CHAPTER 6

ERROR-CORRECTING CODES

6.1. Generalities on codes

The idea behind coding is the following. Suppose you want to send a message to someone through a noisy channel, then your message is likely to get altered along the way and the receiver might not be able to understand the received message. A code is an extra bit of information that you add to your message so that, even if some errors occured during transmission, the receiver can decide whether something went wrong and, in the best case, even reconstruct the original message. The applications of coding theory are numerous: error-correcting codes are used in CDs, DVDs, etc, in transmission of images from spatial probes to Earth, ...

6.1.1. Vocabulary. — To transmit messages, we use a finite alphabet A, with q letters. We send words of a fixed length $n \geq 1$, so that a word is an element of A^n. Most of the time, we choose $A = \mathbb{F}_q$ a finite field (this puts limitations on the possible number of letters/symbols, but there is then a richer structure on A). The set of words A^n is endowed with the Hamming distance $d(-,-)$:

$$\forall w = (w_1, \ldots, w_n), w' = (w'_1, \ldots, w'_n), \qquad d(w, w') = \# \left\{ i \in \{1, \ldots, n\} \ : \ w_i \neq w'_i \right\}.$$

One can show that this is indeed a distance (for a reasonnable definition of distance; namely, it should satisfy axioms like: two words are equal if and only if they are at distance 0 from each other, distance is symmetric, and there is a triangle inequality). If we send a word w to someone, we expect that he receives a word w' which is "close to" the original one, in the sense that most of the letters are correct and that only a few errors have been made during transmission (*i.e.* $d(w, w')$ is small).

A code is a subset $C \subset A^n$ such that $\#C \geq 2$. Elements of C are called codewords (or valid words). The distance of C is defined by:

$$d(C) = \min \left\{ d(w, w'), \ w, w' \in C \text{ s.t. } w \neq w' \right\}.$$

The principle of coding is then the following: one we have chosen a code C, we only send words $w \in C$; if the receiver gets a word $w' \in A^n$, he can then check whether w' is in C or not. If w' is not in C, then he will know that something went wrong during the transmission (*i.e.* that an error has been made).

Better still, if t errors have been made, with $t \leq d(C) - 1$, then we can systematically detect that one or more errors occured. Even better, if t errors occured during transmission and if $2t + 1 \leq d(C)$, then there exists a unique word $\widetilde{w} \in C$ such that $d(w', \widetilde{w}) \leq t$ (proof left as an exercise), so that $\widetilde{w} = w$. And thus, the receiver can reconstruct the original message w from the blurred one he received (if one assumes that not too many errors occured), *i.e.* C allows us to correct t errors. From the previous paragraph one deduces that:

Lemma 6.1. — *Let C be a code with distance $d(C)$. Let t be the maximum number of errors that C can systematically correct. Then*

$$t = \left\lfloor \frac{d(C)-1}{2} \right\rfloor,$$

where $\lfloor . \rfloor$ is the floor function. We then say that C is t-correcting.

There are essentially two qualities that a code should have: it should be able to detect and correct many errors, and it should be very efficient (*i.e.* we shouldn't need add much redundancy to the actual message that we want to send). We won't go into the computational details here, but it should be noted that one usually also wish to work with codes for which coding/decoding is easy and fast.

We thus associate to a code C, two ratios to measure how good C is: first, define the correcting ratio:

$$\gamma(C) := \frac{1}{\#C} \left\lfloor \frac{d(C)-1}{2} \right\rfloor \in [0,1],$$

which measures the proportion of errors in a word that C can correct. Secondly, define the information ratio of C:

$$\tau(C) := \frac{\log \#C}{\log(\#A^n)} = \frac{\log \#C}{n \cdot \log \#A},$$

which measures the proportion of symbols in a codeword that are actually carrying information. Finding a good code means finding a code C with $\tau(C)$ and $\gamma(C)$ as close to 1 as possible.

6.1.2. Examples. — Let us first give a few basic examples of classical codes:

Example 6.2 **(Repetition code).** — The idea behind this code is pretty simple: instead of sending a bit of data once, send it many times (say, 5 times). For simplicity, assume that $A = \{0,1\}$ (*i.e.* we send binary words), and that we wish to encode words of length 4. The code here is

$$C := \left\{ w = (\epsilon_1, \dots, \epsilon_{20}) \in \{0,1\}^{20} \ : \substack{\epsilon_1=\epsilon_5=\cdots=\epsilon_{17},\ \epsilon_2=\epsilon_6=\cdots=\epsilon_{18}, \\ \epsilon_3=\epsilon_7=\cdots=\epsilon_{19},\ \epsilon_4=\epsilon_8=\cdots=\epsilon_{20}} \right\} \subset A^{4\times 5} = A^{20}.$$

If the message is $m = (\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4)$, we send the codeword $w = (m, m, m, m, m) \in C$. Now assume that the receiver gets a word $w' = (x_i) \in \{0,1\}^{20}$, then decoding works as follows: for each $i \in \{1, \dots, 4\}$, form a set $E_i = \{x_i, x_{i+4}, x_{i+8}, x_{i+12}, x_{i+16}\}$. If E_i has more 0's than 1's, then we decide that a 0 was meant in place i, and we set $e_i = 0$ (and vice versa). The decoded message is then $(e_1, \dots, e_4)$. If a majority of bits were correct, the decoded message is the original one. The receiver can thus correct errors of at most 2 bits in a 5-uple E_i. The price to pay is that we need to 5 times as many bits as actually required (*i.e.* we add a lot of redundancy).

It is easy to see that $d(C) = 5 = 2 \times 2 + 1$, so that $t(C) = 2$ and C is 2-correcting. Since $\#C = 2^4 = 16$, we get $\gamma(C) = 1/8 = 12,5\%$ and $\tau(C) = 1/5 = 20\%$.

Example 6.3 **(Parity-check bit).** — Again, we use the alphabet $A = \{0,1\}$. We wish to transmit 4-bit words. Encode a message $(\epsilon_1, \dots, \epsilon_4)$ by adding an extra bit ϵ_5 at the end (called the parity-check bit) such that the sum $\sum_{i=1}^{5} \epsilon_i$ is $\equiv 0 \bmod 2$. We transmit this 5-bit word. The receiver checks whether the received message $(\epsilon'_1, \dots, \epsilon'_5)$ satisfies $\sum_{i=1}^{5} \epsilon'_i \equiv 0 \bmod 2$. If not, then he knows that an error occured during transmission.

This code can detect a 1-bit error in a word, but he can not correct it (since the receiver has no way of knowing where the error is).

Example 6.4 **(Matrix parity-check code).** — Let us build on the previous example. Again, we want to send a binary word $m = (\epsilon_1, \dots, \epsilon_4)$ of length 4. We add some redundancy as follows: form a 3×3 matrix

$$\begin{bmatrix} \epsilon_1 & \epsilon_2 & * \\ \epsilon_3 & \epsilon_4 & * \\ * & * & * \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix},$$

where the values of x_3, x_6, x_7, x_8 and x_9 are determined by the condition that all rows and all colums of the matrix contain an even number of 1's. Given m, we send the word $w = (x_1, \dots, x_9)$. The receiver can then form a 3×3 matrix and check whether the rows and columns add up to 0 modulo 2. If at most one error occured, the receiver can detect it, and even correct it! As an example, assume that one receives

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Then only the second column and the first row do not sum up to zero (modulo 2), so we know that an error was made at their intersection, and we change the bit there. The original message in this case was $(1, 1, 1, 0)$.

As an exercise, you can check that this code has length 9, that $\#C = 16$ and that $d(C) = 4$. Thus:

$$t(C) = 1, \quad \gamma(c) = 1/9 \approx 11,1\%, \qquad \tau(C) = 4/9 \approx 44,4\%.$$

The correction rate is slightly worse than for the repetition code, but the information rate is much better. As a further exercise, explain how to generalize this construction to send longer messages of length ℓ^2 (for $\ell \geq 2$) and compare the parameters of the resulting code.

***Example 6.5* (ISBN code).** — The ISBN code is used to identify books. It consists in a 9-digit number together with an extra symbol which is either a digit or an X (for a total length of 10). Examples are $0 - 412 - 29690 - X$, $0 - 387 - 95432 - 5$, $0 - 387 - 97825 - 3$, ... (Note that the placements of the "–" need not always be the same). The last symbol is a sort of "parity check with weights" and is computed from the 9 first digits as follows. Assume that the first 9 digits are $a_1, a_2, \dots, a_9$, compute

$$x := \sum_{i=1}^{9} i \cdot a_i = a_1 + 2a_2 + 3a_3 + \cdots + 9a_9 \bmod 11,$$

and put $a_{10} = x$ if $x \in \{0, \dots, 9\}$ and $a_{10} = X$ if $x = 10$. Note that $10a_1 + 9a_2 + 8a_3 + \cdots + 2a_9 + x \equiv 0 \bmod 11$.

The two most common errors when copying such a code are: either altering one digit, or transposing two adjacents digits. And the ISBN code is designed precisely to detect if one of these errors occured. Using that the length of a valid ISBN code is less that 11, and that 11 is prime, you can show:

- All possible valid ISBN codes have at least two digits different from each other. That is, the distance of the ISBN code is ≥ 2.
- There are no pairs of valid ISBN codes which have 8 digits in common and two transposed digits.

So the check-digit at the end ensures that it is always possible to detect the two most common mistakes (if either occurs, the result is never a valid ISBN code). The ISBN code can not correct any errors. Note that it is not always possible to detect the alteration of 2 or more digits, or to detect the transposition of 3 or more digits (you can build examples).

***Example 6.6* (BSN number).** — The BSN number for residents of the Netherlands is a 9 digit number, formed by adding to an 8-digit number a control digit at the end. This is very much like the preceding example: if $a_1, \dots, a_8$ are the first 8 digits, then define $a_9 \in \{0, \dots, 9\}$ so that

$$a_9 \equiv 9a_1 + 8a_2 + \cdots + 3a_7 + 2a_8 \bmod 11.$$

If the sum on the right is $\equiv 10 \bmod 11$, the word $a_1, \dots, a_8$ does not give rise to a BSN number, and is to be discarded.

***Example 6.7* (INSEE number).** — The French Institue for Statistic and Economic Studies issues the social security numbers for the French citizens. These codes are also used for surveys of the French population. They are 15-digit codes, constructed as follows:

$$s\ yy\ mm\ llooo\ kkk\ cc,$$

where s encodes the sex (most common: 1 male, 2 female, ...); yy is the year of birth, mm is the month of birth, $llooo$ is a 5-digit administrative code encoding the town of origin (usually, $ll \in \{01, \ldots, 95\}$ is the region of birth and ooo is the code of the city inside the region; for people born abroad $ll = 98$ or 99, etc...), kkk reflects the position of the person among the list of all people born in the same month and the same year in the same town (this number is found on the birth certificate). The interesting bit (for us) is the last 2-digit cc, the control key. It is computed by

$$cc = 97 - (syymmllooookkk \bmod 97).$$

That is, take the remainder R modulo 97 of the number N formed by the first 13 digits and put $cc = 97 - R \in \{01, \ldots, 97\}$. Example: 2 69 05 49 588 157 80.

Since 97 is prime, the same remarks as the ISBN code are valid. Given a 15-digit code as above, by checking whether $N - cc$ is divisible by 97 or not, you can detect if a digit has been altered, or if two digits were transposed. Again, this codes only detects at most two errors, and can not correct them.

6.2. Linear codes

From now, on we only concentrate on the so-called linear codes. The alphabet will always be $A = \mathbb{F}_q$ a finite field (or in bijection with $\mathbb{F}_q$)

Definition 6.8. — A linear code of length n is a $\mathbb{F}_q$-sub vector space of $(\mathbb{F}_q)^n$. We denote by $k(C)$ the dimension of C (as a $\mathbb{F}_q$-vector space).

Let us briefly recap the various invariants associated to such a code $C \subset (\mathbb{F}_q)^n$. The length of the words in C is $n(C) = n$, the dimension of the ambiant vector space. The dimension $k(C)$ of C encodes the "useful part" of codewords. In this case, the information ratio of C is $\tau(C) = k(C)/n(C)$. In the linear case, the Hamming distance becomes much nicer and the distance $d(C)$ (the minimal Hamming distance between two distinct elements of C) can be written as

$$d(C) = \min\{d(x, 0),\ x \in C \smallsetminus \{0\}\}.$$

In other words, the distance of C is the minimal number of nonzero coordinates of a vector $x \in C \smallsetminus \{0\}$. A linear code C (over $\mathbb{F}_q$) with length n, dimension k and distance d will be called a $[n, k, d]$-code.

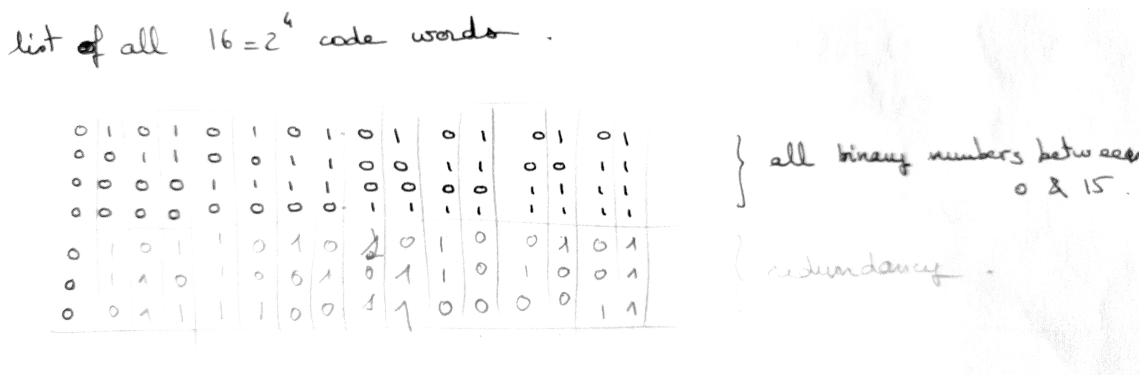
Note that the triple $[n, k, d]$ contains all the information we need about a code C. We remark that $n(C)$ and $k(C)$ are usually easy to compute, while $d(C)$ can be harder to determine (in practice though, lower bounds on $d(C)$ are sufficient).

Most of the previous examples are linear codes. Note that the ISBN code can be seen as a linear code: indeed, the alphabet is in bijection with $\mathbb{F}_{11}$, and the relation between the digits is a $\mathbb{F}_{11}$-linear one, but the first 9 digits of a ISBN code are chosen in $\{0, \ldots, 9\}$ and cannot be "X". The ISBN code is thus a $\mathbb{F}_{11}$-linear code, all of whose codewords are not used.

6.2.1. Hamming code. — Let us give a worked out example of a classical linear code, the Hamming code. We work over $\mathbb{F}_2$. The Hamming code is the subvector space of $(\mathbb{F}_2)^7$ generated by

$$E_0 := \begin{bmatrix}1\\1\\0\\1\\0\\0\\0\end{bmatrix}, \quad E_1 := \begin{bmatrix}0\\1\\1\\0\\1\\0\\0\end{bmatrix}, \quad E_2 := \begin{bmatrix}0\\0\\1\\1\\0\\1\\0\end{bmatrix}, \quad E_3 := \begin{bmatrix}0\\0\\0\\1\\1\\0\\1\end{bmatrix}.$$

It is easily seen that these vectors are independent, and thus C has dimension 4. One can list all $16 = 2^4$ codewords:



list of all $16 = 2^4$ code words.

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

} all binary numbers between 0 & 15.

(redundancy.

Note that the first four coordinates run though the set of all integers $\{0, \ldots, 15\}$, written in binary. And one sees that the minimal number of nonzero coordinates of a nonzero vector in C is 3. So C is a $[7, 4, 3]$-code over $\mathbb{F}_2$.

One nice feature of the Hamming code is that encoding a message and decoding it is very explicit.

Assume that we want to send a binary message $m = (m_0, m_1, m_2, m_3) \in (\mathbb{F}_2)^4$. We transmit the codeword $x = m_0E_0 + m_1E_1 + m_2E_2 + m_3E_3 \in C$, written in coordinates $x = (x_1, \ldots, x_7)$ in the canonical basis of $(\mathbb{F}_2)^7$. To decode the receive message, it is necessary to find explicit equations for C. Using the basis of C, it can be seen that C is given by three equations:

$$C : \begin{cases} x_1 + x_4 + x_6 + x_7 &= 0, \\ x_2 + x_4 + x_5 + x_7 &= 0, \\ x_3 + x_5 + x_6 + x_7 &= 0. \end{cases}$$

In other words, C is the kernel of the linear map $L : (\mathbb{F}_2)^7 \to (F_2)^3$ given by the matrix:

$$L(x_1, \ldots, x_7) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_7 \end{bmatrix}.$$

Now assume that the receiver receives a word $y = (y_1, \ldots, y_7) \in (\mathbb{F}_2)^7$ and that at most one error occured during the transmission (*i.e.* $d(x, y) \leq 1$). We will see how to reconstruct x from y (*i.e.* how to correct the error). Let $e = y - x \in (\mathbb{F}_2)^7$ be the "error vector": by assumption e has at most 1 nonzero coordinate. Notice that $L(e) = L(y - x) = L(y) - L(x) = L(y)$. The procedure for correction goes as follows: upon receiving y, compute $L(y) \in (\mathbb{F}_2)^3$,

- if $L(y) = (0, 0, 0)$, then $y \in C$ and no error has been made, so $y = x$,
- if $L(y) = (1, 0, 0)$, then an error has been made in the first coordinate: change y_1 into $z_1 = y_1 + 1$,
- if $L(y) = (0, 1, 0)$, then an error has been made in the second coordinate: change y_2 into $z_2 = y_2 + 1$,

– more generally, if $L(y) = (\epsilon_1, \epsilon_2, \epsilon_3)$, let $i \in \{0, \dots, 7\}$ be the integer $i = \epsilon_1 + 2\epsilon_2 + 4\epsilon_3$. Then an error has been made in the i-th coordinate: change y_i into $y_i + 1$.

After this step, we have a vector $z = (z_1, \dots, z_7) \in (\mathbb{F}_2)^7$, identical to y except maybe where we "corrected a bit". You can check that this procedure indeed returns a vector $z \in C$ which is closest to y (in terms of Hamming distance). Finally put $m' = (z_1, z_1 + z_2, z_6, z_7)$. If at most one error was made during transmission, one has $m = m'$.

Remark 6.9. — A funny illustration of the Hamming code. The hamming code above suggests that it is possible to reconstruct an element of $(\mathbb{F}_2)^4$ (say, an integer between 0 and 15 in binary notation) from an element of $(\mathbb{F}_2)^7$ (say, seven "YES/NO" informations) if at most one error was made (say, if at most one information is wrong).

In other words, the Hamming code gives the following "magic trick". A person A chooses a secret integer $N \in \{0, \dots, 15\}$; then, person B asks 7 YES/NO questions to A about N; A answers to the questions but A is allowed to lie once; B can then guess N.

Using the procedure above, you can come up with a list of 7 questions that make this trick work. Here is a version: A chooses $N \in \{0, \dots, 15\}$, then B asks the seven questions:

(7) is $N \geq 8$?
(7) is $N \in \{4, 5, 6, 7, 12, 13, 14, 15\}$?
(7) is $N \in \{2, 3, 6, 7, 10, 11, 14, 15\}$?
(7) is N odd?
(7) is $N \in \{1, 2, 4, 7, , 10, 12, 15\}$?
(7) is $N \in \{1, 2, 5, 6, 8, 11, 12, 15\}$?
(7) is $N \in \{1, 3, 4, 6, 8, 10, 13, 15\}$?

Let $(A_1, \dots, A_7) \in (\mathbb{F}_2)^7$ be the list of answers B get ($A_i = 1$ if the answer to question i is YES, and vice versa). Compute

$$x_1 = A_4 + A_5 + A_6 + A_7, \quad x_2 = A_2 + A_3 + A_6 + A_7, \quad x_3 = A_1 + A_3 + A_5 + A_7.$$

If $x_1 = x_2 = x_3 = 0$, then A has not lied. Otherwise, put $L = 4x_1 + 2x_2 + x_3 \in \{0, \dots, 7\}$, and change the L-th answer (A has lied about question L). Let $T = (T_1, \dots, T_7)$ be the list of "true" answers, and $N' = T_4 + 2T_3 + 4T_2 + 8T_1$. Then $N' = N$.

6.3. Hadamard codes

We work over $\mathbb{F}_2$ again. Fix a parameter $r \geq 1$. Write vertically every integer $N \in \{0, 1, \dots, 2^r - 1\}$ in binary expansion, and denote by M_r matrix obtained by juxtaposing these columns. M_R has entries in $\mathbb{F}_2$, it has r rows and 2^r columns (one of which is 0). Let $n = 2^r$, and $H_r \subset (\mathbb{F}_2)^n$ be the subspace generated by the rows of M_r.

Then the dimension of H_r is the rank of M_r: this is easily seen to be exactly r. The distance of H_r is 2^{r-1}, because the first row of M_r is $(0, \dots, 0, 1, \dots, 1)$. In other words, the code H_r is a $[2^r, r, 2^{r-1}]$-code over $\mathbb{F}_2$.

Note that H_r can correct many errors (exercise: the correcting ratio tends to 25% as $r \to \infty$)... but, H_r is not very efficient (exercise: the information ratio tends to 0 when $r \to \infty$).

6.3.1. Bounds. — Before giving further examples and constructions, we prove two bounds which give a rough idea of how good linear codes can be.

***Proposition 6.10* (Singleton bound)**. — *Let C be a $[n, k, d]$-code over $\mathbb{F}_q$. Then*

$$k + d \leq n + 1.$$

A code C such that there is equality in this bound is called a Maximal Distance Separable code (MDS code, for short).

Proof. — Let $\ell = n+1-k$, and $D \subset (\mathbb{F}_q)^n$ be the vector space consisting of all vectors $x = (x_1, \ldots, x_n)$ with $x_{\ell+1} = x_{\ell+2} = \cdots = x_n = 0$. Then D has dimension ℓ and

$$\dim C + \dim D = \ell + k = n + 1 - k + k = n + 1 > n.$$

It follows that $C \cap D$ can not be reduced to $\{0\}$. So there exists a nonzero vector $x \in C \cap D$. Since $x \in D$, x has at most ℓ nonzero coordinates: $d(x, 0) \leq \ell$. Since $x \in C \smallsetminus \{0\}$, we have

$$d = d(C) \leq d(x, 0) \leq \ell = n + 1 - k.$$

This completes the proof. □

Proposition 6.11. — *Let C be a $[n, k, d]$-code, and let $t = \lfloor (d-1)/2 \rfloor$. Then*

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

Proof. — For all $x \in (\mathbb{F}_q)^n$, and $\epsilon \in \{0, \ldots, n\}$, let

$$B(x, \epsilon) := \{y \in (\mathbb{F}_q)^n : \ d(x, y) \leq \epsilon\},$$

be the Hamming ball with center x and radius ϵ. By construction of the distance, one has

$$\#B(x, \epsilon) = \sum_{i=0}^{\epsilon} \binom{n}{i} (q-1)^i.$$

If C is t-correcting (as we assumed), the balls $B(x, t)$ with centers $x \in C$ have to be disjoint (otherwise, ...). Consequently,

$$\#\left(\bigcup_{x \in C} B(x, t) \right) = \sum_{x \in C} \#B(x, t) = \#C \cdot \#B(0, t).$$

On the other hand, the union $\bigcup_{x \in C} B(x, t)$ is a subset of $\mathbb{F}_q^n$, so that

$$\#C \cdot \#B(0, t) = \# \left(\bigcup_{x \in C} B(x, t) \right) \leq \#\mathbb{F}_q^n = q^n.$$

It remains to write that $\#C = q^k$, to use the expression for $\#B(0, t)$, and to reorder terms to get to the promised upper bound. □

6.3.2. Constructions. — One nice feature of linear codes is that we can build up new ones from known codes, using linear algebra. Here is a sample of examples of classical constructions of linear codes:

Constructions from linear algebra : Given two codes with suitable invariants, one can consider their direct sum, their intersection, their tensor product, ... The resulting object is again a linear code, whose invariants can be computed in terms of the invariants of the codes we started with.

Shortened code : Let C be a $[n, k, d]$-code over $\mathbb{F}_q$. For a parameter $m \in \{d, d+1, \ldots, n\}$, consider

$$C^{(m)} := \{(x_1, \ldots, x_m) \in (\mathbb{F}_q)^m \ : \ (x_1, \ldots, x_m, 0, \ldots, 0) \in C\}.$$

Then $C^{(m)}$ is a linear code too: its length is clearly m, and one can show (exercise) that $d(C^{(m)}) \geq d$.

Extended code : Let C be a $[n, k, d]$-code over $\mathbb{F}_q$. One can add to each codeword a "generalized parity-check bit" as follows. Let

$$\overline{C} := \left\{ (x_1, \ldots, x_{n+1}) \in (\mathbb{F}_q)^{n+1} \ : \ (x_1, \ldots, x_n) \in C \text{ and } \sum_{i=1}^{n+1} x_i = 0 \right\}.$$

This new code has length $n+1$, dimension k, and its distance $d(\overline{C})$ satisfies $d(C) \leq d(\overline{C}) \leq d(C)+1$.

Dual code : Let C be a $[n,k,d]$-code over $\mathbb{F}_q$. There is a canonical scalar product on $(\mathbb{F}_q)^n$ given by

$$\forall x = (x_i) \in (\mathbb{F}_q)^n, y = (y_i) \in (\mathbb{F}_q)^n, \qquad \langle x, y, \rangle := \sum_{i=1}^{n} x_i \cdot y_i.$$

Now define the dual of C to be

$$C^* := \{(y_1, \ldots, y_n) \in (\mathbb{F}_q)^n \ : \ \forall x \in C, \ \langle x, y \rangle = 0\}.$$

Then C^* is a linear subspace of $(\mathbb{F}_q)^n$. The length of C^* is n, and its dimension is $n-k$. As an exercise, you can check that the two codes

$$P := \{x = (x_i) \in (\mathbb{F}_q)^n \ : x_1 + x_2 + \cdots + x_n = 0\} \qquad \text{(``parity-check code'')},$$

and

$$R := \{x = (x_i) \in (\mathbb{F}_q)^n \ : x_1 = x_2 = \cdots = x_n\} \qquad \text{(``repetition code'')}$$

are dual to each other.

6.3.3. Cyclic codes. — Let us describe one way to actually construct linear codes.

Definition 6.12. — A cyclic code is an ideal I in the quotient ring $R := \mathbb{F}_q[X]/(X^n - 1)$.

Note that, as a $\mathbb{F}_q$-vector space, R has dimension n: indeed, a $\mathbb{F}_q$-basis for R is $1, X, X^2, \ldots, X^n$. In other words, one can identify R and $(\mathbb{F}_q)^n$ via the map

$$(\mathbb{F}_q)^n \to R, \qquad (a_0, \ldots, a_{n-1}) \mapsto a_0 + a_1 X + \cdots + a_{n-1}X^{n-1}.$$

This map is an isomorphism of $\mathbb{F}_q$-vector spaces (but makes no use of the rign structure on R). Under this isomorphism, an ideal I is R corresponds to a linear subspace of $(\mathbb{F}_q)^n$. In particular, cyclic codes are linear codes.

These codes are called cyclic for the following reason: if $f(X) \in I$ is a codeword, then $X \cdot f(X)$ is also a codeword (since I is an ideal). On the "$(\mathbb{F}_q)^n$-side", this means that, whenever $(a_0, a_1, \ldots, a_{n-1})$ is a codeword, then the cyclic shift $(a_{n-1}, a_0, a_1, \ldots, a_{n-1})$ is also a codeword. The codes one obtains with this construction are thus stable under cyclic shifts of the coordinates.

Example 6.13. — One recovers the parity-check code as a cyclic code as follows. Let $I \subset R$ be the ideal generated by $X - 1 \in \mathbb{F}_q[X]$. A polynomial $f(X) = \sum a_i X^i \in R$ is in I if and only if it vanishes at $X = 1$, which in turn translates as the condition "$\sum_{i=1}^{n} a_i = 0$".

The repetition code can also be seen as a cyclic code. This time, consider the ideal $J \subset R$ generated by $(X^n - 1)/(X - 1) = X^{n-1} + \cdots + X + 1$. The ideal J consists exactly of the scalar multiples of $X^{n-1} + \cdots + X + 1$, and it can be seen that the isomorphism above transports J to the repetition code.

Example 6.14. — Now for a less trivial example, consider the following situation. Let $f \geq 1$ be a parameter, and fix $\alpha \in \mathbb{F}_{2^f}^*$ a generator of the cyclic group $\mathbb{F}_{2^f}^*$. As an element of $\mathbb{F}_{2^f}$, α has a minimal polynomial $\phi(X) \in \mathbb{F}_2[X]$ over $\mathbb{F}_2$. The degree of $\phi(X)$ is $< 2^f$, so let us put $n = 2^f - 1$ and consider the ideal I_α generated by $\phi(X)$ in $R = \mathbb{F}_q[X]/(X^{2^f-1} - 1)$.

The resulting linear code is called a (generalized) binary Hamming code H (with parameter f). It has length $n = 2^f - 1$ (clear) and dimension $k = 2^f - 1 - f$ (exercise). Moreover, since α has order $2^f - 1$ in $\mathbb{F}_{2^f}^*$, there are no nonzero codewords with 2 or less nonzero coordinates. That is, this code has distance $d \geq 3$. A straightforward computation implies that the balls (for the Hamming distance) with radius 1 centered at $h \in H$ are disjoint, that each of them contains 2^f vectors, and that there are 2^{2^f-f} of them. By a further counting argument, one can see that these balls actually cover the whole of R. Thus the distance of H is exactly 3.

In conclusion, the Hamming code is a $[2^f-1, 2^f-f-1, 3]$-code over $\mathbb{F}_2$. You can check that the code in section 6.2.1 corresponds to the choice $f=3$.

6.4. Codes coming from algebraic geometry

6.4.1. Reed-Solomon codes. — Fix a finite field $\mathbb{F}_q$ (the alphabet). Let $\alpha_1, \dots, \alpha_{q-1}$ be an enumeration of the elements of $\mathbb{F}_q^*$. Choose a parameter $r \in \{1, \dots, q-1\}$, and let P_r be the set of polynomials $f \in \mathbb{F}_q[x]$ with $\deg f \leq r-1$. Then P_r is a $\mathbb{F}_q$-vector space of dimension r.

Define an "evaluation map" $\Theta : P_r \to (\mathbb{F}_q)^{q-1}$ by

$$f \mapsto \Theta(f) := (f(\alpha_1), f(\alpha_2), \dots, f(\alpha_{q-1})).$$

The map Θ is $\mathbb{F}_q$-linear and, since $r \leq q-1$, it is injective. We denote by $RS_r \subset (\mathbb{F}_q)^{q-1}$ be the image of Θ. The linear code RS_r is called a Reed-Solomon code. Let us evaluate its parameters.

Clearly RS_r has length $n = q-1$ and dimension $k = \dim RS_r = \dim P_r = r$. Also, the distance d of RS_r is $n-k+1 = q-r$ (left as an exercise, you may use that a polynomial of degree $\leq r-1$ has at most $r-1$ zeroes). In conclusion, RS_r is a $[q-1, r, q-r]$-code over $\mathbb{F}_q$. Notice that RS_r is a MDS code.

This construction provides good codes, but is lacks flexibility. Indeed, once the alphabet is fixed, one can only choose the parameter r and, even then, the length of the words $n = q-1$ is "small" compared to the size of the alphabet $\mathbb{F}_q$. In practice, one would like to work with codes which are long with respect to the alphabet size.

6.4.2. Goppa construction. — The class of Goppa codes was invented some 20 years after the Reed-Solomon codes, and they provide much more flexibilty in the choice of parameters.

Let $\mathbb{F}_q$ be a finite field, and X be a smooth projective curve over $\mathbb{F}_q$. Let $D \in \mathrm{Div}(X)$ be a divisor on X (*i.e.* D is a finite $\mathbb{Z}$-linear combination of $\mathbb{F}_q$-places on X), and let $P_1, \dots, P_n$ be $\mathbb{F}_q$-rational points on X. We assume that "the P_i's do not appear in D", that is to say, $\mathrm{Supp}(D) \cap \{P_1, \dots, P_n\} = \varnothing$ (where $\mathrm{Supp}(D) \subset X(\overline{\mathbb{F}_q})$ is the union of the finitely many points in the finitely many places that have a non zero coefficient in D).

Under this assumption, any rational function $f \in \mathcal{L}(D)$ is regular at P_i, for $i = 1, \dots, n$ (otherwise, f would have a pole at some P_i, which would be accounted for in $\mathrm{div}(f)$, and we wouldn't have $\mathrm{div}(f) \geq -D$). Consequently, we can define a map $\Theta : \mathcal{L}(D) \to (\mathbb{F}_q)^n$ by

$$f \in \mathcal{L}(D) \mapsto \Theta(f) := (f(P_1), \dots, f(P_n)) \in (\mathbb{F}_q)^n.$$

This map is well-defined and clearly $\mathbb{F}_q$-linear. Let $\Gamma := \Gamma(D; P_1, \dots, P_n) \subset (\mathbb{F}_q)^n$ be the image of Θ: it is a linear code, called the Goppa code associated to $X, D, \{P_1, \dots, P_n\}$.

Clearly, the above construction offers more flexibility than that of the Reed-Solomon codes.

6.4.3. Parameters. — Let us now evaluate the parameters of a Goppa code Γ, as above. In what follows, we will always assume that $\deg D < n$.

Let $D \in \mathrm{Div}(X)$ and $\{P_1, \dots, \mathbb{P}_n\} \subset X(\mathbb{F}_q)$ be as above, with $\deg D < n$ (and"the P_i's don't appear in D"). We write Γ, for short, to denote the corresponding code.

The length of Γ is clearly n. Now let $f \in \mathcal{L}(D)$ such that $\Theta(f)$ has $n-d$ zero coordinates (*i.e.* d nonzero coordinates). This means that f has at least $n-d$ zeros among the P_i's: up to renumbering them, we can assume that f vanishes at $P_1, \dots, P_{n-d}$. Another way of encoding this is to write that $\mathrm{div}(f) \geq P_1 + \cdots + P_{n-d}$. Since we already know that $\mathrm{div}(f) \geq -D$, and since the "$\geq$ relation" on divisors is compatible with addition, we deduce that

$$\mathrm{div}(f) \geq P_1 + \cdots + P_{n-d} - D.$$

(Recall that we usually identify $\mathbb{F}_q$-rational points on X and the associated $\mathbb{F}_q$-places of X of degree 1). Taking degrees, we have

$$0 = \deg \mathrm{div}(f) \geq \deg(P_1 + \cdots + P_{n-d}) - \deg D = (n-d) - \deg D.$$

Hence $d \geq n - \deg D > 0$ for any codeword $\Theta(f) \in \Gamma$ with d nonzero coordinates. In conclusion, the distance $d(\Gamma)$ of Γ satisfies $d(\Gamma) \geq n - \deg D > 0$.

It remains to estimate the dimension of Γ. Let us first prove that assuming $n < \deg D$ implies that Θ is injective. In general, the kernel of Θ is formed by rational functions $f \in \mathcal{L}(D)$ such that $f(P_1) = f(P_2) = \cdots = f(P_n) = 0$, which means (using an argument very similar to that used in the previous paragraph),

$$\ker \Theta = \mathcal{L}\left(D - \sum_{i=1}^{n} P_i\right).$$

Let $D' = D - \sum_i P_i \in \operatorname{Div}(X)$: we have $\deg D' = \deg D - n$. As we have seen in the chapter about the Riemann-Roch theorem, if D' has negative degree, $\mathcal{L}(D') = \{0\}$. So, indeed, our assumption that $\deg D < n$ is enough to ensure that Θ is injective. Now, since Θ has no kernel, the dimension of Γ is the same as that of $\mathcal{L}(D)$. Denoting by g the genus of X, he Riemann-Roch theorem yields that

$$\dim \Gamma = \dim \mathcal{L}(D) \geq \deg D + 1 - g,$$

with equality if $\deg D \geq 2g - 1$. Summing up the previous discussion, we arrive at

Proposition 6.15. — *Let X be a smooth projective curve over $\mathbb{F}_q$ of genus g. Let $n \geq 1$ be an integer, $D \in \operatorname{Div}(X)$ with $\deg D < n$ and $P_1, \ldots, P_n \in X(\mathbb{F}_q)$. We assume that $\operatorname{Supp}(D) \cap \{P_1, \ldots, P_n\} = \varnothing$. Denote by Γ the corresponding Goppa code.*

Then Γ is a $[n, k, d]$ code over $\mathbb{F}_q$, with

$$k \geq \deg D + 1 - g \ (\textit{with equality if } \deg D \geq 2g - 1), \quad d \geq n - \deg D.$$

6.4.4. Finding good codes. — Combining the Singleton bound ($k + d \leq n + 1$), with the lower bounds above, we arrive at

$$1 - \frac{1}{n} + \frac{g}{n} \leq \frac{k}{n} + \frac{d}{n} \leq 1 + \frac{1}{n}.$$

To get a good code out of the Goppa construction, we would like to have k/n and/or d/n as big as possible. Since $n \leq \#X(\mathbb{F}_q)$, we need to find curves $X/\mathbb{F}_q$ which have many rational points for a given genus g. The Goppa codes were invented around 1981, and they motivated the search for better bounds on the number of rational points on curves (in terms of their genus). Until the 1980's, little effort had been made to see whether Weil's bound was close to optimality.

6.5. Examples