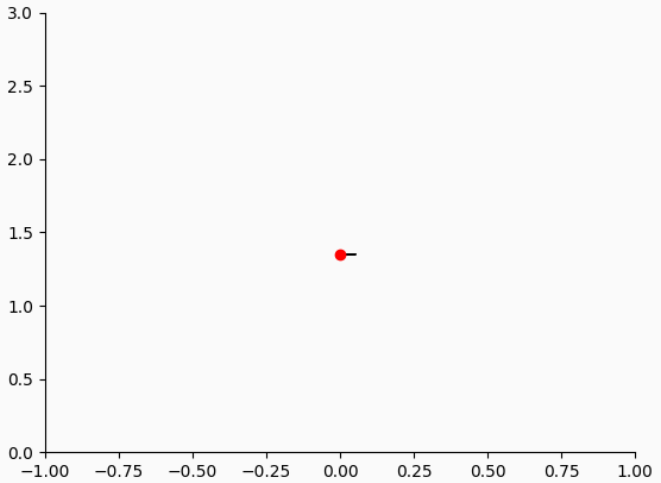
$g \in \nabla f(x) \Leftrightarrow f(x + v) \geq f(x) + \langle v, g \rangle$ for all v

Known Algorithms I

1-dimensional fn:

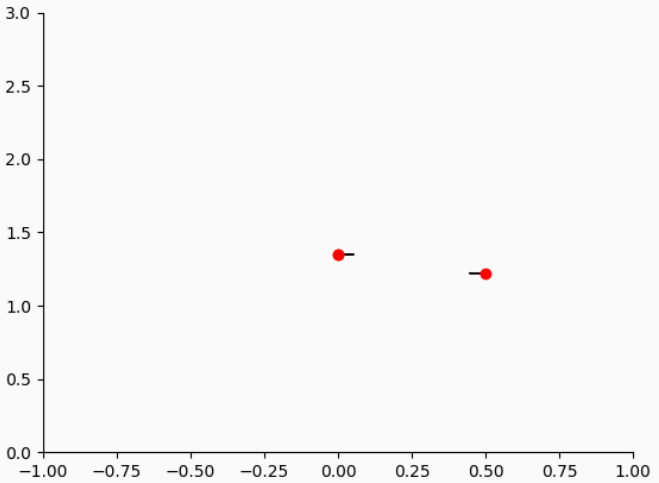
Known Algorithms I

1-dimensional fn:



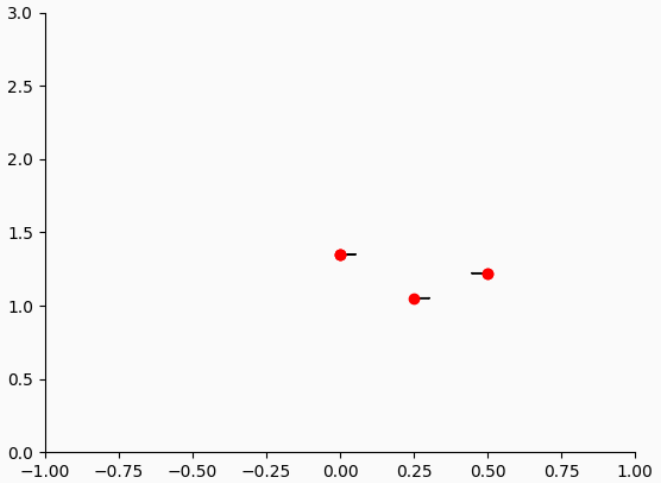
Known Algorithms I

1-dimensional fn:



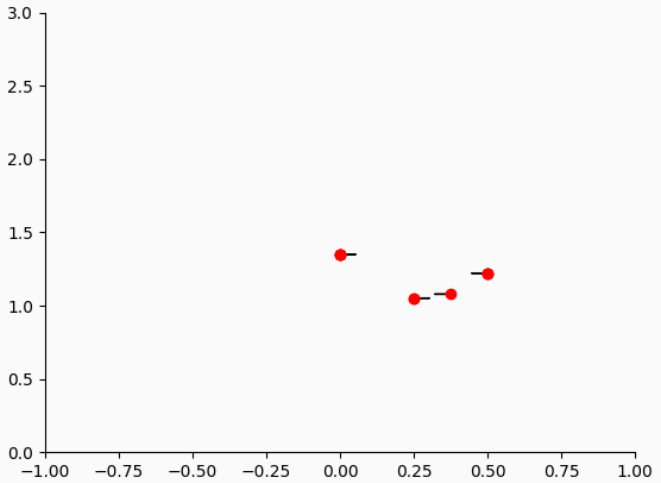
Known Algorithms I

1-dimensional fn:



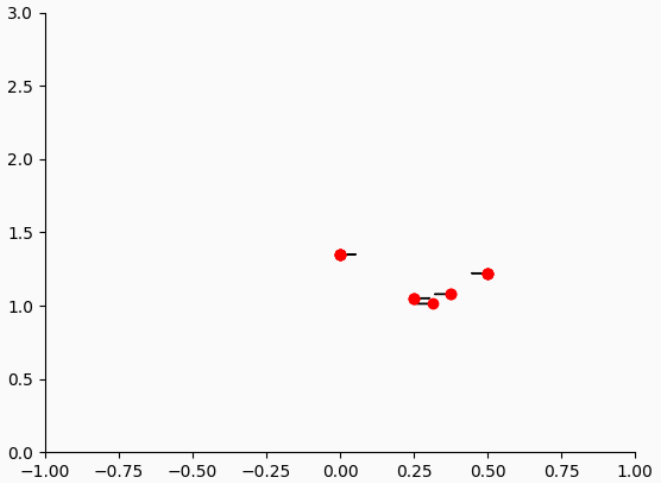
Known Algorithms I

1-dimensional fn:



Known Algorithms I

1-dimensional fn:



Known Algorithms I

1-dimensional fn:

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

Known Algorithms I

n-dimensional fn:

Known Algorithms I

n-dimensional fn:
(Center of Gravity Method)

Known Algorithms I

n-dimensional fn:
(Center of Gravity Method)

$$\approx \log\left(\frac{Vol(B(1))}{Vol(B(\epsilon))}\right)$$

$$= n\log(1/\epsilon) \text{ steps.}$$

Known Algorithms II

Center of Gravity Method

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

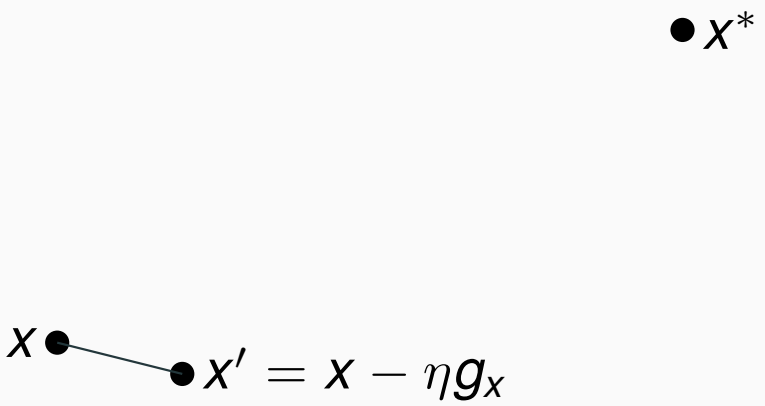
Projected Subgradient Descent

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

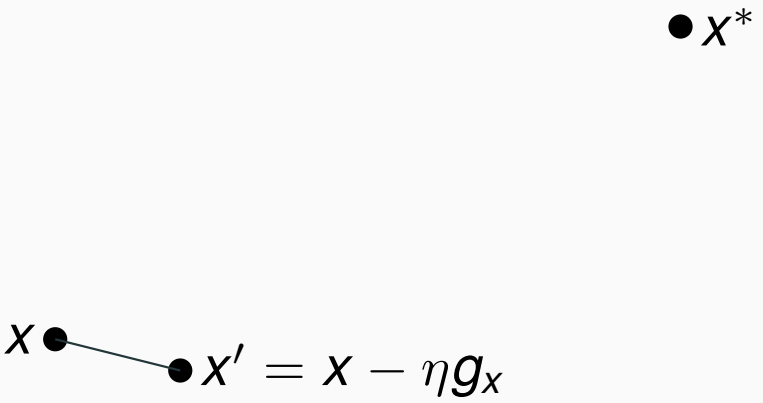


Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent



$$\langle -g_x, x^* - x \rangle \geq f(x) - f(x^*).$$

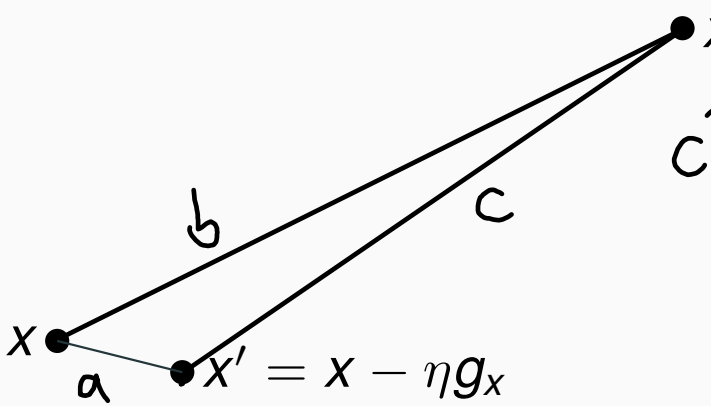
.

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent



$$c^2 = a^2 + b^2 - 2\langle \vec{a}, \vec{b} \rangle$$

$$\langle -g_x, x^* - x \rangle \geq f(x) - f(x^*).$$

$$\|x^* - x'\|^2 \leq \|x^* - x\|^2 + \eta^2 - 2\eta(f(x) - f(x^*)).$$

Set $\eta = \epsilon$ .. If $f(x) - f(x^*) \geq \epsilon$, decr. by ϵ^2

$\therefore$ in $\frac{1}{\epsilon^2}$ steps, reach an ϵ-opt pt.

Known Algorithms II

.

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

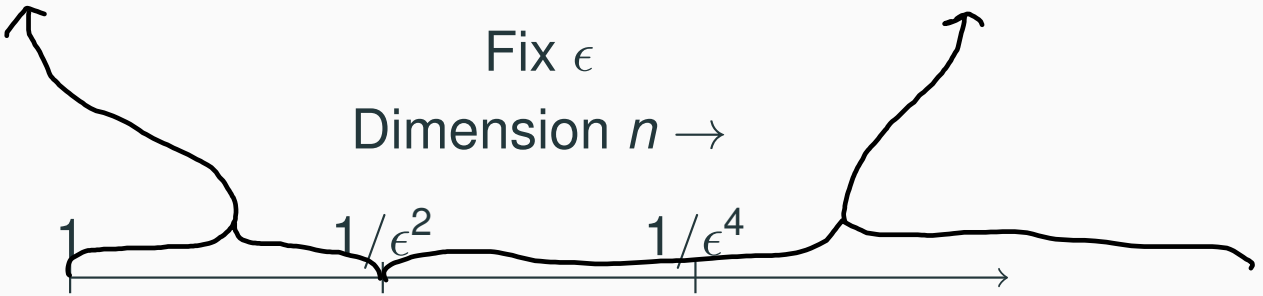
Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.



Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$

Classical: Deterministic

Known Algorithms II

Center of Gravity Method

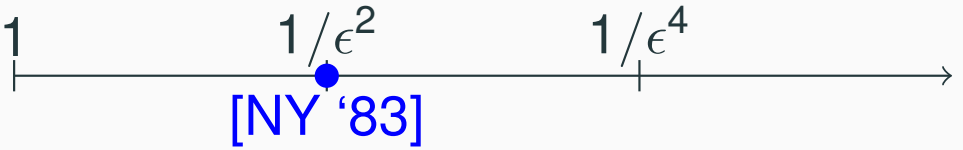
$n \log(1/\epsilon)$ steps.

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$



[NY '83]

$\Omega(1/\epsilon^2)$ queries

Classical: Deterministic

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$

Classical: Randomized

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$

[BJLLS '19]

$\Omega(1/\epsilon^2)$ queries

Classical: Randomized

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$

[BJLLS '19]

$\Omega(1/\epsilon^2)$ queries

Classical: Randomized

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.

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

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

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

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

The Base Function

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

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

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

The Base Function

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

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

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

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

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

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

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

Function Class

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

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

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

Function Class

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

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

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

$$\text{Set } \epsilon = \frac{.9}{\sqrt{n}}. \qquad x \text{ is } \epsilon\text{-opt} \Rightarrow f(x) \leq \frac{-.1}{\sqrt{n}}$$

Function Class

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

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

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

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

The behaviour of f

z_1 z_2 z_3 z_4 z_n

Function Class

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

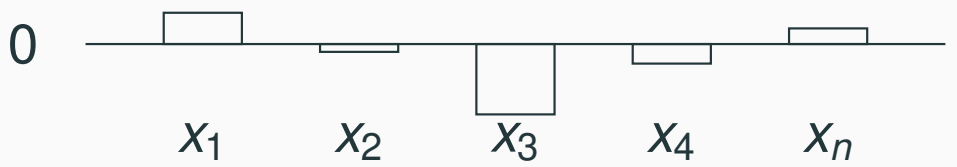
$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

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

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

The behaviour of f

z_1 z_2 z_3 z_4 z_n



Function Class

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

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

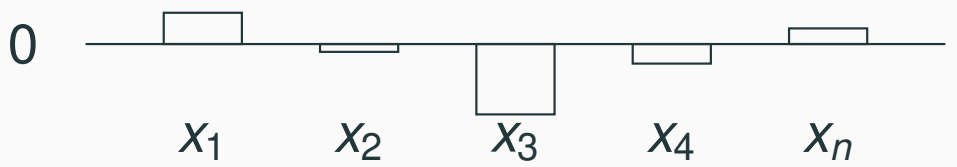
$$\text{Minimum} = -\frac{1}{\sqrt{n}}, \text{ at } x = \left(-\frac{z_1}{\sqrt{n}}, \ldots, -\frac{z_n}{\sqrt{n}}\right).$$

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

The behaviour of f

z_1 z_2 z_3 z_4 z_n

$+$



Function Class

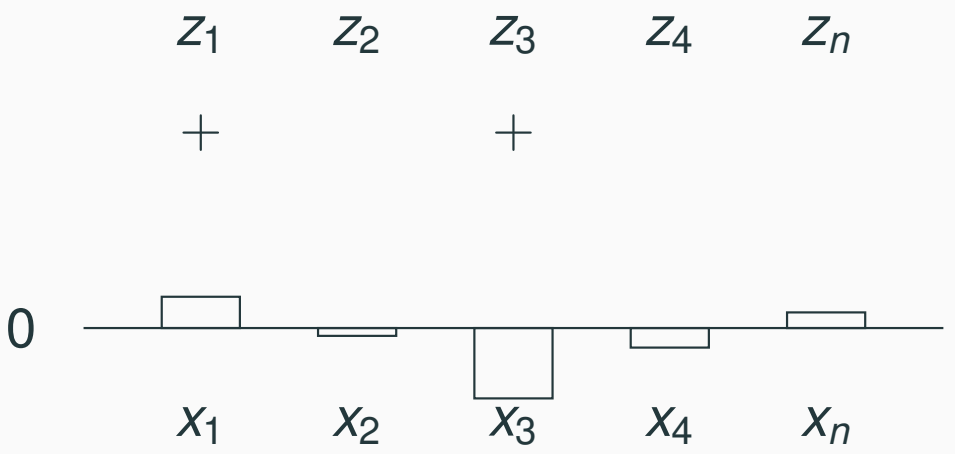
$$z \in \{+1, -1\}^n$$

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

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

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

The behaviour of f



Function Class

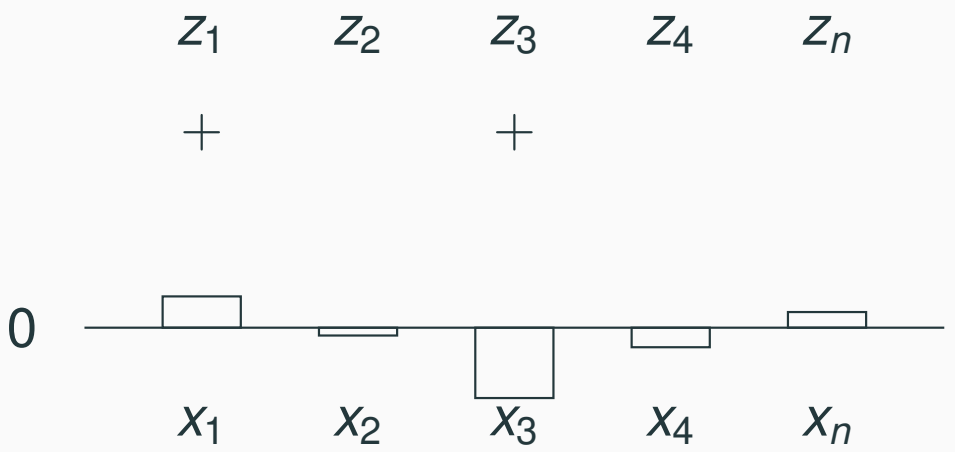
$$z \in \{+1, -1\}^n$$

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

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

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

The behaviour of f



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

Function Class

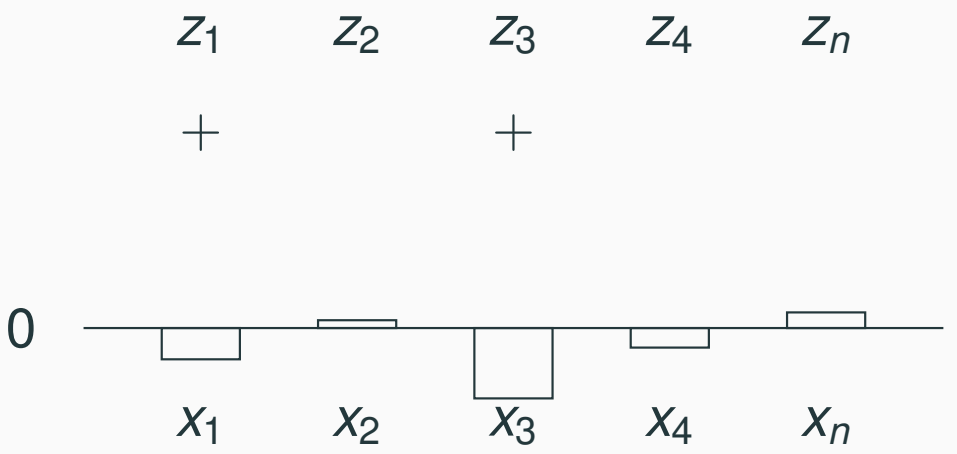
$$z \in \{+1, -1\}^n$$

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

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

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

The behaviour of f



Function Class

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

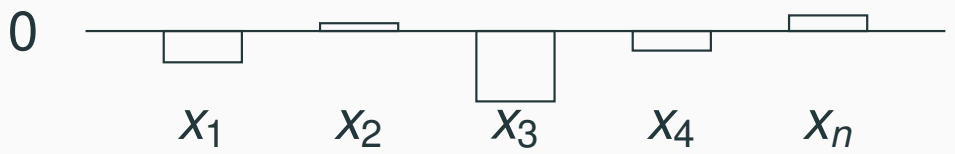
$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

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

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

The behaviour of f

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



Function Class

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

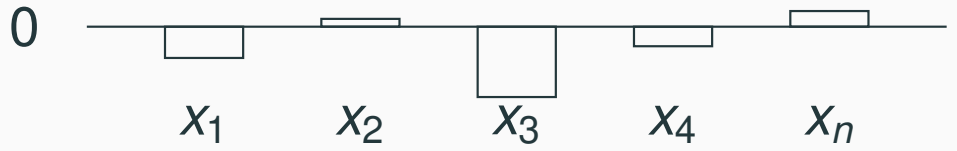
$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

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

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

The behaviour of f

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



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

Need $\Omega(n)$ queries $\left(\Omega\left(\frac{1}{\epsilon^2}\right) \text{ queries}\right)$

Function Class

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

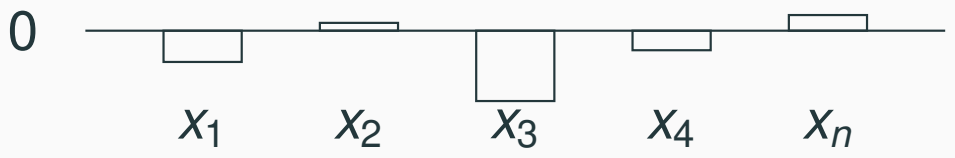
$$f_z(x) = \max\{z_1 x_1, z_2 x_2, \ldots, z_n x_n\}.$$

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

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

The behaviour of f

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

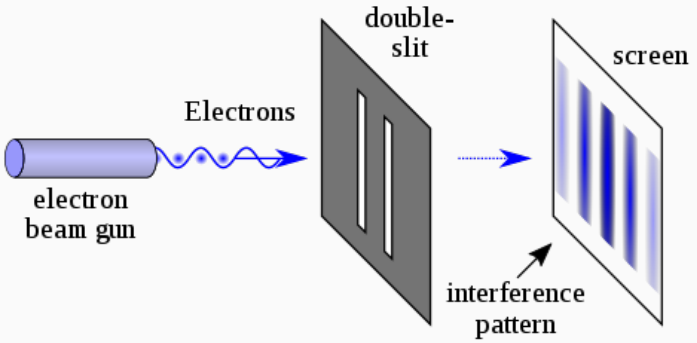


queries.

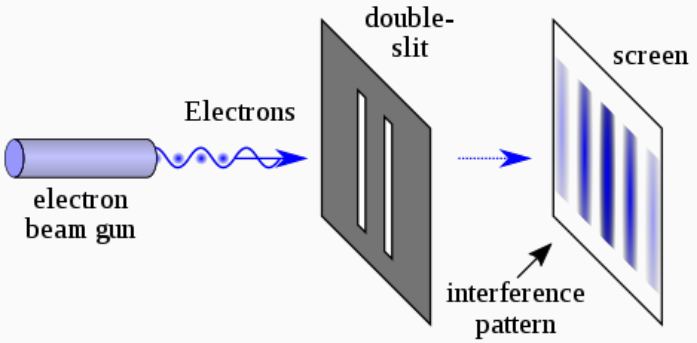
Requires $\Omega(n) = \Omega(1/\epsilon^2)$

Quantum Computing

Quantum Reality

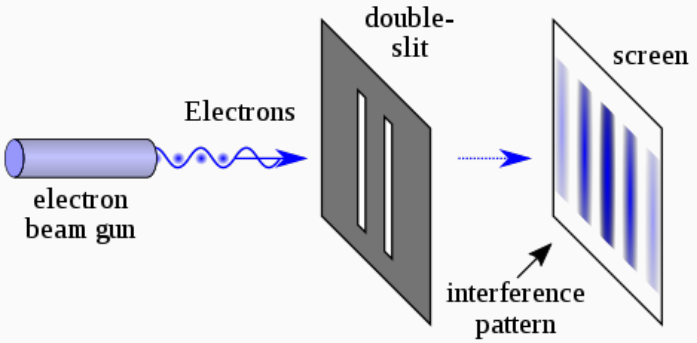


Quantum Reality



Particles can be in a superposition of states.

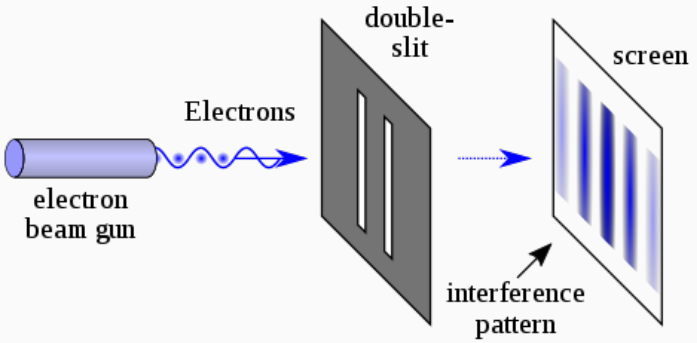
Quantum Reality



Particles can be in a superposition of states.

$$\alpha|\text{Left Slit}\rangle + \beta|\text{Right Slit}\rangle, \text{ with } |\alpha|^2 + |\beta|^2 = 1.$$

Quantum Reality

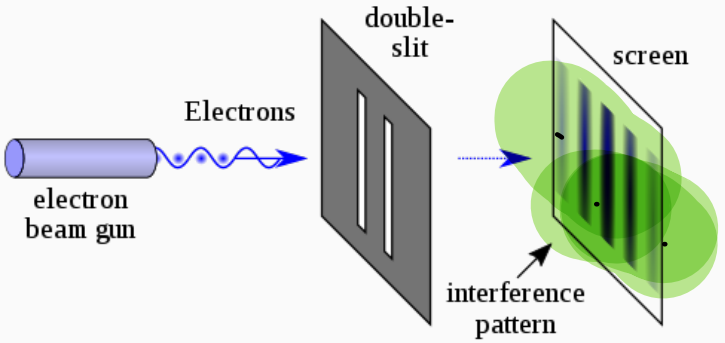


Particles can be in a superposition of states.

$$\alpha|\text{Left Slit}\rangle + \beta|\text{Right Slit}\rangle, \text{ with } |\alpha|^2 + |\beta|^2 = 1.$$

Each component of the superposition “evolves” as it would have without the superposition.

Quantum Reality



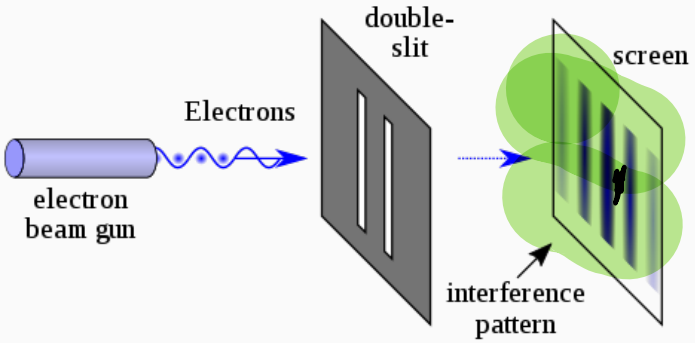
Particles can be in a superposition of states.

$$\alpha|\text{Left Slit}\rangle + \beta|\text{Right Slit}\rangle, \text{ with } |\alpha|^2 + |\beta|^2 = 1.$$

Each component of the superposition "evolves" as it would have without the superposition.

$$|\text{Left Slit}\rangle \mapsto |\text{Left Spread}\rangle \text{ and } |\text{Right Slit}\rangle \mapsto |\text{Right Spread}\rangle.$$

Quantum Reality



Particles can be in a superposition of states.

$$\alpha|\text{Left Slit}\rangle + \beta|\text{Right Slit}\rangle, \text{ with } |\alpha|^2 + |\beta|^2 = 1.$$

Each component of the superposition “evolves” as it would have without the superposition.

$$|\text{Left Slit}\rangle \mapsto |\text{Left Spread}\rangle \text{ and } |\text{Right Slit}\rangle \mapsto |\text{Right Spread}\rangle.$$

The actual state is the sum of the states of each component:
$\alpha|\text{Left Spread}\rangle + \beta|\text{Right Spread}\rangle$.

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

$$|\phi\rangle_{MEMORY}, \text{ where } \phi \in \mathbb{C}^{2^{\text{memory size}}} \text{ and } \|\phi\|_2 = 1.$$

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

$$|\phi\rangle_{MEMORY}, \text{ where } \phi \in \mathbb{C}^{2^{\text{memory size}}} \text{ and } \|\phi\|_2 = 1.$$

- Permitted operations are those that preserve norms.

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

$$|\phi\rangle_{MEMORY}, \text{ where } \phi \in \mathbb{C}^{2^{\text{memory size}}} \text{ and } \|\phi\|_2 = 1.$$

- Permitted operations are those that preserve norms.
- Can apply a unitary M to the state $|\phi\rangle$ to get the new state $|M\phi\rangle$.

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

$$|\phi\rangle_{MEMORY}, \text{ where } \phi \in \mathbb{C}^{2^{\text{memory size}}} \text{ and } \|\phi\|_2 = 1.$$

- Permitted operations are those that preserve norms.
- Can apply a unitary M to the state $|\phi\rangle$ to get the new state $|M\phi\rangle$.
  - Every base state gets mapped to a valid superposition.

$$|m\rangle \longmapsto |\phi_m\rangle$$

Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

$$|\phi\rangle_{MEMORY}, \text{ where } \phi \in \mathbb{C}^{2^{\text{memory size}}} \text{ and } \|\phi\|_2 = 1.$$

- Permitted operations are those that preserve norms.
- Can apply a unitary M to the state $|\phi\rangle$ to get the new state $|M\phi\rangle$.
  - Every base state gets mapped to a valid superposition.
  - Each of these mappings are orthonormal.

$|\phi_{m_1}\rangle$

$|\phi_{m_2}\rangle$

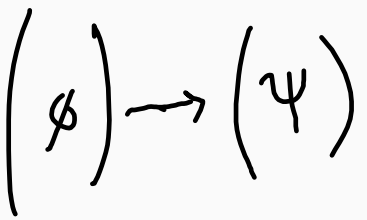
Quantum Computer

- A classical computer has a memory that is in one of $2^{\text{memory size in bits}}$ states.
- A quantum computer's memory can be in a superposition of these 'base' states.

$$|\phi\rangle_{MEMORY} = \sum_{m \in \{0,1\}^{\text{memory size}}} \alpha_m |m\rangle_{MEMORY} \text{ with } \sum |\alpha_m|^2 = 1.$$

$$|\phi\rangle_{MEMORY}, \text{ where } \phi \in \mathbb{C}^{2^{\text{memory size}}} \text{ and } \|\phi\|_2 = 1.$$

- Permitted operations are those that preserve norms.
- Can apply a unitary M to the state $|\phi\rangle$ to get the new state $|M\phi\rangle$.
  - Every base state gets mapped to a valid superposition.
  - Each of these mappings are orthonormal.
  - Just changing the basis.

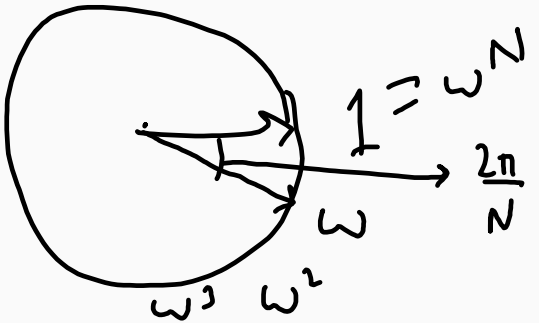


Quantum Tricks

- Can take advantage of Fourier transforms.
- The vectors

$$\frac{1}{\sqrt{N}}\begin{pmatrix}1\\1\\1\\\vdots\\1\end{pmatrix}, \frac{1}{\sqrt{N}}\begin{pmatrix}1\\\omega\\\omega^2\\\vdots\\\omega^{N-1}\end{pmatrix}, \frac{1}{\sqrt{N}}\begin{pmatrix}1\\\omega^2\\\omega^4\\\vdots\\\omega^{2(N-1)}\end{pmatrix}, \frac{1}{\sqrt{N}}\begin{pmatrix}1\\\omega^3\\\omega^6\\\vdots\\\omega^{3(N-1)}\end{pmatrix}, \ldots$$

are all orthonormal where $\omega = e^{i2\pi/N}$ is the principal Nth root of unity.



Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R}$$

Computing Gradients

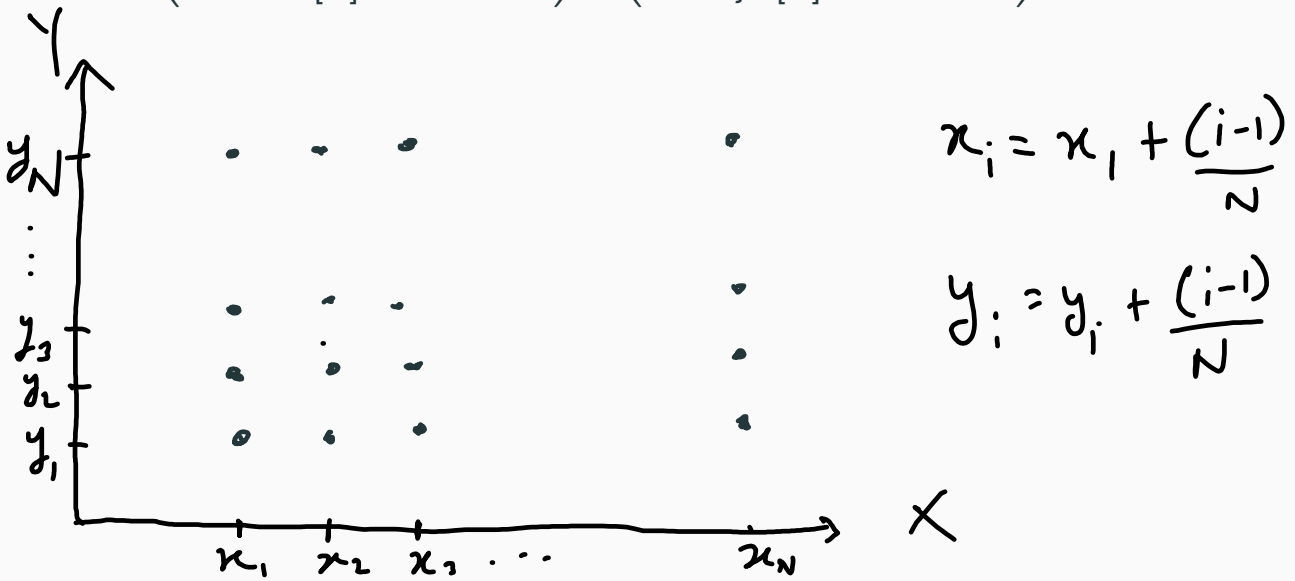
$$f : \mathbb{R}^2 \to \mathbb{R}$$

Example state: $\left(\sum_i \alpha_i |x_i y_i\rangle_{INPUT} \right) |0\rangle_{OUTPUT} |0\rangle_{OTHER\_VARS}$

Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

Initial state: $\left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} |x_i\rangle_{INPUT_1}\right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} |y_j\rangle_{INPUT_2}\right) |0\rangle_{OUTPUT}$



Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

Initial state: $$\left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} |x_i\rangle_{INPUT_1} \right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} |y_j\rangle_{INPUT_2} \right) |0\rangle_{OUTPUT}$$

Query: $$\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$$

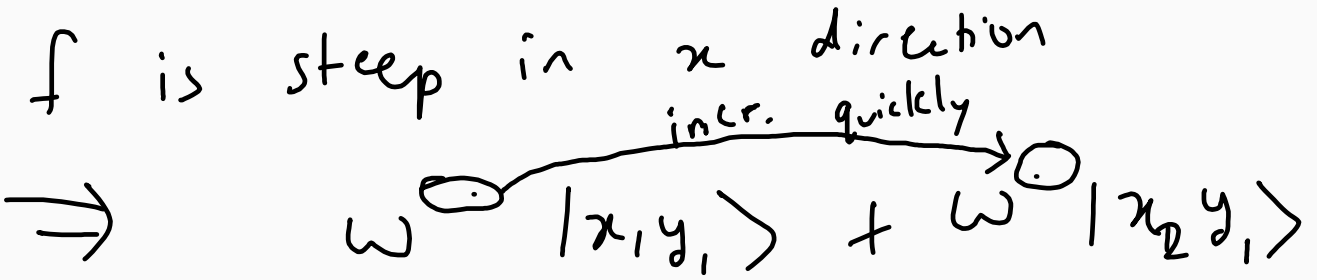
Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

Initial state: $\left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} |x_i\rangle_{INPUT_1}\right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} |y_j\rangle_{INPUT_2}\right) |0\rangle_{OUTPUT}$

Query: $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$

Add phases: $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} \omega^{\frac{f(x_i, y_j) N^2}{3}} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$



Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

Initial state: $\left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} |x_i\rangle_{INPUT_1}\right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} |y_j\rangle_{INPUT_2}\right) |0\rangle_{OUTPUT}$

Query: $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$

Add phases: $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} \omega^{\frac{f(x_i, y_j) N^2}{3}} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$

Query again (to undo): $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} \omega^{\frac{f(x_i, y_j) N^2}{3}} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |0\rangle_{OUTPUT}$

Needs query b/c $|x_1 y_1\rangle|c_1\rangle \mapsto |x_1 y_1\rangle|0\rangle$, $|x_1 y_1\rangle|c_2\rangle \mapsto |x_1 y_1\rangle|0\rangle$ is not unitary.

Only $|x_1 y_1\rangle|f(x_1, y_1)\rangle$ should map to $|x_1 y_1\rangle|0\rangle$

See slide 15 for why query undoes.

Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

Initial state: $\left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} |x_i\rangle_{INPUT_1}\right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} |y_j\rangle_{INPUT_2}\right) |0\rangle_{OUTPUT}$

Query: $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$

Add phases: $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} \omega^{\frac{f(x_i, y_j) N^2}{3}} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |f(x_i, y_j)\rangle_{OUTPUT}$

Query again (to undo): $\frac{1}{N} \sum_{i \in [N]} \sum_{j \in [N]} \omega^{\frac{f(x_i, y_j) N^2}{3}} |x_i\rangle_{INPUT_1} |y_j\rangle_{INPUT_2} |0\rangle_{OUTPUT}$

Equals:

$$\omega^{(ax_1 + by_1)\frac{N^2}{3}} \left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} \omega^{\frac{aiN}{3}} |x_1 + i/N\rangle_{INPUT_1}\right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} \omega^{\frac{bjN}{3}} |y_1 + j/N\rangle_{INPUT_2}\right) |0\rangle_{OUTPUT}$$

$\{0, 1, \ldots, N-1\}$ $\{0, 1, \ldots, N-1\}$

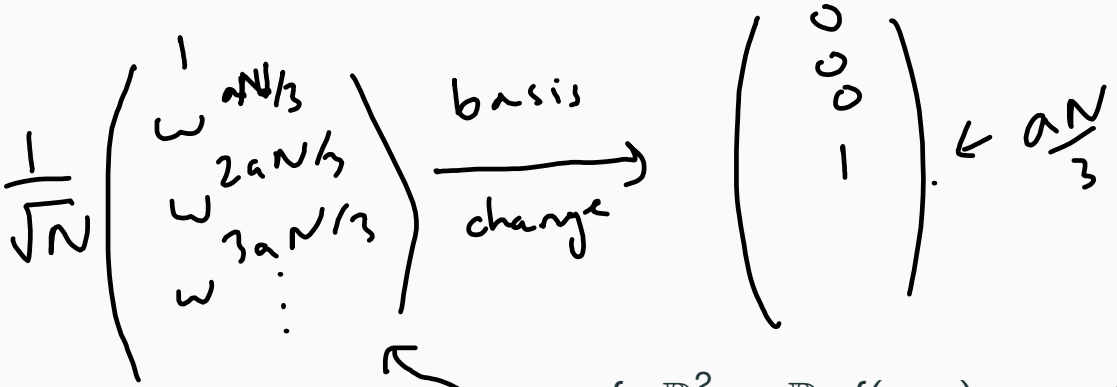
Computing Gradients

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

$$\omega^{(ax_1+by_1)\frac{N^2}{3}} \left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} \omega^{\frac{aiN}{3}} |x_1 + i/N\rangle_{INPUT_1} \right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} \omega^{\frac{bjN}{3}} |y_1 + j/N\rangle_{INPUT_2} \right) |0\rangle_{OUTPUT}$$

$\{0, \ldots, N-1\}$ $\{0, \ldots, N-1\}$

Computing Gradients



$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

$$\omega^{(ax_1+by_1)\frac{N^2}{3}} \left(\frac{1}{\sqrt{N}} \sum_{i \in [N]} \omega^{\frac{aiN}{3}} |x_1 + i/N\rangle_{INPUT_1} \right) \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} \omega^{\frac{bjN}{3}} |y_1 + j/N\rangle_{INPUT_2} \right) |0\rangle_{OUTPUT}$$

Change basis: $\omega^{f(x_1,y_1)\frac{N^2}{3}} |aN/3\rangle_{PHASE_1} \left(\frac{1}{\sqrt{N}} \sum_{j \in [N]} \omega^{\frac{bjN}{3}} |y_1 + j/N\rangle_{INPUT_2} \right) |0\rangle_{OUTPUT}$

Computing Gradients

$f : \mathbb{R}^n \to \mathbb{R}$ 2 queries

$$f : \mathbb{R}^2 \to \mathbb{R},\ f(x, y) = ax + by$$

$$\omega^{(ax_1+by_1)\frac{N^2}{3}} \left(\frac{1}{\sqrt{N}} \sum_{i\in[N]} \omega^{\frac{aiN}{3}} |x_1 + i/N\rangle_{INPUT_1} \right) \left(\frac{1}{\sqrt{N}} \sum_{j\in[N]} \omega^{\frac{bjN}{3}} |y_1 + j/N\rangle_{INPUT_2} \right) |0\rangle_{OUTPUT}$$

Change basis: $\omega^{f(x_1,y_1)\frac{N^2}{3}} |aN/3\rangle_{PHASE_1} \left(\frac{1}{\sqrt{N}} \sum_{j\in[N]} \omega^{\frac{bjN}{3}} |y_1 + j/N\rangle_{INPUT_2} \right) |0\rangle_{OUTPUT}$

For coord 2: $\omega^{f(x_1,y_1)\frac{N^2}{3}} |aN/3\rangle_{PHASE_1} |bN/3\rangle_{PHASE_2} |0\rangle_{OUTPUT}$

Can quantum “do gradient descent” faster?

- Can find gradient with 2 queries to function oracle. [Jordan ‘05]

Can quantum “do gradient descent” faster?

- Can find gradient with 2 queries 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 2 queries 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 2 queries 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 2 queries 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 2 queries 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.

Quantum Speedup

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

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

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

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

Quantum Speedup

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

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

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

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

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

Quantum Speedup

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

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

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

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

A quantum algorithm can ϵ-optimize the above 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

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

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

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

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

A quantum algorithm can ϵ-optimize the above 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.

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$

Quantum

Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.

Fix ϵ

Dimension $n \to$

[CCLW '18]

$\Omega(1/\epsilon)$ queries

Quantum

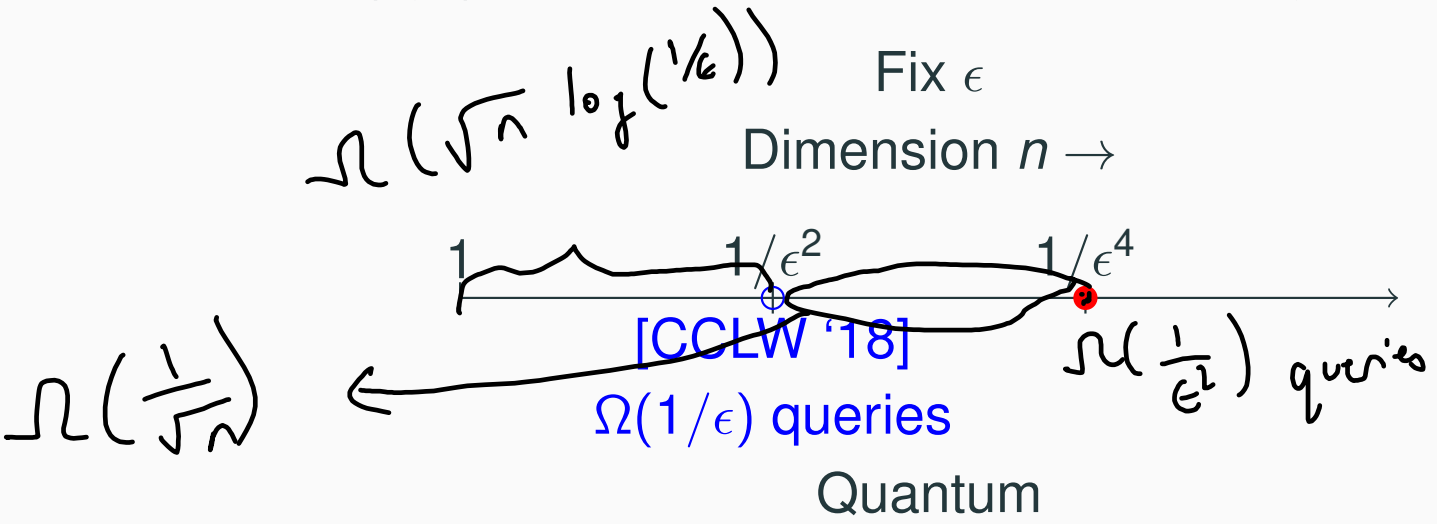
Known Algorithms II

Center of Gravity Method

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

Projected Subgradient Descent

$1/\epsilon^2$ steps.



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.

Quantum Computing

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 $\mathbb{R}^n$.

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

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

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 $\epsilon = \frac{.9}{\sqrt{k}}$. x is ϵ-opt $\Rightarrow f(x) \leq \frac{-.1}{\sqrt{k}}$

The Function Class

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

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

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

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

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

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

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

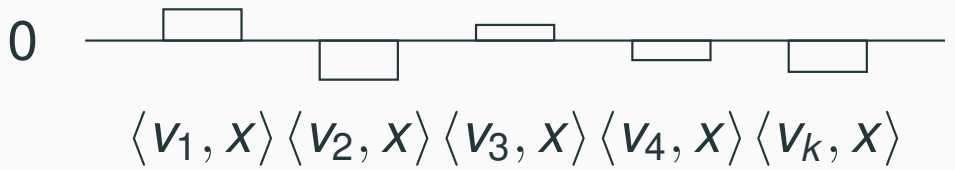
$$\text{Minimum} \approx -\frac{1}{\sqrt{k}}, \text{ at } x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}.$$

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

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

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

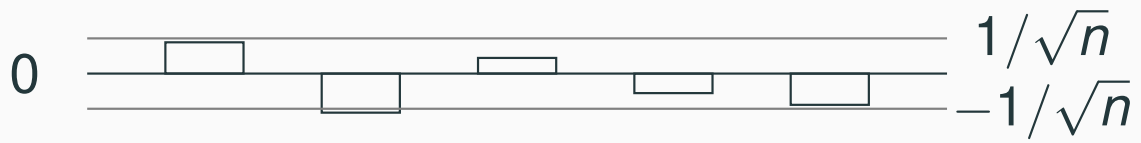
$$f_V(x) = \max\{\langle v_1, x\rangle, \langle v_2, x\rangle - \gamma, \langle v_3, x\rangle - 2\gamma, \ldots, \langle v_k, x\rangle - (k-1)\gamma\}.$$

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

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

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

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

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

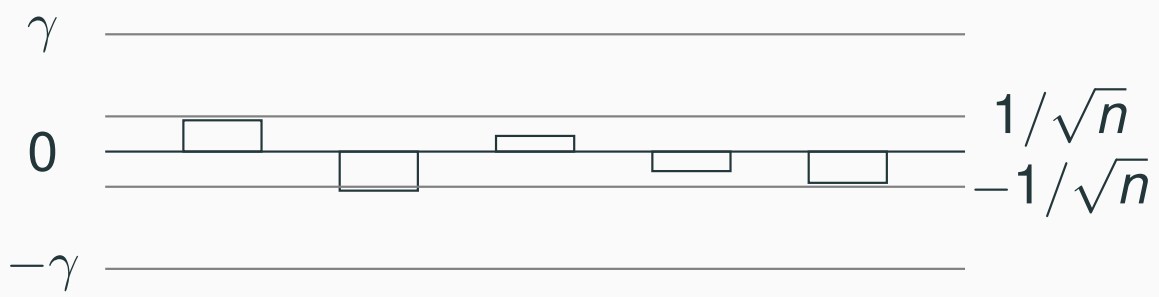
$$\text{Minimum} \approx -\frac{1}{\sqrt{k}}, \text{ at } x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}.$$

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

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

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

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

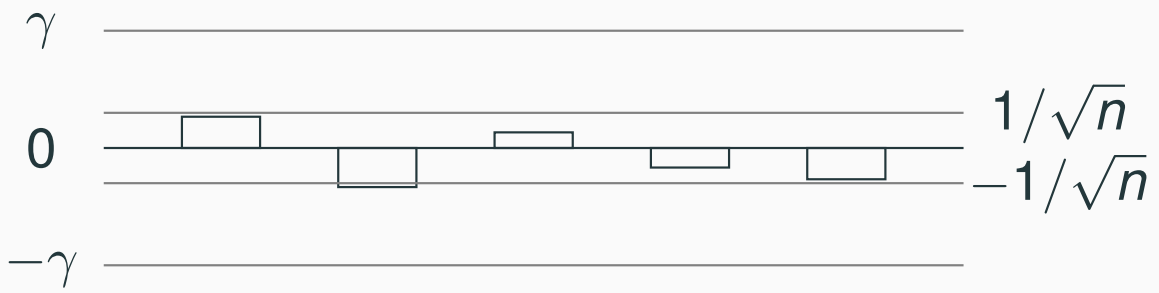
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

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.



$v_2 \notin \{v \mid \langle v, x\rangle > 2\gamma\}$

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 $\mathbb{R}^n$.

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

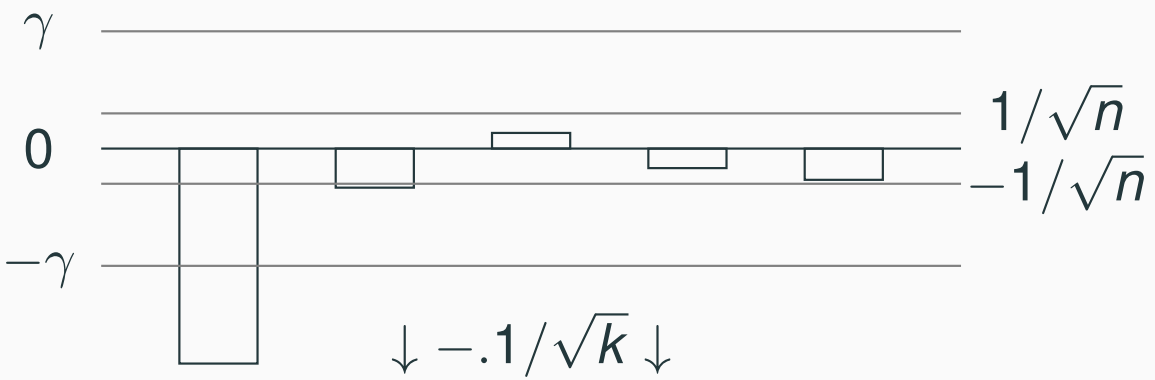
$$\text{Minimum} \approx -\frac{1}{\sqrt{k}}, \text{ at } x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}.$$

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

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

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

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

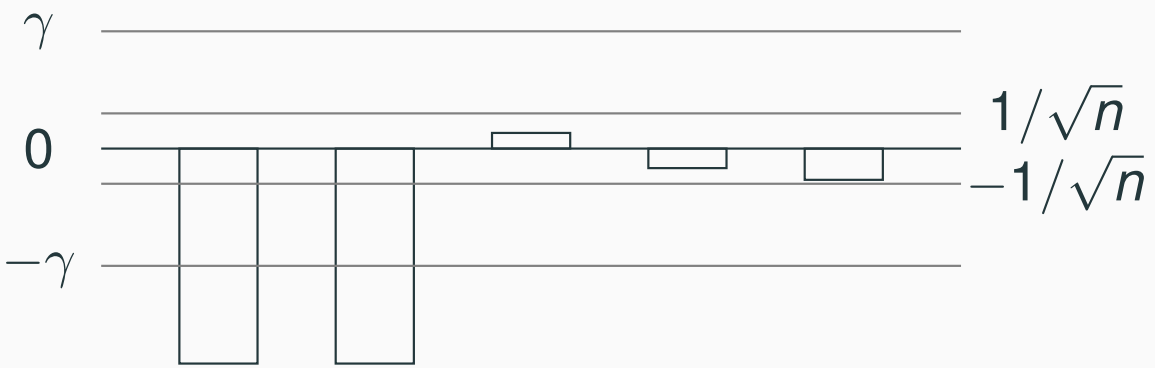
$$\text{Minimum} \approx -\frac{1}{\sqrt{k}}, \text{ at } x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}.$$

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

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

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

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

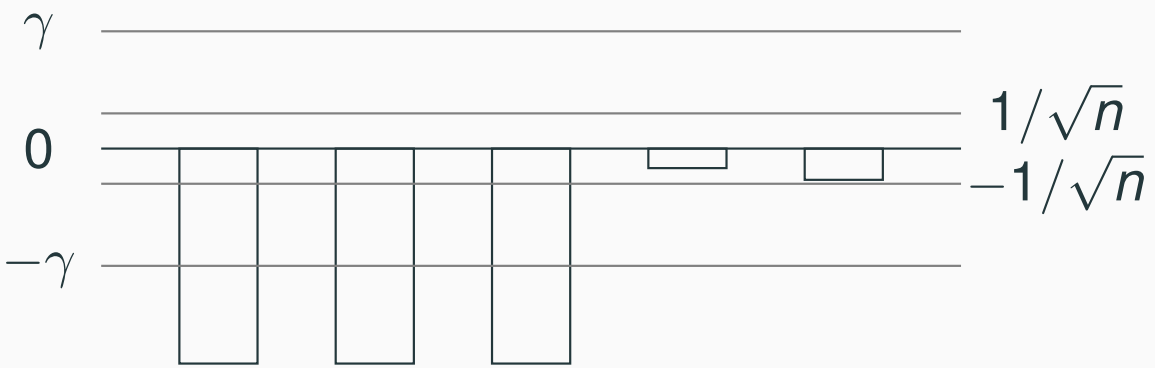
$$\text{Minimum} \approx -\frac{1}{\sqrt{k}}, \text{ at } x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}.$$

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

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

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

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

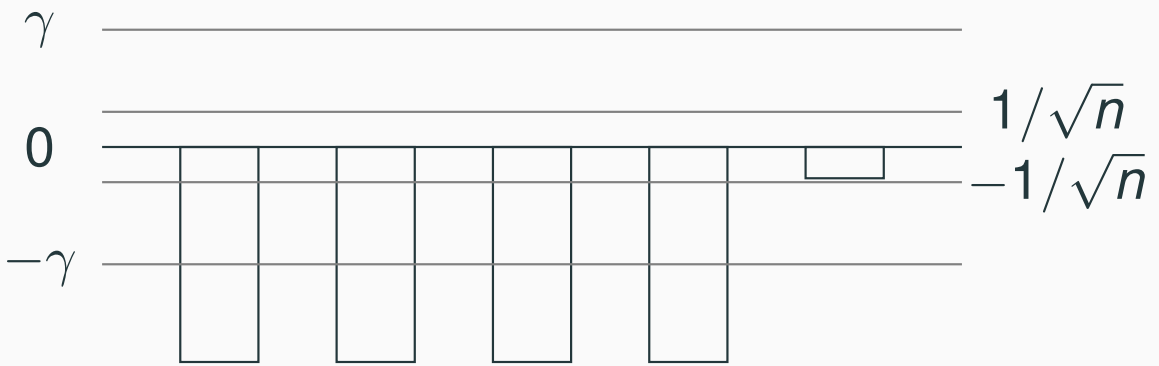
Minimum $\approx -\frac{1}{\sqrt{k}}$, at $x \approx -\frac{v_1}{\sqrt{k}} - \frac{v_2}{\sqrt{k}} \cdots - \frac{v_k}{\sqrt{k}}$.

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

$$
\begin{aligned}
&|x_1\rangle|\phi_1\rangle \\
+&|x_2\rangle|\phi_2\rangle \\
+&|x_3\rangle|\phi_3\rangle \\
+&|x_4\rangle|\phi_4\rangle \\
+&|x_5\rangle|\phi_5\rangle \\
+&\cdots
\end{aligned}
\xrightarrow{\text{pass through oracle for } f_V(x)}
$$

The Hybrid Argument

First query

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

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

First answer

$$
\begin{aligned}
&|x_1\rangle|\psi_1\rangle \\
+&|x_2\rangle|\psi_2\rangle \\
+&|x_3\rangle|\psi_3\rangle \\
+&|x_4\rangle|\psi_4\rangle \\
+&|x_5\rangle|\psi_5\rangle \\
+&\cdots
\end{aligned}
$$

The Hybrid Argument

First query

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

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

w.h.p $f_v(x) = \langle v, x\rangle$

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

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

Corrupted answer

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

The Hybrid Argument

First query

$$
\begin{aligned}
&|x_1\rangle|\phi_1\rangle \\
+&|x_2\rangle|\phi_2\rangle \\
+&|x_3\rangle|\phi_3\rangle \\
+&|x_4\rangle|\phi_4\rangle \\
+&|x_5\rangle|\phi_5\rangle \\
+&\cdots
\end{aligned}
\xrightarrow{\text{pass through oracle for } f_{(v_1)}(x)=\langle v_1, x\rangle}
$$

Corrupted answer

$$
\begin{aligned}
&|x_1\rangle|\psi_1\rangle \\
+&|x_2\rangle|\psi_2\rangle \\
+&|x_3\rangle|\psi_3\rangle \\
+&|x_4\rangle|\psi_4'\rangle \\
+&|x_5\rangle|\psi_5\rangle \\
+&\cdots
\end{aligned}
$$

- Changing oracle #1 barely changes the resulting state after 1 query. (with high probability)
- The actual, corrupted states at the end are also close. (because quantum only does unitaries [which preserve distances])

The Hybrid Argument

First query

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

Corrupted answer

- 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)

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

The Hybrid Argument: Second query

Second query (corrupted)

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

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

Second answer (corrupted)

$$
\begin{aligned}
&|x_1\rangle|\chi_1\rangle \\
+&|x_2\rangle|\chi_2\rangle \\
+&|x_3\rangle|\chi_3\rangle \\
+&|x_4\rangle|\chi_4\rangle \\
+&|x_5\rangle|\chi_5\rangle \\
+&\cdots
\end{aligned}
$$

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 $f_{(v_1,v_2)}(x)=\max\{\langle v_1,x\rangle,\langle v_2,x\rangle-\gamma\}$

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 $f_{(v_1,v_2)}(x)=\max\{\langle v_1,x\rangle,\langle v_2,x\rangle-\gamma\}$

Second answer (twice corrupted)

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

b/c this is ind. of v_2 to v_k, v_2 to v_k are truly at random.

- 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. $1/\sqrt{\epsilon}$ steps

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? $n < \frac{1}{\epsilon^4}$
- What other classes of convex optimization problems get quantum speedups?
- What is the quantum complexity of optimizing the function class

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