Nonbinary block coding, Galois Fields

Coding Technology

Illés Horváth

2025/10/01

Brief history of Error Correction Codes

✓: discussed in this course, ✓: coming up.

- Binary linear block codes
  - repeater, single parity check bit ✓
  - Hamming (1950's) ✓
  - Hadamard (1950's, but ideas were around earlier) ✓
  - binary Golay (1950's) ✓
  - CRC (1960's) ✓
  - Reed-Muller (1960's)
  - polar (2010's)
  - LDPC (basics 1960's ✓, improvements since 1990's)
- Nonbinary linear block codes
  - ternary Golay (1950's)
  - Reed-Solomon (1960's) ✓
  - BCH (1960's) ✓
- Convolutional codes (1950's)
  - sequential decoding (1960's)
  - Viterbi, trellis decoding (1960's)
  - turbo codes (since 1990's)

Brief history of Error Correction Codes

So which code is the best?

In terms of getting close to the Shannon-limit, LDPC codes, turbo codes and polar codes are best, but all require large block sizes.

- LDPC codes are slightly better than turbo codes
- polar codes are potentially better, but only for very large block sizes and only by a small margin, while difficult to implement.

Turbo codes have been used extensively for the last 20+ years for broadband communication.

5G standards include LDPC codes for the data channel and polar codes for the control channel.

For applications where large block sizes are not an option, Reed-Solomon codes are the most popular as they are very versatile and scalable. Hamming, Golay, Reed-Muller and CRC codes are also used for smaller scale applications and specific purposes.

q-ary Symmetric Channel

We aim to switch the arithmetic used from binary to q-ary, with $q > 2$ digits.

Instead of the binary symmetric channel, we switch to the q-ary symmetric channel:

- Input and output alphabets both have q digits, practically labelled as $0, 1, \ldots, q-1$.
- For each digit transmitted, it will be changed with probability p to another digit (uniformly, that is, to any other digit with the same probability), or remain unchanged with probability $1-p$.
- Digits are changed independently from other digits.
- We assume $p < \frac{q-1}{q}$. (For $p = \frac{q-1}{q}$, no information can be transmitted through the channel.)

For a given channel, q is fixed.

q-ary Symmetric Channel

The Hamming-distance of two q-ary vectors of the same length is defined as the number of positions where the two vectors have different digits.

The weight of a q-ary vector is the number of positions with nonzero digits.

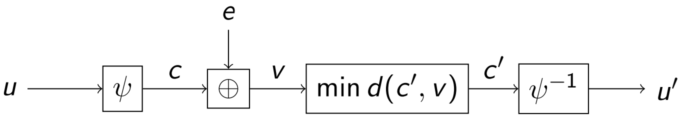
A $C(n, k)$ code maps k-digit messages to n-digit codewords $(n > k)$.

The code rate is k/n.

A code is called systematic if the first k digits of any codeword are the corresponding message vector.

Coding scheme

The general block coding scheme:



e

u ψ c $\oplus$ v $\min d(c', v)$ c' ψ^{-1} u'

(Minor remark: for $\oplus$, we need addition of q-ary digits. It is coming soon, but for now, think about applying e as randomly changing some digits to something else.)

Tetracode

Example. The tetracode is a $C(4,2)$ ternary $(q=3)$ code.

$$\begin{gathered}
(0\,0) \to (0\,0\,0\,0)\\
(0\,1) \to (0\,1\,1\,2)\\
(0\,2) \to (0\,2\,2\,1)\\
(1\,0) \to (1\,0\,1\,1)\\
(1\,1) \to (1\,1\,2\,0)\\
(1\,2) \to (1\,2\,0\,2)\\
(2\,0) \to (2\,0\,2\,2)\\
(2\,1) \to (2\,1\,0\,1)\\
(2\,2) \to (2\,2\,1\,0)
\end{gathered}$$

It is systematic.

Error correction and error detection

Similarly to the binary setup, the error correction and detection capabilities of a code depend on the minimal Hamming-distance between codewords, denoted by $d_{\min}$.

Theorem

Let $d_{\min}$ be the minimal Hamming-distance among codewords. Then the above coding scheme can...

- *detect $d_{\min} - 1$ errors, and*
- *correct $\lfloor \frac{d_{\min}-1}{2} \rfloor$ errors.*

Proof. Identical to the binary case.

Singleton bound

The Singleton bound holds in identical form for the q-ary case. The proof is also the same.

Theorem (Singleton bound)

$$d_{\min} \leq n - k + 1.$$

Proof. Take all q^k codewords, and delete the last $d_{\min} - 1$ bits from each codeword. The resulting truncated codewords must still be all different since originally their Hamming-distance was at least $d_{\min}$.

The q^k truncated codewords have length $n - d_{\min} + 1$, and there are $q^{n-d_{\min}+1}$ different vectors in $\{0, \ldots, q-1\}^{n-d_{\min}+1}$, so

$$q^k \leq q^{n-d_{\min}+1}$$

must hold, and $d_{\min} \leq n - k + 1$ is obtained by rearranging.

Hamming bound

The Hamming bound also remains valid, but in slightly different form.

Theorem (Hamming bound)

Assume a $C(n,k)$ q-ary code can correct $t = \lfloor \frac{d_{\min}-1}{2} \rfloor$ errors. Then

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^{n-k}.$$

Proof. As before, consider all q^k codewords and balls of radius t around them.

Each ball contains $\sum_{i=0}^{t} \binom{n}{i} (q-1)^i$ points, and the balls must be disjoint, so they contain $q^k \sum_{i=0}^{t} \binom{n}{i} (q-1)^i$ points total, while there are q^n points in the entire space $\{0, \ldots, q-1\}^n$, so

$$q^k \sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^n.$$

Perfect and MDS codes

Codes that satisfy the Hamming bound with equality are called perfect codes.

Codes that satisfy the Singleton bound with equality are called maximum distance separable (MDS) codes.

For binary codes, the only MDS code was the repeater code. For q-ary codes with $q > 2$, the MDS property becomes more relevant, as there exist nontrivial MDS codes.

There are also further perfect codes for $q > 2$.

Tetracode

Example. Is the tetracode an MDS code?

The tetracode has the following codewords:

$(0000), (0112), (0221), (1011), (1120), (1202), (2022), (2101), (2210)$

The code parameters are $n = 4, k = 2$. We also need d_{min}.

Pairwise comparison shows $d_{\text{min}} = 3$, and

$$d_{\text{min}} = 3 = n - k + 1 = 4 - 2 + 1$$

holds, so the tetracode is an MDS code.

Is it a perfect code?

$d_{\text{min}} = 3$, so it can correct $\lfloor \frac{d_{\text{min}}-1}{2} \rfloor = 1$ error, and

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i = \binom{4}{0} (3-1)^0 + \binom{4}{1} (2-1)^1 = 9 = q^{n-k} = 3^2,$$

so yes, it is also a perfect code.

q-ary vs binary architecture

Computer and communication hardware use almost exclusively using binary architecture.

How can we make an architecture with q digits work on binary hardware?

The most practical solution is to use $q = 2^\ell$, and then each digit can be represented by ℓ bits.

That said, we will look at codes and examples for other q's as well.

Arithmetics using q digits

At this point, even though we labeled the digits $\{0, 1, \ldots, q-1\}$, they are just symbols in the sense that the only thing relevant so far was whether two digits are equal or different.

It would be convenient to make calculations with the digits, e.g. for linear codes. For that, we need arithmetics using q digits.

Generally, what we expect for an arithmetics is (at least) the 4 base operations: $+, -, *, /$.

Field axioms

A field is a set F with two operations (both on two variables): $+$ and $*$ that satisfy the following axioms:

Addition "$+$"	Multiplication "$*$"
$\alpha, \beta \in F \to \alpha + \beta \in F$	$\alpha, \beta \in F \to \alpha * \beta \in F$
$\alpha + \beta = \beta + \alpha$	$\alpha * \beta = \beta * \alpha$
$(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$	$(\alpha * \beta) * \gamma = \alpha * (\beta * \gamma)$
$\exists\, 0 : \forall \alpha \in F : \alpha + 0 = \alpha$	$\exists\, 1 : \forall \alpha \in F : \alpha * 1 = \alpha$
$\forall \alpha \in F\, \exists \beta : \alpha + \beta = 0;$	$\forall \alpha \in F \backslash \{0\} :\ \exists \beta : \alpha * \beta = 1;$
$\quad \beta = \alpha_a^{-1} = -\alpha$	$\quad \beta = \alpha_m^{-1} = \alpha^{-1}$

$$\alpha * (\beta + \gamma) = \alpha * \beta + \alpha * \gamma$$

Liberty to define "$+$" and "$*$" as long as they satisfy the above axioms.

Fields

Example. The real numbers are a field.

Are the nonnegative real numbers a field? No, there is no additive inverse.

Are the integer numbers a field? No, there is no multiplicative inverse. (E.g. $1/2$ is not an integer number.)

Further examples: the complex numbers and the rational numbers are both fields.

The set $\{0, 1\}$ with the usual binary operations is also a field - a finite field.

A finite field with q elements is called a Galois Field, and denoted GF(q).

GF(3)

Try to construct GF(3).

Start with the addition. Then multiplication.

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

This is known as the mod 3 arithmetics.

mod q arithmetics

Does the mod q arithmetics work for any q? Check for $q = 4$.

Addition works fine, but due to $2 \times 2 = 0$ mod 4, 2 does not have a multiplicative inverse.

In general, the mod q arithmetics only satisfies the field axioms if q is a prime (proof soon).

So does that mean there is no GF(4)?

Not necessarily; all it means is that the mod 4 arithmetics does not work. Maybe there is another arithmetics for GF(4) that works.

Galois Fields

Theorem

For any GF(q), q must be either a prime number or a prime power (p^m, with p prime and $m \geq 2$).

Proof (sketch). Consider the elements

$$1, 1+1, 1+1+1, \ldots$$

Since the field is finite, elements must repeat at some point. If

$$\underbrace{1+\cdots+1}_{j} = \underbrace{1+\cdots+1}_{i} \quad \rightarrow \quad \underbrace{1+\cdots+1}_{p=j-i} = 0.$$

p must be a prime, otherwise if $p = ab$, then

$$\underbrace{(1+\cdots+1)}_{a} \times \underbrace{(1+\cdots+1)}_{b} = 0,$$

and neither term would have a multiplicative inverse.

Galois Fields

The set

$$\{0, 1, 1+1, \ldots, \underbrace{1+\cdots+1}_{p-1}\}$$

is a subfield of the original field.

Does this set include all elements of GF(q)?

- If yes, then $q = p$ and GF(q) has the mod q arithmetics.
- If no, then, using the fact that every field is a vector space over any of its subfields (this is a result from abstract algebra that we use without proof), it follows that the additive structure of GF(q) is identical to the vector space

  $$\{0, 1, \ldots, p-1\}^m$$

  for some $m \geq 2$, and then $q = p^m$.

Galois Fields

Technically, to finish the theorem, we still have to prove that:

- For the $q = p$ case, GF(q) with the mod q arithmetics actually satisfies the field axioms.
- For the $q = p^m$, we have yet to define the multiplicative structure of the arithmetics, and prove that it satisfies the field axioms.

We will address the $q = p$ case very soon, and the $q = p^m$ case in a later lecture.

Further remarks.

- For both the $q = p$ and the $q = p^m$ case, p is called the characteristics of the field.
- For a fixed q, GF(q) is unique up to isomoprhism (that is, if you have two fields of q elements, there is a bijection between the elements that also preserves the arithmetics).

GF(q) for prime q

From among the field axioms, everything except the multiplicative inverse is satisfied for the integer numbers, and the modulo q operation is consistent with integer addition and multiplication, so the corresponding axioms are also satisfied in GF(q) when q is a prime.

So the only thing that actually needs to be checked is multiplicative inverse.

If a, b are nonzero elements from GF(q), then $ab \neq 0$. This is true because $ab = 0$ would imply $ab = nq$ (over the integers), but q is a prime and a, b are both between 1 and $q - 1$, so this is not possible.

For any $a \neq 0$, the numbers $\{a, 2a, \ldots, (q-1)a\}$ are all distinct and nonzero, so one of them must be equal to 1, and if $ab = 1$, then $a^{-1} = b$.

Examples

Let's do a few calculations in GF(7).

$$
\begin{aligned}
5 + 3 &= 1 \pmod 7 \\
5 \times 3 &= 15 = \ 1 \pmod 7 \\
-5 &= 2 \pmod 7 \\
3 - 5 &= 3 + 2 = \ 5 \pmod 7 \\
5^{-1} &= 3 \pmod 7
\end{aligned}
$$

Examples

How to compute the multiplicative inverse?

Example. Compute 5^{-1} in GF(11).

Start with the number 1, then in each step, increase by 11, until we get a number divisible by 5.

$$1, 12, 23, 34, 45$$

Then $45/5 = 9$, and

$$5^{-1} = 9 \pmod{11}$$

(This method only works for small values of q, but that is sufficient for nonbinary coding, as q is usually small. We will need multiplicative inverse mod q for large q values for cryptography later; we will discuss a different method then.)

Examples

Compute 3^{10} in GF(7).

Instead of computing every power of 3 from 3^1 through 3^{10}, we only compute the following:

$$\begin{aligned} 3^2 &= 9 = 2 \pmod 7 \\ 3^4 &= (3^2)^2 = 2^2 = 4 \pmod 7 \\ 3^8 &= (3^4)^2 = 4^2 = 2 \pmod 7, \end{aligned}$$

then

$$3^{10} = 3^{8+2} = 2 \times 2 = 4 \pmod 7.$$

GF(q) for prime q

Theorem

For every nonzero element $a \in GF(q)$,

$$a^{q-1} = 1.$$

Proof. The numbers

$$\{a, 2a, \ldots, (q-1)a\}$$

are all distinct and nonzero, so they are in fact a permutation of $\{1, 2, \ldots, q-1\}$, so

$$a \times 2a \times \cdots \times (q-1)a = 1 \times 2 \times \cdots \times (q-1),$$

and dividing by $1 \times 2 \times \cdots \times (q-1)$ (which is nonzero) gives

$$a^{q-1} = 1.$$

GF(q) for prime q

The order of a is the minimal m for which $a^m = 1$.

Example. The order of nonzero elements in GF(7):

element	powers						order
a	a^1	a^2	a^3	a^4	a^5	a^6	m
1	1	1	1	1	1	1	1
2	2	4	1	2	4	1	3
3	3	2	6	5	4	1	6
4	4	2	1	4	2	1	3
5	5	4	6	2	3	1	6
6	6	1	6	1	6	1	2

GF(q) for prime q

If the order of a is $m = q - 1$, we call a a primitive element.

Theorem

There is always at least one primitive element in GF(q).

(No proof.)

Example. The order of the elements in GF(7):

element	powers						order	
a	a^1	a^2	a^3	a^4	a^5	a^6	m	
1	1	1	1	1	1	1	1	
2	2	4	1	2	4	1	3	
3	3	2	6	5	4	1	6	– primitive element
4	4	2	1	4	2	1	3	
5	5	4	6	2	3	1	6	– primitive element
6	6	1	6	1	6	1	2	

The powers of a primitive element give all nonzero elements in GF(q).