Learning Circuits with Few Negations

Boolean functions are not that monoton(ous).

Eric Blais Clément Canonne Igor Carboni Oliveira Rocco Servedio
Li-Yang Tan

RANDOM – 2015

Introduction

Introduction: learning

Goal: fixed, known class of Boolean functions $\mathcal{C} \subseteq 2^{\{0,1\}^n}$, unknown $f \in \mathcal{C}$. How to learn f efficiently, i.e. output a hypothesis $\hat{f} \simeq f$?

Introduction: learning

Goal: fixed, known class of Boolean functions $\mathcal{C} \subseteq 2^{\{0,1\}^n}$, unknown $f \in \mathcal{C}$. How to learn f efficiently, i.e. output a hypothesis $\hat{f} \simeq f$?

With membership queries: learn f from *queries* of the form $x? \rightsquigarrow f(x)$

$$\Pr_{x \sim \{0,1\}^n}[f(x) \neq \hat{f}(x)] \leq \varepsilon \qquad \text{(w.h.p.)}$$

Introduction: learning

Goal: fixed, known class of Boolean functions $\mathcal{C} \subseteq 2^{\{0,1\}^n}$, unknown $f \in \mathcal{C}$. How to learn f efficiently, i.e. output a hypothesis $\hat{f} \simeq f$?

With membership queries: learn f from *queries* of the form $x? \rightsquigarrow f(x)$

$$\Pr_{x\sim\{0,1\}^n}[f(x) \neq \hat{f}(x)] \leq \varepsilon \qquad \text{(w.h.p.)}$$

Uniform-distribution PAC-learning: learn f from *random examples* $\langle x, f(x)\rangle$, where $x \sim \{0,1\}^n$?

$$\Pr_{x\sim\{0,1\}^n}[f(x) \neq \hat{f}(x)] \leq \varepsilon \qquad \text{(w.h.p.)}$$

Introduction: learning

Goal: fixed, known class of Boolean functions $\mathcal{C} \subseteq 2^{\{0,1\}^n}$, unknown $f \in \mathcal{C}$. How to learn f efficiently, i.e. output a hypothesis $\hat{f} \simeq f$?

With membership queries: learn f from *queries* of the form $x? \rightsquigarrow f(x)$

$$\Pr_{x\sim\{0,1\}^n}[f(x) \neq \hat{f}(x)] \leq \varepsilon \qquad \text{(w.h.p.)}$$

Uniform-distribution PAC-learning: learn f from *random examples* $\langle x, f(x)\rangle$, where $x \sim \{0,1\}^n$?

$$\Pr_{x\sim\{0,1\}^n}[f(x) \neq \hat{f}(x)] \leq \varepsilon \qquad \text{(w.h.p.)}$$

Uniform-distribution learning $\preceq$ learning with queries

Monotone functions (1)

For circuit complexity theorists:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if it is computed by a Boolean circuit with no negations (only AND and OR gates).*

Monotone functions (1)

For circuit complexity theorists:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if it is computed by a Boolean circuit with no negations (only AND and OR gates).*

For analysis of Boolean functions enthusiasts:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if for any $x \preceq y$ in $\{0,1\}^n$, $f(x) \leq f(y)$.*

Monotone functions (1)

For circuit complexity theorists:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if it is computed by a Boolean circuit with no negations (only AND and OR gates).*

For analysis of Boolean functions enthusiasts:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if for any $x \preceq y$ in $\{0,1\}^n$, $f(x) \leq f(y)$.*

For people with a twisted mind:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if $f(0^n) \leq f(1^n)$, and f changes value* at most once *on any increasing chain from 0^n to 1^n.*

(These definitions are equivalent.)

Monotone functions (1)

For circuit complexity theorists:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if it is computed by a Boolean circuit with no negations (only AND and OR gates).*

For analysis of Boolean functions enthusiasts:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if for any $x \preceq y$ in $\{0,1\}^n$, $f(x) \leq f(y)$.*

For people with a twisted mind:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* monotone *if $f(0^n) \leq f(1^n)$, and f changes value* at most once *on any increasing chain from 0^n to 1^n.*

(These definitions are equivalent.)

Majority function (1 iff at least half the votes are positive): more votes cannot make a candidate lose.
s-clique function (1 iff the input graph contains a clique of size s): more edges cannot remove a clique.
Dictator function (1 iff $x_1 = 1$): more voters have no influence anyway.

Monotone functions (2)

Can we learn them?

Learning the class $\mathcal{C}^n$ of monotone Boolean functions from uniform examples (to error ε) can be done in time $2^{\tilde{O}\left(\sqrt{n}/\varepsilon\right)}$. [BT96]

Monotone functions (2)

Can we learn them?

Learning the class $\mathcal{C}^n$ of monotone Boolean functions from uniform examples (to error ε) can be done in time $2^{\tilde{O}\left(\sqrt{n}/\varepsilon\right)}$. [BT96]

Monotone functions (2)

Can we learn them?

Learning the class $\mathcal{C}^n$ of monotone Boolean functions from uniform examples (to error ε) can be done in time $2^{\tilde{O}\left(\sqrt{n}/\varepsilon\right)}$. [BT96]

Can we do better?

Monotone functions (2)

Can we learn them?

Learning the class $\mathcal{C}^n$ of monotone Boolean functions from uniform examples (to error ε) can be done in time $2^{\tilde{O}\left(\sqrt{n}/\varepsilon\right)}$. [BT96]

Can we do better?

Learning the class $\mathcal{C}^n$ *from membership queries* (to error $\frac{1}{\sqrt{n}\log n}$) requires query complexity $2^{\Omega(n)}$. [BT96]

Monotone functions (2)

Can we learn them?

Learning the class $\mathcal{C}^n$ of monotone Boolean functions from uniform examples (to error ε) can be done in time $2^{\tilde{O}\left(\sqrt{n}/\varepsilon\right)}$. [BT96]

Can we do better?

Learning the class $\mathcal{C}^n$ *from membership queries* (to error $\frac{1}{\sqrt{n}\log n}$) requires query complexity $2^{\Omega(n)}$. [BT96]

Are we done here?

Outline of the talk

Introduction

Generalizing monotone functions: $\mathcal{C}_t^n$.

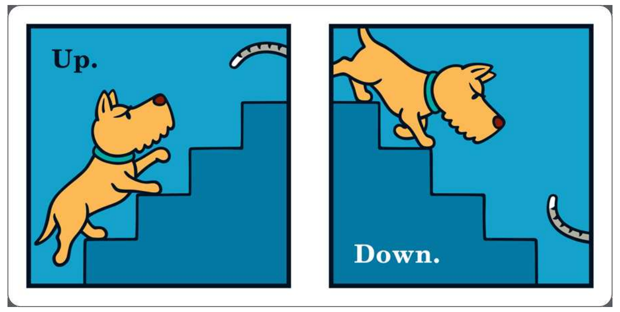
Learning $\mathcal{C}_t^n$: Upper bound.

Learning $\mathcal{C}_t^n$: Lower bound.

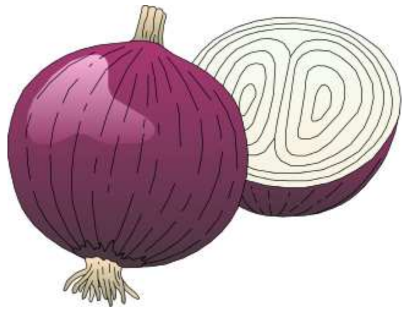
Conclusion and Open Problem(s).

Plan in more detail

- Generalizing monotone functions to "k-alternating:" two views, reconciled by Markov's Theorem.



- A structural theorem: characterizing these new functions as combination of simpler ones $\rightsquigarrow$ upper bound on learning k-alternating functions, almost "for free."



- Lower bound: a succession and combination thereof (from monotone… to monotone to k-alternating: hardness amplification)

Generalizing monotone functions: $\mathcal{C}_t^n$.

k-alternating functions (1)

For circuit complexity theorists:

Definition. $f\colon\{0,1\}^n\to\{0,1\}$ *has* inversion complexity t *if it can be computed by a Boolean circuit with t negations (besides* AND *and* OR *gates), but no less.*

k-alternating functions (1)

For circuit complexity theorists:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *has* inversion complexity t *if it can be computed by a Boolean circuit with t negations (besides* AND *and* OR *gates), but no less.*

For people with a twisted mind:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* k-alternating *if f changes value* at most k times *on any increasing chain from* 0^n *to* 1^n*.*

(Analysis of Boolean functions enthusiasts, stay with us?)

k-alternating functions (1)

For circuit complexity theorists:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *has* inversion complexity t *if it can be computed by a Boolean circuit with t negations (besides* AND *and* OR *gates), but no less.*

For people with a twisted mind:

Definition. $f\colon \{0,1\}^n \to \{0,1\}$ *is* k-alternating *if f changes value* at most k times *on any increasing chain from 0^n to 1^n.*

(Analysis of Boolean functions enthusiasts, stay with us?)

"Not-suspicious" function (1 iff between 50% and 90% of the votes are positive): more than 90%, fishy.
s-clique-but-no-Hamiltonian function (1 iff the input graph contains a clique of size s, but no Hamiltonian cycle): more edges can make things worse.
Highlander function (1 iff exactly one of the x_i's is 1): there shall be only one.

k-alternating functions (2)

But are these definitions the same? Related?

k-alternating functions (2)

But are these definitions the same? Related?

Theorem 4 (Markov's Theorem [Mar57]). *Suppose* $f\colon \{0,1\}^n \to \{0,1\}$ *is not identically* 0. *Then* f *is* k*-alternating iff it has inversion complexity* $O(\log k)$.

k-alternating functions (2)

But are these definitions the same? Related?

Theorem 7 (Markov's Theorem [Mar57]). *Suppose* $f\colon \{0,1\}^n \to \{0,1\}$ *is not identically* 0. *Then* f *is* k*-alternating iff it has inversion complexity* $O(\log k)$.

Refinement of this characterization:

Theorem 8. f *is* k*-alternating iff it can be written* $f(x) = h(m_1(x), \dots, m_k(x))$*, where each* m_i *is monotone and* h *is either the parity function or its negation.*

Corollary 9. *Every* $f \in \mathcal{C}_t^n$ *can be expressed as* $f = h(m_1, \dots, m_T)$ *where* h *is either* Parity_T *or its negation, each* $m_i\colon \{0,1\}^n \to \{0,1\}$ *is monotone, and* $T = O(2^t)$.

k-alternating functions (2)

But are these definitions the same? Related?

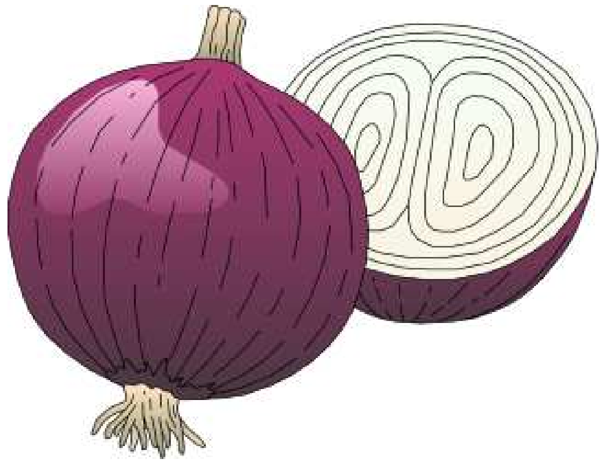
Theorem 10 (Markov's Theorem [Mar57]). *Suppose* $f\colon \{0,1\}^n \to \{0,1\}$ *is not identically* 0. *Then* f *is* k*-alternating iff it has inversion complexity* $O(\log k)$.

Refinement of this characterization:

Theorem 11. f *is* k*-alternating iff it can be written* $f(x) = h(m_1(x), \ldots, m_k(x))$, *where each* m_i *is monotone and* h *is either the parity function or its negation.*

Corollary 12. *Every* $f \in \mathcal{C}_t^n$ *can be expressed as* $f = h(m_1, \ldots, m_T)$ *where* h *is either* Parity_T *or its negation, each* $m_i\colon \{0,1\}^n \to \{0,1\}$ *is monotone, and* $T = O(2^t)$.

Proof (and interpretation). the m_i's are successive nested layers:



□

Learning $\mathcal{C}_t^n$: Upper bound.

Theorem 13. *There is a uniform-distribution learning algorithm which learns any unknown* $f \in \mathcal{C}_t^n$ *from random examples to error* ε *in time* $n^{O(2^t\sqrt{n}/\varepsilon)}$.

Theorem 15. *There is a uniform-distribution learning algorithm which learns any unknown* $f \in \mathcal{C}_t^n$ *from random examples to error* ε *in time* $n^{O(2^t\sqrt{n}/\varepsilon)}$. *(Recall the* $n^{O(\sqrt{n}/\varepsilon)}$ *for monotone functions, i.e.* $t = 0$.*)*

Influence, Low-Degree Algorithm, and a Can of Soup

Theorem 17. *There is a uniform-distribution learning algorithm which learns any unknown $f \in \mathcal{C}_t^n$ from random examples to error ε in time $n^{O(2^t\sqrt{n}/\varepsilon)}$. (Recall the $n^{O(\sqrt{n}/\varepsilon)}$ for monotone functions, i.e. $t = 0$.)*

Proof. Recall that (1) monotone functions have *total influence* $\leq \sqrt{n}$ and that (2) we can learn functions with good *Fourier concentration*:

Theorem 18 (Low-Degree Algorithm ([LMN93])). *Let $\mathcal{C}$ be a class such that for all $\varepsilon > 0$ and $\tau = \tau(\varepsilon, n)$,*

$$\sum_{|S|>\tau} \hat{f}(S)^2 \leq \varepsilon, \qquad \forall f \in \mathcal{C}.$$

Then $\mathcal{C}$ can be learned from uniform random examples in time $\mathrm{poly}(n^\tau, 1/\varepsilon)$.

Decomposition theorem + union bound + massaging + the above: k-alternating functions have total influence $\leq k\sqrt{n}$, and we are done. □

Learning $\mathcal{C}_t^n$: Lower bound.

Three-step program

$$\underbrace{\begin{bmatrix}\text{high-accuracy}\\ \text{monotone}\end{bmatrix}}_{\text{(a)}} \xrightarrow[\text{monotone}]{\bigotimes\text{-like}} \underbrace{\begin{bmatrix}\text{moderate accuracy}\\ \text{monotone}\end{bmatrix}}_{\text{(b)}} \xrightarrow[k\text{-alternating}]{\bigotimes\text{-like}} \underbrace{\begin{bmatrix}\text{moderate accuracy}\\ k\text{-alternating}\end{bmatrix}}_{\text{(c)}}$$

Three-step program

[high-accuracy monotone] $\xrightarrow[\text{monotone}]{\otimes\text{-like}}$ [moderate accuracy monotone] $\xrightarrow[k\text{-alternating}]{\otimes\text{-like}}$ [moderate accuracy k-alternating]

(a) (b) (c)

(a) Monotone functions are hard to learn *well*. (A simple extension of [BT96].)

Learning *monotone* functions to (very small) error $\frac{.1}{\sqrt{n}}$ requires 2^{Cn} queries, for some absolute $C > 0$.

(b) Monotone functions are hard to learn, *period*. (Hardness amplification and the previous result.)

Learning *monotone* functions to (almost any) error ε requires $2^{\Omega(\sqrt{n}/\varepsilon)}$ queries.

(c) k-alternating functions are hard to learn, *too*! (Hardness amplification again + truncated parity.)

Learning *k-alternating* functions to (almost any) error ε requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ queries.

In more detail: ingredients for (b) and (c)

Composition:

- "Inner" function $f\colon \{0,1\}^m \to \{0,1\}$
- + "combining" function $g\colon \{0,1\}^r \to \{0,1\}$
- $\rightsquigarrow$ combined function $(g \otimes f)\colon \{0,1\}^{mr} \to \{0,1\}$

$$(g \otimes f)(x) = g(f(x_1, \ldots, x_m), \ldots, f(x_{(r-1)m+1}, \ldots, x_{rm}))$$

In more detail: ingredients for (b) and (c)

Composition:

- "Inner" function $f\colon \{0,1\}^m \to \{0,1\}$
- + "combining" function $g\colon \{0,1\}^r \to \{0,1\}$
- $\rightsquigarrow$ combined function $(g \otimes f)\colon \{0,1\}^{mr} \to \{0,1\}$

$$(g \otimes f)(x) = g(f(x_1, \ldots, x_m), \ldots, f(x_{(r-1)m+1}, \ldots, x_{rm}))$$

Expected bias: "kill" each variable of f independently by a random restriction. What is the expected bias of the result?

In more detail: ingredients for (b) and (c)

Composition:

- "Inner" function $f\colon \{0,1\}^m \to \{0,1\}$
- + "combining" function $g\colon \{0,1\}^r \to \{0,1\}$
- $\rightsquigarrow$ combined function $(g \otimes f)\colon \{0,1\}^{mr} \to \{0,1\}$

$$(g \otimes f)(x) = g(f(x_1, \dots, x_m), \dots, f(x_{(r-1)m+1}, \dots, x_{rm}))$$

Expected bias: "kill" each variable of f independently by a random restriction. What is the expected bias of the result?

"XOR"-Lemma of [FLS11]: Let $\mathcal{F}$ be a class of m-variable inner functions with "very small bias," and $g\colon \{0,1\}^r \to \{0,1\}$ an outer function with "very small expected bias." Then *if one can learn $g \otimes \mathcal{F}$ efficiently, one can learn $\mathcal{F}$ efficiently-ish.*

In more detail: step (b)

Theorem 19. *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [1/n^{1/6}, .49]$, learning $\mathcal{H}_n$ to error ε requires $2^{\Omega\left(\sqrt{n}/\varepsilon\right)}$ membership queries.*

In more detail: step (b)

Theorem 20. *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [1/n^{1/6}, .49]$, learning $\mathcal{H}_n$ to error ε requires $2^{\Omega\left(\sqrt{n}/\varepsilon\right)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$.

In more detail: step (b)

Theorem 21. *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [1/n^{1/6}, .49]$, learning $\mathcal{H}_n$ to error ε requires $2^{\Omega\left(\sqrt{n}/\varepsilon\right)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$.
- Take the "Mossel–O'Donnell function" g_r [MO03] (a balanced monotone function minimally stable under very small noise)

  (Why? We want $\mathrm{ExpectedBias}_\gamma(g_r) + \epsilon' \leq 1 - \varepsilon$*, and less stable means smaller expected bias)*

In more detail: step (b)

Theorem 22. *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [1/n^{1/6}, .49]$, learning $\mathcal{H}_n$ to error ε requires $2^{\Omega\left(\sqrt{n}/\varepsilon\right)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$.
- Take the "Mossel–O'Donnell function" g_r [MO03] (a balanced monotone function minimally stable under very small noise)
  (Why? We want $\mathrm{ExpectedBias}_\gamma(g_r) + \epsilon' \leq 1 - \varepsilon$*, and less stable means smaller expected bias)*
- Apply the hardness amplification theorem on $g_r \otimes \mathcal{G}_m$, $\mathcal{G}_m$ being the "hard class" from Step (a).

In more detail: step (b)

Theorem 23. *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [1/n^{1/6}, .49]$, learning $\mathcal{H}_n$ to error ε requires $2^{\Omega\left(\sqrt{n}/\varepsilon\right)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$.
- Take the "Mossel–O'Donnell function" g_r [MO03] (a balanced monotone function minimally stable under very small noise)

  (Why? We want $\mathrm{ExpectedBias}_\gamma(g_r) + \epsilon' \leq 1 - \varepsilon$, *and less stable means smaller expected bias)*
- Apply the hardness amplification theorem on $g_r \otimes \mathcal{G}_m$, $\mathcal{G}_m$ being the "hard class" from Step (a).
- Hope all the constants and parameters work out.

In more detail: step (b)

Theorem 24. *There exists a class $\mathcal{H}_n$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [1/n^{1/6}, .49]$, learning $\mathcal{H}_n$ to error ε requires $2^{\Omega\left(\sqrt{n}/\varepsilon\right)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$.
- Take the "Mossel–O'Donnell function" g_r [MO03] (a balanced monotone function minimally stable under very small noise)

  (Why? We want $\text{ExpectedBias}_\gamma(g_r) + \epsilon' \leq 1 - \varepsilon$*, and less stable means smaller expected bias)*
- Apply the hardness amplification theorem on $g_r \otimes \mathcal{G}_m$, $\mathcal{G}_m$ being the "hard class" from Step (a).
- Hope all the constants and parameters work out.

□

In more detail: step (c)

Theorem 25. *For any $k = k(n)$, there exists a class $\mathcal{H}^{(k)}$ of balanced k-alternating Boolean functions (on n variables) such that, for n big enough and (almost) any $\varepsilon > 0$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

In more detail: step (c)

Theorem 26. *For any $k = k(n)$, there exists a class $\mathcal{H}^{(k)}$ of balanced k-alternating Boolean functions (on n variables) such that, for n big enough and (almost) any $\varepsilon > 0$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$ and $r \approx k^2$.

In more detail: step (c)

Theorem 27. *For any $k = k(n)$, there exists a class $\mathcal{H}^{(k)}$ of balanced k-alternating Boolean functions (on n variables) such that, for n big enough and (almost) any $\varepsilon > 0$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$ and $r \approx k^2$.
- Take $\mathsf{Parity}_{k,r}$, the "k-Truncated Parity function on r variables" as combining function, *in lieu* of the previous g_r.

(Why? We want it to be k-alternating, and very little stable)

In more detail: step (c)

Theorem 28. *For any $k = k(n)$, there exists a class $\mathcal{H}^{(k)}$ of balanced k-alternating Boolean functions (on n variables) such that, for n big enough and (almost) any $\varepsilon > 0$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$ and $r \approx k^2$.
- Take $\mathsf{Parity}_{k,r}$, the "k-Truncated Parity function on r variables" as combining function, *in lieu* of the previous g_r.

  (Why? We want it to be k-alternating, and very little stable)
- Apply the hardness amplification theorem on $\mathsf{Parity}_{k,r} \otimes \mathcal{H}_m$, $\mathcal{H}_m$ coming from Step (b).

In more detail: step (c)

Theorem 29. *For any $k = k(n)$, there exists a class $\mathcal{H}^{(k)}$ of balanced k-alternating Boolean functions (on n variables) such that, for n big enough and (almost) any $\varepsilon > 0$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$ and $r \approx k^2$.
- Take $\mathsf{Parity}_{k,r}$, the "k-Truncated Parity function on r variables" as combining function, *in lieu* of the previous g_r.

  (Why? We want it to be k-alternating, and very little stable)
- Apply the hardness amplification theorem on $\mathsf{Parity}_{k,r} \otimes \mathcal{H}_m$, $\mathcal{H}_m$ coming from Step (b).
- *Really* hope all the constants and parameters work out.

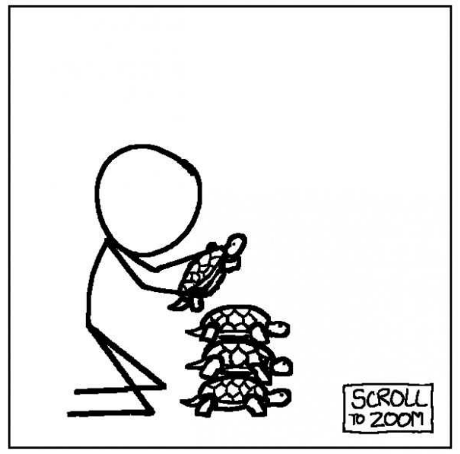
In more detail: step (c)

Theorem 30. *For any $k = k(n)$, there exists a class $\mathcal{H}^{(k)}$ of balanced k-alternating Boolean functions (on n variables) such that, for n big enough and (almost) any $\varepsilon > 0$, learning $\mathcal{H}^{(k)}$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(k\sqrt{n}/\varepsilon)}$ membership queries.*

Sketch.

- Choose suitable $m, r = \omega(1)$ such that $mr = n$ and $r \approx k^2$.
- Take $\mathsf{Parity}_{k,r}$, the "k-Truncated Parity function on r variables" as combining function, *in lieu* of the previous g_r.

  (Why? We want it to be k-alternating, and very little stable)
- Apply the hardness amplification theorem on $\mathsf{Parity}_{k,r} \otimes \mathcal{H}_m$, $\mathcal{H}_m$ coming from Step (b).
- *Really* hope all the constants and parameters work out.



Conclusion and Open Problem(s).

Open problems

Weak Learning: can one learn $\mathcal{C}_t^n$ to error $\frac{1}{2} - \frac{1}{\mathrm{poly}(n)}$ ("barely better than random") in polynomial time?

(Related) Fourier spectrum: Can we get any further understanding of the Fourier spectrum of k-alternating functions?

Concrete example:

Let f, g be monotone Boolean functions, and $h = \mathsf{Parity}(f, g)$. Can we prove

$$\sum_{|S|\leq 2} \hat{h}(S)^2 \geq \frac{1}{\mathrm{poly}(n)}?$$

Or even $\sum_{|S|\leq 2} \hat{h}(S)^2 > 0$?

Thank you.

Any question?

References

[BT96] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.

[FLS11] V. Feldman, H. K. Lee, and R. A. Servedio. Lower bounds and hardness amplification for learning shallow monotone formulas. *Journal of Machine Learning Research - Proceedings Track*, 19:273–292, 2011.

[LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[Mar57] A. A. Markov. On the inversion complexity of systems of functions. *Doklady Akademii Nauk SSSR*, 116:917–919, 1957.

[MO03] E. Mossel and R. O'Donnell. On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23(3):333–350, 2003.