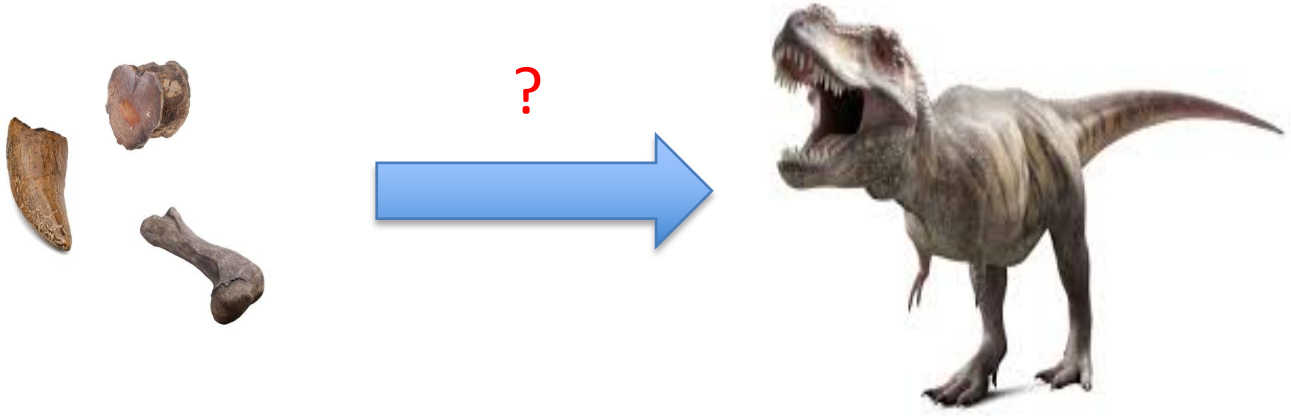
Boolean function property testing: the next generation



Rocco Servedio

Columbia University

STOC workshop

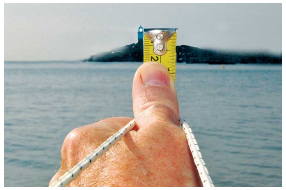
June 2026

Outline of this talk

Background on Boolean function property testing

Next generation of problems and models:

- *Distance estimation* / *Tolerant* testing



- *Relative-error* testing

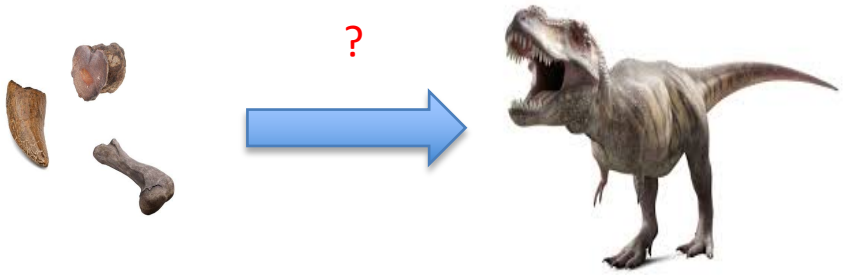
Focus on recent results, open questions, directions for future work

Background: Property testing [BLR93]

High level idea: determine whether an unknown "massive object"

- **has** some property of interest
- or **is far from having** the property

while inspecting only a *tiny fraction* of the object



Our interest: testing **Boolean functions**

Unknown $f : \{0,1\}^n \to \{0,1\}$ is a truly massive object, of size 2^n.

Background: Boolean function property testing

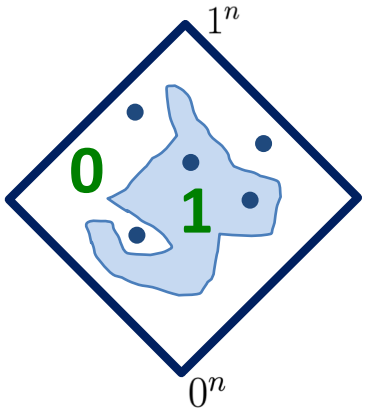
Our access to f is via **queries** for $f(x)$ at chosen values of $x \in \{0,1\}^n$

Testing algorithm

$$x_1 \in \{0,1\}^n \longrightarrow$$
$$\longleftarrow f(x_1)$$
$$\vdots \qquad \vdots \qquad \vdots$$
$$x_q \in \{0,1\}^n \longrightarrow$$
$$\longleftarrow f(x_q)$$

YES or NO

Function f



If f has the property

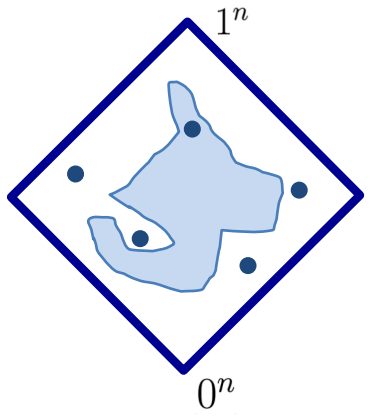
If f is “far from having the property”

Adaptive versus non-adaptive testing algorithms

Non-adaptive algorithm: all q queries made "in parallel"

$$x_1 \in \{0,1\}^n \longrightarrow$$
$$\vdots \qquad\qquad \vdots$$
$$x_q \in \{0,1\}^n \longrightarrow$$
$$\longleftarrow f(x_1) \ \ldots \ f(x_q)$$

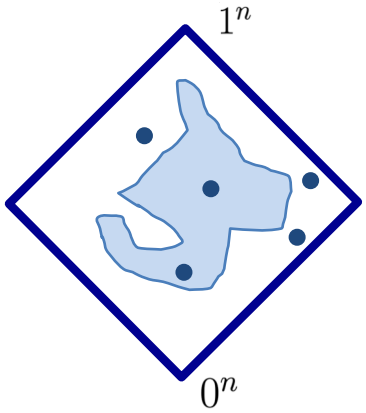
YES or NO

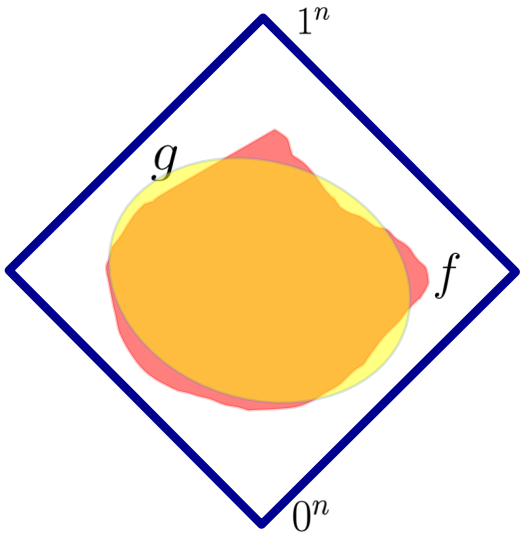


Adaptive algorithm: later queries can depends on response to earlier ones

$$x_1 \in \{0,1\}^n \longrightarrow$$
$$\longleftarrow f(x_1)$$
$$x_2 \in \{0,1\}^n \longrightarrow$$
$$\longleftarrow f(x_2)$$
$$\vdots \qquad \vdots \qquad \vdots$$

YES or NO

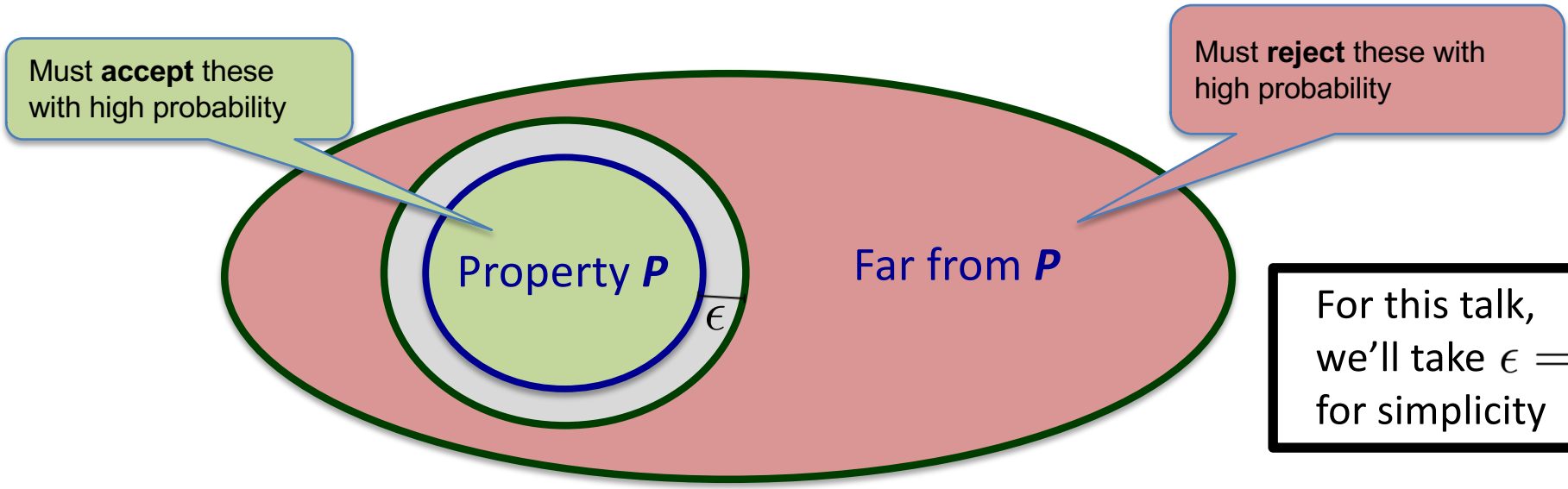




Distance to property $\mathcal{P}$ = distance to closest function with the property:

$$d(f, \mathcal{P}) = \min_{g \in \mathcal{P}} d(f, g)$$

$$d(f, g) = \frac{\square + \square}{2^n}$$

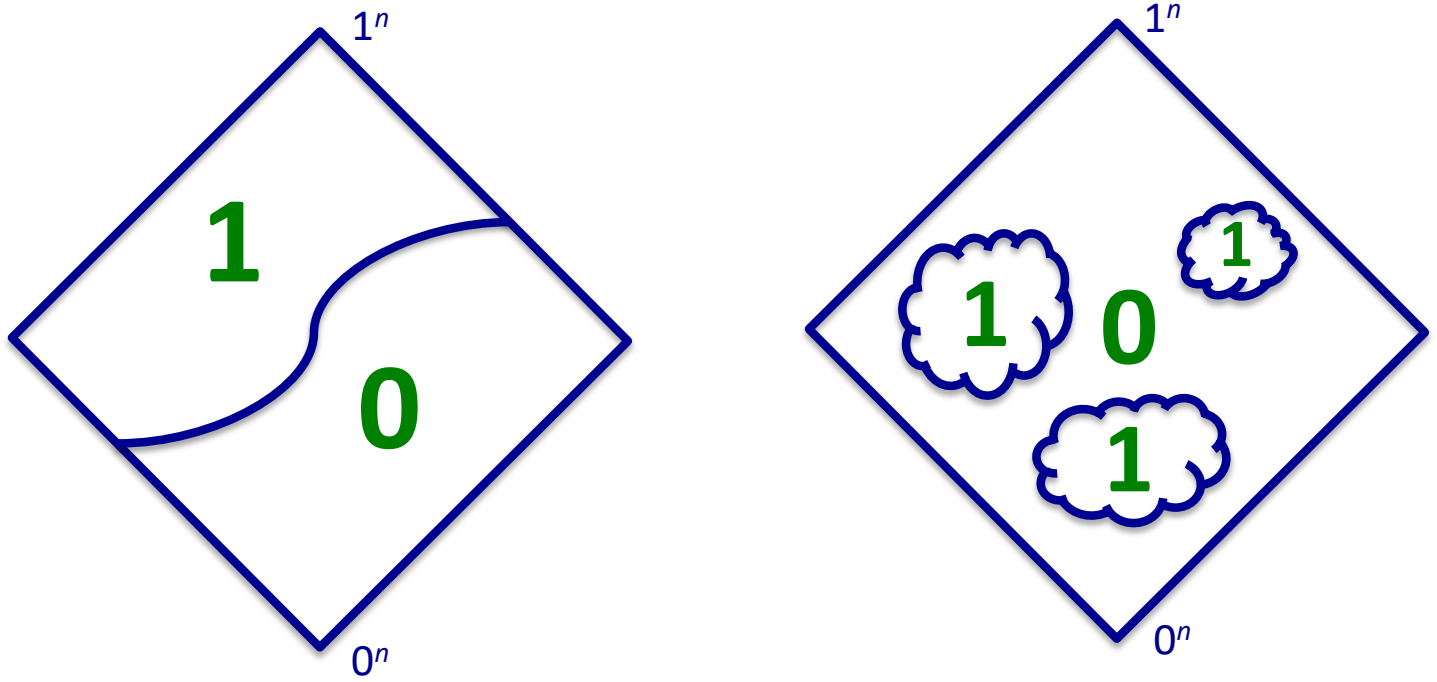


For this talk, we'll take $\epsilon = 0.01$ for simplicity

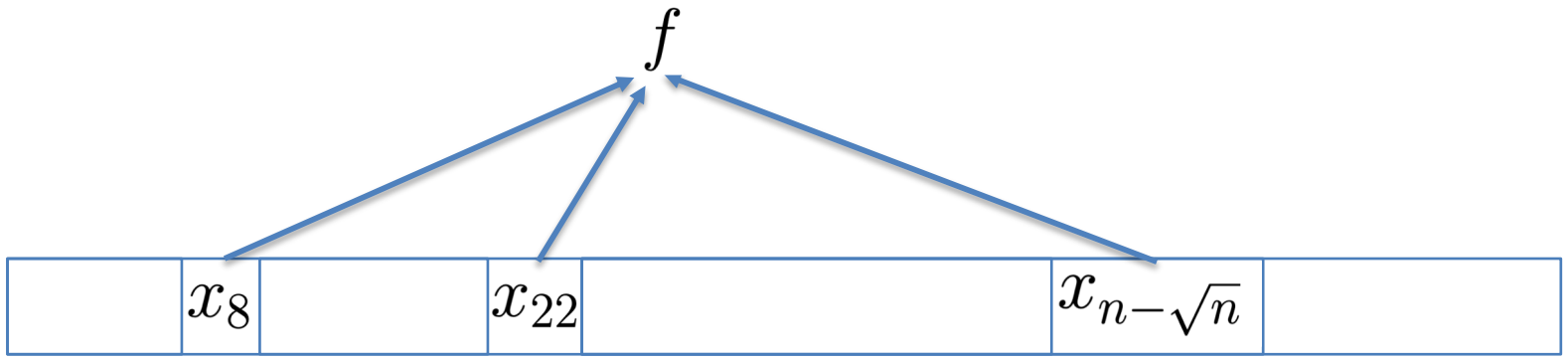
All Boolean functions $f : \{0,1\}^n \to \{0,1\}$

First-generation Boolean function property testing: some flagship results

Monotonicity



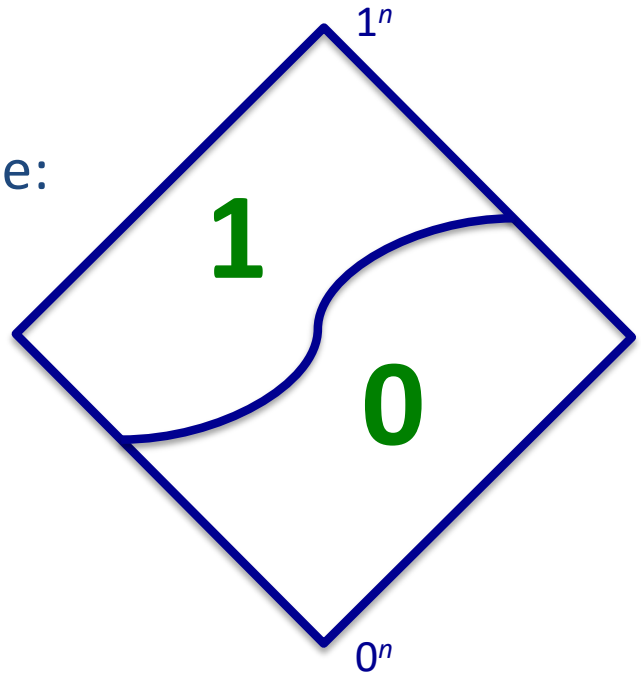
Juntas



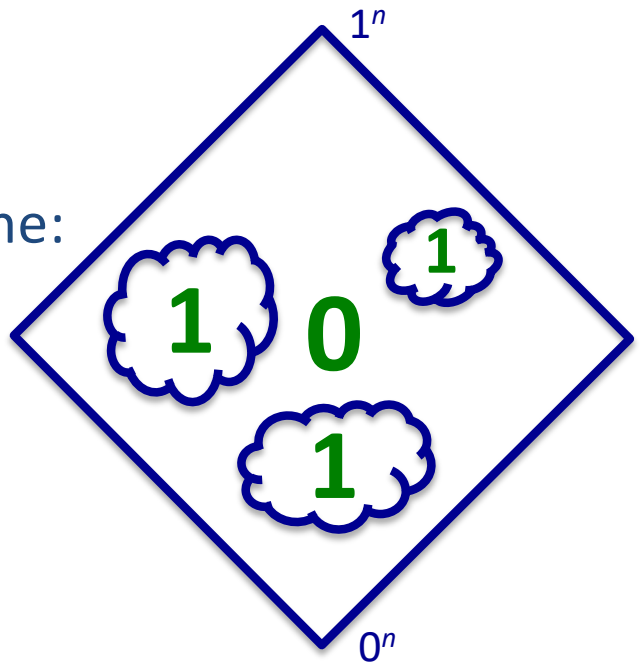
A property testing classic: **Monotonicity**

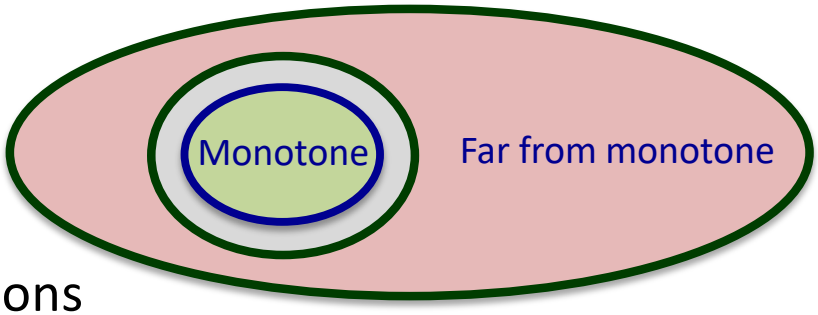
Definition: A function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \geq f(y)$ whenever $x \geq y$.

Monotone:



Far from monotone:

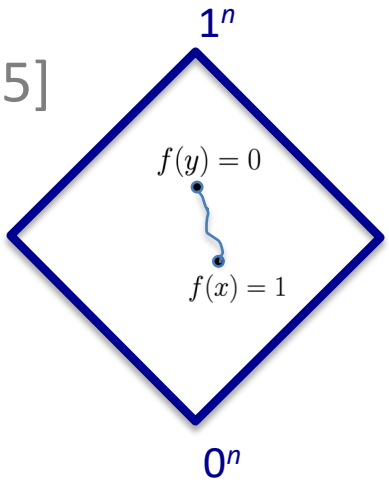




All Boolean functions

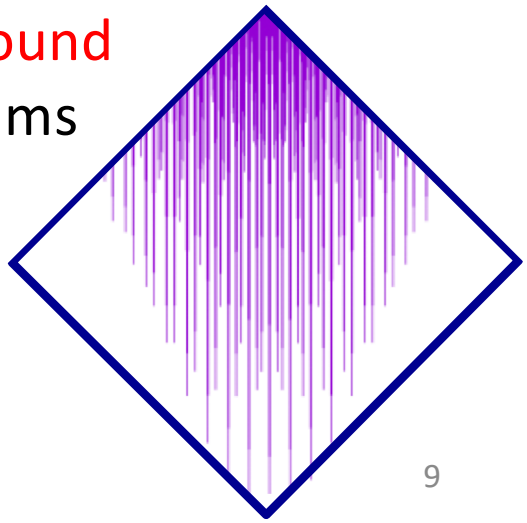
Monotonicity testing: We now have tight bounds!

Best known algorithm: $\tilde{O}(\sqrt{n})$-query algorithm [KMS15]
(non-adaptive, one-sided)



- improved earlier algorithms making $O(n), \tilde{O}(n^{7/8}), \tilde{O}(n^{5/6})$ queries
  [GGLRS00], [CS13], [CST14]
- Key ingredients: directed isoperimetric theorems about the hypercube

Best known lower bound: $n^{1/2-o(1)}$-query lower bound
[CCCPS26, M26], even for *adaptive, two-sided* algorithms

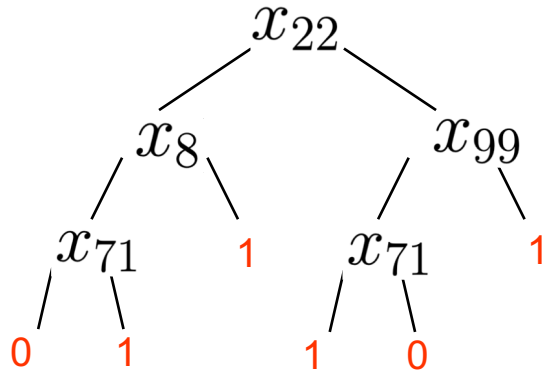


- Based on extremal monotone Boolean functions for which constant fraction of $\{0,1\}^n$ is "robustly on the boundary"

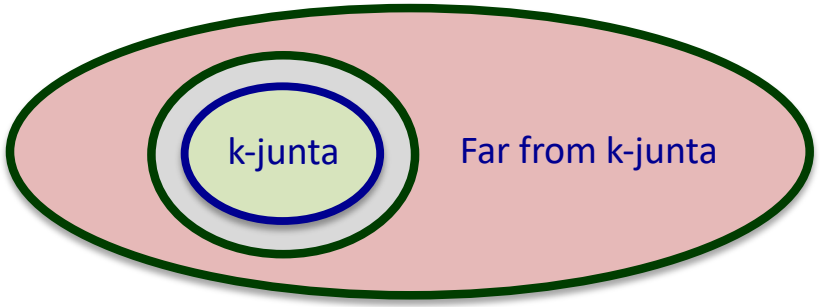
Another classic: **Juntas**

Definition: A **k-junta** is a function $f : \{0,1\}^n \to \{0,1\}$ that depends on at most k of the n input variables.

Example: Here's a 4-junta:



- $2^k + k \log n$-query algorithm is known which can *learn* k-juntas, so can *test* k-juntas with that many queries
- Is dependence on n necessary? Exponential dependence on k?



Junta testing: We have tight bounds!

queries needed: *completely independent of ambient dimension n*

Best known algorithm: $\tilde{O}(k)$-query algorithm [B09]

- Improved earlier $\tilde{O}(k^{3/2}), \tilde{O}(k^2)$ -query algorithms [B08,FKRSS04]
- For *non-adaptive* algorithms, $\tilde{O}(k^{3/2})$ -query algorithm given in [B08]

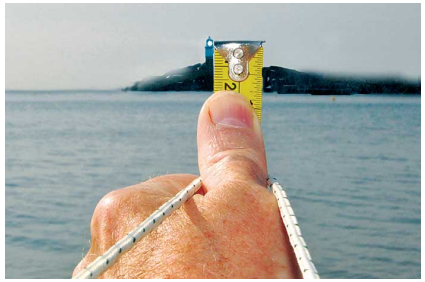
Best known lower bound: $\Omega(k)$-query lower bound [CG04]

- For *non-adaptive* algorithms, $\tilde{\Omega}(k^{3/2})$ -query lower bound is known [CSTWX17]

The take-away so far: Some of the classic property testing problems are now well-understood in the original model.

Rest of this talk: Much less is known in other frameworks!

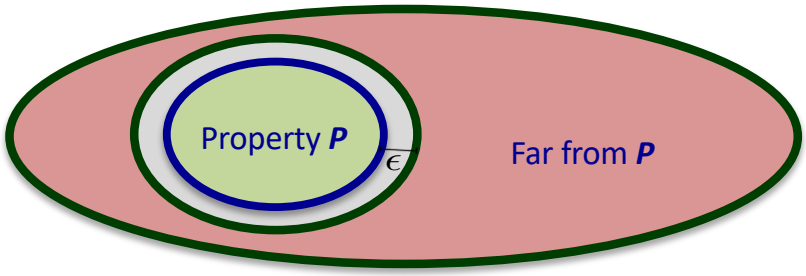
- *Tolerant* testing / distance estimation



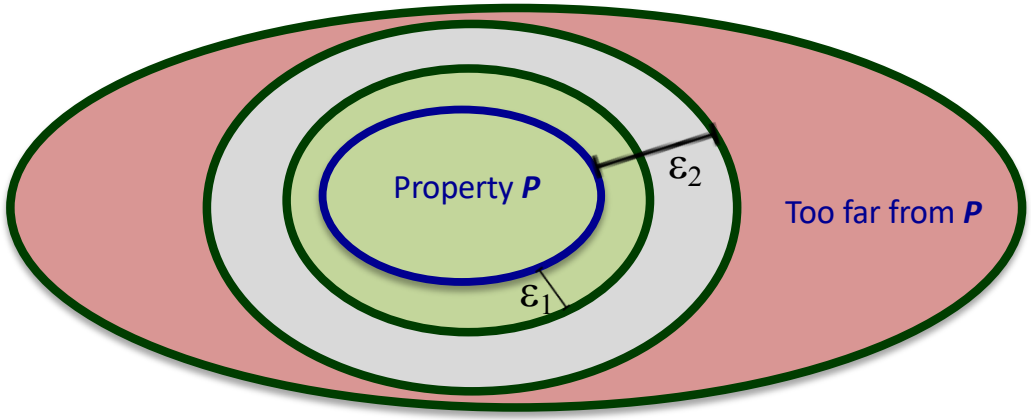
- *Relative-error* testing

Making testing harder: Tolerant testing [PRR04]

Usual testing setup: must distinguish
$d(f, \mathcal{P}) = 0$ vs $d(f, \mathcal{P}) > \epsilon$



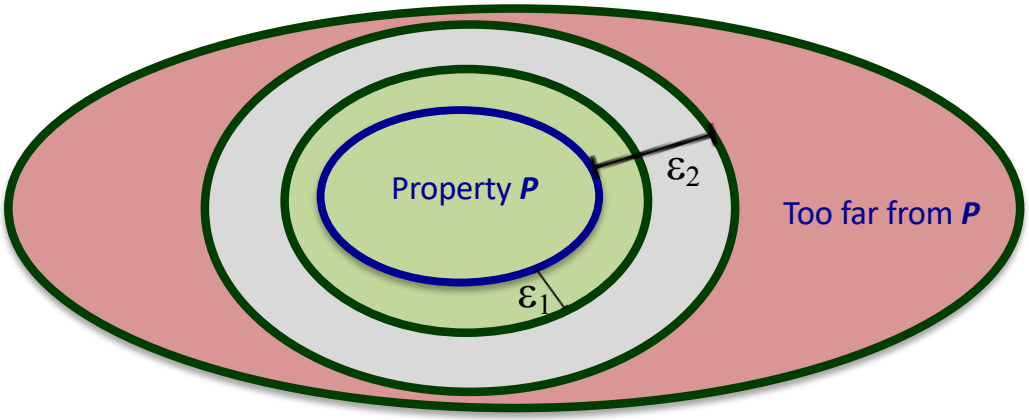
The *tolerant testing* setup:
must distinguish $d(f, \mathcal{P}) \leq \epsilon_1$
vs $d(f, \mathcal{P}) \geq \epsilon_2$

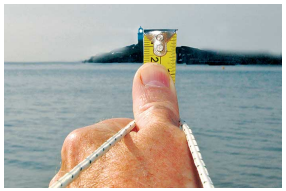


Harder algorithmic problem: less structure in yes-case.

Tolerant testing and distance estimation

"Fully tolerant" tester: can handle any $\varepsilon_1 < \varepsilon_2$



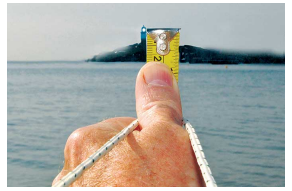


Equivalent to *distance estimation:* given black-box access to f, estimate distance to property $\mathcal{P}$, i.e. distance to closest function with the property:

$$d(f, \mathcal{P}) = \min_{g \in \mathcal{P}} d(f, g)$$

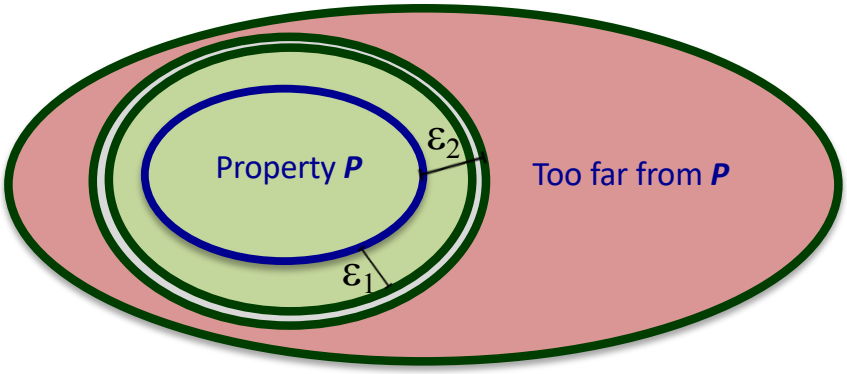
- Well motivated: may be unrealistic to hope that "good" functions *perfectly* have the property of interest.
- Surge of recent research progress

Tolerant monotonicity testing: recent lower bound progress!

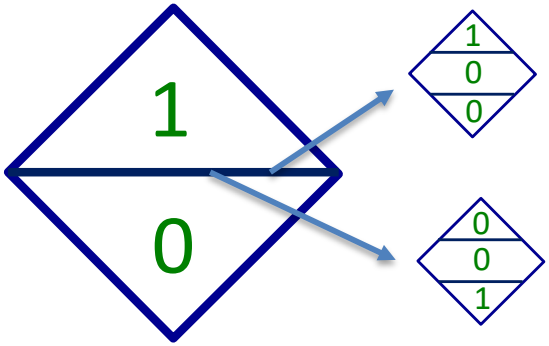


Folklore upper bound: $2^{\tilde{O}(\sqrt{n})}$ queries (nonadaptive) via agnostic learning [BT94,KKMS05]

[PRW19]: $2^{\Omega(n^{0.01})}$ -query non-adaptive lower bound if $\epsilon_2 - \epsilon_1$ is **tiny**



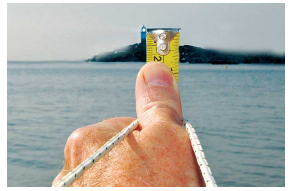
Key idea: "hiding" somewhat-monotone structure using boundary of the **majority function**



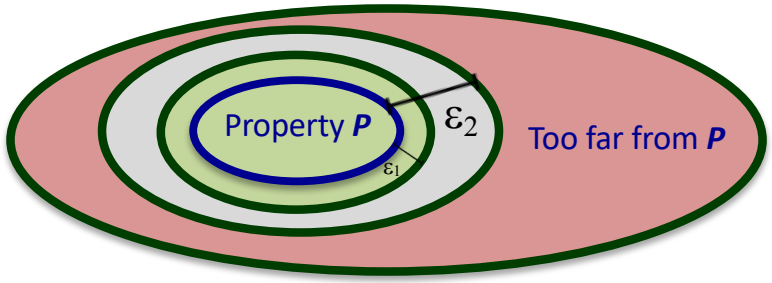
hypercube over unknown set of n/2 "control variables"

hypercubes over the other n/2 "action variables"

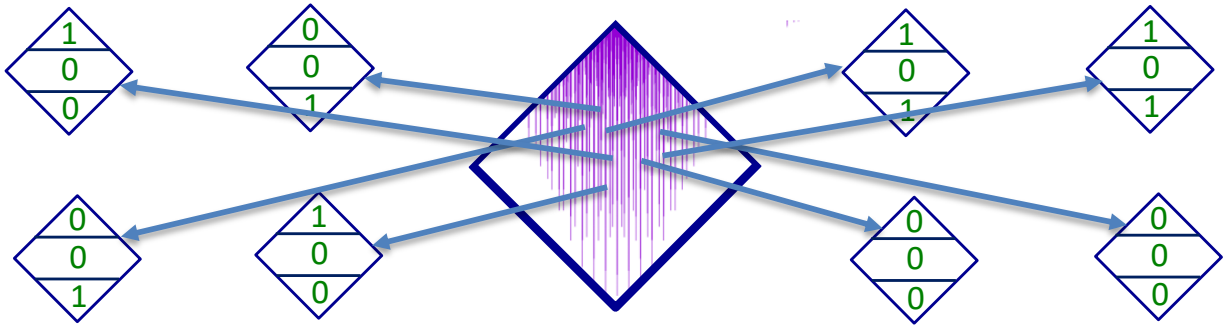
Tolerant monotonicity testing: recent lower bound progress!



[CDLNS24]: $2^{\Omega(n^{1/4})}$-query non-adaptive l.b. even if $\epsilon_2 - \epsilon_1$ is **constant**



Hides somewhat-monotone structure using the boundary of the **extremal monotone function** mentioned earlier



hypercubes over the other $\sqrt{n}$ "action variables"

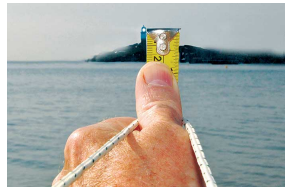
hypercube over unknown set of $n - \sqrt{n}$ "control variables"

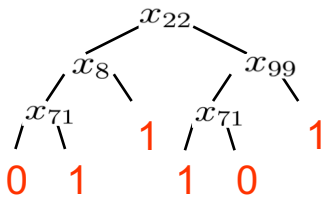
What we don't have for tolerant monotonicity testing: positive results!

- As we said earlier, via learning, get $2^{\tilde{O}(\sqrt{n})}$-query tester
  - [LRV22]: better dependence on $\varepsilon_1, \varepsilon_2$

- Can we do better in the dependence on n?

Tolerant junta testing: recent algorithmic progress!



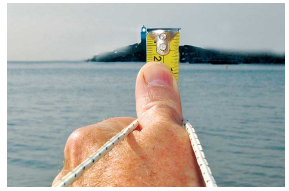


Initial work gave nontrivial but not fully tolerant algorithms [BCELR16]

More recently, fully tolerant testing algorithms have been achieved:

- [DMN19] breakthrough: 2^k -query algorithm (*adaptive*)
- [ITW21]: $2^{\tilde{O}(\sqrt{k})}$ *adaptive* queries
- [NP24]: $2^{\tilde{O}(\sqrt{k})}$ *nonadaptive* queries (tight for nonadaptive!)
- [CPS26]: $2^{\tilde{O}(k^{1/3})}$ *adaptive* queries

[DMN19]: 2^k-query adaptive distance estimation



Let $g : \{-1,1\}^n \to \{-1,1\}$ be the closest k-junta to f.
[DMN19] *learns* g to high accuracy (up to knowing which variables it depends on).

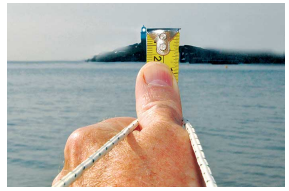
- Variables with very low influence $(\ll 1/k)$ don't matter. Say variable x_i is *interesting* if it is influential* in f, i.e. $\sum_{S \ni i} \widehat{f}(S)^2$ is large.

Key goal: Get oracle access to the interesting variables.

Accomplished using

- Random restrictions of f to ``knock down'' Fourier weight to level 1: restriction f_ρ has $|\widehat{f_\rho}(i)|$ large.
- Hastad-inspired operator that simulates oracle access to x_i

[DMN19]:2^k-query adaptive distance estimation, cont.



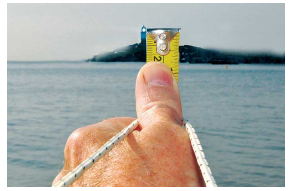
[DMN19] give a way to obtain a set of poly(k) "coordinate oracles" which include oracles for all of the *interesting* variables.

Net effect: It's enough to solve junta testing problem over $\{-1, 1\}^{\mathrm{poly}(k)}$

Now can use brute force:

- Try every size-k subset S of these $\mathrm{poly}(k)$ coordinate oracles
  - For each such S, empirically estimate distance of best junta-over-S to f
- Algorithm's output: minimum estimated distance over all S

[NP24]: $2^{\tilde{O}(\sqrt{k})}$-query nonadaptive distance estimation



Builds on the [DMN19] "coordinate oracle" ideas:
it's enough to solve distance-to-k-junta estimation for (n=poly(k))-variable Boolean functions. Here we'll pretend n=2k.

Simple but crucial insight: Junta distance estimation is closely connected to mean estimation:

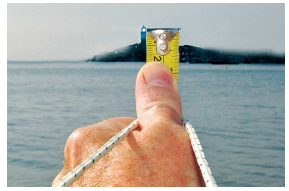
Indeed, a connection between the mean of a function and distance to junta is direct as for a set $S \subseteq [n]$ with $|S| = k$, we have that

$$\mathrm{dist}(f, \mathcal{J}_S) = \frac{1}{2} - \frac{1}{2} \mathop{\mathbf{E}}_{\boldsymbol{x} \in \{\pm 1\}^S} \left[\left| \mathop{\mathbf{E}}_{\boldsymbol{y} \in \{\pm 1\}^{[n] \setminus S}} [f(\boldsymbol{x} \sqcup \boldsymbol{y})] \right| \right]. \tag{1}$$

Key new idea in [NP24]: r-*local estimator* for $|\mathbf{E}[f]|$.

- Takes as input f's values *only on radius-*r Hamming ball around x
- Outputs estimate of $|\mathbf{E}[f]|$

[NP24]: $2^{\tilde{O}(\sqrt{k})}$-query nonadaptive distance estimation, cont.



r-local estimator for $|\mathbf{E}[f]|$:

- Takes as input f's values *only on radius-r* Hamming ball around x
- Outputs estimate of $|\mathbf{E}[f]|$

Key result from [NP24]: Can do this with $r = \tilde{\Theta}(\sqrt{n})$

- Extremal polynomials, "approximate inclusion-exclusion" machinery

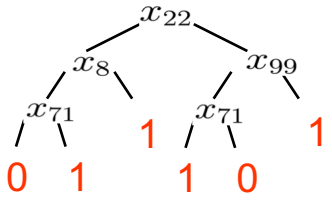
Using this for junta testing:

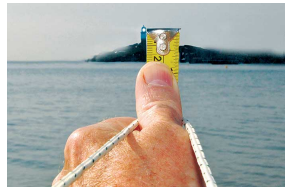
- Try all k-variable subsets $S \subset [2k]$
- For each one, run local estimator to estimate

$$\mathrm{dist}(f, \mathcal{J}_S) = \frac{1}{2} - \frac{1}{2} \mathop{\mathbf{E}}_{\boldsymbol{x} \in \{\pm 1\}^S} \left[\left| \mathop{\mathbf{E}}_{\boldsymbol{y} \in \{\pm 1\}^{[n] \setminus S}} [f(\boldsymbol{x} \sqcup \boldsymbol{y})] \right| \right].$$

Crucially, this can be done for all S using same set of $2^{\tilde{\Theta}(\sqrt{k})}$ nonadaptive queries!

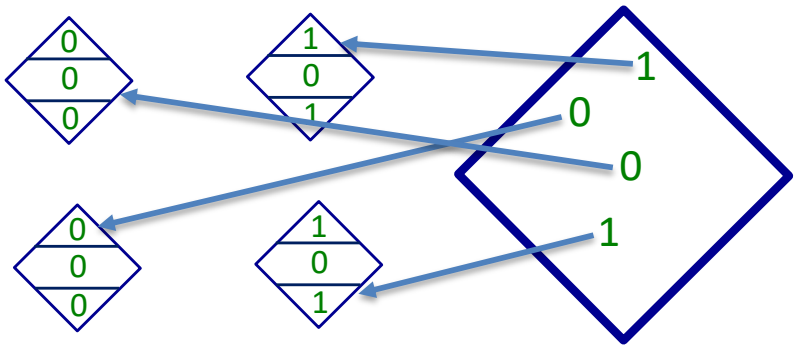
Tolerant junta testing: recent progress on lower bounds!





Non-adaptive problem: settled!

- [CDLNS24]: $2^{\Omega(\sqrt{k})}$-query lower bound (sharpened by [NP24])
  - Construction is similar in spirit to tolerant monotonicity lower bound



Structured functions over n/2-dimensional hypercube of "action variables";
distance to constant differs in yes-case vs no-case

Random function over n/2-dimensional hypercube of "control variables"

Adaptive lower bounds: harder, but some progress:

- [CP23]: $k^{\Omega(\log(1/\varepsilon))}$-query lower bound, where $\varepsilon = \varepsilon_2 - \varepsilon_1$

Future work, tolerant testing:

- **Narrowing the gap** for adaptive tolerant junta testing?
  - $2^{\tilde{O}(k^{1/3})}$ -query algorithm, but lower bound is only $k^{\Omega(\log(1/\varepsilon))}$

- Tolerant testers for **other classes**?

Outline of this talk

- Distance estimation / Tolerant testing
- **Relative-error testing**

Relative-error testing

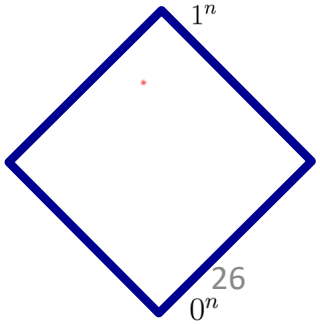
Motivation: what about "small" massive objects?

10^{17} cells 

 5 x 10^6 cells

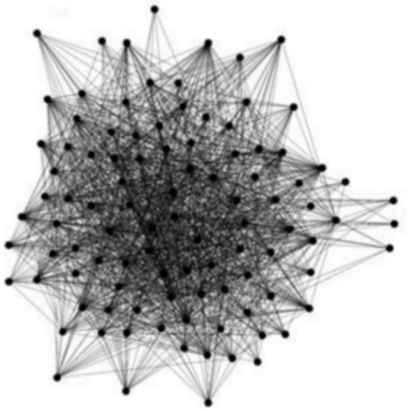
Many interesting Boolean functions $f : \{0,1\}^n \to \{0,1\}$ output 1 on only a *vanishingly small fraction* of all inputs in $\{0,1\}^n$, e.g., on $2^{n/2}$ inputs (a $2^{-n/2}$ fraction)

But **all "sparse" functions are close to the constant-0 function** – at least, vis-à-vis our usual distance notion…

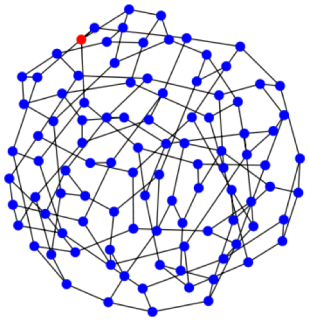


Inspiration from graph property testing

"Dense graph" property testing model:
Distance between two N-node graphs is fraction of all $\binom{N}{2}$ potential edges that need to be added or deleted to transform one graph into the other.



Not suitable for testing *sparse graphs* with $o(N^2)$ edges.

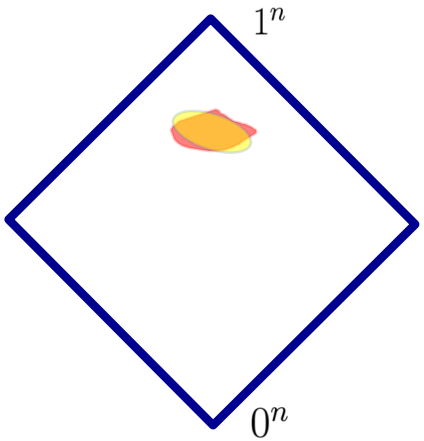


Alternative model for sparse graphs [GR02]: for d-regular graphs, distance between two graphs measured relative to dN, not $\binom{N}{2}$.

Relative-error testing over $\{0,1\}^n$

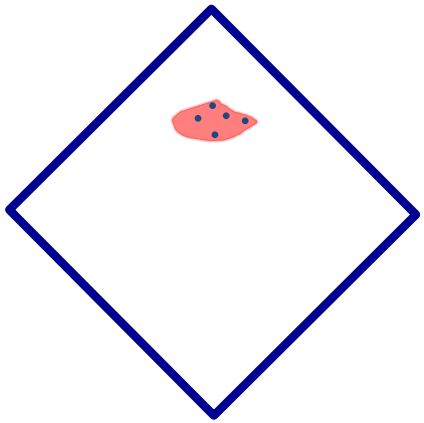
Standard model distance notion: $\text{ham-dist}(f,g) := \dfrac{|f^{-1}(1) \triangle g^{-1}(1)|}{2^n}$

- **Relative-error** distance notion: $\text{rel-dist}(f,g) := \dfrac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|}$



$$d(f,g) = \frac{\square + \square}{\square + \square}$$

- Algorithm also has access to independent uniform satisfying assignments of f



Early work in similar model by [RT10]: testing sparse image properties over domain $[m] \times [m]$

How does this relate to the standard model?

Comparison with standard model:

- Relative-error **never easier** than standard-model testing

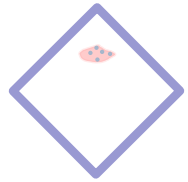
- Relative-error **can be much harder** than standard-model testing:
  - [CDHLNSY24]: There is a simple property that's trivially testable (with zero queries!) in standard model, but requires $2^{\Omega(n)}$ queries in relative-error model.

Relative-error testing over $\{0,1\}^n$

Relative-error distance notion:

$$\text{rel-dist}(f,g) := \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|}$$

Algorithm also has access to uniform sat assts



What's the deal with relative-error testing
for our favorite properties?

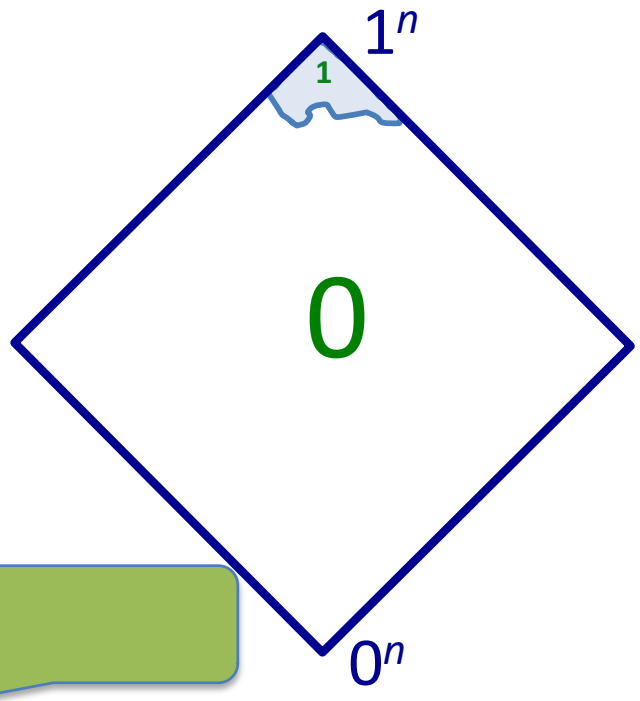
Some recent work on relative-error testing of well-studied properties

Some results to date:

1. Monotonicity
2. Halfspaces
3. Juntas and junta subclasses
4. Conjunctions and Decision Lists
5. DNF formulas

Relative-error monotonicity testing

Question: How easily can we test *sparse functions* for monotonicity?

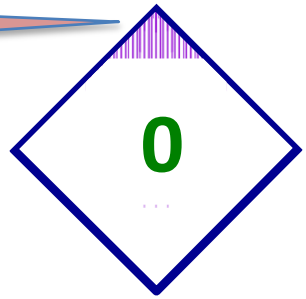


Let S = number of satisfying assignments to unknown f.

Never worse than n --- **at most polynomially harder** than standard-model testing

[CDHLNSY24]: Tester that uses $O(\log S)$ samples + queries

With sparsity, can "confine" hard functions to a small region of the cube where there is "less monotonicity to detect"

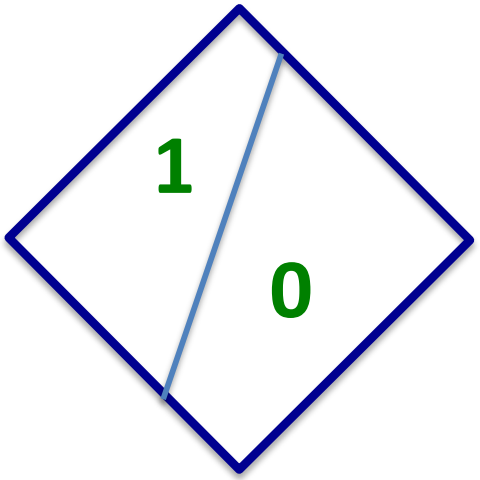


[CDHLNSY24]: $\Omega((\log S)^{2/3})$ samples + queries needed, even for adaptive algs.
(Improved to $\Omega((\log S)^{1-c})$ by new [CCCPS26] result!)

Testing halfspaces

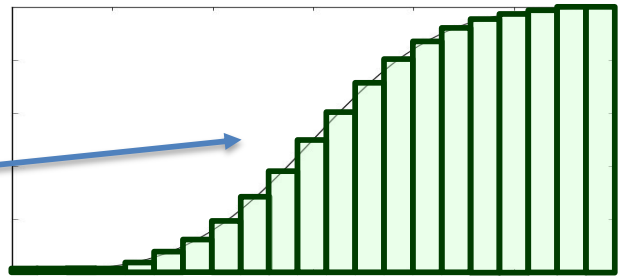
Another basic class: halfspaces over $\{0,1\}^n$

$$f(x) = \mathbf{1}[w_1 x_1 + \cdots + w_n x_n \geq \theta]$$



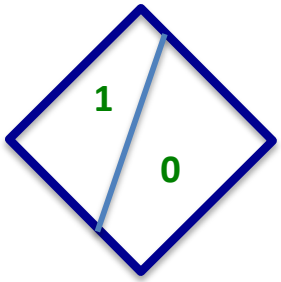
[MORS10]: In standard model, halfspaces are testable with *constantly many* queries, independent of n

Algorithm combines junta-testing techniques and central limit theorems



Relative-error halfspace testing

$$f(x) = \mathbf{1}[w_1 x_1 + \cdots + w_n x_n \geq \theta]$$



Relative-error model: Some results are known [CDHNSY24]:

- $\tilde{O}(n^{1/2})$ -query alg. for halfspaces over $\mathbb{R}^n$ under *Gaussian distribution* (with some fine print…)
  - Ingredients: robust extremal isoperimetric properties of halfspaces under Gaussian distribution, Gaussian standard-model halfspace testing
- $\tilde{\Omega}(\log n)$-query lower bound for halfspaces over $\{0,1\}^n$

Much harder than standard setting - big contrast with monotonicity!

Intuition: central limit theorems give *absolute-error* bounds, which are very poor in terms of relative-error performance "way out at the tails."

Testing juntas and junta subclasses: Standard model

Many Boolean functions can naturally be viewed as *subclasses* of k-juntas:

- Decision trees of size k
- Branching programs of size k
- Boolean circuits of size k
- Etc

Standard model: Intensively studied.

- Testing k-juntas: $\tilde{O}(k/\varepsilon)$-query algorithms [B09]
- Testing subclasses of k-juntas: $\tilde{O}\left(\frac{k+\log|P^*(k)|}{\varepsilon}\right)$ queries [DLMORSW07,B20]
  - "testing by implicit learning"

Relative-error testing juntas and junta subclasses

Relative-error model: [CPPS25] showed testing juntas, junta subclasses *with relative error* is **no harder*** than in standard model!

- Testing k-juntas: $\tilde{O}(k/\varepsilon)$-query algorithm
  - Some twists on distribution-free junta testing approach of [B20]
- Testing subclass of k-juntas: $\tilde{O}\left(\frac{k \cdot \log |P^*(k)|}{\varepsilon}\right)$ queries
  - More twists: need high accuracy "even where D doesn't put any weight"

Testing Conjunctions and Decision Lists: Standard model

Conjunctions: We all know what a conjunction is.

Decision lists: $x_6 \rightarrow x_4 \rightarrow \ldots\ldots\ldots\ldots \rightarrow x_{11} \rightarrow 1$

$\downarrow$ $\downarrow$ $\downarrow$

1 0 0

Standard model:

- $O(1/\varepsilon)$-query-algorithm for conjunction testing [PRS02]

- $\tilde{O}(1/\varepsilon)$-query algorithm for decision list testing [DLMORSW07,B20]
  - "testing by implicit learning" for functions that are *approximated* by juntas (every decision list is ε-close to a $\log(1/\varepsilon)$-junta)

Relative-error testing of conjunctions and decision lists

Relative-error model: [CPPS25] showed testing conjunctions and decision lists is **no harder** than in standard model!

- Testing conjunctions: $O(1/\varepsilon)$-query algorithm

- Testing decision lists: $\tilde{O}(1/\varepsilon)$ queries
  - Uses new (but simple) decomposition of "sparse" decision lists:

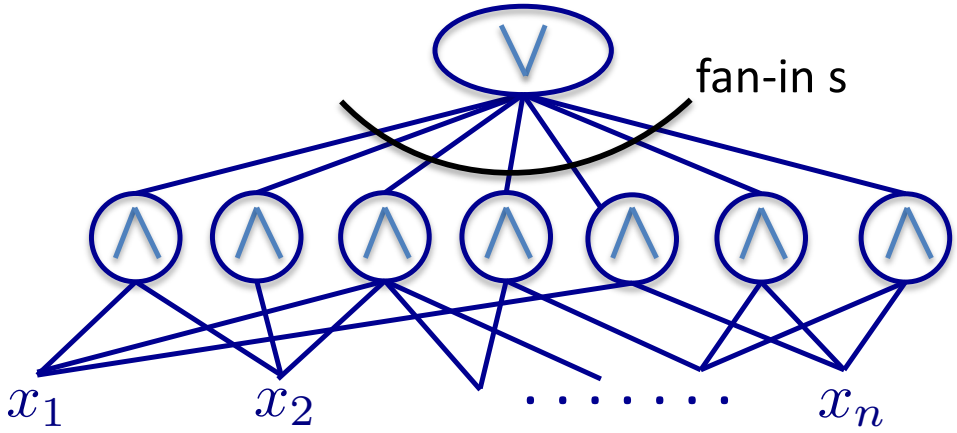
$x_6 \to x_4 \to x_7 \ldots\ldots\ldots \to x_1 \to x_3 \to \ldots \to x_{11} \to 1$

$\downarrow 0 \quad \downarrow 0 \qquad\qquad \downarrow 0 \quad \downarrow 1 \qquad\qquad \downarrow 0$

"head" is a (possibly long) conjunction,
"tail" is a decision list that's ε-close to a $\log(1/\varepsilon)$-junta

Testing DNF formulas: Standard model

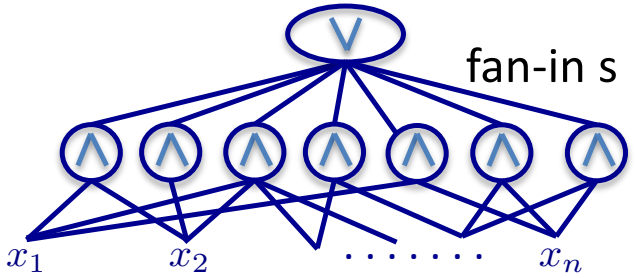
Definition: An **s-term DNF formula** is an s-way OR of ANDs (of any size) of literals:



Standard model:

- $\tilde{O}(s/\varepsilon)$-query-algorithm for testing s-term DNFs [DLMORSW07,B20]
  - "testing by implicit learning" for functions that are *approximated* by juntas (every s-term DNF is ε-close to a $s\log(s/\varepsilon)$-junta)

Testing DNF formulas with relative error



Relative-error model: [CPPS25] showed testing s-term DNF is **at most polynomially harder** than in standard model!

- Testing s-term DNF: $\mathrm{poly}(s/\varepsilon)$-query algorithm
  - New structural decomposition of s-term DNF into “clusters” (sub-DNFs)
    - This decomposition is not known to the tester!
  - “Factor out” a long conjunction from each cluster: what’s left is a “junta DNF”
  - Tackle each ”junta DNF” using relative-error-for-junta-subclass technology
    - Presence of other clusters introduces significant complications…

Future work, relative-error testing:
General conditions that suffice for relative-error testability?

Goal: Unified way to go beyond junta subclasses?

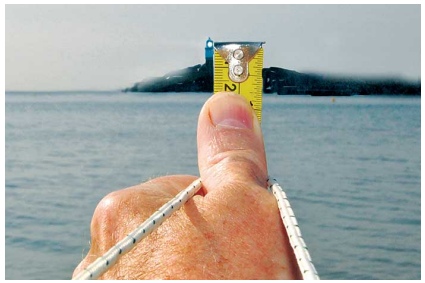
- Results for conjunctions, decision lists, DNFs, are somewhat ad hoc…

- …is there an analogue to standard-model "testing by implicit learning" results for functions *approximated by* juntas?

Wrapping up

New(ish) models of Boolean function property testing:

- *Tolerant* testing / distance estimation



- *Relative-error* testing

Clean and appealing open questions

Question: Can these models be combined:
Tolerant relative-error testing?

Thank you!