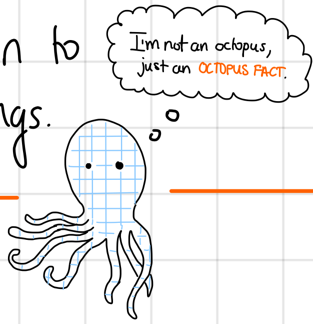
CS250/EE387 - LECTURE 7 - Efficiently decoding concatenated codes.

TODAY'S OCTOPUS FACT

Octopuses can change the color and even the texture of their skin to blend in with their surroundings.



AGENDA

1. ZYABLOV BOUND
2. EFFICIENTLY DECODING CONCATENATED CODES

Recall the GOAL from last lecture:

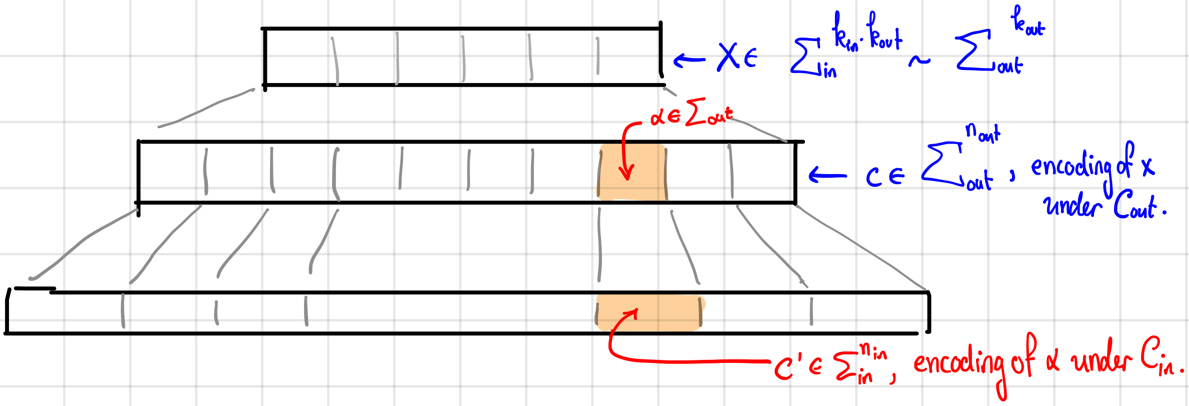
GOAL. Obtain EXPLICIT (aka, efficiently constructible), ASYMPTOTICALLY GOOD families of BINARY CODES, ideally with fast algorithms.

(for today)

Today, we'll see how to use CONCATENATED CODES to achieve this goal.

RECALL:

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

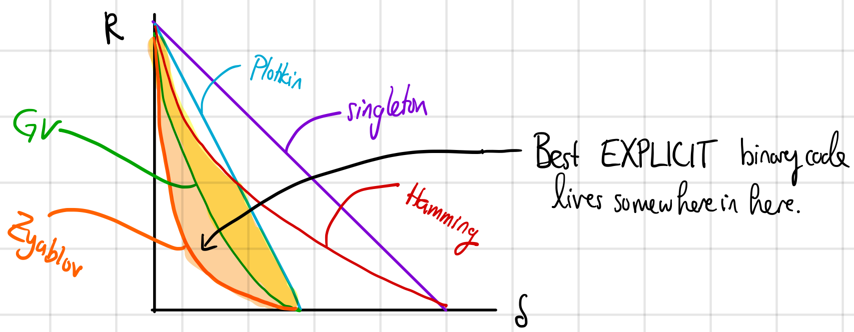


① ZYABLOV BOUND

THM. For any $\varepsilon > 0$, there is a family of explicit binary linear codes of rate R and distance δ, satisfying

$$R \geq \sup_{0 < r < 1 - H_2(\delta) - \epsilon} \left(r \cdot \left(1 - \frac{\delta}{H_2^{-1}(1-r-\epsilon)} \right) \right)$$

That is called the Zyablov Bound.



pf. The construction will be:

- $\mathcal{C}_{out}$ = RS code with rate R_{out}, dist. $\delta_{out} = 1 - R_{out}$
- $\mathcal{C}_{in}$ = Binary linear code on the GV bound, with rate $r \geq 1 - H_2(\delta_{in}) - \epsilon$.
  - We still need to say how to get this... we'll get there.
- The concatenated code has:

RATE: $R_{out} \cdot r$

DISTANCE: $\delta_{out} \cdot \delta_{in} = (1 - R_{out}) \, H_2^{-1}(1 - r - \epsilon)$. ← $R_{out} = 1 - \frac{\delta_{in}\delta_{out}}{H_2^{-1}(1-r-\epsilon)}$

(↑ at least)

Hence $R = R_{out} \cdot r = r \cdot \left(1 - \frac{\delta_{in} \cdot \delta_{out}}{H_2^{-1}(1-r-\epsilon)} \right)$,

and then we get to choose r.

So that gives us the combinatorial bound.
Next, the algorithmic bit.

continued...

proof continued...

Suppose the evaluation pts for the RS code are $\mathbb{F}_q^*$, where $q = 2^{k_{in}}$.

So $k_{in} = \lg(q)$, and $n_{out} = q-1$

ALG 1. Search over all $\mathbb{F}_2$-linear codes of rate r and dimension k_{in}.

There are approximately $2^{n_{in} \cdot k_{in}} = 2^{k_{in}^2/r} = 2^{\lg^2(q)/r}$ such codes

$$q = n_{out} + 1 = \frac{n}{n_{in}} + 1 = \frac{rn}{k_{in}} + 1 \Rightarrow n \sim \frac{1}{r} q \lg(q)$$

(final block length ↙ n)

So $\lg^2(q) = \Theta(\lg^2(n))$, so that's $2^{\Theta(\lg^2(n))} = n^{\Theta(\lg(n))}$, which is **NOT** polynomial time. :(

ALG 2. In class, we will give an alg to construct binary linear codes on the GV bound w/ rate r, dim k_{in} in time $2^{O_r(k_{in})}$, instead of $2^{O_r(k_{in}^2)}$. This will fix the above, and proves the theorem.

So we have acheived part of our goal. (Explicit codes — no algs yet).

\BEGIN{ASIDE}

The following bit about the Wozencraft ensemble is <u>bonus</u>, not in the videos. We may discuss it in class. Feel free to skip it.

HOWEVER, this version of "explicit" [can compute it in polynomial time] may be unsatisfying.

WHAT IF I WANTED "explicit" meaning: "Give me a short, useful description"

Formally, I'd like to be able to compute any entry G_{ij} in time polylog(n).

IDEA: Instead of using the <u>same</u> inner code at every position and requiring it to be good, we'll use a different inner code in each position.

We won't actually know which of these inner codes is good, but we'll know that enough of them are good.

THM. Let $\varepsilon > 0$, fix any k. There is an ensemble of binary linear codes

$$C_{in}^1, C_{in}^2, \ldots, C_{in}^N \subseteq \mathbb{F}_2^{2k}$$

of rate $1/2$, with $N = 2^k - 1$, so that for at least $(1-\varepsilon)N$ values of i, C_{in}^i has distance at least $H_2^{-1}(1/2 - \varepsilon)$.

This is called the **WOZENCRAFT ENSEMBLE**.

proof idea. For $x \in \mathbb{F}_2^k$, treat it as an element of $\mathbb{F}_{2^k}$.
Then for each $\alpha \in \mathbb{F}_{2^k}^*$, let the α^{th} code C_{in}^α be the image of the encoding map

$$E_{in}^\alpha : x \longmapsto (x, \alpha \cdot x)$$

($\alpha \cdot x$: multiplication in $\mathbb{F}_{2^k}$; $(x, \alpha \cdot x)$: treat these as $2k$ bits.)

FUN EXERCISE: finish the proof!

Using the Wozencraft ensemble, we can implement the idea above to obtain the **JUSTESAN CODE**.

DEF (JUSTESEN CODE)

Let $k > 0$ [k will be the dimension of the inner codes in the Wozencraft Ensemble]

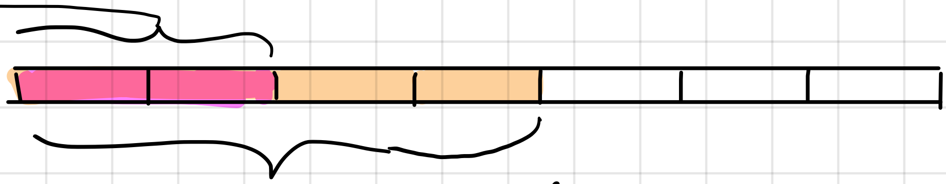
Let $C_{out} = RS_{2^k}\left(\mathbb{F}_{2^k}^*,\ 2^k-1,\ R_{out}\cdot(2^k-1)\right)$

Use the Wozencraft Ensemble as the inner code:

$$C = \left\{ \left\langle E_{in}^{\alpha}(f(\alpha)) \right\rangle_{\alpha \in \mathbb{F}_{2^k}^*} : f \in \mathbb{F}_{2^k}[X],\ \deg(f) < R_{out}(2^k-1) \right\}$$

CLAIM. Let R_{out} be constant. Then [C] is asymptotically good!

pf. (sketch) The rate is $R_{out}/2$, and it's a binary linear code, so we just have to consider the minimum wt to compute the distance. Choose $\varepsilon > \frac{1-R_{out}}{2}$. Consider any codeword:



- At least $(1-R_{out}) \geq 2\epsilon$ fraction of the chunks are the encodings of nonzero symbols.
- At most an ε-fraction of chunks have "bad" inner codes, so at least an $2\varepsilon - \varepsilon = \varepsilon$-fraction of chunks are the encodings of nonzero symbols with a "good" inner code.

For each of those, since the inner code has distance $\geq H_2^{-1}(\frac{1}{2}-\epsilon) = \Theta(1)$, a constant fraction of the bits in each of a constant fraction of blocks are nonzero.

$\Rightarrow$ Each nonzero codeword has relative weight larger than some constant. Thus the code is asymptotically good.

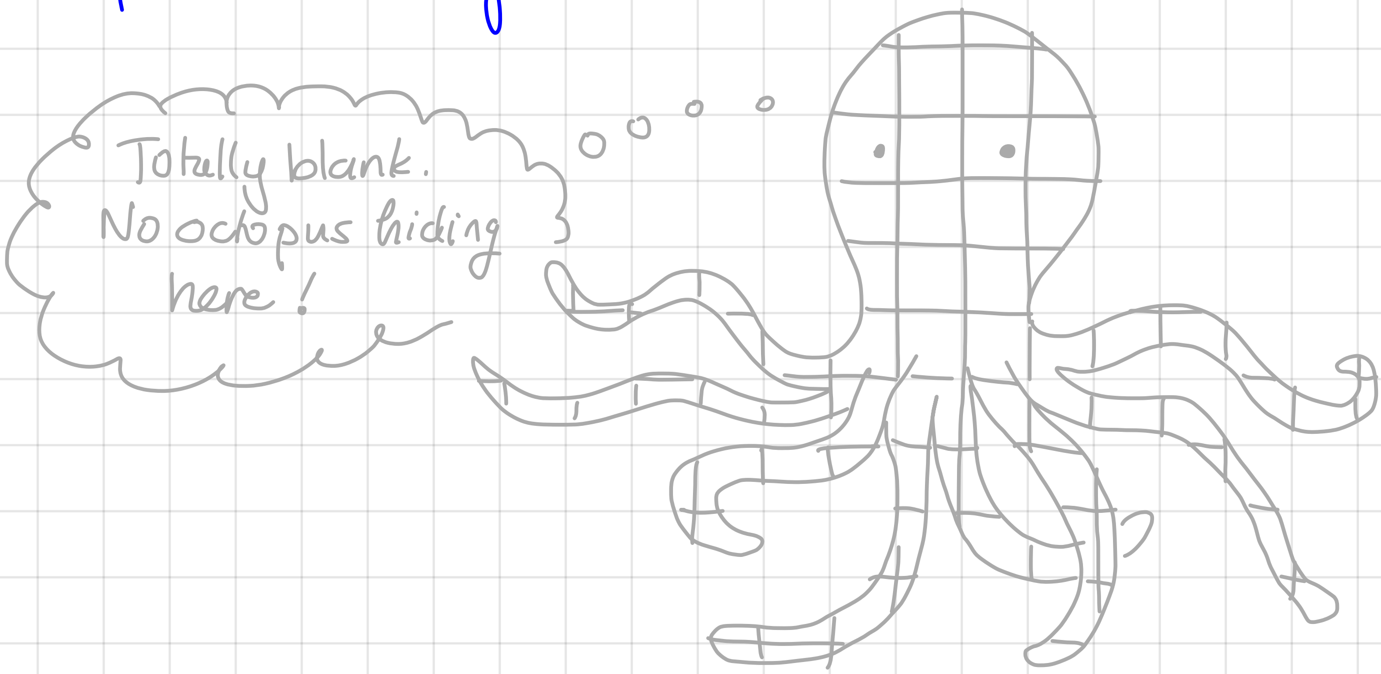
So the JUSTESEN CODE is "EXPLICIT" in the way we wanted.
The α'th block is given by $(f(\alpha), \alpha \cdot f(\alpha)) \in \mathbb{F}_q^2 \sim \mathbb{F}_2^{2\lg(q)}$.
That's pretty explicit!

FUN EXERCISE. What is the best rate/distance trade-off you can get w/ the Justesen code?

FUN EXERCISE. What happens to the Wozencraft ensemble if you do
$x \mapsto (x, \alpha \cdot x, \alpha^2 \cdot x, \ldots, \alpha^r \cdot x)$?

\END{Aside}

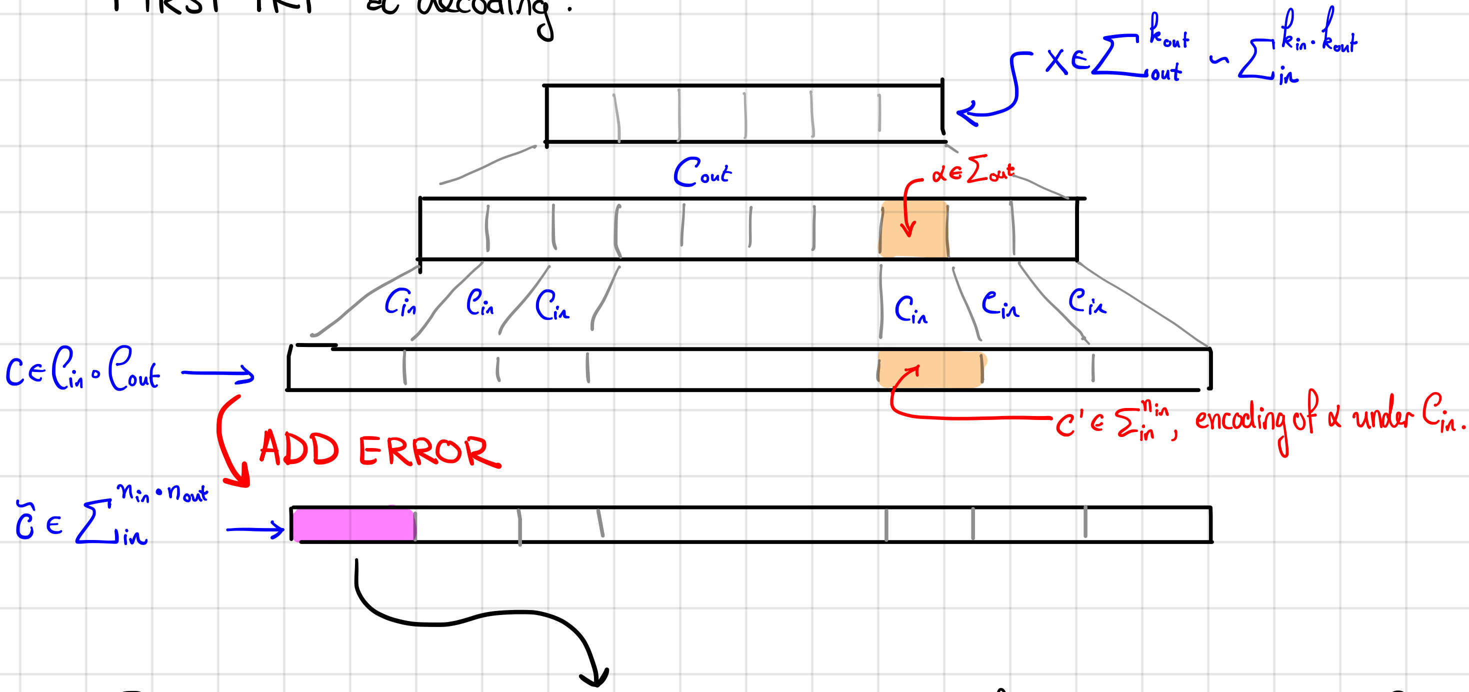
This space intentionally blank



② EFFICIENT DECODING ALGS for CONCATENATED CODES.

Now that we have explicit constructions of asymptotically good codes (and in particular efficient encoding algs), what about efficient DECODING algs?

FIRST TRY at decoding:



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

② Convert the "corrected" chunks [orange] $\in C_{in}$ into [α] $\in \Sigma_{out}$

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 [pink] "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.
If a block is NOT BAD, then the inner code works.
Thus we win provided (# BAD BLOCKS) $\leq \lfloor \frac{d_{out}-1}{2} \rfloor$

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

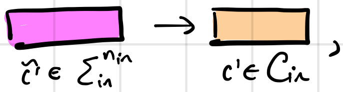
$$e \leq \left\lfloor \frac{d_{in}-1}{2} \right\rfloor \left\lfloor \frac{d_{out}-1}{2} \right\rfloor \approx \frac{d_{in} \cdot d_{out}}{4}$$

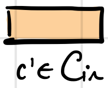
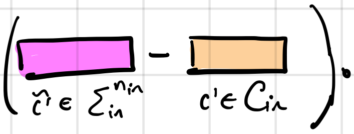
Indeed, that's what happens when there are exactly $\lfloor \frac{d_{in}-1}{2} \rfloor$ errors in each bad block.

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

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

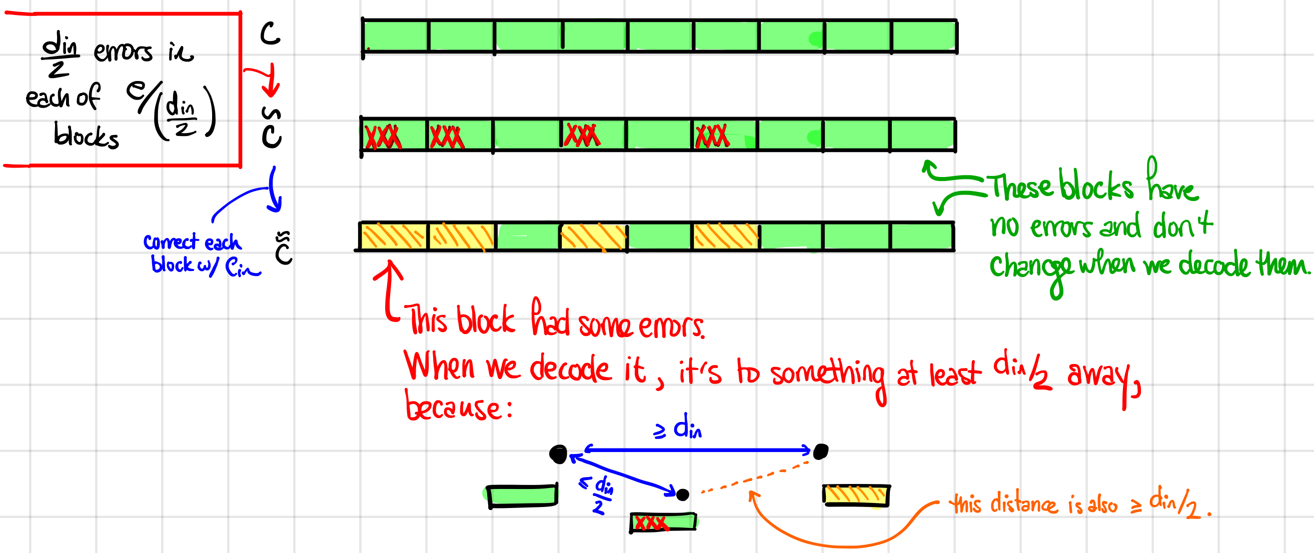
WHAT ARE WE LEAVING ON THE TABLE?

Key observation: When we decode the inner code 

we learn more than just  ; we also know wt().

SOME MOTIVATING EXAMPLES:

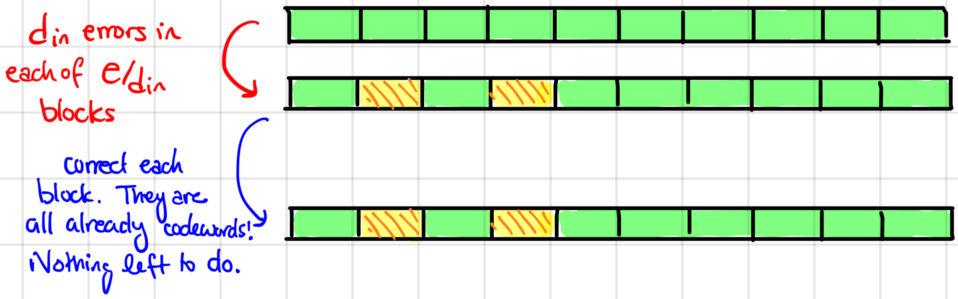
① Each block either has 0 or $d_{in}/2$ errors. [This is the bad example from before].



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.



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 $\beta_i = \bot$
- Else:
  - Set β_i s.t. $E_{in}(\beta_i) = w_i'$

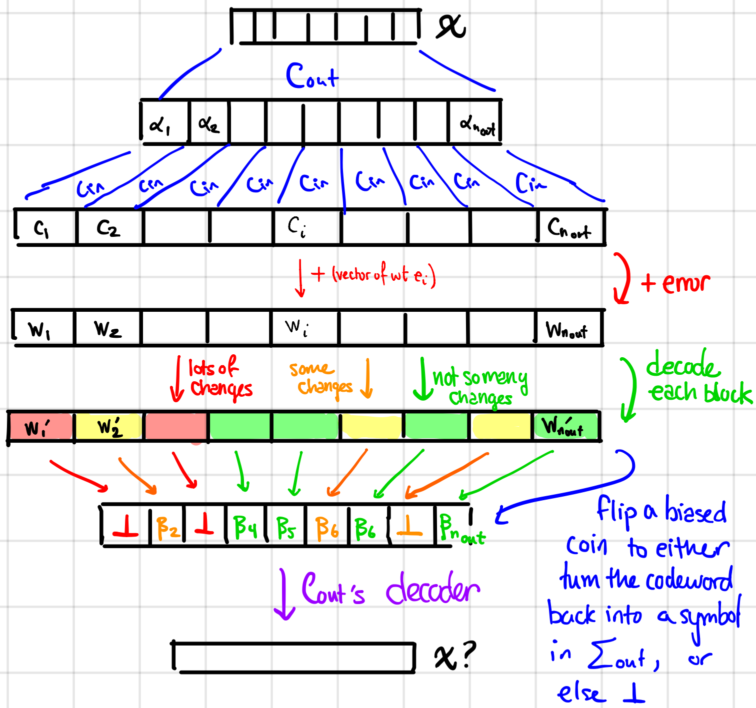
Run C_{out}'s (error + erasure) decoder on $(\beta_1, \dots, \beta_{n_{out}})$, RETURN the result.

Why does this algorithm work?

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

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

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



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

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

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

(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 $\mathbb{E}[X_i^{\bot}] \le \frac{2\cdot\Delta(w_i, w_i')}{d_{in}} = \frac{2\Delta(w_i, c_i)}{d_{in}} = \frac{2e_i}{d_{in}}$

CASE 2. $c_i \neq w_i'$ As before, $\mathbb{E}[X_i^{\bot}] = \frac{\min(2\cdot\Delta(w_i, w_i'), d_{in})}{d_{in}}$, and

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

SUBSUBCLAIM. $2e_i + \min(2\Delta(w_i, w_i'), d_{in}) \ge 2d_{in}$.

Given the SUBSUBCLAIM,

$$\mathbb{E}[2X_i^e + X_i^{\bot}] = 2(1 - \mathbb{E}[X_i^{\bot}]) + \mathbb{E}[X_i^{\bot}]$$
$$= 2 - \mathbb{E}[X_i^{\bot}]$$
$$\le 2 - \frac{1}{d_{in}} \min(2\cdot\Delta(w_i, w_i'), d_{in})$$
$$\le 2 - \frac{1}{d_{in}}(2d_{in} - 2e_i) \quad \text{(SUBSUBCLAIM)}$$
$$= 2e_i / d_{in}, \text{ as desired.}$$

(Proof of SUBSUBCLAIM on next page).

SUBSUBCLAIM. If $c_i \neq w_i'$, $2e_i + \min(2\Delta(w_i, w_i'), d_{in}) \geq 2 \cdot d_{in}$

Proof: Suppose that $2\Delta(w_i, w_i') \leq d_{in}$. Then the SUBSUBCLAIM reads:

$$2e_i + 2\Delta(w_i, w_i') \geq 2d_{in}$$

$$e_i + \Delta(w_i, w_i') \geq d_{in}$$

$$\Delta(w_i, c_i) + \Delta(w_i, w_i') \geq d_{in}$$

which is true since

$$d_{in} \leq \Delta(c_i, w_i') \leq \Delta(w_i, c_i) + \Delta(w_i, w_i')$$

(first inequality: since $c_i, w_i' \in \mathcal{C}_{in}$ and $c_i \neq w_i'$; second inequality: triangle inequality.)

On the other hand, if $d_{in} < 2\Delta(w_i, w_i')$, then the SUBSUBCLAIM reads:

$$2e_i + d_{in} \geq 2d_{in} \quad \text{aka} \quad e_i \geq \frac{d_{in}}{2}.$$

But this must be true because we are in the setting where $c_i \neq w_i'$. Indeed, if $e_i < \frac{d_{in}}{2}$, then the inner code's decoder would have worked correctly and we would have $c_i = w_i'$.

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 $\vec{w} = (w_1, w_2, \ldots, w_{n_{out}}) \in (\mathbb{F}_{q_{in}}^{n_{in}})^{n_{out}}$, 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.

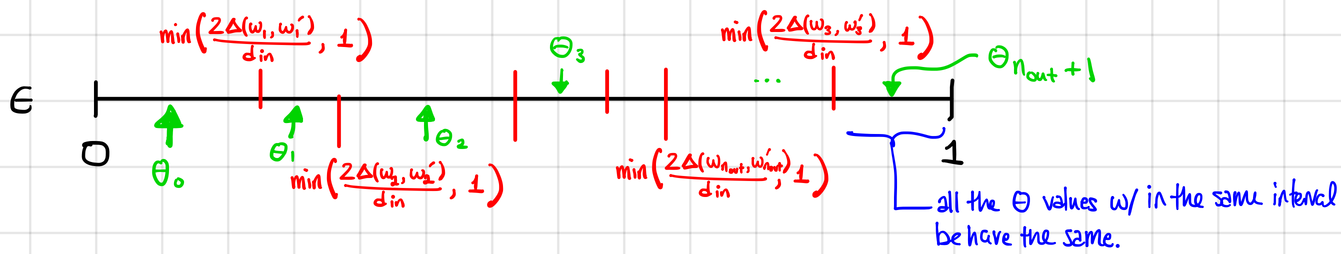
For each $i = 1, \ldots, n_{out}$:

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

Run C_{out}'s (error + erasure) decoder on $(\beta_1, \ldots, \beta_{n_{out}})$, 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 $\Delta(\mathrm{Enc}(\hat{x}), w) \le \lfloor \frac{d-1}{2} \rfloor$:
  - RETURN $\hat{x}$

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

there exists some $\Theta \in [0,1]$ so that $2(\#\text{errs}) + (\#\text{erasures}) \le d_{out}$, 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 ↓

- C_{out} = RS code with rate R_{out}, dist. $\delta_{out} = 1 - R_{out}$
- 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.

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

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 $O(n_{in} \cdot |C_{in}|) = O(n_{in} \cdot 2^{k_{in}}) = \text{poly}(n)$
  - 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?