Problem 1: Singleton Bound

Lemma 3 (Singleton Bound) *If C is an $(n, k, d)_q$ code then $d \leq n - k + 1$.*

Proof

Let C be a $(n, k, d)_q$ code. Then it has q^k codewords of length n. Project C to a $k-1$ dimensional space by only considering the first $k-1$ entries of each code words (see Figure 1). There are at most q^{k-1} possible strings of length $k-1$ in $\mathbb{F}_q^{k-1}$. However, we have in total q^k code words so there must exist at least two codewords with the same projections. Thus the distance between these two code words is $n - (k-1)$. Hence $d \leq n - (k-1) = n - k + 1$. ■

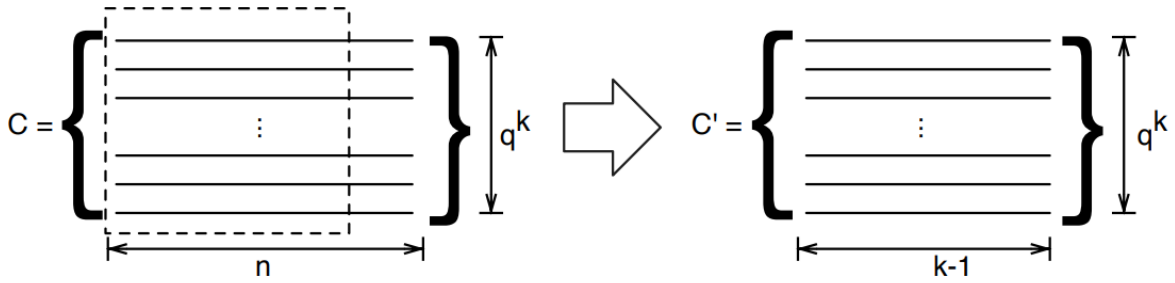

Figure 1: Projection used in proof of Lemma 3.

lect16.pdf (purdue.edu)

Problem 2: Plotkin bound

Theorem 4 *Let C be a binary code C of block length n and distance d.*

1. *If $d > n/2$, then $|C| \leq \frac{2d}{2d-n}$.*
2. *If $d \geq n/2$, then $|C| \leq 2n$.*

PROOF: Let $m = |C|$ and let $c_1, c_2, \ldots, c_m \in \{0,1\}^n$ be the codewords of C. By hypothesis $\Delta(c_i, c_j) \geq d$ for $1 \leq i < j \leq m$. We will map the codewords into unit vectors $v_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, m$, such that the angle between every pair of vectors is at least 90 degrees (i.e., their dot product $\langle v_i, v_j \rangle < 0$). These vectors are defined as follows:

$$v_i = \frac{1}{\sqrt{n}}\left((-1)^{c_i[1]}, (-1)^{c_i[2]}, \cdots, (-1)^{c_i[n]}\right) ,$$

where $c_i[\ell]$ is the ℓ'th bit of the codeword c_i. It is easy to check that

$$\langle v_i, v_j \rangle = \frac{1}{n}(n - 2\Delta(c_i, c_j)) \leq \frac{n-2d}{n} .$$

When $d \geq n/2$, these dot products are non-positive, and by the second part of Lemma 3, we can bound $m \leq 2n$.

For the first part, if $2d > n$, then $\langle v_i, v_j \rangle \leq -\frac{2d-n}{n} < 0$, and therefore by the first part of Lemma 3, we can bound

$$m \leq 1 + \frac{n}{2d-n} = \frac{2d}{2d-n} .$$

□

notes4.pdf (cmu.edu)

Problem 3: Fermat's Little Theorem

Theorem: (Fermat). If p is a prime and a is any number not divisible by p, then

$$a^{p-1} \equiv 1 \bmod p$$

For example, we know from this, without calculating, that $3^{22} \equiv 1 \bmod 23$.

It's more convenient to prove

$$a^p \equiv a \bmod p \text{ for all } a.$$

This clearly follows from the above congruence by multiplying it by a. And Fermat's little theorem follows from this congruence by canceling a which is allowed if p does not divide a.

The proof uses the binomial theorem. Clearly, $1^p \equiv 1 \bmod p$. Now

$$2^p = (1+1)^p = 1 + \binom{p}{1} + \binom{p}{2} + \cdots + \binom{p}{p-1} + 1 \equiv 1 + 0 + 0 + \cdots + 0 + 1 = 2 \bmod p.$$

Once we have $2^p \equiv 2 \bmod p$, we use the binomial theorem again to find 3^p:

$$3^p = (1+2)^p = 1 + \binom{p}{1}2 + \binom{p}{2}2^2 + \cdots + \binom{p}{p-1}2^{p-1} + 2^p \equiv 1+0+0+\cdots+0+2 = 3 \bmod p.$$

This process can be continued indefinitely to prove the result. (Technically, the result $a^p \equiv a \bmod p$ is found by induction on a.)

https://www.math.nyu.edu/~hausner/fermat.pdf

Problem 4: A finite field of order 3^2

Step 1: choose a primitive polynomial (ex: $x^2 + 1$)

Step 2: find a primitive element (ex: $y = x + 1$)

A primitive element generates the group mod the primitive polynomial

y^0	y^1	y^2	y^3	y^4	y^5	y^6	y^7	y^8
1	$x+1$	$2x$	$2x+1$	2	$2x+2$	x	$x+2$	1

Addition Table

+	0	y^0	y^1	y^2	y^3	y^4	y^5	y^6	y^7
0	0	y^0	y^1	y^2	y^3	y^4	y^5	y^6	y^7
y^0	y^0	y^4	y^7	y^3	y^5	0	y^2	y^1	y^6
y^1	y^1	y^7	y^5	y^0	y^4	y^6	0	y^3	y^2
y^2	y^2	y^3	y^0	y^6	y^1	y^5	y^7	0	y^4
y^3	y^3	y^5	y^4	y^1	y^7	y^2	y^6	y^0	0
y^4	y^4	0	y^6	y^5	y^2	y^0	y^3	y^7	y^1
y^5	y^5	y^2	0	y^7	y^6	y^3	y^1	y^4	y^0
y^6	y^6	y^1	y^3	0	y^0	y^7	y^4	y^2	y^5
y^7	y^7	y^6	y^2	y^4	0	y^1	y^0	y^5	y^3

Multiplication Table

*	0	y^0	y^1	y^2	y^3	y^4	y^5	y^6	y^7
0	0	0	0	0	0	0	0	0	0
y^0	0	y^0	y^1	y^2	y^3	y^4	y^5	y^6	y^7
y^1	0	y^1	y^2	y^3	y^4	y^5	y^6	y^7	y^0
y^2	0	y^2	y^3	y^4	y^5	y^6	y^7	y^0	y^1
y^3	0	y^3	y^4	y^5	y^6	y^7	y^0	y^1	y^2
y^4	0	y^4	y^5	y^6	y^7	y^0	y^1	y^2	y^3
y^5	0	y^5	y^6	y^7	y^0	y^1	y^2	y^3	y^4
y^6	0	y^6	y^7	y^0	y^1	y^2	y^3	y^4	y^5
y^7	0	y^7	y^0	y^1	y^2	y^3	y^4	y^5	y^6

Problem 5: Peeling Algorithm

H =

0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

<u>Erasure pattern 1: {1, 2, 3}</u>

C_1 =

0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

With Erasures:

e	e	e	0	0	0	0	1	1	1	1	1	1	1	1

Parity check # for erasures

E_1: 4

E_2: 3

E_3: 3,4

Decoding fails as there are no dangling erasures

<u>Erasure pattern 2: {3, 4, 9}</u>

C_2 =

0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

With Erasures:

0	0	e	e	1	1	1	0	e	0	0	1	1	1	1

Parity check # for erasures

E_1: 3,4

E_2: 2

E_3: 1,4

We have a dangling bit for parity check 1,2, and 3 so we calculate the parity at the parity check node and set the erasure to that value. We successfully decode:

0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

Erasure pattern 3: { 4, 5, 13}

C_3 =

1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

With Erasures:

1	0	1	e	e	0	1	0	1	0	1	0	e	0	1

Parity check # for erasures

E_1: 2

E_2: 2,4

E_3: 1,2,4

We have a dangling bit for parity check 1, which has current parity 1, we update our codeword to

1	0	1	e	e	0	1	0	1	0	1	0	1	0	1

Parity check # for erasures

E_1: 2

E_2: 2,4

We have a dangling bit for parity check 4, which has current parity 1, we update our codeword to

1	0	1	e	1	0	1	0	1	0	1	0	1	0	1

Parity check # for erasures

E_1: 2

We have a dangling bit for parity check 2, which has current parity 0, we update our codeword to

1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Decoding was successful.