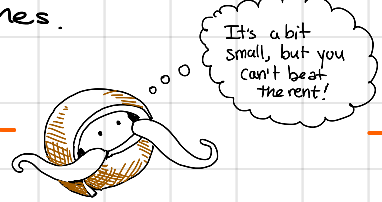
CS250/EE387 – LECTURE 6 – MAKING RS CODES BINARY

AGENDA

1. BCH Codes
2. Reed-Muller Codes [part I]
3. Concatenated Codes [part I]

TODAY'S OCTOPUS FACT

Octopuses use tools! Some octopuses have been observed picking up coconut shells and using them as mobile homes.



The story so far is:

Reed-Solomon Codes have the optimal trade-off between rate and distance, and they have efficient decoding algorithms!

That's a pretty good story. (Maybe we should just stop here?)

HOWEVER, there are some downsides to RS codes. A big one is the alphabet size.

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

for today

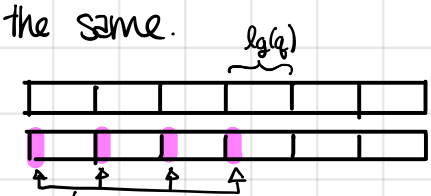
Today we will see a few ways to approach this problem.

[The first several won't work, but they are independently interesting and will come back later.

STRAWMAN. Replace every symbol of $\mathbb{F}_q$ with $\log_2(q)$ bits.
If we start w/ n', k' RS code:

NOTE: This is actually done in practice!

$$\text{Rate} = \frac{k' \cdot \log_2(q)}{n' \cdot \log_2(q)} = k'/n' \quad \text{stays the same.}$$



$$\text{Relative Distance} = \frac{n'-k'+1}{n' \cdot \log_2(q)} \sim \frac{1-R}{\log(n')}$$

if $n' = q-1$

corrupting d bits also can corrupt d symbols. So the distance is $d \geq n'-k'+1$

$$\sim \frac{1-R}{\log\left(\frac{n}{\lg(n)}\right)} \sim \frac{1-R}{\log(n)}$$

where $n = n' \lg(n')$ is the block length of the binary code.

So the STRAWMAN is NOT asymptotically good.
If R is constant, then $\delta \to 0$.

① BCH Codes.

BCH = **B**ose and **R**ay-**C**haudhuri, **H**ocquenghem

What if we just take $RS(n,k) \cap \{0,1\}$?

DEF. Let $n = 2^m - 1$, let γ be a primitive element of $\mathbb{F}_{2^m}$.
Then for $d \le n$, define

$$BCH(n,d) = \left\{ (c_0, \ldots, c_{n-1}) \in \mathbb{F}_2^n \;\middle|\; c(\gamma^j) = 0 \quad \forall j = 1, \ldots, d-1 \right\}$$

$c(X) = \sum_{i=0}^{n-1} c_i \cdot X^i$

Notice that this is exactly the same as our def. of RS codes, except that we restrict $(c_0, \ldots, c_{n-1}) \in \mathbb{F}_2^n$ instead of $\mathbb{F}_{2^m}^n$. In particular, $dist(BCH(n,d)) = d$.

b/c $BCH(n,d) \subseteq RS(n, n-d+1)$ and $RS(n,n-d+1)$ has dist d.

NOTE. BCH codes make sense if you replace "2" with "q" [any prime power].
We focus on binary codes for today.

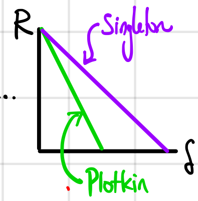
FALSE

CLAIM. BCH codes are linear codes with dimension $\ge n - d + 1$.

BAD **Proof.** $c(\gamma^j) = 0$ is a linear constraint; there are $d-1$ such constraints, so the dimension is at least $n - (d-1)$.

GREAT! So, BCH codes are binary codes that meet the Singleton bound!

Erm....



doesn't that violate the Plotkin bound??

(Yes).

What's wrong?!

The problem is that the constraints $c(\gamma^j)=0$ are linear over $\mathbb{F}_{2^m}$, not over $\mathbb{F}_2$. Fortunately, BCH codes ARE still linear over $\mathbb{F}_2$:

CLAIM. $BCH(n,d)$ is $\mathbb{F}_2$-linear with dimension $\geq n-(d-1)\lg(n+1)$.

proof. Each constraint $c(\gamma^j)=0$ is actually $m=\log_2(n)$ linear constraints over $\mathbb{F}_2$. To see this, we'll use the fact that $\mathbb{F}_{2^m}$ is actually a vector space over $\mathbb{F}_2$ of dimension m:

$\mathbb{F}_{2^m}$ has the same additive structure as $\mathbb{F}_2^m$.

So it makes sense to write elements $\alpha \in \mathbb{F}_{2^m}$ as vectors $v_\alpha \in \mathbb{F}_2^m$, as long as we're only going to be adding them or multiplying by scalars in $\mathbb{F}_2$.

Then $c(\gamma^j)=0$ means:

$c_i \in \{0,1\}$

$$\sum_{i=0}^{n-1} c_i \gamma^{ij} = 0 \iff \sum_{i=0}^{n-1} c_i \cdot \underbrace{\boxed{}}_{v(\gamma^{ij}) \in \mathbb{F}_2^m} = 0$$

$$\iff m\left\{\underbrace{\begin{bmatrix} | & | & | & & | \\ v_1 & v_{\gamma^i} & v_{\gamma^{2i}} & \cdots & v_{\gamma^{(n-1)i}} \\ | & | & | & & | \end{bmatrix}}_{n} \right. \begin{bmatrix} \\ \vec{c} \\ \end{bmatrix} = 0$$

binary matrix

$\vec{c} \in \mathbb{F}_2^n$

So each $\mathbb{F}_{2^m}$-linear constraint is actually m $\mathbb{F}_2$-linear constraints, and altogether we have $m\cdot(d-1)$ $\mathbb{F}_2$-linear constraints.

$$n - m(d-1) = n - \underbrace{\log_2(n+1)}_{n=2^m-1 \Rightarrow m=\lg(n+1)} \cdot (d-1),$$ as claimed.

In fact, we can do even better:

CLAIM. $BCH(n,d)$ is $\mathbb{F}_2$-linear with dimension $\geq n - \lceil \frac{d-1}{2} \rceil \lg(n+1)$.

Proof. We'll show that the linear constraints $c(\gamma^j)=0$ are actually redundant: $c(\gamma^j)=0 \iff c(\gamma^{2j})=0$. This cuts the number of constraints in half, which gives the bound.

SUB CLAIM. For any $c \in \mathbb{F}_2[X]$, $\alpha \in \mathbb{F}_{2^m}$, $c(\alpha)=0 \iff c(\alpha^2)=0$.

pf.

$$c(\alpha) = 0$$

$\iff [c(\alpha)]^2 = 0$ — Since $0^2=0$

$\iff \left[\sum_{i=0}^{n-1} c_i \alpha^i\right]^2 = 0$ — Def. of $c(X)$

$\iff \sum_{i=0}^{n-1} c_i^2 \alpha^{2i} = 0$ — Since in $\mathbb{F}_{2^m}$, $1+1=0$, so $(a+b)^2 = a^2+ab+ab+b^2 = a^2+b^2$ and more generally $(\sum_i a_i)^2 = \sum_i a_i^2$.

$\iff \sum_{i=0}^{n-1} c_i \alpha^{2i} = 0$ — Since $c_i \in \{0,1\}$, so $c_i^2 = c_i$.

$\iff c(\alpha^2) = 0$ — Def. of $c(X)$ again.

and by the above reasoning the SUB CLAIM proves the CLAIM.

Is this good? $R \geq \frac{n - \lceil \frac{d-1}{2} \rceil \cdot \lg(n+1)}{n} \sim 1 - \frac{\delta}{2}\lg(n)$, aka

BCH codes

$$\boxed{\text{Rate } R, \text{ distance } \delta \sim \frac{2}{\lg(n)} \cdot (1-R).}$$

So BCH codes do slightly better than our STRAWMAN:

STRAWMAN

$$\boxed{\text{Rate } R, \quad \text{distance } \delta \sim \frac{1-R}{\lg(n)}}$$

But still not asymptotically good :(.

However! Note that BCH codes can be decoded with either Berlekamp-Welch or Berlekamp-Massey! So that's pretty cool.

(2) BINARY REED-MULLER CODES

(Silly) idea: just do RS codes over $\mathbb{F}_2$ directly! $\quad RS_2(n,k) = \left\{ (f(\alpha_1),\ldots,f(\alpha_n)) \;\middle|\; \begin{array}{l} f\in\mathbb{F}_2[X] \\ \deg(f)<k \end{array} \right\}$

This is obviously silly since (a) $\deg(f) \le q-1 = 1$ to be interesting

(b) $\alpha_1,\ldots,\alpha_n$ should be distinct pts in $\mathbb{F}_2$, so $n \le 2$.

However, one fix is to add more variables.

DEF. The BINARY m-VARIATE REED-MULLER CODE of DEGREE r is

$$RM_2(m,r) = \Big\{ \underbrace{\big(f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_{2^m})\big)}_{\substack{\{\alpha_1,\ldots,\alpha_{2^m}\} = \mathbb{F}_2^m, \\ \text{in any pre-determined order.}}} : \underbrace{f \in \mathbb{F}_2[X_1,\ldots,X_m]}_{\substack{m\text{-variate polynomials over } \mathbb{F}_2.\ \text{eg,} \\ f(X_1,X_2) = 1 + X_1X_2 + X_1}}, \underbrace{\deg(f) \le r}_{\substack{\deg \text{ means total degree. eg,} \\ \deg(X_1X_2)=2.}} \Big\}$$

Parameters:

Block length $n = 2^m$

Dimension $k = \sum_{j=0}^{r} \binom{m}{j} = \mathrm{Vol}_2(r,m)$. ← This is the number of coefficients in $f(X_1,\ldots,X_m) = \sum_{\substack{S\subseteq[m] \\ |S|\le r}} c_S \cdot \prod_{i\in S} X_i$, a generic degree $\le r$ m-variate polynomial over $\mathbb{F}_2$.

Distance $d = ?$

LEMMA (Binary Schwartz-Zippel)

Let $f \in \mathbb{F}_2[X_1,\ldots,X_m] \neq 0$, with $\deg(f) \le r$.

Then $\sum_{\alpha\in\mathbb{F}_2^m} \mathbb{1}\{f(\alpha)\neq 0\} \ge 2^{m-r}$.

We may do the proof later for a more general version, but if you haven't seen this before it's a FUN EXERCISE!

So $\mathrm{dist}(RM_2(m,r)) \geq 2^{m-r}$. This is because $RM_2(m,r)$ is linear, and so as usual we only need to look at the minimum wt of a codeword.

And, it turns out this is the correct answer: consider $f(X_1,\dots,X_m) = X_1 \cdot X_2 \cdots X_r$. This vanishes whenever any of $X_1, \dots, X_r = 0$, and so

$$\left| \{ \alpha \in \mathbb{F}_2^m : f(\alpha) \neq 0 \} \right| = \left| \{ \alpha \in \mathbb{F}_2^m : \alpha_1 = \dots = \alpha_r = 1 \} \right| = 2^{m-r}.$$

So for $RM_2(m,r)$:
$$n = 2^m \qquad\qquad$$
$$k = \mathrm{Vol}_2(r,m) \quad \Rightarrow \quad R = \mathrm{Vol}_2(r,m)/2^m$$
$$d = 2^{m-r} \qquad\qquad \delta = 1/2^r$$

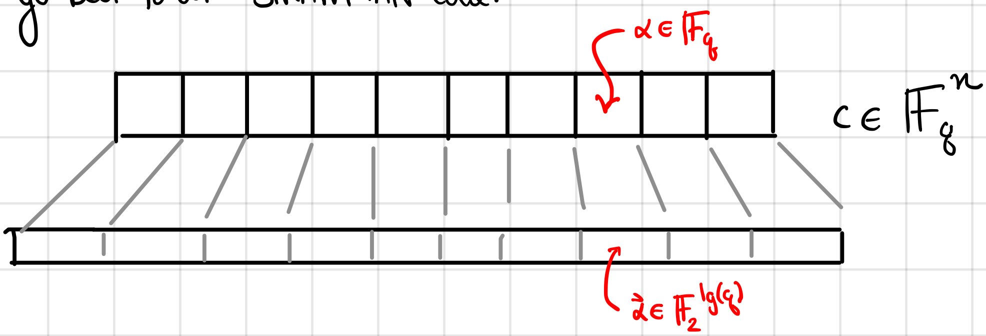
RM codes also admit efficient decoding algs. We'll see some of these later in the course.

Unfortunately, this isn't asymptotically good either. If $\delta = \Theta(1)$, then r is constant but $m \nearrow$, so $R \searrow 0$.

So this doesn't acheive the GOAL either... :(

③ CONCATENATED CODES.

Let's go back to our STRAWMAN code:

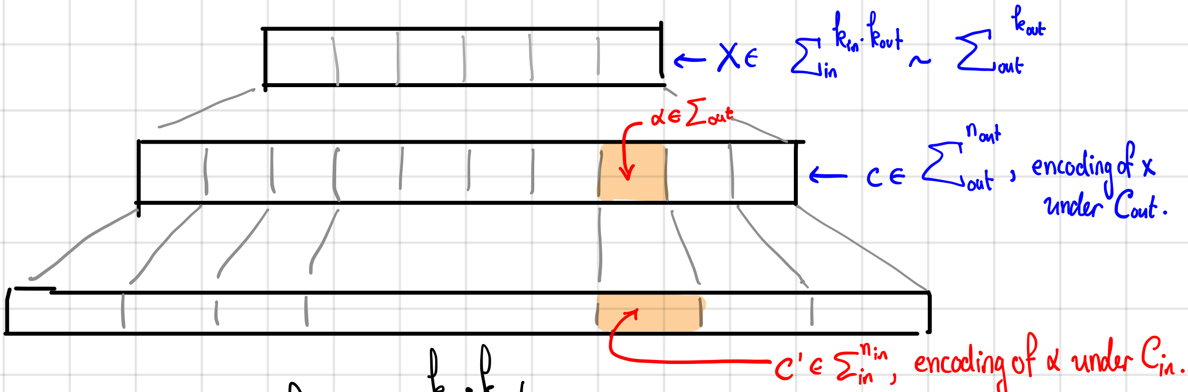


This wasn't a good idea, because if I were a bad guy, I'd mess up one bit from each of $d+1$ of the length $\lg(q)$ blocks. So these $\underbrace{\boxed{}}_{\lg(q)}$ blocks were not very robust.

IDEA. Let's encode each block $\vec{\alpha} \in \mathbb{F}_2^{\lg_2(q)}$ with a good binary ECC!

DEF. Let $C_{out} \subseteq \Sigma_{out}^{n_{out}}$ be a q_{out}-ary code of dimension k_{out} and distance d_{out}.
Let $C_{in} \subseteq \Sigma_{in}^{n_{in}}$ be the same thing with "in" subscripts, so that $q_{out} = q_{in}^{k_{in}}$.

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



Formally, the encoding of $x \in \Sigma_{q_{in}}^{k_{in} \cdot k_{out}}$ is given by:

- Treat $x \in \left[\Sigma_{in}^{k_{in}}\right]^{k_{out}} \cong \Sigma_{out}^{k_{out}}$
- Let $y = C_{out}(x) \in \Sigma_{out}^{n_{out}} \cong \left[\Sigma_{in}^{k_{in}}\right]^{n_{out}}$
- Let $c = \underbrace{\mathrm{Enc}_{in}(y_1) \circ \cdots \circ \mathrm{Enc}_{in}(y_{n_{out}})}_{n_{in} \cdot n_{out} \text{ symbols}}$ where $\mathrm{Enc}_{in} : \Sigma_{in}^{k_{in}} \to \Sigma_{in}^{n_{in}}$, and $\circ$ is concatenation.

Parameters of Concatenated Codes:

alphabet: Σ_{in}

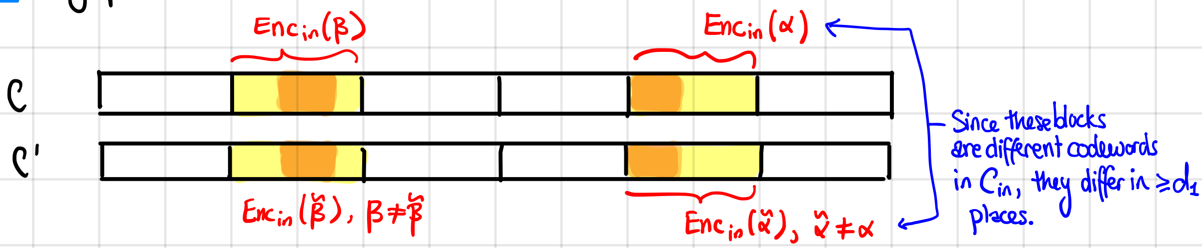
message length: $k_{in} \cdot k_{out}$

Codeword length: $n_{out} \cdot n_{in}$

} So the rate is $\frac{k_{in} \cdot k_{out}}{n_{in} \cdot n_{out}} = R_{in} \cdot R_{out}$

PROPOSITION. The distance of $C_{in} \circ C_{out}$ is at least $d_{in} \cdot d_{out}$.

pf. by picture: Let $c, c' \in C_{in} \circ C_{out}$:



- At least d_{out} blocks of c, c' are encoding different symbols.
- In each of those, there are at least d_{in} symbols in Σ_{in} that differ.
- So that's $d_{out} \cdot d_{in}$ differences total.

[in particular, the relative distance is $\delta = \delta_1 \cdot \delta_2$]

DEF. The DESIGNED DISTANCE of a concatenated code as in is $d_{in} d_{out}$

Finally! Progress to our GOAL.

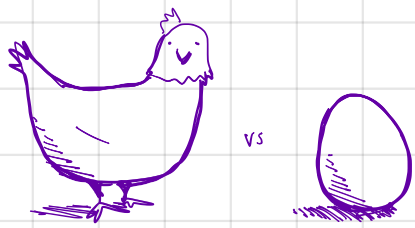
To obtain an EXPLICIT, ASYMPTOTICALLY GOOD BINARY CODES:

1. Set C_{out} = Reed-Solomon Code
2. Set C_{in} = EXPLICIT, ASYMPTOTICALLY GOOD BINARY CODE.

D'oh.

But actually it's OKAY!

The secret is that C_{in} will be short enough that we can do exhaustive stuff efficiently. We'll see how this works next time!



QUESTIONS to PONDER

1. How would you efficiently decode a concatenated code?
2. How would you efficiently decode Reed-Muller codes?