1

Thu 02/28

Comments about Polar Codes:

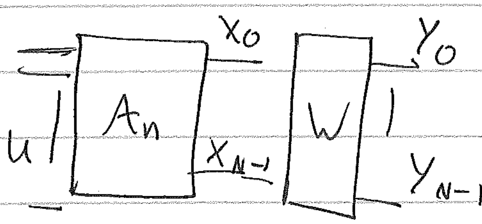
1) Polar Codes give an alternate proof of Shannon's Theorem for Symmetric Channels.

NB. Polar Codes are linear.

2) Method has been generalized to prime alphabets. (and maybe also general ones?)

Source Coding:

Example) Take $Y = \text{null}$



$$X_0^{N-1} \longrightarrow M_n X_0^{N-1} = u_0^{N-1}.$$

but $H(u_i \mid u_0^{i-1}, \underbrace{Y_0^{N-1}}_{=0}) \longrightarrow 0$ (for most indices)

$$\sum_{i=0}^{N-1} H(u_i \mid u_0^{i-1}) = H(u_0^{N-1}) = H(X_0^{N-1}) = N \cdot H(X).$$

Only reveal u_i for which $H(u_i \mid u_0^{i-1})$ is not $\approx 0 \Rightarrow$ We get Compression $\sim n\,H(X)$.

$\Rightarrow$ Another proof for the (bit-) Source Coding Theorem.

3) Which indices polarize to $\underline{0}$?

$$\sum H(U_i \mid U_0^{i-1}) = N \cdot H(X)$$

i.e., find the "Frozen bits".

It might "feel" reasonable that $H(U_i \mid U_0^{i-1})$ is $\simeq 1$ for small i and ≈ 0 for large i.

* But life is not so simple.

+ There is an algorithm to figure out which bits to freeze.

4) Polar Codes are Versatile!

Useful for Slepian–Wolf, Wyner–Ziv, Gelfand–Finsker, etc.

References: Arıkan's original paper, Şaşoglu's thesis (Chap 2) Upcoming writeup with P. Xia.

③

$\boxed{\text{New Topic}}$

* Moser's "entropy Compression" argument.

Of course, if $Pr(\overline{E_i})$ is small

$\Rightarrow Pr(\bigcup \overline{E_i})$ is small by union bound.

$\Rightarrow \bigcap E_i$ can happen.

Lovasz Local Lemma (LLL) gives another criterion.
(under limited dependence).

k-SAT: n Boolean var's $x_1, \ldots, x_n$
m clauses $C_1, \ldots, C_m$

$C_i : \; x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k}$
(possibly with negations)

Q: Given a k-SAT instance, is there a way of assigning x_i s.t. all clauses are satisfied?

Sufficient Condition for Satisfiability of k-SAT:

1) Clauses are disjoint. (don't share var's)

2) If clauses overlap a lot, may not be satisfiable.
(e.g., take all 2^k clauses over k bits).

<u>Theorem:</u> ~~Suppose~~ Every instance of k-SAT where each clause overlaps with $\leq 2^{k-c}$ clauses is <u>always</u> satisfiable. (for some const c) ($c=3$ works).

<u>Algorithmic Proof (Moser 2009)</u>

1) Pick a random assignment to the n var's.
If all satisfied, done!
else let T be the set of unsat clauses.

2) For each clause $s \in T$, <u>$\text{Fix}(s)$.</u>

<u>$\text{Fix}(s)$</u> 1) If A satisfies s, do nothing.
2) Replace the bits of A on support of s by k random bits.
3) Find all clauses s' which overlap with s & which the new A violates. Call the set B.

(s may be still in B).

For each $s' \in B$ (in some fixed order),

$\text{Fix}(s')$.

---

* <u>Claim:</u> If $\text{fix}(S)$ terminates, then the new A satisfies S, and ~~it will contain~~ if $\tilde{S}$ was satisfied by the old A before $\text{fix}(S)$ was called, then the new A will also satisfy $\tilde{S}$.

($\Rightarrow$ each application of $\text{fix}(\cdot)$ decreases the # of unsat clauses).

Pf: Obvious!

---

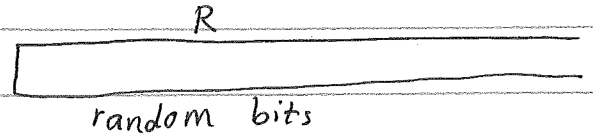
* <u>Corollary:</u> If the main algo. terminates, the instance was satisfiable (A being a satisfying assignment).

* All that remains is to prove $\text{fix}(\cdot)$ terminates early enough.

⑥

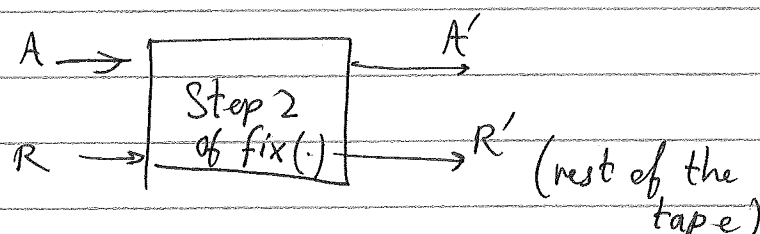
Idea: (Entropy Compression)

Give the alg. a long tape R of random bits.



each step consumes ~~about~~ k bits of randomness.

Q: Given A' and R', can you recover (A, R)?



But if we knew which S was being fixed, we could reverse the process. (since only a unique assignment fails to satisfy S).

* Now imagine a log file where you record the sequence of clauses on which fix is called.

Suppose the alg. runs for more than M runs of Step 2 of Fix(·) ~~steps~~ & in M calls of Step 2 of Fix() we consume $|R| = Mk$ bits of randomness.

~~Suppose~~ Then the information in the log and A' (and $R' = \phi$ in the end) can be used to recover (A, R).

* Naive coding of history takes $M (\log m)$ bits. To get an "impossible compression" we need $M(\log m) < M \cdot k$

$$\Rightarrow m < 2^k.$$ Not so good! (true but trivial to proof).

More clever way?

Knowing s, specifying s' needs only ~~log~~ $k - c$ bits!

Big insight: Start by using $m \log m$ bits to encode clauses in T in the start.
For other clauses s' called from $\mathrm{fix}(s)$, record s' via $k-c$ bits which encodes neighbors of s.

8

* We also need termination symbols $\overset{\text{(constant \# of bits)}}{\vee}$ when Fix() returns.

⇒ Total size of log =

$$|H'| = O(m \log m) + M(k - c + O(1))$$

⇒ We compress random $\underbrace{(A+R)}_{\cancel{n}+Mk \text{ bits}}$ to $\underbrace{(A', H')}_{\cancel{n}+O(m \log m)+M(k-c+O(1))}$.

$$\cancel{n} + Mk \text{ bits} \le \cancel{n} + O(m \log m) + M(k-c+O(1))$$

⇒ we get an upper bound on M.