CS250/EE387 - LECTURE 7 - *Efficiently decoding concatenated codes.*

1/30/2018

AGENDA

AKA, Finishing Lecture 6.
- ⓪ CONCATENATED CODES and the ZYABLOV BOUND
- ½ ASIDE: JUSTESEN CODES

1. EFFICIENTLY DECODING CONCATENATED CODES

GASTROPOD FACT.

Land slugs and land snails have a single lung, called a pneumostome, which opens directly to the outside. The opening opens and closes about once every few minutes in a fully hydrated land slug, but it speeds up if the slug gets dehydrated.

Recall the GOAL from last lecture:

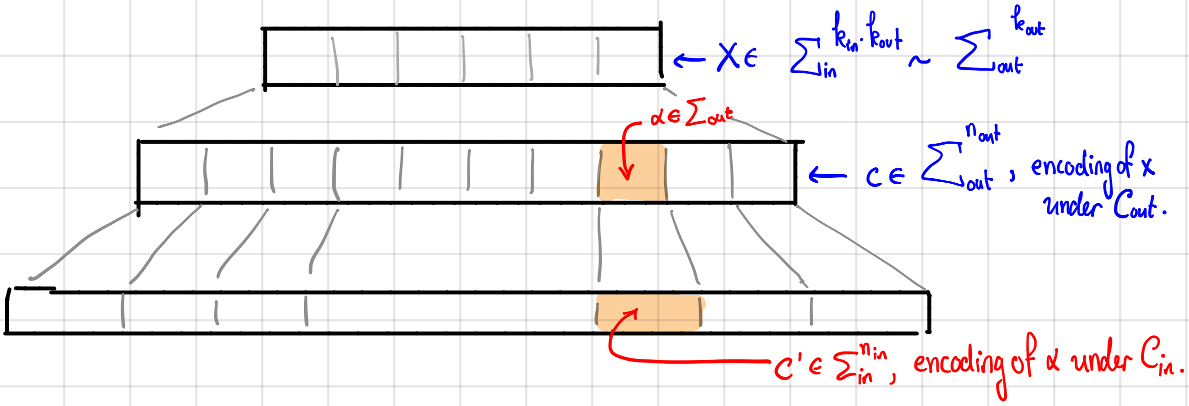
GOAL. Obtain EXPLICIT (aka, efficiently constructible), ASYMPTOTICALLY GOOD families of BINARY CODES, ideally with fast algorithms.

for today

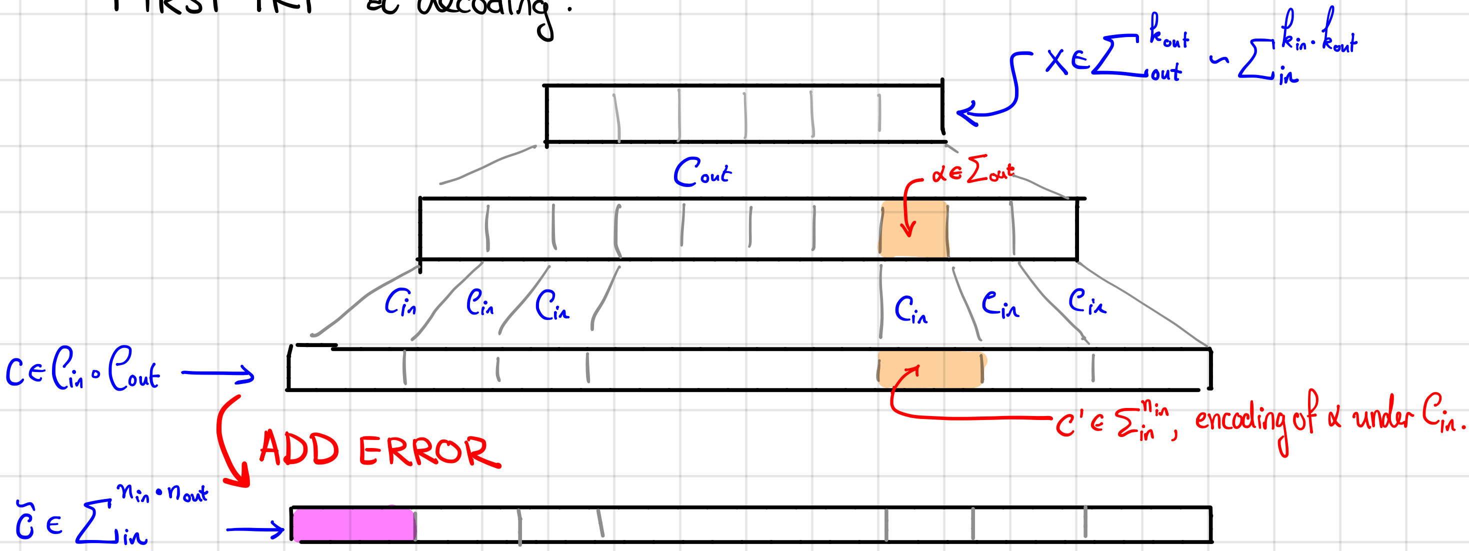
We just saw how to use CONCATENATED CODES to do all of that except the "fast algorithms" part. Now we will do that.

RECALL:

The CONCATENATED CODE $C_{in} \circ C_{out} \subseteq \Sigma_{in}^{n_{out} \cdot n_{in}}$ is defined [by picture...] by



FIRST TRY at decoding:



1. Decode each of these $\boxed{}$ blocks: that is, find the codeword $c' \in C_{in}$ which is the closest to the received word.

2. Convert the "corrected" chunks $\boxed{} \in C_{in}$ into $\boxed{\alpha} \in \Sigma_{out}$

3. Decode C_{out} to get the original message.

CLAIM.* The above works PROVIDED that the number of errors e is $< \frac{d_{in} \cdot d_{out}}{4}$

* Maybe with some floors and/or ± 1's.

NOTICE: $d = d_{in} \cdot d_{out}$ is the designed distance of the concatenated code.
So we'd really like $e \leq \lfloor \frac{d-1}{2} \rfloor$, not $d/4$.
But let's prove the claim anyway, to understand why this approach might fail.

pf (ish). Let's call a block $\boxed{}$ "**BAD**" if there are more than $\lfloor \frac{d_{in}-1}{2} \rfloor$ errors in that block.

If there are e errors total, at most $e / \lfloor \frac{d_{in}-1}{2} \rfloor$ blocks are BAD.

Indeed, that's what happens when there are exactly $\lfloor \frac{d_{in}-1}{2} \rfloor$ errors in each bad block.

If a block is NOT BAD, then the inner code works.

Thus we win provided $(\#\text{BAD BLOCKS}) \leq \lfloor \frac{d_{out}-1}{2} \rfloor$

aka $$e / \left\lfloor \frac{d_{in}-1}{2} \right\rfloor \leq \left\lfloor \frac{d_{out}-1}{2} \right\rfloor$$

$$e \leq \left\lfloor \frac{d_{in}-1}{2} \right\rfloor \left\lfloor \frac{d_{out}-1}{2} \right\rfloor \approx \frac{d_{in} \cdot d_{out}}{4}$$

The proof shows that this might NOT be a good idea.

If the adversary JUST BARELY messes up as many blocks as he can, this decoder will fail on $\lfloor \frac{d-1}{2} \rfloor$ errors.

WHAT ARE WE LEAVING ON THE TABLE?

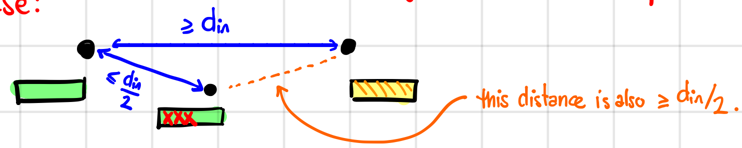
Key observation: When we decode the inner code $\tilde{c} \in \Sigma_{in}^{n_{in}} \to c' \in C_{in}$,

we learn more than just $c' \in C_{in}$; we also know $wt(\tilde{c} \in \Sigma_{in}^{n_{in}} - c' \in C_{in})$.

SOME MOTIVATING EXAMPLES:

① Each block either has 0 or $d_{in}/2$ errors. [This is the bad example from before].

This block had some errors.
When we decode it, it's to something at least $d_{in}/2$ away, because:

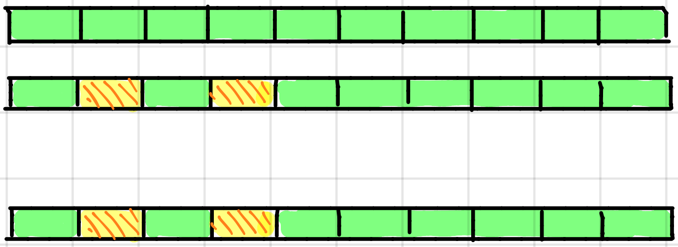


Thus, even though the blocks are incorrect, we can detect that they were incorrect.

So the thing we should do in this case is treat the blocks as ERASURES. We can handle twice as many of those! So our error tolerance is actually about $d/2$ in this case, which is what we wanted.

② MOTIVATING EXAMPLE #2. The bad guy tries to foil our previous example by adding error d_{in} to some blocks, turning them into other codewords.

d_{in} errors in each of e/d_{in} blocks



correct each block. They are all already codewords! Nothing left to do.

Now we can't detect anything! BUT, there are only e/d_{in} corrupted blocks. Again we save a factor of 2 and can correct up to $e \approx d/2$ errors.

We would like to interpolate between these two extremes.

CLAIM. We can efficiently decode $RS_q(n,k)$ from e errors and s erasures, as long as $2e + s < n-k+1$. (aka, the distance of the RS code).

pf-ish. Throw out the s erasures. You are left with $RS(n-s,k)$. Since $2e < (n-s)-k+1$, run Berlekamp-Welch to correct the errors. Now you have an $RS(n,k)$ codeword w/ s erasures. Since $s < n-k+1$, correct the erasures (via linear algebra).

This inspires the following algorithm:

ALGORITHM: **(NOT THE FINAL VERSION).**

Given $\vec{w} = (w_1, w_2, \dots, w_{n_{out}}) \in \left(\mathbb{F}_{q_{in}}^{n_{in}}\right)^{n_{out}}$, s.t. $\Delta(w,c) < \frac{d_{in} \cdot d_{out}}{2}$ for some $c \in C_{in} \circ C_{out}$

For each $i = 1, \dots, n_{out}$:

- Let $w_i' = \underset{y \in C_{in}}{\operatorname{argmin}} \left(\Delta(y, w_i)\right)$
- With probability $\min\left(\frac{2\Delta(w_i, w_i')}{d_{in}}, 1\right)$:
  - set $\beta_i = \perp$
- Else:
  - Set β_i s.t. $E_{in}(y_i) = w_i'$

Run C_{out}'s (error + erasure) decoder on $(\beta_1, \dots, \beta_{n_{out}})$, RETURN the result.

Why does this algorithm work?

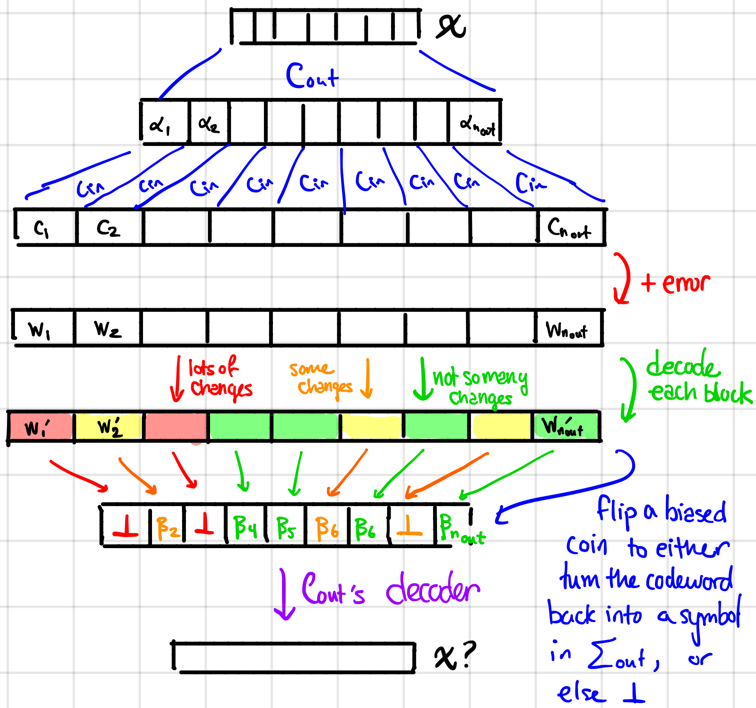
CLAIM. $\underset{\text{algorithm's randomness}}{\mathbb{E}}\left[\left(\#y_i \text{ that } = \perp\right) + 2\cdot\left(\#y_i \text{ that are not correct}\right)\right] < d_{out}$.

Proof (sketch). Let $e_i = \Delta(w_i, c_i)$, so

$$e = \sum_i e_i < \frac{d_{in}\cdot d_{out}}{2}.$$

Let $X_i^\perp = \mathbb{1}\{\beta_i = \perp\}$

$X_i^e = \mathbb{1}\{\beta_i \neq \perp \text{ and } \beta_i \neq \alpha_i\}$



SUBCLAIM. $\mathbb{E}\left[2X_i^e + X_i^\perp\right] \le \frac{2e_i}{d_{in}}$.

(Notice that the SUBCLAIM proves the CLAIM, by linearity of expectation).

pf. of SUBCLAIM:

CASE 1. $c_i = w_i'$. So $X_i^e = 0$, and $\mathbb{E}[X_i^\perp] \le \frac{2\cdot\Delta(\omega_i,\omega_i')}{d_1} = \frac{2\Delta(\omega_i,c_i)}{d_1} = \frac{2e_i}{d_1}$

CASE 2. $c_i \neq w_i'$. As before, $\mathbb{E}[X_i^\perp] = \frac{2\min(\Delta(\omega_i,\omega_i'), d_1)}{d_1}$, and

$\mathbb{E}[X_i^e] = 1 - \mathbb{E}[X_i^\perp]$, since if we didn't find $\perp$, then we made a mistake.

SUBSUBCLAIM. $e_i + \min\left(\Delta(\omega_i,\omega_i'), d_1\right) \ge d_1$.

Given the SUBSUBCLAIM, $\mathbb{E}[2X_i^e + X_i^\perp] = 2(1-\mathbb{E}[X_i^\perp]) - \mathbb{E}[X_i^\perp]$

$= 2 - \mathbb{E}[X_i^\perp]$

$\le \frac{2}{d_1}\left(d_1 - \min(\Delta(\omega_i,\omega_i'), d_1)\right)$

$\le 2e_i/d_1$. By SUBSUBCLAIM.

Proof of SUBSUBCLAIM.

If the min is attained by $\Delta(\omega_i,\omega_i')$, then the claim reads

$\Delta(c_i,\omega_i) + \Delta(\omega_i,\omega_i') \ge d_1$, which is true by the Δ-ineq. (Since $\Delta(c_i,\omega_i') \ge d_1$)

And if it's attained by d_1 then we are done.

So the CLAIM implies that the algorithm works "in expectation."

We could try to turn this into a high probability result (repeat a bunch of times), but instead we will actually be able to DERANDOMIZE it.

STEP 1. We will reduce the necessary randomness by a little bit.

ALGORITHM VERSION 2

Given $\vec{w} = (w_1, w_2, \dots, w_{n_{out}}) \in \left(\mathbb{F}_{q_{in}}^{n_{in}}\right)^{n_{out}}$, s.t. $\Delta(w, c) < \frac{d_{in} \cdot d_{out}}{2}$ for some $c \in C_{in} \circ C_{out}$

CHOOSE $\theta \in [0,1]$ UNIFORMLY AT RANDOM.

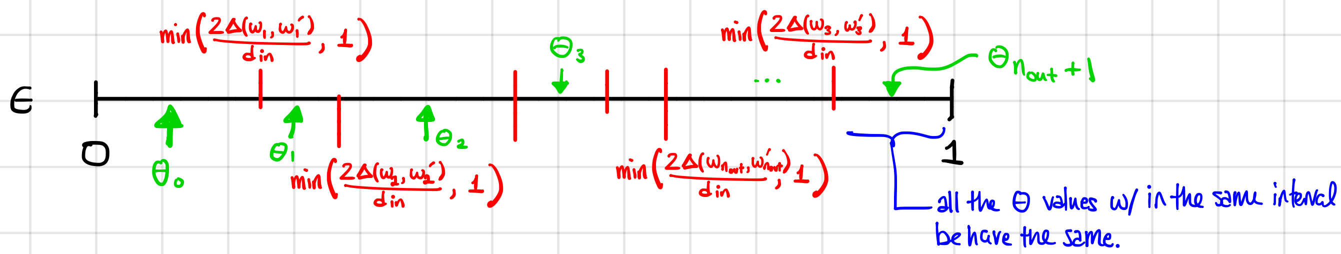
For each $i = 1, \dots, n_{out}$:

- Let $w_i' = \underset{y \in C_{in}}{\operatorname{argmin}} \left(\Delta(y, w_i) \right)$
- **IF** $\theta \le \min\left(\frac{2\Delta(w_i, w_i')}{d_{in}}, 1 \right)$:
  - set $\beta_i = \perp$
- Else:
  - Set β_i s.t. $E_{in}(y_i) = w_i'$

Run C_{out}'s (error + erasure) decoder on $(\beta_1, \dots, \beta_{n_{out}})$, RETURN the result.

That is, we never used the fact that our draws for β_i were independent. So let's make them not at all independent.

Our next step will be to search over all possible θ's. In fact, we only need to look at $n_{out} + 2$ values of θ:



This is called FORNEY'S GENERALIZED MINIMUM DISTANCE DECODER.

ALGORITHM: FINAL VERSION

Given $\vec{w} = (w_1, w_2, \dots, w_{n_{out}}) \in \left(\mathbb{F}_{q_{in}}^{n_{in}}\right)^{n_{out}}$, s.t. $\Delta(w, c) < \frac{d_{in} \cdot d_{out}}{2}$ for some $c \in \mathcal{C}_{in} \circ \mathcal{C}_{out}$

COMPUTE THE $n_{out}+2$ RELEVANT VALUES of Θ, $\Theta_0, \dots, \Theta_{n_{out}+1}$

FOR $j = 0, \dots, n_{out}+1$:

- For each $i = 1, \dots, n_{out}$:
  - Let $w_i' = \underset{y \in \mathcal{C}_{in}}{\operatorname{argmin}} \left(\Delta(y, w_i) \right)$
  - **IF $\Theta_j <$** $\min\left(\frac{2\Delta(w_i, w_i')}{d_{in}}, 1 \right)$:
    - set $\beta_i = \perp$
  - Else:
    - Set β_i s.t. $E_{in}(y_i) = w_i'$
- Run $\mathcal{C}_{out}$'s (error + erasure) decoder on $(\beta_1, \dots, \beta_{n_{out}})$, to obtain $\hat{x}$
- **IF $\Delta(\mathrm{ENC}(\hat{x}), w) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$:**
  - **RETURN $\hat{x}$**

The fact that this algorithm is correct follows from our earlier claim. Since $\underset{\Theta}{\mathbb{E}}\left[2 \cdot (\#\text{errs}) + (\#\text{erasures}) \right] \leq d_{out}$,

there exists some $\Theta \in [0, 1]$ so that $2(\#\text{errs}) + (\#\text{erasures}) \leq d_{out}$, aka so that the alg. finds the correct $\hat{x}$.

Thus, our algorithm above, which tries ALL values of Θ, must find that good value and return the correct answer.

What is the running time of this algorithm?

Depends on the codes. Let's choose our explicit construction from last time:

Recall we had $n_{out} = q_{out} - 1$, and $q_{out} = 2^{k_{in}}$.

- $\mathcal{C}_{out}$ = RS code with rate R_{out}, dist. $\delta_{out} = 1 - R_{out}$
- $\mathcal{C}_{in}$ = Binary linear code on the GV bound, with rate $r \geq 1 - H_2(\delta_{in}) - \epsilon$.

You showed/will show how to find this in time poly(n) on your homework.

The expensive bits of the alg are:

For $O(n_{out})$ choices of Θ:
  For $i = 1, \ldots, n_{out}$:
  - Decode the inner code (length $n_{in} = O(k_{in}) = O(\log(n_{out}))$) by brute force // Time $O(n_{in} \cdot |\mathcal{C}_{in}|) = O(n_{in} \cdot 2^{k_{in}}) = \mathrm{poly}(n)$
  - Run the RS decoder. // Time poly(n)

So altogether the whole thing runs in polynomial time.

We have proved

THM For every $R \in (0,1)$, there is a family $\mathcal{C}$ of EXPLICIT BINARY LINEAR CODES that lies at or above the Zyablov bound. Further, $\mathcal{C}$ can be decoded from errors up to half the Zyablov bound in time poly(n).

AKA, we have achieved our goal! Hooray!

To RECAP the story of Concatenated Codes:

- We considered (RS code) $\circ$ (Binary Linear Code on the GV bound)

- Because the inner code is so small, we can find a good one by brute force in time poly(n).

- We can be a little more clever with the Justesen Code, if we want something asymptotically good and STRONGLY explicit.

- (RS) $\circ$ (Binary code on the GV bd) met the "Zyablov Bound", which was defined as "the bound that these codes meet."

- We saw how to use Forney's GMD decoder to efficiently decode these codes up to half the minimum distance.

QUESTIONS to PONDER:

1. When does code concatenation give distance STRICTLY LARGER than $d_{in} \cdot d_{out}$?

2. Do there exist concatenated codes on the GV bound?

   SPOILER ALERT: YES, see [Thomesson 1983]. (It's a randomized construction)

3. Can we decode these efficiently?

   SPOILER ALERT: ALSO YES. It uses list decoding, we may see it later.

4. Can you do better than the Zyablov bound for EXPLICIT CODES with EFFICIENT ALGS?