Making Hard Problem Harder

Nilkamal Adak
CS11M037

IITM

April 19, 2012

Notion of Hard Functions

- What do you mean by hard functions ?
- Worst case $s - hard$ functions.
- Average case $s - hard$ functions.
- Why are we interested to make hard functions harder ?

Hardness Condenser & Hardness Extractor

- Hardness Condenser
- Hardness Extractor
- What if we have an 'efficient' hardness condenser ?

Some Notations

- $f : m \rightarrow n$ to denote $f : \{0,1\}^m \rightarrow \{0,1\}^n$
- To identify a boolean function f on n bits, we use 2^n bit strings .ie the truth table of the functions.
- hard functions for size s : A function f is said to be a hard function for size s if no circuit of size s can compute it correctly.
- $\delta-$hard functions for size s : A function f is said to be a $\delta-$hard function for size s if no circuit of size s that compute a function that agree with f on more than $1 - \delta$ fractions of the inputs.

Hardness Condenser and Extractor

Hardness Condenser

Let A and B be complexity models(eg. deterministic circuits, nondeterministic circuits, circuits with oracle D, formulas, branching programs etc.) $An(n, s, n', s')$ *hardness condenser with advice length l from worst-case (resp.* $\delta - average - case$*) hardness for A to worst-case (resp.* $\delta' - average - case$*) hardness for B is a function*

$$C_n : 2^n \times l \rightarrow 2^{n'} \tag{1}$$

with $n' < n$ *such that if* $f : n \rightarrow 1$ *requires (resp. is* $\delta - hard\ for$*) size s in A, then there is a string* $y \in \{0, 1\}^l$ *for which* $C_n(f, y)$ *requires (resp. is* $\delta - hard\ for$ *) size* s' *in B.*

Hardness Condenser

An (n, s, n', s') *hardness condenser with advice length* l *is a function*

$$C_n : 2^n \times l \to 2^{n'}$$

with $n' < n$ *such that if* $f : n \to 1$ *requires size* s*, then there is a string* $y \in \{0,1\}^l$ *for which* $C_n(f, y)$ *requires size* s' *.*

We would like (n, s, n', s') hardness condensers with advise length l.

- l is as small as possible.
- s' is as close to s as possible.
- n' is as close to $log(s')$ as possible.

Hardness Extractor

Hardness Extractor

An (n, s) *hardness extractor is an* (n, s, n', s') *hardness condenser with advice length* $O(log(n))$ *for which* $n' = \Omega(log(s)$ *and* $s' \geq 2^{n'}/n'$

Hardness Extractor is hardness condenser with close to ideal parameters.

Ideal Case

Suppose we have a explicit hardness condenser without any advise, then we can use it to give a explicit function that is as hard as possible in the following manner.
We start with a hard function f_1 and apply our hardness condenser to get an explicit function f_2 that is more harder and repeatedly apply this procedure to get greater levels of hardness.
Recall that we know there are boolean functions that has circuit lower bound $2^n/n$ but we don't know any explicit functions of that family. So if we have an efficient hardness condenser , we can explicitly give functions of this nature.

Negative Result

We cannot achieve the ideal case. That is we cannot eliminate the advise the strings for general hardness condensers.

Theorem

Any (n, s, n', s') relativizing hardness condenser requires l bits of advice where

$$2^l > \frac{2^n + \Omega(s') - s}{2^{n'+1}}$$

Positive Result

We give two special class of functions for which there are efficient hardness condensers without advise.

1. Biased functions : The output is biased to 0 (or 1)
2. Average Case Hard Functions : A function f is said to be a $\gamma-$hard function for size s if no circuit of size s that compute a function that agree with f on more than $1-\gamma$ fractions of the inputs.

Biased Functions

Theorem

For some constant c, there is an explicit $(n, s, n - \lfloor c \log \frac{1}{H(\alpha_F)} \rfloor, \Omega((s-n)/n))$ *hardness condenser from worst-case hardness to worst-case hardness that requires no advice.*

Proof strategy

- Interesting application of Pairwise independent hash family
- construction of f' from f using hash function from Pairwise independent hash family
- Argue about efficient computation of the condenser
- Argue about bound on s'
- Argue that n' is small.

Average case hard Functions

Definition : Covering Codes

A (K, N, R) *covering code is a function* $A : K \rightarrow N$ *such that for each* $y \in \{0,1\}^N$, *there is some string* $x \in \{0,1\}^K$ *such that the Hamming distance between* y *and* $A(x)$ *is at most* R.

Definition : t-local efficient recoverable covering

A (K, N, R) *covering code* A *is* $t - local$ *if for each* $x \in \{0,1\}^K$ *there is a circuit of size* $poly(t, log(N))$ *with oracle access to* x *which , given as input an index* i *between* 1 *and* N, *outputs the ith bit of* $A(x)$. .

A is efficiently recoverable if there is a polynomial time procedure rec_A which given a string $y \in \{0,1\}^N$ as input, outputs a string $x \in \{0,1\}^K$ such that the Hamming distance between $A(x)$ and y is at most R

continued...

Theorem

If there is a $t-$ local efficiently recoverable $(2^k, 2^n, R)$ covering code, then there is a constant c such that there is an explicit $(n, s, k, s/(t+n)^c)$ hardness condenser from $R/2^n$ average-case hardness for deterministic circuits to worst-case hardness for deterministic circuits to worst-case hardness for deterministic circuits.

Proof strategy

- Interesting application of Coding Theory
- construction of f' from f via $t-$local efficiently decodable code
- Argue about efficient computation of the condenser
- Argue about bound on s'

Thank You!