Adaptivity helps for testing juntas

Rocco Servedio, Li-Yang Tan, John Wright

Columbia TTIC CMU

Adaptivity helps for testing juntas

Rocco Servedio, Li-Yang Tan, John Wright
Columbia TTIC CMU

(work done while I was visiting Columbia)

Juntas

$f : \{0,1\}^n \to \{0,1\}$

Juntas

$f : \boxed{0 \mid 0 \mid 1 \mid 0 \mid 1 \mid 1 \mid 1 \mid 0 \mid 1 \mid 1} \rightarrow \{0,1\}$

Juntas

$$f : \boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{1} \to \{0,1\}$$

k-junta: f only depends on **k** bits

Juntas

$$f : 0\;0\;\boxed{1}\;0\;1\;\boxed{1}\;\boxed{1}\;0\;1\;1 \to \{0,1\}$$

k-junta: f only depends on **k** bits

Juntas

$$f: 0\ 0\ \boxed{1}\ 0\ 1\ \boxed{1}\ \boxed{1}\ 0\ 1\ 1 \to \{0,1\}$$

(a 3-junta)

k-junta: f only depends on **k** bits

Juntas

$$f : \; 0 \;\; 0 \;\; \boxed{1} \;\; 0 \;\; 1 \;\; \boxed{1} \;\; \boxed{1} \;\; 0 \;\; 1 \;\; 1 \;\to \{0,1\}$$

(a 3-junta)

k-junta: f only depends on **k** bits

(**k = 1**: f is a dictator)

Juntas

$$f : \; 0 \; 0 \; \boxed{1} \; 0 \; 1 \; \boxed{1} \; \boxed{1} \; 0 \; 1 \; 1 \; \to \{0,1\}$$

(a 3-junta)

k-junta: f only depends on **k** bits

(**k = 1**: f is a dictator)

Key question: how to tell if f is a **k**-junta?

Queries

Given: ability to make queries

$$\mathbf{x} \rightarrow f(\mathbf{x})$$

Queries

Given: ability to make queries

$$\mathbf{x} \rightarrow f(\mathbf{x})$$

Nonadaptive: fix queries in advance

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \cdots$$

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \to f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Queries

Given: ability to make queries

$$\mathbf{x} \rightarrow f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \rightarrow f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \dots \to f(\mathbf{x}_1), f(\mathbf{x}_2), \dots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1$$

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \to f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1 \to f(\mathbf{x}_1)$$

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \to f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1 \to f(\mathbf{x}_1), \mathbf{x}_2$$

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \to f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1 \to f(\mathbf{x}_1), \mathbf{x}_2 \to f(\mathbf{x}_2)$$

Queries

Given: ability to make queries

$$\mathbf{x} \to f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \to f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1 \to f(\mathbf{x}_1), \mathbf{x}_2 \to f(\mathbf{x}_2), \mathbf{x}_3$$

Queries

Given: ability to make queries

$$\mathbf{x} \rightarrow f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \rightarrow f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1 \rightarrow f(\mathbf{x}_1), \mathbf{x}_2 \rightarrow f(\mathbf{x}_2), \mathbf{x}_3 \rightarrow f(\mathbf{x}_3)$$

Queries

Given: ability to make queries

$$\mathbf{x} \rightarrow f(\mathbf{x})$$

Nonadaptive: fix queries in advance

$$\mathbf{x}_1, \mathbf{x}_2, \ldots \rightarrow f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots$$

Adaptive: choose queries based on answers

$$\mathbf{x}_1 \rightarrow f(\mathbf{x}_1), \mathbf{x}_2 \rightarrow f(\mathbf{x}_2), \mathbf{x}_3 \rightarrow f(\mathbf{x}_3), \ldots.$$

Property testing

Goal: distinguish whether (unknown) **f** is

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

Property testing

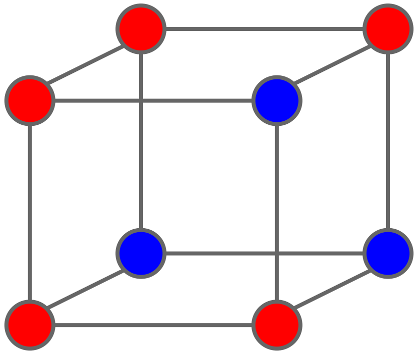
Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

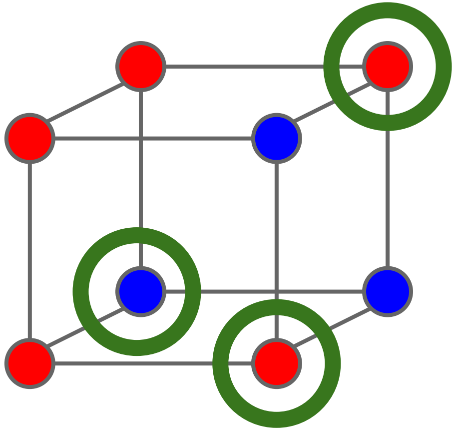
f: 

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

f:

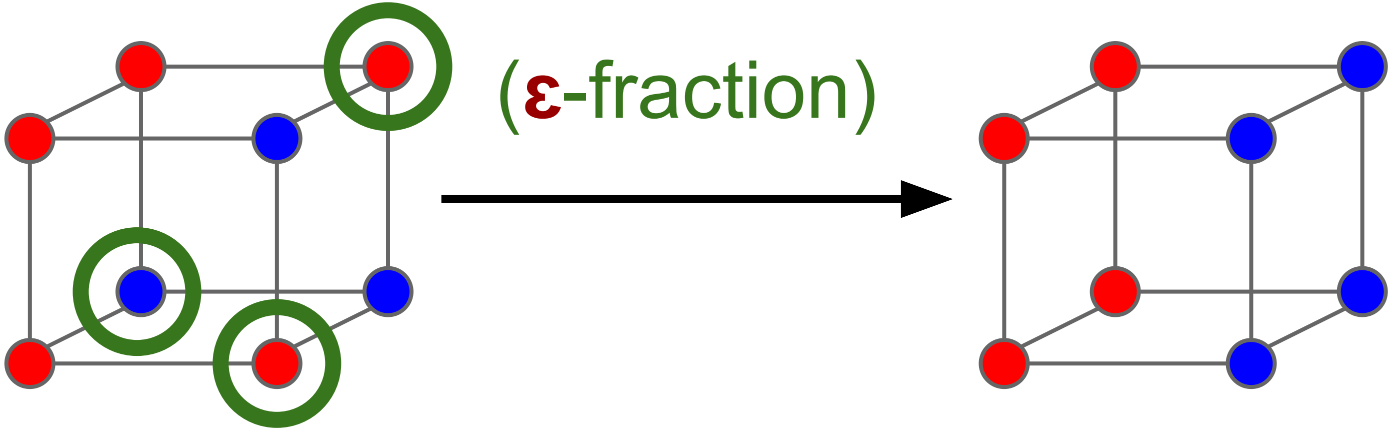


Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

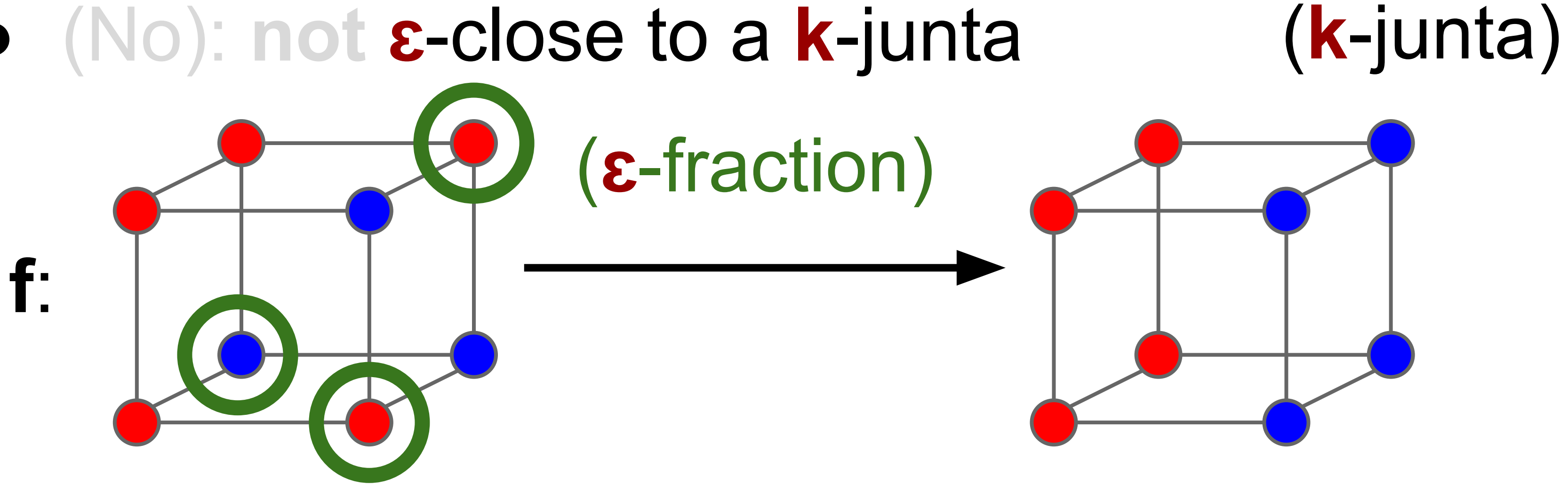
f:



Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta



Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **not** **ε**-close to a **k**-junta

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **ε**-far from all **k**-juntas

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **ε**-far from all **k**-juntas

Resources: Minimize query count **q**

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **ε**-far from all **k**-juntas

Resources: Minimize query count **q**

in terms of **k** and **ε**

Property testing

Goal: distinguish whether (unknown) **f** is

- (Yes): a **k**-junta
- (No): **ε**-far from all **k**-juntas

Resources: Minimize query count **q**

in terms of **k** and **ε** (no dependence on **n**!)

Junta testing motivation

- Boolean function version of finding a **low rank** model for **high dimensional** data

Junta testing motivation

- Boolean function version of finding a **low rank** model for **high dimensional** data
- For **k = 1**, equivalent to **dictatorship testing**, a basic topic in hardness of approximation

Junta testing motivation

- Boolean function version of finding a **low rank** model for **high dimensional** data
- For **k = 1**, equivalent to **dictatorship testing**, a basic topic in hardness of approximation
- One of the most basic Boolean function properties.

Prior work

nonadaptive

adaptive

Prior work

nonadaptive **adaptive**

[Bla08] $O(k^{3/2}\log(k)^3/\varepsilon)$

Prior work

nonadaptive **adaptive**

[Bla08] $O(k^{3/2}\log(k)^3/\varepsilon)$

[Bla08] $\Omega(k/(\varepsilon \log(k/\varepsilon)))$

Prior work

nonadaptive **adaptive**

[Bla08] $O(k^{3/2}\log(k)^3/\varepsilon)$

[Bla08] $\Omega(k/(\varepsilon \log(k/\varepsilon))$

[BGSMdW13] $\Omega(k \log(k))$

Prior work

nonadaptive | **adaptive**

[Bla08] $O(k^{3/2}\log(k)^3/\varepsilon)$

$O(k \log(k) + k/\varepsilon)$ [Bla09]

[Bla08] $\Omega(k/(\varepsilon \log(k/\varepsilon)))$

[BGSMdW13] $\Omega(k \log(k))$

Prior work

nonadaptive

[Bla08] $O(k^{3/2}\log(\mathbf{k})^3/\boldsymbol{\varepsilon})$

[Bla08] $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon}\log(\mathbf{k}/\boldsymbol{\varepsilon})))$

[BGSMdW13] $\Omega(\mathbf{k}\log(\mathbf{k}))$

adaptive

$O(\mathbf{k}\log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ [Bla09]

[CG04] $\Omega(\mathbf{k})$

Prior work

nonadaptive

[Bla08] $O(k^{3/2}\log(k)^3/\varepsilon)$

[Bla08] $\Omega(k/(\varepsilon \log(k/\varepsilon)))$

[BGSMdW13] $\Omega(k \log(k))$

adaptive

$O(k \log(k) + k/\varepsilon)$ **[Bla09]**

[CG04] $\Omega(k)$

$O(\mathbf{k} \log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ [Bla09]

[Bla08] $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$

[BGSMdW13] $\Omega(\mathbf{k} \log(\mathbf{k}))$

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

$O(k \log(k) + k/\varepsilon)$ [Bla09]

[Bla08] $\Omega(k/(\varepsilon \log(k/\varepsilon)))$

[BGSMdW13] $\Omega(k \log(k))$

Annoyance: adaptive **UB** ≥ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

$O(\mathbf{k} \log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ **[Bla09]**

[Bla08] $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon}))$

[BGSMdW13] $\Omega(\mathbf{k} \log(\mathbf{k}))$

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

$O(\mathbf{k} \log(\mathbf{k}) + \mathbf{k}/\varepsilon)$ **[Bla09]**

[Bla08] $\Omega(\mathbf{k}/(\varepsilon \log(\mathbf{k}/\varepsilon)))$

[BGSMdW13] $\Omega(\mathbf{k} \log(\mathbf{k}))$

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

It should: **[Bla09]**’s $O(\mathbf{k}\log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ adaptive algorithm uses **binary search**.

Annoyance: adaptive **UB** ≥ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

It should: **[Bla09]**'s $O(k \log(k) + k/\varepsilon)$ adaptive algorithm uses **binary search**.

Adaptivity also helps for testing **signed majority functions**, **read-once width-two OBDDs**.

Annoyance: adaptive **UB** ≥ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

It should: **[Bla09]**'s $O(k \log(k) + k/\varepsilon)$ adaptive algorithm uses **binary search**.

Adaptivity also helps for testing **signed majority functions**, **read-once width-two OBDDs**. Adaptive algos use **binary search**.

Annoyance: adaptive **UB** ≥ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

It should: [Bla09]'s $O(k \log(k) + k/\varepsilon)$ adaptive algorithm uses **binary search**.

Adaptivity also helps for testing **signed majority functions**, **read-once width-two OBDDs**. Adaptive algos use **binary search**.

Annoyance: adaptive **UB** ≥ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

$O(\mathbf{k} \log(\mathbf{k}) + \mathbf{k}/\varepsilon)$ **[Bla09]**

[Bla08] $\Omega(\mathbf{k}/(\varepsilon \log(\mathbf{k}/\varepsilon)))$

[BGSMdW13] $\Omega(\mathbf{k} \log(\mathbf{k}))$

Annoyance: adaptive **UB** ≥ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

$O(\mathbf{k}\log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ **[Bla09]**

[Bla08] $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon}\log(\mathbf{k}/\boldsymbol{\varepsilon})))$

[BGSMdW13] $\Omega(\mathbf{k}\log(\mathbf{k}))$

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

Our work: yes it does

$O(\mathbf{k} \log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ **[Bla09]**

[Bla08] $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$

[BGSMdW13] $\Omega(\mathbf{k} \log(\mathbf{k}))$

Annoyance: adaptive **UB** $\geq$ nonadaptive **LB**

[Bla09]: does adaptivity **even help?**

Our work: yes it does. new nonadaptive **LB**

$O(\mathbf{k}\log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$ **[Bla09]**

[Bla08] $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon}\log(\mathbf{k}/\boldsymbol{\varepsilon})))$

[BGSMdW13] $\Omega(\mathbf{k}\log(\mathbf{k}))$

Main result

Any **nonadaptive** algorithm requires

$$\mathbf{q} \geq \frac{\mathbf{k} \log(\mathbf{k})}{\boldsymbol{\varepsilon}^{c} \log(\log(\mathbf{k})/\boldsymbol{\varepsilon}^{c})}$$

queries

Main result

Any **nonadaptive** algorithm requires

$$\mathbf{q} \geq \frac{\mathbf{k}\log(\mathbf{k})}{\boldsymbol{\varepsilon}^{c}\log(\log(\mathbf{k})/\boldsymbol{\varepsilon}^{c})}$$

queries (for any $0 < c < 1$).

Main result

Any **nonadaptive** algorithm requires

$$\mathbf{q} \geq \frac{\mathbf{k} \log(\mathbf{k})}{\boldsymbol{\varepsilon}^c \log(\log(\mathbf{k})/\boldsymbol{\varepsilon}^c)}$$

queries (for any $0 < c < 1$).

Set $\boldsymbol{\varepsilon} = 1/\log(\mathbf{k})$.

Main result

Any **nonadaptive** algorithm requires

$$\mathbf{q} \geq \frac{\mathbf{k} \log(\mathbf{k})}{\boldsymbol{\varepsilon}^{c} \log(\log(\mathbf{k})/\boldsymbol{\varepsilon}^{c})}$$

queries (for any $0 < c < 1$).

Set $\boldsymbol{\varepsilon} = 1/\log(\mathbf{k})$.

Adaptive UB = $O(\mathbf{k} \log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon})$

Main result

Any **nonadaptive** algorithm requires

$$q \geq \frac{k \log(k)}{\varepsilon^c \log(\log(k)/\varepsilon^c)}$$

queries (for any $0 < c < 1$).

Set $\varepsilon = 1/\log(k)$.

Adaptive UB = $O(k \log(k) + k/\varepsilon) = O(k \log(k))$

Main result

Any **nonadaptive** algorithm requires

$$\mathbf{q} \geq \frac{\mathbf{k}\log(\mathbf{k})}{\boldsymbol{\varepsilon}^c \log(\log(\mathbf{k})/\boldsymbol{\varepsilon}^c)}$$

queries (for any $0 < c < 1$).

Set $\boldsymbol{\varepsilon} = 1/\log(\mathbf{k})$.

Adaptive UB $= O(\mathbf{k}\log(\mathbf{k}) + \mathbf{k}/\boldsymbol{\varepsilon}) = O(\mathbf{k}\log(\mathbf{k}))$

Our **nonadapt** LB $= \mathbf{k}\log(\mathbf{k})^{1+c}/\log(\log(\mathbf{k}))$

Our techniques

- Basic ideas come from **[CG04]**'s $\Omega(\mathbf{k})$ **adaptive** lower bound

Our techniques

- Basic ideas come from **[CG04]**'s $\Omega(\mathbf{k})$ **adaptive** lower bound
- **[Bla08]**'s $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$ **nonadaptive** lower bound based on **[CG04]**'s lower bound

Our techniques

- Basic ideas come from [CG04]'s $\Omega(\mathbf{k})$ **adaptive** lower bound
- [Bla08]'s $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$ **nonadaptive** lower bound based on [CG04]'s lower bound
- We give a new analysis of [Bla08]'s **LB**.

Our techniques

- Basic ideas come from **[CG04]**'s $\Omega(\mathbf{k})$ **adaptive** lower bound
- **[Bla08]**'s $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$ **nonadaptive** lower bound based on **[CG04]**'s lower bound
- We give a new analysis of **[Bla08]**'s **LB**.

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

[CG04] considers two distributions on $\mathbf{n} = (\mathbf{k+1})$-variable functions:

D_{yes}:

- Pick $\mathbf{i} \sim \{\mathbf{1},\ldots,\mathbf{k+1}\}$ uar.

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate **i**.

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate **i**.

$$f_{yes}(x_1,\ldots,\underset{i}{0},\ldots,x_{k+1}) = f_{yes}(x_1,\ldots,\underset{i}{1},\ldots,x_{k+1})$$

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate **i**.

$$f_{yes}(x_1,\ldots,\underset{\mathbf{i}}{\mathbf{0}},\ldots,x_{k+1}) = f_{yes}(x_1,\ldots,\underset{\mathbf{i}}{\mathbf{1}},\ldots,x_{k+1}) = \text{random } \{\mathbf{0},\mathbf{1}\}$$

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate **i**.

$$f_{yes}(x_1,\ldots,\underset{\mathbf{i}}{\mathbf{0}},\ldots,x_{k+1}) = f_{yes}(x_1,\ldots,\underset{\mathbf{i}}{\mathbf{1}},\ldots,x_{k+1}) = \text{random } \{\mathbf{0},\mathbf{1}\}$$

(for all $x_1,\ldots,x_{k+1}$)

[CG04] considers two distributions on $n = (k+1)$-variable functions:

D_{yes}:

- Pick $i \sim \{1,\ldots,k+1\}$ uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate i.

$$f_{yes}(x_1,\ldots,\underset{i}{0},\ldots,x_{k+1}) = f_{yes}(x_1,\ldots,\underset{i}{1},\ldots,x_{k+1}) = \text{random } \{0,1\}$$

(for all $x_1,\ldots,x_{k+1}$)

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate **i**.

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:
- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \to \{0,1\}$ uar subject to **not** depending on coordinate **i**.

D_{no}:
- Set $f_{no}:\{0,1\}^{k+1} \to \{0,1\}$ uar.

[CG04] considers two distributions on $n = (k+1)$-variable functions:

D_{yes}:
- Pick $i \sim \{1,...,k+1\}$ uar.
- Set $f_{yes}:\{0,1\}^{k+1} \to \{0,1\}$ uar subject to **not** depending on coordinate i.

(a k-junta)

D_{no}:
- Set $f_{no}:\{0,1\}^{k+1} \to \{0,1\}$ uar.

[CG04] considers two distributions on $n = (k+1)$-variable functions:

D_{yes}:
- Pick $i \sim \{1,\ldots,k+1\}$ uar.
- Set $f_{yes} : \{0,1\}^{k+1} \to \{0,1\}$ uar subject to **not** depending on coordinate i.

(a **k**-junta)

D_{no}:
- Set $f_{no} : \{0,1\}^{k+1} \to \{0,1\}$ uar.

(**usually** far from a **k**-junta)

[CG04] considers two distributions on
$n = (k+1)$-variable functions:

D_{yes}:

- Pick $i \sim \{1,\ldots,k+1\}$ uar.
- Set $f_{yes} : \{0,1\}^{k+1} \rightarrow \{0,1\}$ uar subject to **not** depending on coordinate i.

(a k-junta)

D_{no}:

- Set $f_{no} : \{0,1\}^{k+1} \rightarrow \{0,1\}$ uar.

(**usually** far from a k-junta)

[CG04] considers two distributions on **n** = (**k+1**)-variable functions:

D_{yes}:
- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \to \{0,1\}$ uar subject to **not** depending on coordinate **i**.

D_{no}:
- Set $f_{no}:\{0,1\}^{k+1} \to \{0,1\}$ uar.

[CG04] considers two distributions on
$n = (k+1)$-variable functions:

D_{yes}:
- Pick $i \sim \{1,\ldots,k+1\}$ uar.
- Set $f_{yes}:\{0,1\}^{k+1} \to \{0,1\}$ uar subject to **not** depending on coordinate i.

D_{no}:
- Set $f_{no}:\{0,1\}^{k+1} \to \{0,1\}$ uar.

[CG04 THM]: Need $\Omega(k)$ queries to distinguish these distributions

Given f, how to tell if from D_{yes} or D_{no}?

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**: • Pick **x** uar.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

- Pick **x** uar.
- Query f on **x** and **x**$\oplus$**i**.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

- Pick **x** uar.
- Query f on **x** and **x⊕i**.

Differ only on coord **i**.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:
- Pick **x** uar.
- Query f on **x** and **x⊕i**.

Differ only on coord **i**.

Def: x and **x⊕i** form an **i-twin**.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:
- Pick **x** uar.
- Query f on **x** and **x**⊕**i**.

Differ only on coord **i**.

Def: **x** and **x**⊕**i** form an **i-twin**.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

- Pick **x** uar.
- Query f on **x** and **x**$\oplus$**i**.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

- Pick **x** uar.
- Query f on **x** and **x**⊕**i**.
- If f(**x**) ≠ f(**x**⊕**i**), output **relevant**.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

- Pick **x** uar.
- Query f on **x** and **x⊕i**.
- If f(**x**) ≠ f(**x⊕i**), output **relevant**.
- Repeat 10 log(**k**) times.

Given f, how to tell if from D_{yes} or D_{no}?

Idea: See if it has any **irrelevant** coords.

For coord **i**:

- Pick **x** uar.
- Query f on **x** and **x⊕i**.
- If f(**x**) ≠ f(**x⊕i**), output **relevant**.
- Repeat 10 log(**k**) times.
- Output **irrelevant**.

If **i** is **relevant**:

If **i** is **relevant**: $f(\mathbf{x})$ $f(\mathbf{x} \oplus \mathbf{i})$

If **i** is **relevant**: f(**x**) f(**x**⊕**i**)

uar {**0**,**1**}

If **i** is **relevant**: $f(\mathbf{x})$ $f(\mathbf{x}\oplus\mathbf{i})$

uar {**0**,**1**}

uar {**0**,**1**}

If **i** is **relevant**: $f(\mathbf{x})$ $f(\mathbf{x} \oplus \mathbf{i})$

uar {**0**,**1**}

uar {**0**,**1**}

If **i** is **relevant**: $f(\mathbf{x})$ $f(\mathbf{x}\oplus\mathbf{i})$

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

∴ will conclude **relevant** after O(1) **i**-twins

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x}\oplus\mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x}\oplus\mathbf{i})$ w/prob 1/2

∴ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**:

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$\therefore$ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ always

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$\therefore$ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ always

$\therefore$ will query O(log(**k**)) **i**-twins

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$\therefore$ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ always

$\therefore$ will query O(log(**k**)) **i**-twins

Query cost:

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$\therefore$ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ always

$\therefore$ will query O(log(**k**)) **i**-twins

Query cost: (**k+1**) * O(1)

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

∴ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ always

∴ will query O(log(**k**)) **i**-twins

Query cost: (**k+1**) * O(1) + O(log(**k**))

If **i** is **relevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

$f(\mathbf{x}) \neq f(\mathbf{x} \oplus \mathbf{i})$ w/prob 1/2

∴ will conclude **relevant** after O(1) **i**-twins

If **i** is **irrelevant**: $f(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{i})$ always

∴ will query O(log(**k**)) **i**-twins

Query cost: $(\mathbf{k+1}) * O(1) + O(\log(\mathbf{k})) = O(\mathbf{k})$

[CG04]'s $\Omega(k)$ lower bound

[CG04]'s $\Omega(k)$ lower bound

Key idea:

- Suppose you query f on $x_1,\ldots,x_q$

[CG04]'s $\Omega(k)$ lower bound

Key idea:

- Suppose you query f on $x_1,\ldots,x_q$
- Want to test: is coord **i relevant**?

[CG04]'s Ω(**k**) lower bound

Key idea:

- Suppose you query f on $\mathbf{x}_1,\ldots,\mathbf{x}_q$
- Want to test: is coord **i relevant**?
- Then $\mathbf{x}_1,\ldots,\mathbf{x}_q$ must have an **i**-twin

[CG04]'s Ω(**k**) lower bound

Key idea:

- Suppose you query f on $\mathbf{x}_1,\ldots,\mathbf{x}_q$
- Want to test: is coord **i relevant**?
- Then $\mathbf{x}_1,\ldots,\mathbf{x}_q$ must have an **i**-twin

LB: **q** points can have **i**-twins for ≤ **q-1** coords.

[CG04]'s Ω(**k**) lower bound

Key idea:
- Suppose you query f on $\mathbf{x_1,\ldots,x_q}$
- Want to test: is coord **i relevant**?
- Then $\mathbf{x_1,\ldots,x_q}$ must have an **i**-twin

LB: **q** points can have **i**-twins for ≤ **q-1** coords.

$\therefore$ **q** = Ω(**k**).

Matching upper and lower bounds?

Matching upper and lower bounds?

Algorithm was **adaptive**:

Matching upper and lower bounds?

Algorithm was **adaptive**:

- for **relevant** coords **i**, query O(1) **i**-twins

Matching upper and lower bounds?

Algorithm was **adaptive**:

- for **relevant** coords **i**, query O(1) **i**-twins
- for **irrelevant** coords **i**, query O(log(**k**))

Matching upper and lower bounds?

Algorithm was **adaptive**:

- for **relevant** coords **i**, query O(1) **i**-twins
- for **irrelevant** coords **i**, query O(log(**k**))

Can’t plan this in advance:

Matching upper and lower bounds?

Algorithm was **adaptive**:

- for **relevant** coords **i**, query O(1) **i**-twins
- for **irrelevant** coords **i**, query O(log(**k**))

Can't plan this in advance: $x_1,\ldots,x_q$ need O(log(**k**)) **i**-twins in **all** **k+1** directions.

Matching upper and lower bounds?

Algorithm was **adaptive**:

- for **relevant** coords **i**, query O(1) **i**-twins
- for **irrelevant** coords **i**, query O(log(**k**))

Can't plan this in advance: $\mathbf{x_1,\ldots,x_q}$ need O(log(**k**)) **i**-twins in **all** **k+1** directions.

$\therefore$ $\mathbf{q} = \Omega(\mathbf{k} \log(\mathbf{k}))$ **nonadaptive** **LB**?

Matching upper and lower bounds?

Algorithm was **adaptive**:

- for **relevant** coords **i**, query O(1) **i**-twins
- for **irrelevant** coords **i**, query O(log(**k**))

Can't plan this in advance: $\mathbf{x_1},\dots,\mathbf{x_q}$ need O(log(**k**)) **i**-twins in **all** **k+1** directions.

∴ $\mathbf{q} = \Omega(\mathbf{k} \log(\mathbf{k}))$ **nonadaptive** **LB**?

(not quite)

A nonadaptive algorithm.

[Fra83]: there are $q = O(k \log(k) / \log \log(k))$ points $x_1, \ldots, x_q$ with $\log(k)$ i-twins for each i.

A nonadaptive algorithm.

[Fra83]: there are $q = O(k \log(k) / \log\log(k))$ points $x_1,\ldots,x_q$ with $\log(k)$ i-twins for each i.

Recall our LB: $$q \geq \frac{k \log(k)}{\varepsilon^c \log(\log(k)/\varepsilon^c)}$$

A nonadaptive algorithm.

[Fra83]: there are $q = O(k \log(k) / \log\log(k))$ points $x_1, \ldots, x_q$ with $\log(k)$ i-twins for each i.

Recall our LB: $$q \geq \frac{k \log(k)}{\varepsilon^c \log(\log(k)/\varepsilon^c)}$$

Goal: • show [Fra83] is optimal

A nonadaptive algorithm.

[Fra83]: there are $q = O(k \log(k) / \log\log(k))$ points $x_1,\ldots,x_q$ with $\log(k)$ i-twins for each i.

Recall our LB: $$q \geq \frac{k \log(k)}{\varepsilon^c \log(\log(k)/\varepsilon^c)}$$

Goal:
- show [Fra83] is optimal
- extend to general ε

A nonadaptive algorithm.

[Fra83]: there are $q = O(k \log(k) / \log\log(k))$ points $x_1,\ldots,x_q$ with $\log(k)$ i-twins for each i.

Recall our LB: $$q \geq \frac{k \log(k)}{\varepsilon^c \log(\log(k)/\varepsilon^c)}$$

Goal:
- show [Fra83] is optimal
- extend to general ε

New distributions.

New distributions.

D_{no}:

- Set $f_{no}:\{0,1\}^{k+1} \to \{0,1\}$ random ε-biased.

New distributions.

$f(\mathbf{x})$ is independent from all other $f(\mathbf{x}')$,

D_{no}:

- Set $f_{no}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ random ε-biased.

New distributions.

f(**x**) is independent from all other f(**x**'), satisfies **Pr**[f(**x**) = 1] = **ε**

D_{no}:

- Set $f_{no}:\{0,1\}^{k+1} \to \{0,1\}$ random **ε**-biased.

New distributions.

$f(\mathbf{x})$ is independent from all other $f(\mathbf{x}')$,
satisfies $\mathbf{Pr}[f(\mathbf{x}) = 1] = \boldsymbol{\varepsilon}$

D_{no}:
- Set $f_{no}:\{0,1\}^{\mathbf{k+1}} \to \{0,1\}$ random $\boldsymbol{\varepsilon}$-biased.

New distributions.

D_{no}:

- Set f_{no}:$\{0,1\}^{k+1} \to \{0,1\}$ random **ε**-biased.

New distributions.

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.

D_{no}:

- Set $f_{no}:\{0,1\}^{\mathbf{k+1}} \to \{0,1\}$ random **ε**-biased.

New distributions.

D_{yes}:

- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set $f_{yes}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ random **ε**-biased subject to **not** depending on coordinate **i**.

D_{no}:

- Set $f_{no}:\{0,1\}^{k+1} \rightarrow \{0,1\}$ random **ε**-biased.

New distributions.

D_{yes}:
- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set f_{yes}:$\{0,1\}^{\mathbf{k+1}} \to \{0,1\}$ random **ε**-biased subject to **not** depending on coordinate **i**.

(a **k**-junta)

D_{no}:
- Set f_{no}:$\{0,1\}^{\mathbf{k+1}} \to \{0,1\}$ random **ε**-biased.

New distributions.

D_{yes}:
- Pick **i** ~ {**1**,...,**k+1**} uar.
- Set f_{yes}: $\{0,1\}^{\mathbf{k+1}} \to \{0,1\}$ random **ε**-biased subject to **not** depending on coordinate **i**.

(a **k**-junta)

D_{no}:
- Set f_{no}: $\{0,1\}^{\mathbf{k+1}} \to \{0,1\}$ random **ε**-biased.

(**usually** **ε**-far from a **k**-junta)

Distributions studied in [Bla08].

Distributions studied in [Bla08].

His LB: $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$

Distributions studied in [Bla08].

His LB: $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$

Main tool: **Edge-isoperimetric inequality**

Distributions studied in [Bla08].

His LB: $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon}))$

Main tool: **Edge-isoperimetric inequality**:

points $\mathbf{x}_1,\ldots,\mathbf{x}_q$ can only have $O(\mathbf{q} \log(\mathbf{q}))$ $\mathbf{i}$-twins

Distributions studied in [Bla08].

His LB: $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon})))$

Main tool: **Edge-isoperimetric inequality**:
points $\mathbf{x}_1,\ldots,\mathbf{x}_q$ can only have $O(\mathbf{q} \log(\mathbf{q}))$ **i**-twins

- Only about **total** # of **i**-twins.

Distributions studied in [Bla08].

His LB: $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon}))$

Main tool: **Edge-isoperimetric inequality**:
points $\mathbf{x}_1,\ldots,\mathbf{x}_q$ can only have $O(\mathbf{q} \log(\mathbf{q}))$ **i**-twins

- Only about **total** # of **i**-twins.
- Could be **few** directions have lots of **i**-twins.

Distributions studied in [Bla08].

His LB: $\Omega(\mathbf{k}/(\boldsymbol{\varepsilon} \log(\mathbf{k}/\boldsymbol{\varepsilon}))$

Main tool: **Edge-isoperimetric inequality**:
points $\mathbf{x_1,\ldots,x_q}$ can only have $O(\mathbf{q} \log(\mathbf{q}))$ **i**-twins

- Only about **total** # of **i**-twins.
- Could be **few** directions have lots of **i**-twins.
- Want edge-iso ineq. about **most** directions.

Our main tool: **New edge-iso inequality**

Our main tool: **New edge-iso inequality**

Suppose $\mathbf{x}_1,\ldots,\mathbf{x}_q$ have **m** **i**-twins in **d** directions.
Then $\mathbf{q} \geq \mathbf{md} / \log(\mathbf{m})$.

Our main tool: **New edge-iso inequality**

Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_q$ have **m** **i**-twins in **d** directions.
Then $\mathbf{q} \geq \mathbf{md} / \log(\mathbf{m})$.

- $\mathbf{m} = \log(\mathbf{k})$ and $\mathbf{d} = \mathbf{k}$ gives

Our main tool: **New edge-iso inequality**

Suppose $x_1,\ldots,x_q$ have **m i**-twins in **d** directions.
Then $q \geq md / \log(m)$.

- **m** = log(**k**) and **d** = **k** gives

$$q \geq k \log(k)/\log(\log(k))$$

Our main tool: **New edge-iso inequality**

Suppose $\mathbf{x}_1,\ldots,\mathbf{x}_q$ have **m i**-twins in **d** directions.
Then $\mathbf{q} \geq \mathbf{md} / \log(\mathbf{m})$.

- $\mathbf{m} = \log(\mathbf{k})$ and $\mathbf{d} = \mathbf{k}$ gives

$$\mathbf{q} \geq \mathbf{k} \log(\mathbf{k})/\log(\log(\mathbf{k}))$$

- Generalization of **[Fra83]**

Other ideas

- Proofs are quite technical

Other ideas

- Proofs are quite technical
- Answer you get for single query $f(\mathbf{x}_j)$ is **not too important**

Other ideas

- Proofs are quite technical
- Answer you get for single query $f(\mathbf{x}_j)$ is **not too important**
- Analyze a specific **martingale** w/r/t

$$f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_q)$$

Other ideas

- Proofs are quite technical
- Answer you get for single query $f(\mathbf{x}_j)$ is **not too important**
- Analyze a specific **martingale** w/r/t

$$f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_q)$$

- Use McDiarmid's inequality **(with bad events)**

Open problem

Prove a separation between **adapative** and **nonadaptive** when **ε** = **const**.

Thanks!