Testing Basic Boolean Formulae

Michal Parnas Dana Ron Alex Samorodinsky

The Goal :

- Finding Property Testers for:

    1. Singletons.
    2. Monomials.
    3. DNF functions.

- Having query complexity better than corresponding Learning algorithms.

The functions :

- $f : \{0,1\}^n \rightarrow \{0,1\}$.
- Singletons : $f(x) = x_i$ / $f(x) = \bar{x}_i$
- k-Monomials : $f(x) = x_{i_1} \wedge \bar{x}_{i_2} \wedge \cdots \wedge x_{i_k}$
- ℓ-term DNF : $f(x) = (x_{i_1} \wedge x_{i_2}) \vee \cdots \vee (x_{i_1} \wedge x_{i_j} \wedge x_{i_n})$

Motivation :

- 2 views of Property Testing :

  1. Relaxation of exact decision.
     Testing combinatorial objects – Graphs.

  2. Learning perspective.
     Testing- Functions.

The Results :

- **Sub-Linear** query complexity
- **Independent** of the input size – n.
- For **all 3** Property Testers :

1. Singletons – $O(1/\varepsilon)$.
2. Monomials – $O(1/\varepsilon)$.
3. Monotone ℓ-term DNFs – $\tilde{O}(\ell^2/\varepsilon)$.

This Talk :

- Singletons – Natural Algorithm - $O(1/\varepsilon^2)$.
- Monomials – Generalization - $O(1/\varepsilon^2)$.

- Singletons – Second Algorithm - $O(1/\varepsilon)$.
- Monomials – Structure - $O(1/\varepsilon)$.

Property Tester :

- Testing algorithm :
  - Is given :
    1. Distance parameter – ε. $(0< \varepsilon<1)$
    2. Query access to - f.
  - Outputs: (with probability at least 2/3)
    1. Accept - for any $f \in F$.
    2. Reject - for any f that is ε-far from F.

Distance definitions :

- Distance between functions - f , g :

$$dist(f,g) \equiv \Pr_{x \in \{0,1\}^n}[f(x) \neq g(x)]$$

- Distance between function and group - f , F :

$$dist(f,F) \equiv \min_{g \in F} dist(f,g)$$

Testing Singleton Functions

Testing Algorithm I

Claim 1:

- f is a monotone k-monomial if and only if :

1. $\Pr[\ f\ = 1] = 1 / 2^{k}$
2. $\forall\, x, y$:

$$f(x \wedge y) = f(x) \wedge f(y)$$

Claim 1 – proof :

1. $f(x) = 0 \qquad , \forall \mid x \mid < k$
2. Let – $y = \wedge_{x \in F_1} x$

   $\Rightarrow \; f(y) = f(\wedge_{x \in F_1} x) = \wedge_{x \in F_1} f(x) = 1$

   $\Rightarrow \; \mid y \mid \geq k$

   $\Rightarrow \; y_{i_1} = y_{i_2} = \cdots = y_{i_k} = 1$

   $\Rightarrow \; x_{i_1} = x_{i_2} = \cdots = x_{i_k} = 1 \quad , \quad \forall x \in F_1$
3. Let – $g(x) = x_{i_1} \wedge \cdots \wedge x_{i_k}$

   $\Rightarrow \; F_1 = G_1$

   $\Rightarrow \; f(x) = g(x)$

Natural Algorithm :

- A natural algorithm :
  - Uniformly samples pairs - x , y.
    1. Verify - $\Pr[f=1] \approx 1/2$
    2. Verify – $f(x \wedge y) = f(x) \wedge f(y) \quad , \forall x, y$

Size test

Closure under intersection

- This algorithm is proven correct for :
  - Functions that are not too far from singletons.

$$\delta = dist(f, F) \leq 1/2 - \lambda$$

Algorithm I :

- Testing singletons with lower bound – λ :
  1. Size test :
     I. Uniformly sample m=Θ(1/ε²).
     II. If – $|a - 1/2| > \varepsilon/4$ – Reject.

     a=fraction of samples with f(x)=1
  2. Closure under intersection test :
     I. Repeat Θ(1/(ελ)) :
        » uniformly select a pair - x , y.
        » If - $f(x \wedge y) \neq f(x) \wedge f(y)$ – Reject.
  3. If no step caused rejection – Accept.

Algorithm I - proof :

- If f is a singleton :
  1. Pass Size test with probability at least 2/3:

     Pr[Reject] <1/3 , for sample m=(15/ε²) .

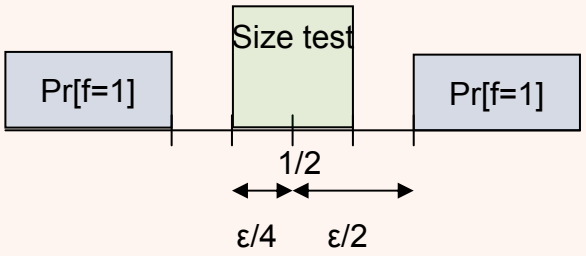
     $$\Pr[|a-1/2|>\varepsilon/4]\leq 2e^{-2(15/\varepsilon^2)(\varepsilon^2/4^2)}$$

  2. Always pass Closure under intersection test.

- Algorithm – f is Accepted w.p at least 2/3.

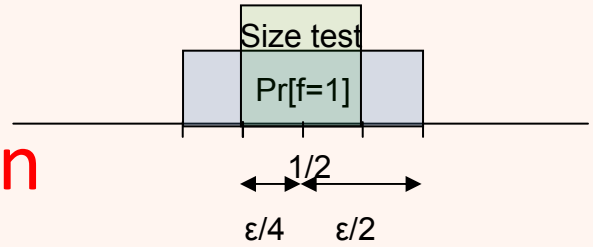
Algorithm I - proof :

- If f is ε-far from any singleton
- And If - $|\Pr[f=1]-1/2|>\varepsilon/2$
  1. f is Rejected in Size test w.p at least 5/6:



- Algorithm – f is Rejected w.p at least 5/6.

Algorithm I - proof :



- If f is ε-far from any singleton
- And - $|\Pr[f=1]-1/2| \le \varepsilon/2 < \delta/2$

1. f is Rejected in second step w.p >= 5/6:

$$\Pr(violating.pair) \ge \delta/4(1/2-\delta) \ge \lambda\varepsilon/4$$

This is not bounded from below for distance close to ½.

- Algorithm – f is Rejected w.p >= 5/6.

Natural Algorithm :

- This Algorithm can be generalized for k-monomials :
  - $\delta = dist(f, F) \leq \frac{1}{2^k} - \lambda$
  - Query complexity : $O(1/\varepsilon^2)$.

Testing Algorithm II

Singletons using Linearity :

- Testing Singletons using Linearity test :
  1. Linearity Test .
  2. Singletons Test assuming Linearity.

- Testing Algorithm :
  - No distance constraints.
  - 1 sided error.
  - Query complexity : $O(1/\varepsilon)$.

Linearity test :

- Parity function : (Linear over GF(2))

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

$$f(x) = x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k}$$

- All Singletons are Parity functions.

Singletons test :

- Part 1 : Parity test [Blum, Luby and Rubinfeld]
  - Uniformly picks O(1/ε) and Checks that

$$f(x \oplus y) = f(x) \oplus f(y)$$

- f is parity – test Accepts w.p=1.
- f is ε-far – test Rejects w.p at least 9/10.

Singletons test :

- Part 2 : Singletons test.
  - The analysis of this part depends on the fact that the function is parity.
  - What if f passed the parity test,

    But it is just ε-close to Parity?

    → Use Self-corrector.

Singletons test :

- Self-corrector :
  - For each x –

    Returns g(x) - The value on x of the linear function that is closest to f(x).

- f is parity – returns f w.p=1.
- f is 1/4-close to parity g – returns g w.p >= 9/10.

Singletons test :

- Part 2 : Singletons test:
  - Uniformly select m=64 pairs x,y and Check that –

$$SC(x \wedge y) = SC(x) \wedge SC(y)$$

- g is singleton – test Accepts w.p=1.
- g (parity) is ε-far – test Rejects w.p >= 9/10.

Singletons proof :

- Part 2 : proof:
  - g is s parity:
  - $\Pr[g(x \wedge y) = g(x) \wedge g(y)] = \frac{1}{2} + \frac{1}{2^{|s|+1}}$, $|s|$–*even*
  - $\Pr[g(x \wedge y) = g(x) \wedge g(y)] = \frac{1}{2} + \frac{1}{2^{|s|}}$, $|s|$–*odd*
  - g is a non-singleton s parity → |s|>=2 :
  - $\Rightarrow \quad \Pr[g(x \wedge y) \neq g(x) \wedge g(y)] > 1/8$
  - → Pr(rejecting a pair in step 2 | parity) >= 1/8.

Singletons proof :

- Part 2 : f non-singleton, ε-close to parity - g
  - Pr(rejecting a pair in Part 2) =

    Pr(rejecting a pair in Part 2 | parity)*

    Pr(SC returns g on all 3 calls)

    >= (1/8)*(9/10)^3 > 1/16.
  - In sample=64 :
    - Pr(rejecting f in Part 2) >=9/10.

Algorithm II :

- Testing singletons Algorithm :

1. Parity with min{1/5,ε}, if rejected – Reject.
2. All Zeros - If – $SC(\overrightarrow{1}) = 0$ – Reject :
3. Singleton test :
   I. Uniformly select 64 pairs.
   II. If $-SC(x \wedge y) \neq SC(x) \wedge SC(y)$ – Reject.
4. If no step caused rejection – Accept.

Algorithm II - proof :

- If f is a singleton :
  - f passes all steps w.p =1.
- Algorithm – f is Accepted w.p=1.

Algorithm II - proof :

- If f is ε-far from any singleton
  - Pr(f accepted) < 1/3 :
    1. Pr(ε-far from Parity and pass step 1) <= 1/10
    2. Pr(ε-close to the "all 0s" and pass step 2) <= 1/10
    3. Pr(ε-close to Parity (s>1) and pass step 3) <= 1/10
- Algorithm – f is Rejected w.p at least 2/3.

Algorithm II - Summary :

- If f is a singleton :
  - f is Accepted w.p = 1.
- If f is ε-far from any singleton :
  - f is Rejected w.p at least 2/3.
- Query complexity : $O(1/\varepsilon)$.

Testing Monomial Functions

Affinity test :

- Affine subset : if and only if :

$$for\ \ every\ \ y_1, y_2, y_3 \in H\ \ we\ \ have\ \ y_1 \oplus y_2 \oplus y_3 \in H$$

$$for\ \ every\ \ y_1, y_2 \in H, y_4 \notin H\ \ we\ \ have\ \ y_1 \oplus y_2 \oplus y_4 \notin H$$

- For all Monomials F_1 is an Affine subset:
  - y_1 ={11.....}
  - y_2 ={11.....}
  - y_3 ={11.....}
  - y_4 ={10.....}

$for – k = 2$

Algorithm III :

- Testing Monomials :

1. Size test, Verify $\Pr[f=1] \approx 1/2^k$, else - Reject.
2. Affinity test, if rejected – Reject.
3. Closure under intersection test :
   I. Uniformly select 32 $x \in F_1$ and- 2^{k+3} points y.
   II. If – $SC(x \wedge y) \neq SC(x) \wedge SC(y)$ – Reject.
4. If no step caused rejection – Accept.

Algorithm III - Summary :

- If f is a Monomial :
- f is Accepted w.p at least 2/3.

- If f is ε-far from any Monomial :
- f is Rejected w.p at least 2/3.

- Query complexity : $O(1/\varepsilon)$.

Home Work

- If we would like to test Singletons
  Under a different, unknown distribution
  Other than the Uniform.

  1. Could we use the tests from the first Algorithm?
  2. Could we use the tests from the second Algorithm?
  3. Same question for testing Monomials.