Cyclic codes

Rodrigo Ribeiro de Aguiar

MAGEF 2023/2024

1 Introduction

When information is transmitted in some way, some of it may be lost. If we are talking with someone we may mishear some words. If we are watching a movie from a DVD, some of the information that encodes the movie may be lost by scratching the disc (and thus some seconds of the movie may not contain audio). In all these cases someone sent information through a communication channel (in the first case our hearing and speaking systems and in the second case our DVD and its reading/writing systems) and some of it was lost. There is, then, a need of trying to recover some of this lost information, so that we can associate our received message to, lets say, a message which makes sense to receive (in the first case this can be done in many cases by trying to guess the misheard word so it fits the context of the rest of the received phrase).

In the first example we were able to guess the correct word because the language we use in a daily basis has some degree of redundancy, so one would guess that adding such redundancy to a message we want to transmit makes it more likely for the receiver to guess what the correct message is, in the case that the noise in the used communication channel caused some error to happen in one of the components of the message. One way of doing this is using the theory of linear codes.

2 Linear Codes

A code is a function $c : X^* \to A^*$, which receives a sequence of symbols from a set X - the original message - and has as an output a sequence of symbols from a set A - the encoded message. We call the set A the encoding alphabet and we can, then, define the set of the encoded messages we expect to receive (informally speaking) as the image of this function. Such set is called the set of *codewords*.

One of the simplest and most efficient ways to encode messages may be achieved by using linear algebra: We can see our original message a as a vector of dimension k, and by multiplying our vector with a $k \times n$ matrix G we get a codeword aG, containing n symbols from the encoding alphabet A. Such matrix is called a *generator matrix* and contains in its rows a set of k codewords. It is also needed for n to be bigger than k, so that we have some degree of redundancy in the encoded message (as it contains more symbols than the original message). Such code is denoted as an $[n, k]$-*linear* code. One can also define a parity-check matrix H, which is a $(n-k) \times n$ matrix such that a word $c \in A^*$ is a codeword if and only if $Hc^T = 0$. This matrix is particularly useful because it enables us to check if a codeword is in C, and if it is not what is its error pattern. Note that, if $c \in C$, and e is the error pattern, $H(c+e)^T = Hc^T + He^T = He^T$, and one could find e by finding the word x such that $Hx^T = He^T$. The original codeword may be obtained by subtracting e to the received word. The name given to He^T is the *syndrome* of $(c+e)$ and the name of this procedure is *syndrome decoding*. This procedure is only possible if the minimal distance between 2 codewords (we define distance between two words as the number of coordinates in which they differ) is more than two times the number of coordinates in which an error happened, as if that were not the case there could be two words x_1 and x_2 such that $Hx_1^T = Hx_2^T = He^T$. We say, then, that an $[n, k, d]$-linear code is *t-error correcting*, where $t = \lfloor \frac{d-1}{1} \rfloor$ and d is the minimal distance between two codewords.

In the scope of Linear codes, it is particularly convenient to define our alphabets to be *finite fields*.

3 Finite Fields

A field is a ring where multiplication is commutative and all elements (except zero) have a multiplicative inverse. As we saw in the course, $\mathbb{Z}_q$ defines a field (which shall be denoted $\mathbb{F}_q$) if and only if q is a prime

number. Despite this fact, there is a way of constructing fields with non-prime order. Let $\mathbb{F}_q[x]$ be the set of polynomials with coefficients in $\mathbb{F}_q$, that is:

$$\mathbb{F}_q[x] = \{a_0x^0 + ... + a_mx^m : a_i \in \mathbb{F}_q, \forall i \in \{1, ..., m\}\} \tag{1}$$

for all $m \geq 0$. We can, using one of these polynomials, define a quotient field $\mathbb{F}_q[x]/(f(x))$, where $f(x) \in \mathbb{F}_q[x]$ (taking $f(x)$ to be irreducible in $\mathbb{F}_q$). Defining a quotient with a polynomial is a simple generalization of what we have seen in the course: in our new quotient field, a polynomial $g(x) = f(x)h(x)$ will be taken to be equal to $h(x)$ (where $g(x)$ and $h(x)$ are also in $\mathbb{F}_q[x]$). We can determine what every polynomial from $\mathbb{F}_q[x]$ is equal to in this new field by applying the usual polynomial division algorithm.

Let α be a root of a degree m polynomial $f(x)$, that is, $f(\alpha) = 0$. Then, we know that $\alpha^m = -\frac{1}{a_m}(a_0 + ... + a_{m-1}\alpha^{m-1})$. This means that if we try to get the k^{th} power of α we end up with another element of the field. It can then be seen that this new quocient group is really just the set of all polynomials of the form:

$$p(x) = a_0 + ... + a_{m-1}x^{m-1} : a_i \in \mathbb{F}_q, \forall i \in \{1, ..., k\} \tag{2}$$

and since there are q choices for the values of each of the a_i, we observe that our new field has order q^m. This construction is also valid if the order of the field we take the quotient from is not prime, as long as that field is itself the extension of some prime order field.

4 Cyclic Codes

Cyclic codes are one of the most important class of linear codes. A linear code $C \subset \mathbb{F}^n$ over a finite field $\mathbb{F}$ is called a *cyclic code* if applying a cyclic shift of the form $(u_0, ..., u_{n-1}) \to (u_{n-1}, u_0, ..., u_{n-2})$ to any codeword $c \in C$ gives us another codeword in C. Note that the field $\mathbb{F}^n$ is isomorphic to $\mathbb{F}[x]/(x^n - 1)$ by the isomorphism

$$\rho((u_0, ..., u_{n-1})) = \sum_{i=0}^{n-1} u_i x^i \tag{3}$$

and, thus, every codeword in C can be represented by a polynomial.

It can be proven that a code in $\mathbb{F}[x]/(x^n - 1)$ is cyclic if and only if it is an ideal of this field, and that every ideal of $\mathbb{F}[x]/(x^n - 1)$ can be generated by a single element (that is, there is $u \in C$ such that for every $c \in C$, $c = uv$ for some $v \in \mathbb{F}[x]/(x^n - 1)$. With these two results in mind, it can be proven that every code $C \subset \mathbb{F}[x]/(x^n - 1)$ has a generator polynomial $g(x)$ such that $g(x), xg(x), ..., x^{k-1}g(x)$ is a basis for C, where k is the maximal natural number such that the different polynomials in the basis are linearly independent (so they can form a basis). Note that multiplying $g(x)$ by x is the same as doing the mentioned cyclic shift!

One of the biggest advantages of these codes is that the decoding is much more powerful, in the sense that the original message can be obtained even if the number of transmission errors is bigger than $\lfloor \frac{d-1}{1} \rfloor$, as long as they occur close to each other: A *l-burst* is a vector such that the shortest cyclic interval containing all non-zero coordinates has lenght l. For example, 0001010 and 1000010 are 3-bursts. A cyclic code can correct any l-burst if each of its cosets contains at most one burst of lenght l or less. As sometimes the maximal number l such that this happens is bigger than t, there is a big advantage in using cyclic codes, and that is why they are the base for many types of error-correcting codes that are used nowadays: Reed-Solomon Codes are used in some NASA missions; Many binary Hamming codes are equivalent to cyclic codes and share their properties (in particular the Hamming(7,4) code, which is still used nowadays); The Quantum error-correcting Steane Code uses cyclic codes to correct qubit flip and phase flip errors. A lot of work has been (and is still) done in order to get more efficient and powerful error-correcting codes, so they can be less time and resource expensive, especially in quantum computation, as nowadays it is still hard to have a big number of coherent (and thus available for computations) qubits, and so the theory of error-correcting codes is a very relevant application of what we've learnt about rings and fields in this course.