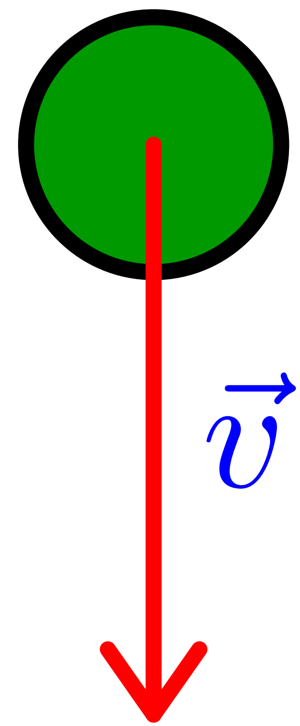
Testing for Concise Representations

Krzysztof Onak
MIT, CSAIL

Joint work with **Ilias Diakonikolas, Homin Lee, Kevin Matulef, Ronitt Rubinfeld, Rocco Servedio, Andrew Wan**

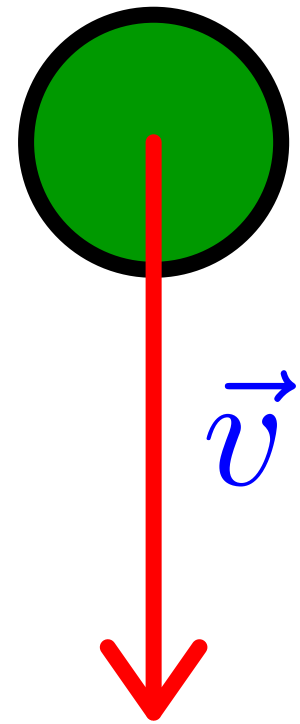
Physics Review

Free fall:



Physics Review

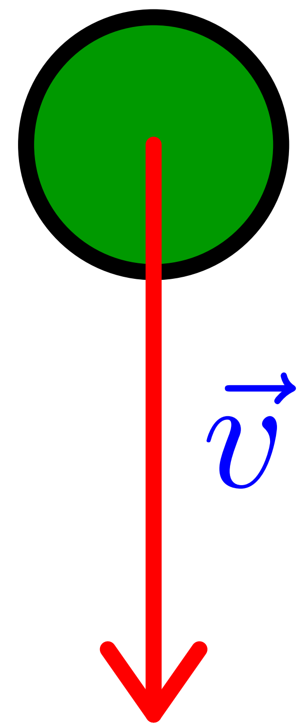
Free fall:



All objects fall at constant speed.

Physics Review

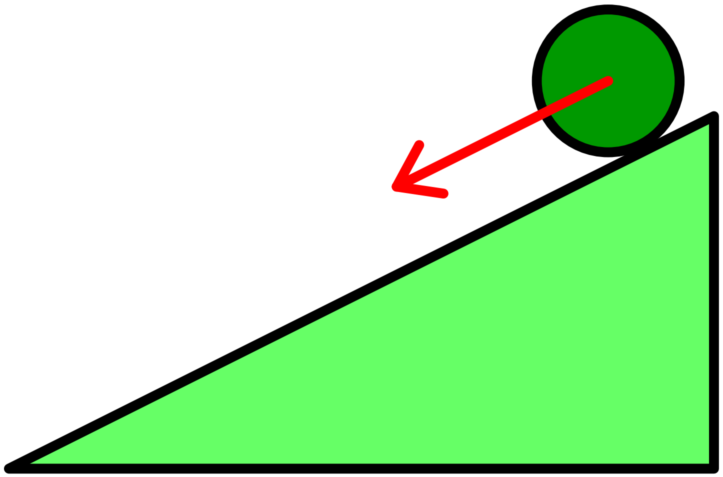
Free fall:

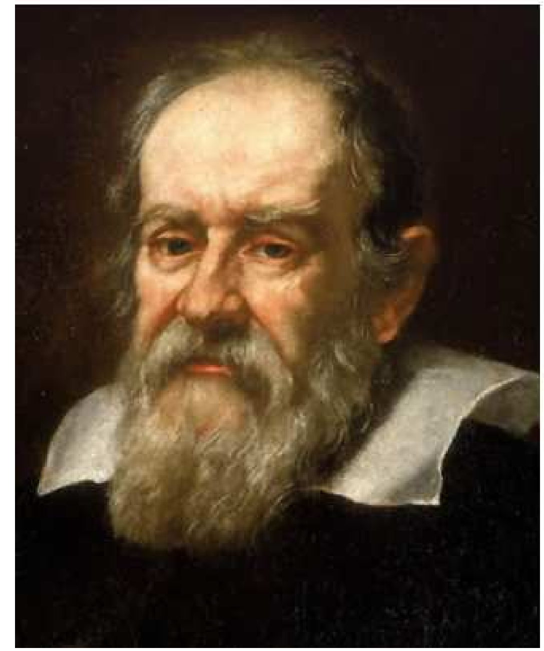


All objects fall
at constant speed.

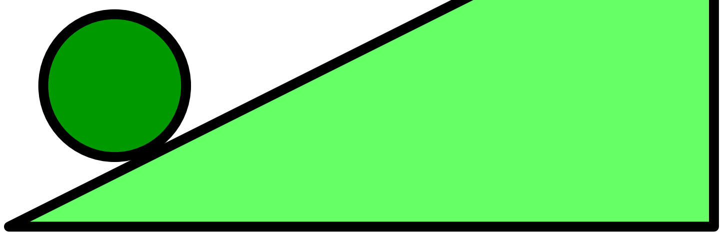
Aristotle

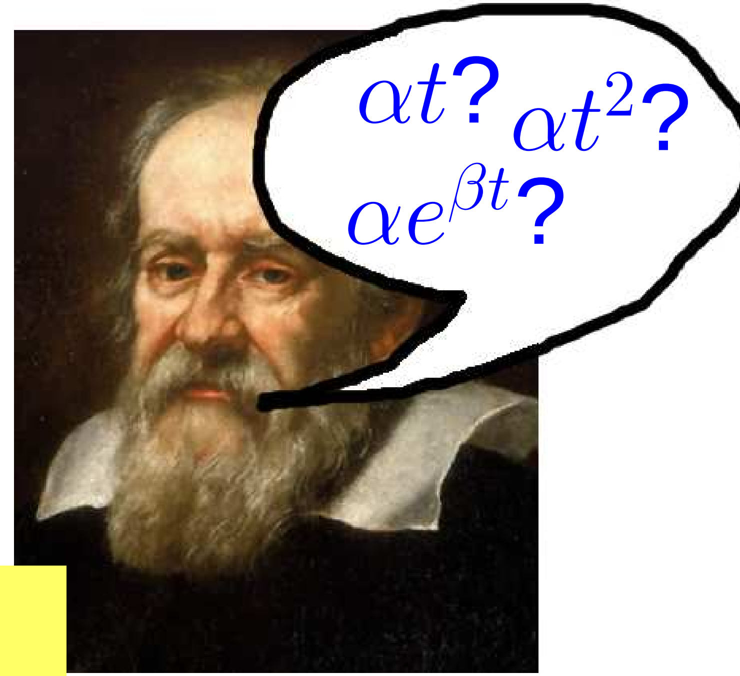
A Physicist Discovering the World

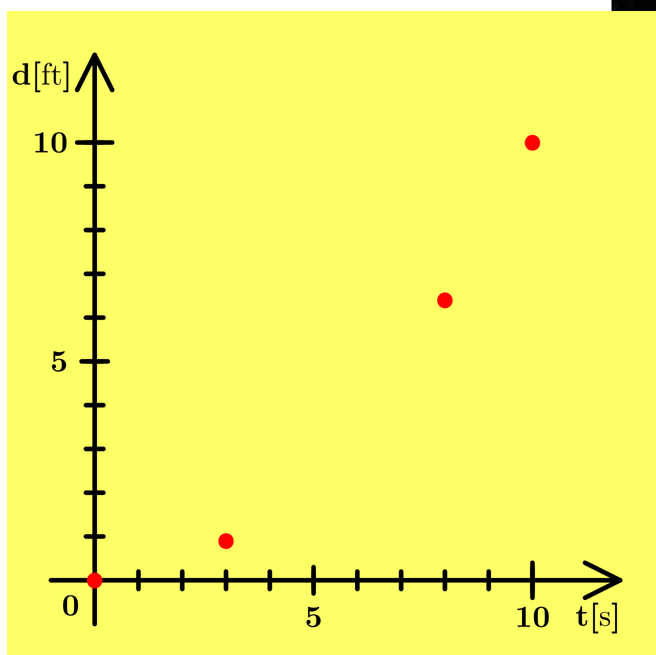




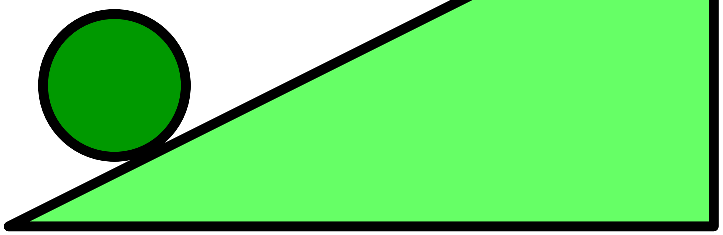
A Physicist Discovering the World

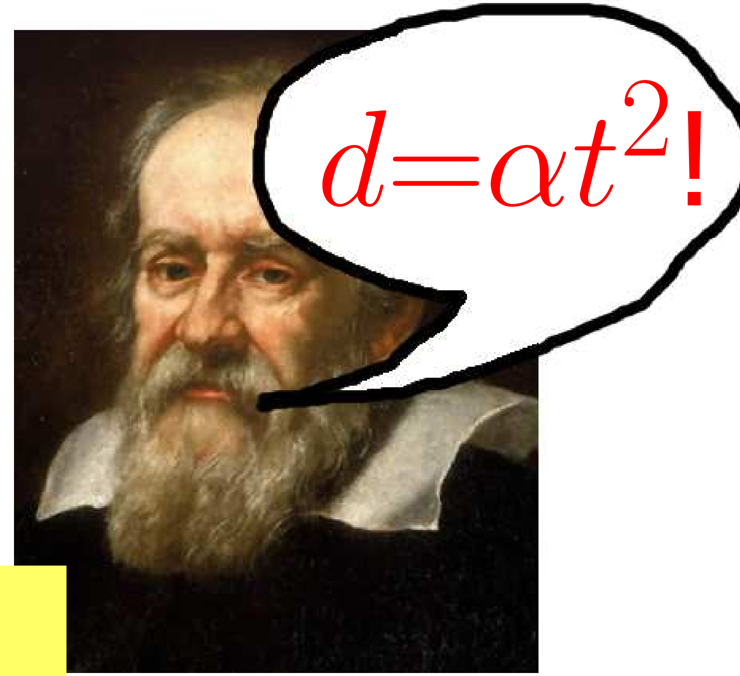


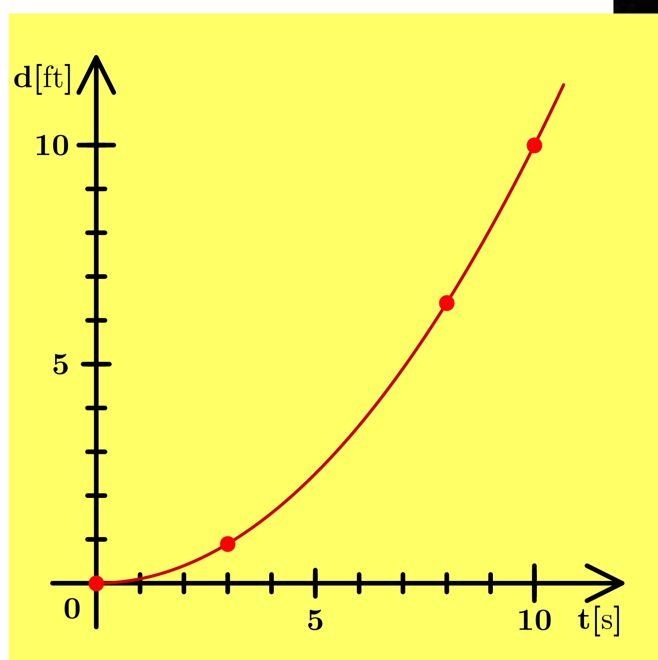




A Physicist Discovering the World





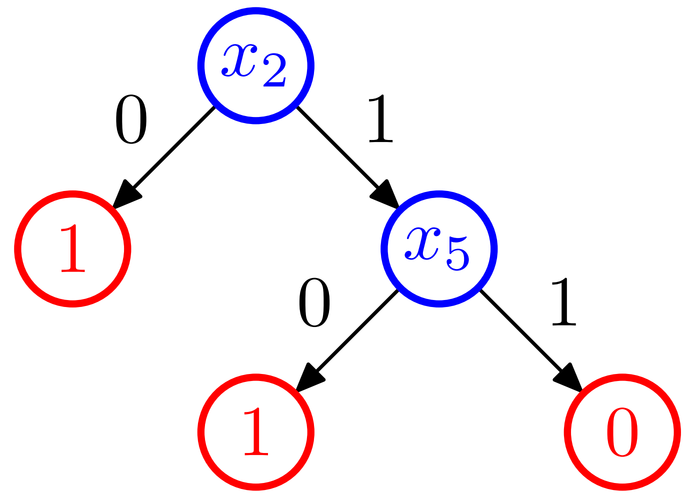


Computer Scientist's Discrete World

- Access to an unknown function $f : \{0,1\}^n \to \{0,1\}$.

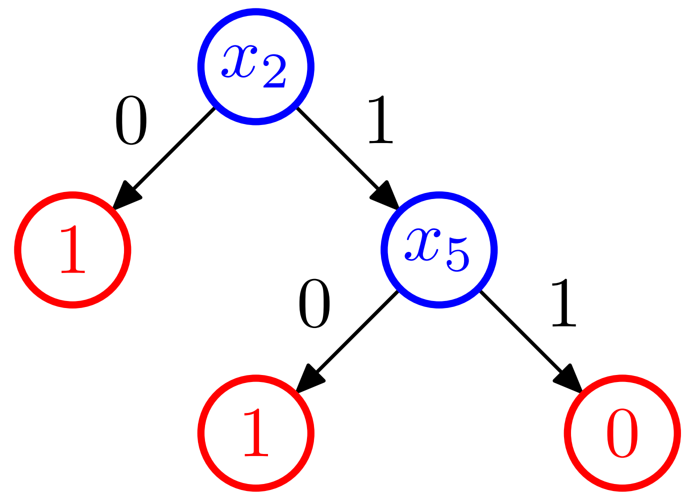
Computer Scientist's Discrete World

- Access to an unknown function $f : \{0,1\}^n \to \{0,1\}$.
- Can it be represented as
  - a small decision tree?
  - a small DNF formula?
  - a small Boolean circuit?



Computer Scientist's Discrete World

- Access to an unknown function $f : \{0,1\}^n \to \{0,1\}$.
- Can it be represented as
  - a small decision tree?
  - a small DNF formula?
  - a small Boolean circuit?



- This could help you choose the right representation, if you wanted to learn the function.

The Model

- Query access to a function $f : \Omega^n \to X$.

The Model

- Query access to a function $f : \Omega^n \to X$.
- A class C of functions.

The Model

- Query access to a function $f : \Omega^n \to X$.
- A class C of functions.
- Want an algorithm that
  - outputs **YES** w.p. $\geq 2/3$, if $f \in C$
  - outputs **NO** w.p. $\geq 2/3$, if f disagrees with each function in C on at least an ϵ-fraction of inputs (i.e. f is ϵ-far from any function in C)

The Model

- Query access to a function $f : \Omega^n \to X$.
- A class C of functions.
- Want an algorithm that
  - outputs **YES** w.p. $\geq 2/3$, if $f \in C$
  - outputs **NO** w.p. $\geq 2/3$, if f disagrees with each function in C on at least an ϵ-fraction of inputs (i.e. f is ϵ-far from any function in C)
- Primary objective: minimize the number of queries

Plan of the Talk

- Basic definitions
- Testing vs. learning
- Previous results and our results
- Review of the junta test
- Our techniques:
  - Junta test + learning
  - Classes of functions close to juntas
  - Non-Boolean ranges
- Open questions

Political Systems

- An alternative look at a function $f : \{0,1\}^n \to \{0,1\}$

Political Systems

- An alternative look at a function $f : \{0,1\}^n \rightarrow \{0,1\}$
- Final decision is a function of input values

Political Systems

- An alternative look at a function $f : \{0,1\}^n \to \{0,1\}$
- Final decision is a function of input values
- Dictatorship: single variable decides

$f = x_i$ or $f = \neg x_i$

Political Systems

- An alternative look at a function $f : \{0,1\}^n \to \{0,1\}$
- Final decision is a function of input values
- Dictatorship: single variable decides

$$f = x_i \text{ or } f = \neg x_i$$

- k-Junta: k variables decide

$$f = f'(x_{i_1}, \ldots, x_{i_k})$$

Political Systems

- An alternative look at a function $f : \{0,1\}^n \to \{0,1\}$
- Final decision is a function of input values
- Dictatorship: single variable decides

$$f = x_i \text{ or } f = \neg x_i$$

- k-Junta: k variables decide

$$f = f'(x_{i_1}, \ldots, x_{i_k})$$

- direct democracy $\approx$ majority

Testing vs. Learning

- Suppose you know f is a dictatorship.
  Need $\Omega(\log n)$ queries to learn the relevant variable.

Testing vs. Learning

- Suppose you know f is a dictatorship.
  Need $\Omega(\log n)$ queries to learn the relevant variable.

- $O(1/\epsilon)$ queries suffice to test if f is a dictatorship
  [Parnas, Ron, Samorodnitsky 2001]

Testing vs. Learning

- Suppose you know f is a dictatorship.
  Need $\Omega(\log n)$ queries to learn the relevant variable.

- $O(1/\epsilon)$ queries suffice to test if f is a dictatorship
  [Parnas, Ron, Samorodnitsky 2001]

- Can check if f is a dictatorship, not learning who is the dictator!!!

Testing vs. Learning

- Suppose you know f is a dictatorship.
  Need $\Omega(\log n)$ queries to learn the relevant variable.

- $O(1/\epsilon)$ queries suffice to test if f is a dictatorship
  [Parnas, Ron, Samorodnitsky 2001]

- Can check if f is a dictatorship, not learning who is the dictator!!!

- Implicit learning: Learn the structure of f, but not which variables it depends on.

Selected Previous Results (1/2)

- Parnas, Ron, Samorodnitsky 2001:
  is a Boolean function $f : \{0,1\}^n \to \{0,1\}$:
  - a dictator? $O(1/\epsilon)$ queries

$$f(x_1, \ldots, x_{2007}) = \neg x_7$$

Selected Previous Results (1/2)

- Parnas, Ron, Samorodnitsky 2001:
  is a Boolean function $f : \{0,1\}^n \to \{0,1\}$:
  - a dictator? $O(1/\epsilon)$ queries

    $$f(x_1, \ldots, x_{2007}) = \neg x_7$$

  - a conjunction? $O(1/\epsilon)$ queries

    $$f(x_1, \ldots, x_{2007}) = x_2 \wedge \neg x_4 \wedge x_{10}$$

Selected Previous Results (1/2)

- Parnas, Ron, Samorodnitsky 2001:
  is a Boolean function $f : \{0,1\}^n \to \{0,1\}$:
  - a dictator? $O(1/\epsilon)$ queries

    $$f(x_1, \ldots, x_{2007}) = \neg x_7$$

  - a conjunction? $O(1/\epsilon)$ queries

    $$f(x_1, \ldots, x_{2007}) = x_2 \wedge \neg x_4 \wedge x_{10}$$

  - an s-term monotone DNFs? $\tilde{O}(s^2/\epsilon)$ queries

    $$f(x_1, \ldots, x_{2007}) = (x_2 \wedge x_3 \wedge x_{20} \wedge x_{37}) \vee x_{21} \vee (x_2 \wedge x_5)$$

Selected Previous Results (2/2)

- Fischer, Kindler, Ron, Safra, Samorodnitsky 2002
  - testing J-juntas $f : \Omega^n \rightarrow \{0,1\}$
    - non-adaptive one-sided test: $\tilde{O}(J^4/\epsilon)$
    - adaptive one-sided test: $\tilde{O}(J^3/\epsilon)$
    - non-adaptive two-sided test: $\tilde{O}(J^2/\epsilon)$
  - $\tilde{\Omega}(\sqrt{J})$ lower bound for non-adaptive testing juntas

Selected Previous Results (2/2)

- Fischer, Kindler, Ron, Safra, Samorodnitsky 2002
  - testing J-juntas $f : \Omega^n \to \{0, 1\}$
    - non-adaptive one-sided test: $\tilde{O}(J^4/\epsilon)$
    - adaptive one-sided test: $\tilde{O}(J^3/\epsilon)$
    - non-adaptive two-sided test: $\tilde{O}(J^2/\epsilon)$
  - $\tilde{\Omega}(\sqrt{J})$ lower bound for non-adaptive testing juntas
- Chockler, Gutfreund 2004
  - $\Omega(J)$ lower bound for testing juntas

Our Results (1/2)

- Generic tester for many classes of Boolean functions:
  - s-term DNFs (open problem in [PRS]): $\tilde{O}(s^4/\epsilon^2)$
  - size-s decisions trees, size-s branching programs, size-s Boolean formulas, s-sparse polynomials over $\mathbb{F}_2$: $\tilde{O}(s^4/\epsilon^2)$
  - size-s Boolean circuits: $\tilde{O}(s^6/\epsilon^2)$
  - decision lists: $\tilde{O}(1/\epsilon^2)$
  - functions with Fourier degree $\leq d$: $\tilde{O}(2^{6d}/\epsilon^2)$

Our Results (2/2)

- Extension of the junta test to functions with non-Boolean ranges

Our Results (2/2)

- Extension of the junta test to functions with non-Boolean ranges
  - same complexity as for the Boolean range (only a constant-factor overhead)

Our Results (2/2)

- Extension of the junta test to functions with non-Boolean ranges
  - same complexity as for the Boolean range (only a constant-factor overhead)
  - also generalizes the generic algorithm:
    - s-sparse polynomials over field $\mathbb{F}$: $\tilde{O}((s|\mathbb{F}|)^4/\epsilon^2)$
    - size-s algebraic circuits, and size-s algebraic computation trees over $\mathbb{F}$: $\tilde{O}(s^4 \log^4 |\mathbb{F}|/\epsilon^2)$

Our Results (2/2)

- Extension of the junta test to functions with non-Boolean ranges
  - same complexity as for the Boolean range (only a constant-factor overhead)
  - also generalizes the generic algorithm:
    - s-sparse polynomials over field $\mathbb{F}$: $\tilde{O}((s|\mathbb{F}|)^4/\epsilon^2)$
    - size-s algebraic circuits, and size-s algebraic computation trees over $\mathbb{F}$: $\tilde{O}(s^4 \log^4 |\mathbb{F}|/\epsilon^2)$
- Lower bounds:
  - s-sparse polynomials over $\mathbb{F}_2$: $\tilde{\Omega}(\sqrt{s})$
  - functions with Fourier degree $\leq d$: $\tilde{\Omega}(\sqrt{d})$
  - s-sparse polynomials over field $\mathbb{F}$: $\tilde{\Omega}(\sqrt{s})$ for $|\mathbb{F}| = O(1)$

The (Simplest) Junta Test [FKRSS]

- Assign input variables to $O(J^2)$ buckets at random.

$$x_2\, x_7 \;\Big|\; x_8 \;\Big|\; x_1\, x_5\, x_6 \;\Big|\; x_3\, x_4$$

The (Simplest) Junta Test [FKRSS]

- Assign input variables to $O(J^2)$ buckets at random.
- For each bucket, do $\tilde{O}(J^2/\epsilon)$ times the following:

$$x_2\, x_7 \;\big|\; x_8 \;\big|\; x_1\, x_5\, x_6 \;\big|\; x_3\, x_4$$

The (Simplest) Junta Test [FKRSS]

- Assign input variables to $O(J^2)$ buckets at random.
- For each bucket, do $\tilde{O}(J^2/\epsilon)$ times the following:
  - Random assignment to the variables not in the bucket.

$x_2\,x_7$	x_8	$x_1\,x_5\,x_6$	$x_3\,x_4$
$1\quad 0$	1		$0\quad 0$

The (Simplest) Junta Test [FKRSS]

- Assign input variables to $O(J^2)$ buckets at random.
- For each bucket, do $\tilde{O}(J^2/\epsilon)$ times the following:
  - Random assignment to the variables not in the bucket.
  - Two random assignments to the variables in the bucket.

x_2	x_7	x_8	x_1	x_5	x_6	x_3	x_4
1	0	1	0	1	1	0	0
			1	0	0		

The (Simplest) Junta Test [FKRSS]

- Assign input variables to $O(J^2)$ buckets at random.
- For each bucket, do $\tilde{O}(J^2/\epsilon)$ times the following:
  - Random assignment to the variables not in the bucket.
  - Two random assignments to the variables in the bucket.
  - If the value of the function on the two assignments is different, mark the bucket.

$$
\begin{array}{l}
\begin{array}{cc|c|ccc|cc}
x_2 & x_7 & x_8 & x_1 & x_5 & x_6 & x_3 & x_4 \\
\end{array} \\
f(\; 1 \;\; 0 \;|\; 1 \;|\; 0 \;\; 1 \;\; 1 \;|\; 0 \;\; 0 \;) = 1 \\
f(\; 1 \;\; 0 \;|\; 1 \;|\; 1 \;\; 0 \;\; 0 \;|\; 0 \;\; 0 \;) = 0
\end{array}
$$

The (Simplest) Junta Test [FKRSS]

- Assign input variables to $O(J^2)$ buckets at random.
- For each bucket, do $\tilde{O}(J^2/\epsilon)$ times the following:
  - Random assignment to the variables not in the bucket.
  - Two random assignments to the variables in the bucket.
  - If the value of the function on the two assignments is different, mark the bucket.
- If more than J buckets marked, output **NO**. Otherwise, output **YES**.

$$x_2\, x_7 \;\Big|\; x_8 \;\Big|\; x_1\, x_5\, x_6 \;\Big|\; x_3\, x_4$$

Variation [FKRSS]

- The variation of $f : \Omega^n \to \{0,1\}$ on a subset of variables I is

$$\mathrm{Vr}_f(I) = \mathbb{E}_{\substack{\text{random}\\\text{assignment}\\\text{to vars}\\\text{not in } I}} \mathbb{V}_{\substack{\text{random}\\\text{assignment}\\\text{to vars}\\\text{in } I}} f\left(\begin{array}{c}\text{concatenation of}\\\text{assignments}\end{array}\right)$$

Variation [FKRSS]

- The variation of $f : \Omega^n \to \{0,1\}$ on a subset of variables I is

$$\mathrm{Vr}_f(I) = \mathbb{E}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{not in } I}} \mathbb{V}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{in } I}} f\left(\begin{array}{c}\text{concatenation of}\\ \text{assignments}\end{array}\right)$$

- Measures sensitivity of f to the values of variables in I.

Variation [FKRSS]

- The variation of $f : \Omega^n \to \{0,1\}$ on a subset of variables I is

$$\mathrm{Vr}_f(I) = \mathbb{E}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{not in } I}} \mathbb{V}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{in } I}} f\left(\begin{array}{c}\text{concatenation of}\\ \text{assignments}\end{array}\right)$$

- Measures sensitivity of f to the values of variables in I.
- Plays a central role in the proof that the junta test works.

Testing Classes of Juntas

- Let C be a class of functions such that
  - each function in C is a J-junta
  - C is closed under permutations

Testing Classes of Juntas

- Let C be a class of functions such that
  - each function in C is a J-junta
  - C is closed under permutations
- Algorithm for testing if $f \in C$:

Testing Classes of Juntas

- Let C be a class of functions such that
  - each function in C is a J-junta
  - C is closed under permutations
- Algorithm for testing if $f \in C$:
  - Run the junta test, make sure that f is a J-junta, and identify at most J subsets of variables such that each subset contains at most one relevant variable, and no relevant variable is left out.

$$x_2\, x_7 \;\Big|\; x_8 \;\Big|\; x_1\, x_5\, x_6 \;\Big|\; x_3\, x_4$$

Testing Classes of Juntas

- Let C be a class of functions such that
  - each function in C is a J-junta
  - C is closed under permutations
- Algorithm for testing if $f \in C$:
  - Run the junta test, make sure that f is a J-junta, and identify at most J subsets of variables such that each subset contains at most one relevant variable, and no relevant variable is left out.
  - Collect sufficiently many samples for the function restricted to the relevant variables (see next slide).

Implicit Learning

$x_2\, x_7 \mid x_8 \mid x_1\, x_5\, x_6 \mid x_3\, x_4$

Testing Classes of Juntas

- Let C be a class of functions such that
  - each function in C is a J-junta
  - C is closed under permutations
- Algorithm for testing if $f \in C$:
  - Run the junta test, make sure that f is a J-junta, and identify at most J subsets of variables such that each subset contains at most one relevant variable, and no relevant variable is left out.
  - Collect sufficiently many samples for the function restricted to the relevant variables (see next slide).
  - Test them against all restrictions of functions in C to relevant variables. If at least one restriction survives, output **YES**. Otherwise, output **NO**.

Implicit Learning

$x_2\, x_7 \;|\; x_8 \;|\; x_1\, x_5\, x_6 \;|\; x_3\, x_4$

Collecting a Sample

- Pick a random assignment $\mathbf{x} = (x_1, \ldots, x_n)$. Need to read off the hidden relevant variables.

$$\begin{array}{rc|c|c|cl} & x_2\;\; x_7 & x_8 & x_1\;\; x_5\;\; x_6 & x_3\;\; x_4 & \\ f(& 0\;\;\; 0 & 1 & 1\;\;\; 0\;\;\; 1 & 0\;\;\; 0 &) = 0 \end{array}$$

Collecting a Sample

- Pick a random assignment $\mathbf{x} = (x_1, \ldots, x_n)$. Need to read off the hidden relevant variables.
- Let I be a set of variable indices. At most one variable is relevant.

$$
\begin{array}{c|c|c|c}
x_2\ x_7 & x_8 & x_1\ x_5\ x_6 & x_3\ x_4 \\
f(\ 0\ \ 0 & 1 & 1\ \ 0\ \ 1 & 0\ \ 0\ ) = 0 \\
 & & & \\
f'(\ \ 0 & & ? &) = 0
\end{array}
$$

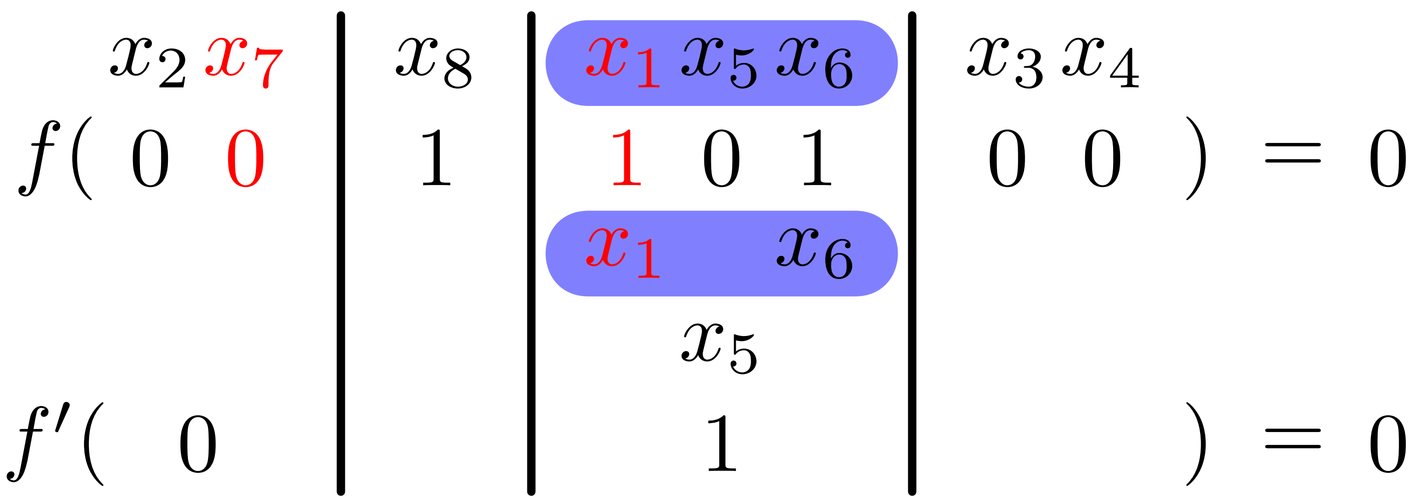
Collecting a Sample

- Pick a random assignment $\mathbf{x} = (x_1, \ldots, x_n)$. Need to read off the hidden relevant variables.
- Let I be a set of variable indices. At most one variable is relevant.
- Check as in the junta test which of $\{i : x_i = 0\} \cap I$ and $\{i : x_i = 1\} \cap I$ contains an index of a relevant variable. If you don't detect a relevant variable in any, pick a random value as the assignment on this set.

	$x_2\ x_7$	x_8	$x_1\ x_5\ x_6$	$x_3\ x_4$	
$f($	$0\ \ 0$	1	$1\ \ 0\ \ 1$	$0\ \ 0$	$) = 0$
			$x_1\ \ \ \ x_6$		
			x_5		
$f'($	0		$?$		$) = 0$

Collecting a Sample

- Pick a random assignment $\mathbf{x} = (x_1, \ldots, x_n)$. Need to read off the hidden relevant variables.
- Let I be a set of variable indices. At most one variable is relevant.
- Check as in the junta test which of $\{i : x_i = 0\} \cap I$ and $\{i : x_i = 1\} \cap I$ contains an index of a relevant variable. If you don't detect a relevant variable in any, pick a random value as the assignment on this set.



Approximation by Juntas

- s-term DNFs are not a class of small juntas.

$$f(x_1, \ldots, x_{1000}) = (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5) \vee (x_6 \wedge x_7 \wedge \ldots \wedge x_{1000})$$

Approximation by Juntas

- s-term DNFs are not a class of small juntas.

$$f(x_1, \ldots, x_{1000}) = (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5) \vee (x_6 \wedge x_7 \wedge \ldots \wedge x_{1000})$$

- If a DNF term $x_1 \wedge x_2 \wedge \ldots \wedge x_k$ is long, it becomes almost irrelevant. The probability of difference in a random assignment is $\leq 2^k$.

$$f(x_1, \ldots, x_{1000}) \approx (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5)$$

Approximation by Juntas

- s-term DNFs are not a class of small juntas.

$$f(x_1, \ldots, x_{1000}) = (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5) \vee (x_6 \wedge x_7 \wedge \ldots \wedge x_{1000})$$

- If a DNF term $x_1 \wedge x_2 \wedge \ldots \wedge x_k$ is long, it becomes almost irrelevant. The probability of difference in a random assignment is $\leq 2^k$.

$$f(x_1, \ldots, x_{1000}) \approx (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5)$$

- Can drop such a term for large k's. Each of our arguments to f is random.

Approximation by Juntas

- s-term DNFs are not a class of small juntas.

$$f(x_1,\ldots,x_{1000}) = (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5) \vee (x_6 \wedge x_7 \wedge \ldots \wedge x_{1000})$$

- If a DNF term $x_1 \wedge x_2 \wedge \ldots \wedge x_k$ is long, it becomes almost irrelevant. The probability of difference in a random assignment is $\leq 2^k$.

$$f(x_1,\ldots,x_{1000}) \approx (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5)$$

- Can drop such a term for large k's. Each of our arguments to f is random.
- Suffices to focus on DNFs that are J-juntas for sufficiently large J.

Approximation by Juntas

- s-term DNFs are not a class of small juntas.

$$f(x_1, \ldots, x_{1000}) = (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5) \vee (x_6 \wedge x_7 \wedge \ldots \wedge x_{1000})$$

- If a DNF term $x_1 \wedge x_2 \wedge \ldots \wedge x_k$ is long, it becomes almost irrelevant. The probability of difference in a random assignment is $\leq 2^k$.

$$f(x_1, \ldots, x_{1000}) \approx (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_4 \wedge \neg x_5)$$

- Can drop such a term for large k's. Each of our arguments to f is random.
- Suffices to focus on DNFs that are J-juntas for sufficiently large J.
- If want to stay τ-close, suffices to take $J = s \log(s/\tau)$.

Non-Boolean Ranges ($f : \Omega^n \rightarrow X$)

- We prove that the junta test works with only a constant factor overhead.

Non-Boolean Ranges ($f : \Omega^n \rightarrow X$)

- We prove that the junta test works with only a constant factor overhead.
- Our testing techniques can be applied as well.

Non-Boolean Ranges ($f : \Omega^n \rightarrow X$)

- We prove that the junta test works with only a constant factor overhead.
- Our testing techniques can be applied as well.
- Need new tools to prove that. The variation only works for the Boolean range.

Non-Boolean Ranges ($f : \Omega^n \to X$)

- Maybe mapping to $\{0,1\}$? Won't lose much? Which mapping?

$$\mathbb{E}_{\substack{\text{random}\\\text{assignment}\\\text{to vars}\\\text{not in } I}} \max_{\phi : X \to \{0,1\}} \mathbb{V}_{\substack{\text{random}\\\text{assignment}\\\text{to vars}\\\text{in } I}} (f \circ \phi) \left(\begin{array}{c}\text{concatenation of}\\\text{assignments}\end{array} \right)$$

$$\max_{\phi : X \to \{0,1\}} \mathbb{E}_{\substack{\text{random}\\\text{assignment}\\\text{to vars}\\\text{not in } I}} \mathbb{V}_{\substack{\text{random}\\\text{assignment}\\\text{to vars}\\\text{in } I}} (f \circ \phi) \left(\begin{array}{c}\text{concatenation of}\\\text{assignments}\end{array} \right)$$

Non-Boolean Ranges ($f : \Omega^n \to X$)

- Maybe mapping to $\{0, 1\}$? Won't lose much? Which mapping?

$$\mathbb{E}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{not in } I}} \max_{\phi:X\to\{0,1\}} \mathbb{V}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{in } I}} (f \circ \phi) \left(\begin{array}{c} \text{concatenation of} \\ \text{assignments} \end{array} \right)$$

$$\max_{\phi:X\to\{0,1\}} \mathbb{E}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{not in } I}} \mathbb{V}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{in } I}} (f \circ \phi) \left(\begin{array}{c} \text{concatenation of} \\ \text{assignments} \end{array} \right)$$

- Need both! But they are within a constant factor.

Non-Boolean Ranges ($f : \Omega^n \to X$)

- Maybe mapping to $\{0, 1\}$? Won't lose much? Which mapping?

$$\mathbb{E}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{not in } I}} \max_{\phi:X\to\{0,1\}} \mathbb{V}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{in } I}} (f \circ \phi)\left(\begin{array}{c}\text{concatenation of}\\ \text{assignments}\end{array}\right)$$

$$\max_{\phi:X\to\{0,1\}} \mathbb{E}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{not in } I}} \mathbb{V}_{\substack{\text{random}\\ \text{assignment}\\ \text{to vars}\\ \text{in } I}} (f \circ \phi)\left(\begin{array}{c}\text{concatenation of}\\ \text{assignments}\end{array}\right)$$

- Need both! But they are within a constant factor.
- Can make all the proofs work.

Open Questions

- We gave a generic algorithm. Can improve the query complexity for any of the considered classes of functions?
- Can improve the exponential running time by, for instance, replacing the exponential implicit learning step with a more efficient algorithm?

Questions?