Introduction to Coding Theory **Técnico–ULisboa, 2026/2027**

Part 1: Fundamental concepts and challenges in coding theory

Lecturer: João Ribeiro

These notes are heavily based on parts of Chapters 1 and 2 of Essential Coding Theory, and on an (yet unpublished) informal article I have written on error-correcting codes and sphere packing.

Additional recommended reading: Essential Coding Theory, Sections 1.1 to 1.6.

1 The noisy communication problem

Our course begins with a love story that is about to turn sour. Alice wishes to send a message to Bob, but they are only connected by a *noisy* communication channel. Because the channel is noisy, a message m sent by Alice may be received as a corrupted message $\widetilde{m} \neq m$ by Bob. This is bad, because, in the current setup, Bob is even unable to decide whether Alice meant to actually send $\widetilde{m}$ (and so no errors were introduced by the channel), or if she meant to send some other message (and so the channel did introduce errors), *let alone discover the exact locations of these errors*. While many errors are harmless, even one tiny error can be enough to ruin someone's day, as illustrated in Figure 1. Assuming that Alice wants to be able to send *any* message of a given length, this problem is impossible to solve in the current form.

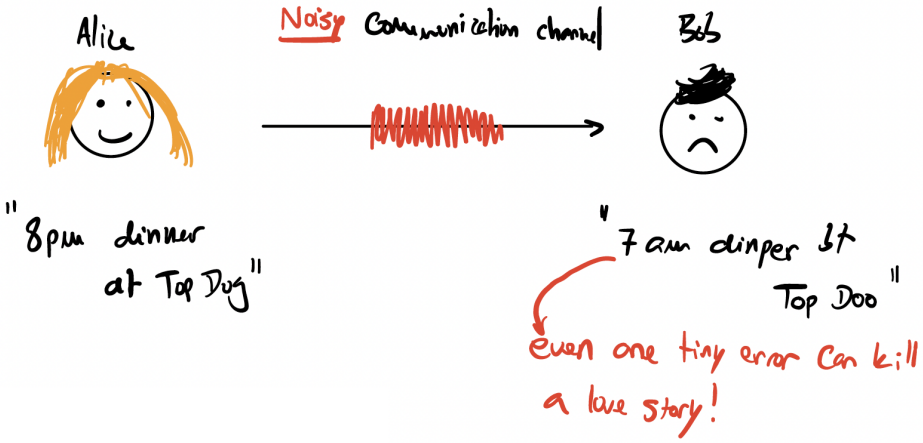


Figure 1: Basic communication setup over a noisy channel.

How can we help Alice and Bob? Clearly, transmitting messages as is does not work, so we need to update the communication setup by allowing Alice to incorporate *redundant information* into her transmission so that errors can be corrected. More precisely, suppose that Alice wishes to transmit an arbitrary message $m \in \Sigma^k$, where Σ is some *alphabet* (think of the English or Portuguese

alphabets, or ASCII, or binary...). Instead of sending m as is through the channel, Alice first *encodes* m into a *codeword* $c \in \Sigma^n$, with $n > k$. Of course, we require that different messages are encoded to different codewords, and so we may informally see the codeword c as "the message plus $n - k$ redundant symbols". The codeword c is sent through the channel, yielding the corrupted output $\widetilde{c}$. Ideally, the encoding procedure should have been designed so that Bob can *decode* the corrupted transmission $\widetilde{c}$ and recover the correct message m. Here, we assume that the encoding and decoding methods have been previously agreed on by Alice and Bob. This setup is illustrated in Figure 2.

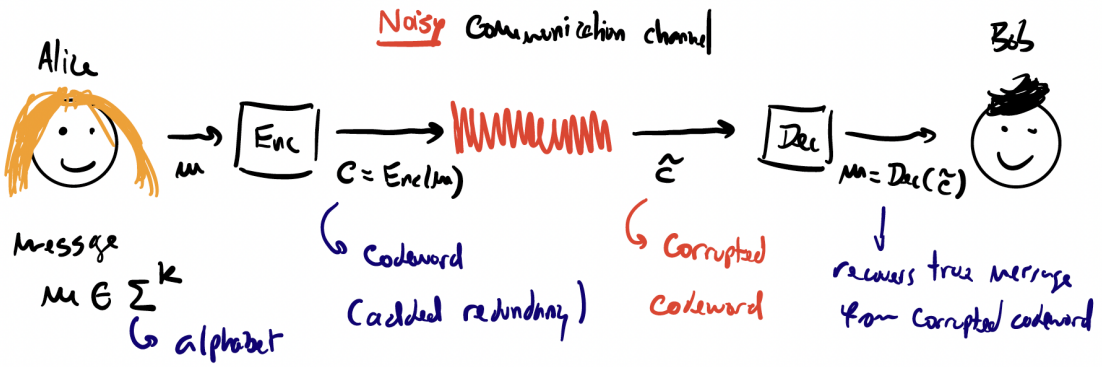


Figure 2: Communication setup over a noisy channel with error-correction method.

This problem is more than just a pure thought exercise. Noise is everywhere. Phone calls, video streaming, Internet packet transmissions, and deep space communications are constantly subjected to various noise sources. Data storage systems are affected by physical phenomena inherent to their writing and reading processes and to the degradation of the underlying physical media. Computational processes are affected by various types of errors.

We call the encoding and decoding procedures that allow Alice and Bob to communicate correctly even over a noisy channel *error-correction methods*, and the set of codewords resulting from encoding these messages is called an *error-correcting code*. Because we want error-correcting codes to be used in practice, we are led to some natural criteria for what constitutes a good solution to the problem above:

1. We would like to minimize the number of symbols transmitted through the noisy channel;
2. We would like for the encoding and decoding procedures to be pretty fast.

This is motivated by the fact that communicating and computing take time and cost money.

For now, we will focus mostly on the first fundamental goal of minimizing the number of transmitted symbols. Clearly, thinking about this requires fixing a noise model for our channel.

1.1 Modelling noise

There are many ways we could go about specifying the behavior of our noisy channel. One way would be to have a very specific real-world use case in mind and carry out a detailed statistical study of its underlying noise sources. This would lead to a very precise but complex model, meaning that its mathematical analysis would be less "clean". Plus, because of fine-grained specialization, its applicability would most likely be limited to that use case.

Instead, we will flip this and study some simple error models, which allow for clean mathematical analysis without being constrained to any specific application. I say this, of course, armed with decades of hindsight. The models we are about to see were first introduced in the late 1940s by Richard Hamming and Claude Shannon, and they have turned out to be extremely impactful in practice, in the sense that they have consistently guided us to theoretical results that carry out quite well to the real world.

I emphasize, though, that there is no unique "right answer" when it comes to developing theoretical noise models. Sometimes simplicity is useful, but sometimes introducing more details in the noise model leads to more efficient solutions, and that is important too. There is a whole amazing world between the two extremes laid out above.

On to our simple error models. To simplify things even further, we will focus on the binary alphabet $\Sigma = \{0, 1\}$, although these models generalize quite easily to larger alphabets.

- **Hamming's adversarial error model:** For a given number of errors t, the channel is allowed to replace any up to t coordinates of c by other symbols. When $\Sigma = \{0, 1\}$, this means that the channel is allowed to introduce up to t *bit flips* (replacing a 0 by a 1 or vice-versa). This model was first introduced by Hamming in 1950 [Ham50], motivated by detecting and correcting errors in computations carried by old and clunky computers.
- **Shannon's probabilistic error model:** For a given error probability $p \in [0, 1]$, the channel independently flips each transmitted symbol with probability p. This model was first introduced by Shannon in his 1948 landmark work that kickstarted the field of information theory [Sha48a, Sha48b].[1]

We can now phrase the first fundamental problem more precisely (but still not 100% formal) for each of these models. In Hamming's adversarial model, our first main goal is to understand how large n (the codeword length) needs to be as a function of the message length k and the number of errors t in order to allow us to correct any pattern of up to t errors. In Shannon's probabilistic model we need to allow for a small probability of decoding error (as there is some probability that the channel introduces many errors). Then, our first main goal in Shannon's model is is to understand how large n needs to be as a function of the message length k, the error probability p, and the target decoding error probability ε, to allow for error-correction with probability at least $1 - \varepsilon$ against error probability p.

Coding theory, the subject of this course, was born out of the study of these questions. In the

[1] Shannon's model is more general than this, but we will leave that discussion for later. The channel mentioned here is an important special case, called the *binary symmetric channel*.

lecturer's very biased opinion, it features a beautiful combination of algebra, combinatorics, graph theory, and probability theory, straddling the fuzzy borders of mathematics, computer science, and electrical engineering. Beyond the beautiful mathematics, it has an impressive track record of impactful applications that form the backbone of many technologies we take for granted today.

2 An easier challenge: detecting errors

In the first part of our course we will focus on Hamming's adversarial model. We will begin by studying the simplest possible setting: *detecting* 1 *adversarial error*. More precisely, our goal here is to devise a way to encode messages $m \in \{0,1\}^k$ into codewords $c \in \{0,1\}^n$ so that Bob can correctly decide, based on the channel output $\widetilde{c} \in \{0,1\}^n$ he received through the adversarial 1-error channel, whether the channel introduce 1 error or no error. Note that this is never harder than *correcting* 1 adversarial error (make sure you see why). With our first fundamental problem in mind, we want to devise an error-detecting method such that the redundancy $n-k$ is as small as possible.

2.1 Detecting one error with minimal redundancy

One natural first approach is to repeat each message symbol. More precisely, this corresponds to the encoding

$$m = (m_1, m_2, \ldots, m_k) \mapsto c = (m_1, m_1, m_2, m_2, \ldots, m_k, m_k).$$

For example, if $m = 0101$, then the corresponding codeword would be 00110011. From a different but equivalent perspective, the *repetition code* $\mathcal{C}_{\text{rep}}$ associated with this encoding procedure is the set of all codewords, i.e.,

$$\mathcal{C}_{\text{rep}} = \{(m_1, m_1, m_2, m_2, \ldots, m_k, m_k) : (m_1, \ldots, m_k) \in \{0,1\}^k\}.$$

The repetition code $\mathcal{C}_{\text{rep}}$ detects 1 error, because if an error is introduced in a codeword $c \in \mathcal{C}_{\text{rep}}$, yielding a corrupted codeword $\widetilde{c}$, then there will be an odd index i such that $\widetilde{c}_i \neq \widetilde{c}_{i+1}$. This never happens if no errors are introduced, in which case $\widetilde{c} = c$. If the message length is k, then codewords of the repetition code have length $n = 2k$, and so the repetition code has redundancy

$$n - k = 2k - k = k.$$

Can we do better?

Yes! Consider the *parity encoding*

$$m = (m_1, m_2, \ldots, m_k) \mapsto c = \left(m_1, m_2, \ldots, m_k, \sum_{i=1}^{k} m_i \pmod 2\right),$$

which induces the *parity code*

$$\mathcal{C}_{\text{par}} = \left\{(c_1, \ldots, c_k, c_{k+1}) \in \{0,1\}^{k+1} : \sum_{i=1}^{k} c_i = c_{k+1} \pmod 2\right\}.$$

Put differently, $\mathcal{C}_{\text{par}}$ consists of all length-$(k+1)$ binary vectors with an even number of 1s. We claim that $\mathcal{C}_{\text{par}}$ detects 1 error. To see this, note that if $\widetilde{c}$ is obtained by flipping some coordinate of $c \in \mathcal{C}_{\text{par}}$, then the parity of the number of 1s in $\widetilde{c}$ differs from that of c, and so $\widetilde{c} \notin \mathcal{C}_{\text{par}}$. Moreover, the redundancy of $\mathcal{C}_{\text{par}}$ is $k+1-k=1$. Remarkably, its redundancy does not depend on k and is clearly the best possible (convince yourself of this).

2.2 A geometric perspective

We will now develop a geometric perspective on the previous discussion on error-detection. This geometric perspective is fundamental in coding theory, capturing various coding tasks in a clean and unified manner. We start by formalizing some basic concepts.

Definition 1 (Code) *Fix an alphabet Σ (a non-empty finite set[2]). Then, we call any subset $\mathcal{C} \subseteq \Sigma^n$ a* code *over Σ, and we call n the* block length *of $\mathcal{C}$.*

We have already informally used this "subset perspective" on codes above, by working with the set of encodings of all messages. We now define error-detection more formally.

Definition 2 (Error-detecting code) *We say that a code $\mathcal{C} \subseteq \Sigma^n$ is* t-error-detecting *if for any $\widetilde{c} \in \Sigma^n$ obtained by applying t' adversarial errors to some $c \in \mathcal{C}$, for some $t' \in \{1, 2, \ldots, t\}$, it holds that $\widetilde{c} \notin \mathcal{C}$.*

We will now see that constructing error-detecting codes is really a geometric problem under the appropriate lens. Consider the following distance between vectors in Σ^n.

Definition 3 (Hamming distance and Hamming ball) *The* Hamming distance *between two vectors $x, y \in \Sigma^n$, denoted by $d(x,y)$, is defined as*

$$d(x,y) = |\{i : x_i \neq y_i\}|.$$

In other words, $d(x,y)$ counts the number of coordinates on which x and y disagree. We can then define the radius-t Hamming ball in Σ^n centered at x, *denoted by $B_t(x)$, as*[3]

$$B_t(x) = \{y \in \Sigma^n : d(x,y) \leq t\}.$$

We shall see in a moment that error-detection boils down to constructing codes whose elements are sufficiently far apart in Hamming distance. Informally, the reason for this is that $d(x,y)$ corresponds exactly to the number of errors that must be applied to x to obtain y, and $B_t(x)$ corresponds to the set of vectors that can be obtained by applying up to t errors to x. Motivated by this, we introduce the notion of the minimum distance of a code.

[2] We will usually denote $|\Sigma|$ by q. Without loss of generality we may assume that $\Sigma = \{0, 1, \ldots, q-1\}$, although often it will be useful to imbue more structure into Σ (e.g., interpret it as a finite field of size q).

[3] We will keep the dependence on n and Σ in the notation for the Hamming ball mostly silent for the sake of simplicity, as they are almost always clear from context.

Definition 4 (Minimum distance) *The* minimum (Hamming) distance *of a code $\mathcal{C} \subseteq \Sigma^n$, denoted by $d(\mathcal{C})$, is given by*

$$d(\mathcal{C}) = \min_{c,c' \in \mathcal{C} : c \neq c'} d(c, c').$$

It is also useful to rephrase the definition of minimum distance above in terms of Hamming balls: a code $\mathcal{C}$ has minimum distance larger than d if and only if $B_d(c) \cap \mathcal{C} = \{c\}$ for all $c \in \mathcal{C}$. We now formally connect the minimum distance of a code to its error-detection capabilities.

Theorem 1 *A code $\mathcal{C} \subseteq \Sigma^n$ is t-error-detecting if and only if $d(\mathcal{C}) > t$.*

Proof: By definition, $\mathcal{C}$ is t-error-detecting if and only if any $\widetilde{c}$ obtained by applying $t' \in \{1, \ldots, t\}$ errors to some $c \in \mathcal{C}$ satisfies $\widetilde{c} \notin \mathcal{C}$. This is the same as saying that $B_t(c) \cap \mathcal{C} = \{c\}$, which in turn, by the discussion above, is the same as saying that $d(\mathcal{C}) > t$. ■

Based on this simple theorem, we conclude that constructing error-detecting codes is equivalent to constructing sets of vectors in Σ^n that are all sufficiently far apart from each other (in Hamming distance). Recall that our fundamental goal was to construct error-detecting methods that have redundancy as small as possible. This is the same as constructing error-detecting codes that are as large as possible. This motivates the following basic concepts.

Definition 5 (Redundancy and rate of a code) *The* redundancy *of a code $\mathcal{C} \subseteq \Sigma^n$ with $|\Sigma| = q$ is given by $n - \log_q |\mathcal{C}|$. The* rate *of $\mathcal{C}$ is given by $\frac{\log_q |\mathcal{C}|}{n}$.*

Redundancy and rate measure the same thing (the size of a code relative to the space it lives in) in two different ways. We define them both because sometimes it is useful to think in terms of redundancy (when the number of errors is small compared to the block length), and sometimes it is useful to think in terms of rate (when the number of errors is relatively large).

Let's recollect our goal regarding error-detection, rephrased in terms of our newly-defined concepts:

> *Our first fundamental goal is to, given a block length n, alphabet Σ, and distance parameter d, construct a set $\mathcal{C} \subseteq \Sigma^n$ with minimum distance $d(\mathcal{C}) \geq d$ that is as large as possible.*

We already saw above that the parity code is optimal for detecting one error over a binary alphabet. Let's rephrase our result in our new language.

Theorem 2 *Fix a block length n. Then, the parity code*

$$\mathcal{C}_{\text{par}} = \left\{ (c_1, \ldots, c_n) \in \{0,1\}^n : \sum_{i=1}^{n} c_i = 0 \pmod 2 \right\}$$

has block length n, size 2^{n-1}, and minimum distance 2. In particular, it has redundancy 1, rate $1 - \frac{1}{n}$, and detects 1 error.

Proof: The claims about the block length and size of $\mathcal{C}_{\text{par}}$ are clear. To see that $d(\mathcal{C}_{\text{par}}) = 2$, first note that any two distinct $c, c' \in \mathcal{C}_{\text{par}}$ must differ in at least two coordinates, since both c and c' have an even number of 1s. This implies that $d(\mathcal{C}_{\text{par}}) \geq 2$. To see that $d(\mathcal{C}_{\text{par}}) \leq 2$, note that $(0, 0, \ldots, 0)$ and $(1, 1, 0, \ldots, 0)$ are both codewords of $\mathcal{C}_{\text{par}}$, and they are at Hamming distance 2. ■

3 Correcting errors

We now move on to the more challenging problem of *correcting* errors.

Definition 6 (Error-correcting code) *We say that a code $\mathcal{C} \subseteq \Sigma^n$ is* t-error-correcting *if for any two distinct codewords $c_1, c_2 \in \mathcal{C}$ and any $\widetilde{c_1}, \widetilde{c_2}$ obtained by applying any up to t errors to c_1 and c_2, respectively, it holds that $\widetilde{c_1} \neq \widetilde{c_2}$.*

It turns out that just like error-detection is controlled solely by the minimum distance of the code, so is error-correction!

Theorem 3 *A code $\mathcal{C} \subseteq \Sigma^n$ is t-error-correcting if and only if $t < d(\mathcal{C})/2$.*

Proof: This is a simple consequence of the fact that $\mathcal{C}$ is t-error-correcting if and only if $B_t(c) \cap B_t(c') = \varnothing$ for all distinct $c, c' \in \mathcal{C}$. ■

Consequently, our first fundamental goal remains the same: construct large sets in Σ^n with large minimum distance.

3.1 Digression 1: Error-correction and sphere packing

As noted in Theorem 3, correcting t errors is equivalent to requiring that $B_t(c) \cap B_t(c') = \varnothing$ for all distinct $c, c' \in \mathcal{C}$. If we see the codewords of $\mathcal{C}$ as the centers of their radius-t Hamming balls, then constructing an as-large-as-possible t-error-correcting code is the same as packing as many non-overlapping radius-t Hamming balls as possible into Σ^n.

This can be seen as a discrete variant of the classical *sphere packing problem*, whose study goes back at least to Harriot and Kepler in 1611. Informally, the sphere packing problem in n dimensions asks what is the largest fraction of $\mathbb{R}^n$ that can be covered by non-overlapping spheres with the same radius (with respect to the Euclidean distance). Despite more than 400 years of study, it remains a source of fascinating open problems. Kepler conjectured that in 3 dimensions the face-centered cubic packing (Figure 3) is optimal. An incomplete solution to the 2-dimensional case was presented by Thue in 1890, and a complete proof was only given much later by Fejes Tóth in 1940. Kepler's conjecture was only proved in 1998 by Hales. More recently, in an amazing breakthrough in 2016, the sphere packing problem in 8 dimensions was settled by Viazovska, and soon after in 24 dimensions in collaboration with Cohn, Kumar, Miller, and Radchenko. Viazovska was awarded the Fields Medal in 2022 for her contributions.

Figure 3: The best way of packing spheres in 3 dimensions (face-centered cubic packing). Reproduced from Wikipedia under a Creative Commons Attribution 2.0 Generic license. Author: Funkdooby, https://commons.wikimedia.org/wiki/File:Rye_Castle,_Rye,_East_Sussex,_England-6April2011_(1).jpg.

The connection between the sphere packing problem and error-correcting codes runs deeper than this. For example, techniques that were originally developed to understand the limits of error-correcting codes by Delsarte and McEliece–Rodemich–Rumsey–Welch in the 1970s served as an inspiration for the Cohn–Elkies linear programming bound on sphere packings, which formed the basis for Viazovska's more recent breakthrough. Also, error-correcting codes themselves play an important role in the construction of good sphere packings.

If you would like to read more about sphere packing, see the excellent Conway-Sloane book [CS99], Hales's 1999 survey, and Cohn's more recent survey about recent breakthroughs [Coh17].

3.2 Digression 2: Other types of errors

The equivalence between error-correction capabilities in Hamming's adversarial error model and the minimum Hamming distance of a code is not that surprising, because we tailored the definition of Hamming distance already with Hamming's model in mind. In a sense, most of the legwork so far has been accomplished by introducing the "right" definitions.

There are other core types of errors that don't fit into Hamming's model, such as *deletions* and *insertions* (e.g., deleting the second coordinate of 1010 yields 110, and inserting a 1 after the second coordinate of 1010 yields 10110). We still would like to study these errors from a geometric perspective, which boils down to changing the distance we operate under from the Hamming distance to the *edit distance*, which in turn is connected to combinatorial concepts such as longest common subsequences of a set of strings. In sum, we can adapt to the types of errors we are interested in by changing the underlying distance.

Nevertheless, due to the limited time we have, our course will mostly focus on the Hamming distance.

3.3 Correcting one bit flip

Let's get back to our main storyline, and start with the simplest goal of correcting 1 adversarial error (in Hamming's model) over the binary alphabet $\Sigma = \{0, 1\}$.

3.3.1 A simple approach

Previously, we first used a repetition code to detect 1 error (and then showed how to do better than that). It turns out that repetition codes can also be used to correct errors. Indeed, consider the "3-repetition code" $\mathcal{C}_{\text{3rep}}$ obtained by repeating each symbol of the message 3 times, i.e.,

$$\mathcal{C}_{\text{3rep}} = \{(m_1, m_1, m_1, m_2, m_2, m_2, \ldots, m_k, m_k, m_k) : (m_1, \ldots, m_k) \in \{0,1\}^k\}.$$

The following theorem is easy to prove, and we leave it as an exercise.

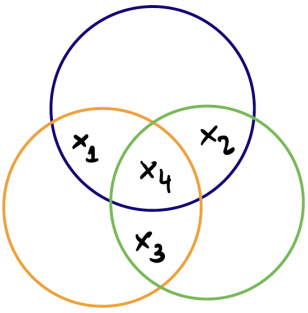
Theorem 4 *The code $\mathcal{C}_{\text{3rep}}$ has block length $n = 3k$, size 2^k, and minimum distance 3. In particular, it has redundancy $2k$, rate $1/3$, corrects 1 error, and detects 2 errors.*

3.3.2 The length-7 Hamming code

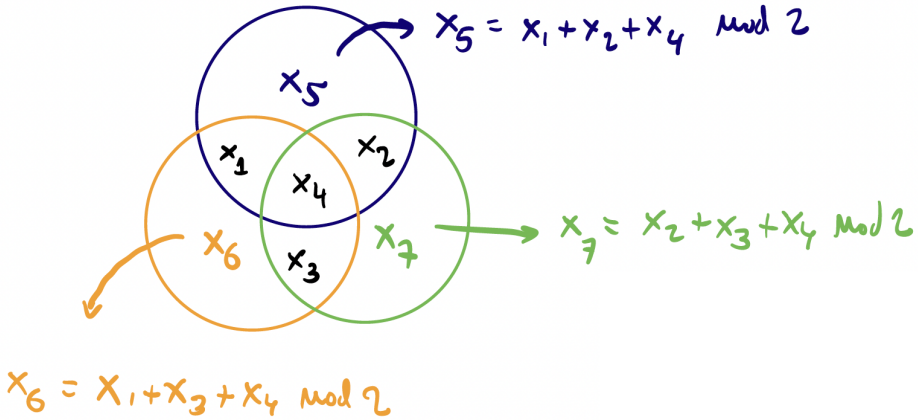
We will now discuss a code that does better than the 3-repetition code above. At first it will seem a bit magical. Later on we will develop an alternative perspective that demystifies it.

We make things even more concrete, and for now assume that we want to encode binary messages of length 4 (i.e., our code will have size $2^4 = 16$). The 3-repetition code above encodes length-4 messages into length-12 codewords, adding 8 bits of redundancy and achieving rate $1/3$. We will now see an improved construction of a 1-error-correcting code that encodes length-4 messages into length-7 codewords, adding only 3 bits of redundancy and achieving rate $4/7$.

Say that we want to encode the message $x = (x_1, x_2, x_3, x_4) \in \{0,1\}^4$. We will add three carefully chosen "parity bits" x_5, x_6, x_7 as follows. First, draw some mysterious circles and place the message bits in the intersections.



Then, to each circle we add a parity bit corresponding to the message bits contained in that circle.



In the end, the encoding of $x = (x_1, x_2, x_3, x_4)$ is $c = (x_1, \ldots, x_7)$ with

$$\begin{aligned} x_5 &= x_1 + x_2 + x_4 \pmod 2, \\ x_6 &= x_1 + x_3 + x_4 \pmod 2, \\ x_7 &= x_2 + x_3 + x_4 \pmod 2. \end{aligned}$$

This encoding induces the *Hamming code* $\mathcal{C}_{\text{Ham}} \subseteq \{0,1\}^7$ defined as

$$\mathcal{C}_{\text{Ham}} = \{(x_1, \ldots, x_7) : x_1 + x_2 + x_4 + x_5 = x_1 + x_3 + x_4 + x_6 = x_2 + x_3 + x_4 + x_7 = 0 \pmod 2\}. \quad (1)$$

The following theorem, stating the error-correcting capabilities of the Hamming code, can be proved by case analysis. That is not very satisfying, though. Later on we will see a much more illuminating way of establishing this result (and a more general one, even).

Theorem 5 *The Hamming code $\mathcal{C}_{\text{Ham}}$ has minimum distance 3. In particular, it has redundancy 3, rate 4/7, and corrects 1 error.*

3.3.3 The length-7 Hamming code is optimal!

It is natural to wonder whether we can do better than the Hamming code. In other words, can we find a larger subset of $\{0,1\}^7$ that has minimum distance at least 3? We will see now that the answer is "no"!

How can we prove that the Hamming code is optimal? We will do this by thinking in terms of sphere packings.

Fix an arbitrary 1-error-correcting code $\mathcal{C} \subseteq \{0,1\}^n$. Then, any two distinct $c, c' \in \{0,1\}$ we have $B_1(c) \cap B_1(c') = \varnothing$. Because these balls are disjoint and $\bigcup_{c \in \mathcal{C}} B_1(c) \subseteq \{0,1\}^n$, we conclude that

$$\sum_{c \in \mathcal{C}} |B_1(c)| = \left| \bigcup_{c \in \mathcal{C}} B_1(c) \right| \leq |\{0,1\}^n| = 2^n.$$

Now, the "volume" of a radius-1 Hamming ball in $\{0,1\}^n$ is

$$|B_1(c)| = |B_1(0)| = 1 + \binom{n}{1} = n + 1,$$

and it does not depend on the center of the ball (here, 0 denotes the all-0s vector in $\{0,1\}^n$). Combining these two observations shows that

$$|\mathcal{C}| \leq \frac{2^n}{n+1}.$$

This discussion is summarized in the following result.

Theorem 6 (Hamming bound) *Every 1-error-correcting code $\mathcal{C} \subseteq \{0,1\}^n$ (equivalently, every code of minimum distance at least 3) satisfies*

$$|\mathcal{C}| \leq \frac{2^n}{n+1}.$$

In particular, $\mathcal{C}_{\text{Ham}}$ has maximal size amongst all 1-error-correcting codes of block length 7.

3.4 A linear-algebraic perspective on the Hamming code

The construction of the length-7 Hamming code above was very ad hoc. It is not clear where it came from, nor how we can extend it to other block lengths. We will now discuss another perspective, based on linear algebra, that will lead to a cleaner and more general construction and analysis of error-correcting codes.

The first thing to note is that $\mathcal{C}_{\text{Ham}}$ is defined in Equation (1) as the set of solutions of a system of three linear equations, if we are thinking of operations (addition and multiplication) modulo 2. Given this, it is perhaps not surprising (and not hard to check) that $\mathcal{C}_{\text{Ham}}$ is *linear*, in the sense that if $c, c' \in \mathcal{C}_{\text{Ham}}$, then

$$c + c' = (c_1 + c'_1 \pmod 2, \ldots, c_7 + c'_7 \pmod 2) \in \mathcal{C}_{\text{Ham}}.$$

Now, if you recall your linear algebra course, you may remember that we can represent systems of linear equations as matrices. Writing down the system of equations from Equation (1) in matrix form yields the 3×7 matrix

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

Then, we get that

$$\mathcal{C}_{\text{Ham}} = \{x \in \{0,1\}^7 : Hx = 0\} = \ker(H),$$

where, again, operations are done modulo 2. We call H the *parity-check matrix* of $\mathcal{C}_{\text{Ham}}$. We will see later that every linear code can be defined as the kernel of some matrix, and so has a parity-check matrix.

This linear-algebraic perspective is useful for various reasons. First, it gives us a systematic, usually simpler, way of certifying the minimum distance of a linear code. Second, it helps us see how to correctly generalize the length-7 Hamming code to larger block lengths. Third, linear codes are important because, as we shall discuss in more detail later, they have compact representations and, by default, come with relatively efficient encoding algorithms. Fourth, although efficient decoding is not a given (and we will work hard for it later on), their algebraic structure does often allow for the design of efficient decoding algorithms.

Minimum distance of binary linear codes. The minimum distance of a binary linear code admits a simpler characterization in terms of the columns of its parity-check matrix H. To discuss this we need to introduce some definitions.

Definition 7 (Hamming weight and minimum weight of a code) *The* Hamming weight *of a vector $x \in \{0,1\}^n$, denoted by* $\mathrm{wgt}(x)$*, is defined as*

$$\mathrm{wgt}(x) = |\{i : x_i \neq 0\}| = d(x, 0).$$

The minimum weight *of a code $\mathcal{C} \subseteq \{0,1\}^n$, denoted by* $\mathrm{wgt}(\mathcal{C})$*, is defined as*

$$\min_{c \in \mathcal{C} : c \neq 0} \mathrm{wgt}(c).$$

For a linear code, the minimum distance and minimum weight coincide.

Theorem 7 *For every binary linear code $\mathcal{C} \subseteq \{0,1\}^n$ it holds that $d(\mathcal{C}) = \mathrm{wgt}(\mathcal{C})$.*

Proof: The fact that $d(\mathcal{C}) \leq \mathrm{wgt}(\mathcal{C})$ follows easily from the fact that we always have $0 \in \mathcal{C}$ by linearity of $\mathcal{C}$. To see that $d(\mathcal{C}) \geq \mathrm{wgt}(\mathcal{C})$, note that for any distinct $c, c' \in \mathcal{C}$ we have[4]

$$d(c, c') = d(c - c', 0) = \mathrm{wgt}(c - c'),$$

and observe that $c - c' \in \mathcal{C}$, again by linearity. ■

We now give a linear-algebraic characterization of the minimum distance of a binary linear code. First, we need to define the concept of linear independence, which is analogous to what you already saw in your linear algebra course (over $\mathbb{R}^n$).

Definition 8 *We say that a collection of r vectors $v_1, \ldots, v_r \in \{0,1\}^n$ is* linearly independent *if*

$$\sum_{i=1}^{r} \alpha_i v_i \neq 0 \pmod 2$$

for all nonzero vectors $\alpha = (\alpha_1, \ldots, \alpha_r) \in \{0,1\}^r$. Otherwise, we say that the collection is linearly dependent*.*

[4] It is true that $a + b = a - b$ modulo 2, so we could have written $c + c'$ instead of $c - c'$. But since we will later generalize this beyond the binary alphabet, it is useful to emphasize now that this is a difference and not a sum.

Theorem 8 *Let $\mathcal{C} \subseteq \{0,1\}^n$ be a linear code with parity-check matrix H (i.e., $\mathcal{C} = \ker(H)$). Then, $d(\mathcal{C}) = d$ if and only if*

- *Every subset of $d-1$ columns of H is linearly independent;*
- *There exist d columns of H that are linearly dependent.*

Proof: We prove the "only if" direction of this equivalence and leave the "if" direction as an exercise.

By Theorem 7, we can focus on the minimum weight $\mathrm{wgt}(\mathcal{C})$. Suppose that $\mathrm{wgt}(\mathcal{C}) = d$. To see the first condition, fix any subset of $d-1$ columns of H with indices $i_1 < i_2 < \cdots < i_{d-1}$. We may write any nonzero linear combination of these columns as Hx for a vector $x \in \{0,1\}^n$ whose nonzero coordinates form a non-empty subset of $\{i_1, \ldots, i_{d-1}\}$. Since $\mathrm{wgt}(x) \leq d-1 < d$, we conclude that $Hx \neq 0$, as otherwise we would have $x \in \mathcal{C}$. Because the linear combination was arbitrary, we conclude that the r columns are linearly independent. To see the second condition, note that there exists some $c \in \mathcal{C}$ with $\mathrm{wgt}(c) = d$. Because $Hc = 0$, we get that the d columns corresponding to the d nonzero coordinates of c are linearly dependent. ■

Let's use Theorem 8 to prove the claim about the minimum distance of $\mathcal{C}_{\mathrm{Ham}}$.

Theorem 9 *$\mathcal{C}_{\mathrm{Ham}}$ has minimum distance 3.*

Proof: By Theorem 8 it suffices to check that every pair of columns of the parity-check matrix H described in Equation (2) are linearly independent and that there exist three linearly dependent columns. The former holds because all columns of H are distinct and non-zero. The latter follows by noting that $(1,0,0) + (0,1,0) + (1,1,0) = 0$. ■

Generalizing the length-7 Hamming code. We now discuss how to extend the length-7 Hamming code to larger block lengths n.

In the proof of Theorem 9, we saw that to get a code $\mathcal{C}$ with minimum distance at least 3 it suffices to construct an $h \times n$ parity-check matrix H whose columns are non-zero and distinct, and define $\mathcal{C} = \ker(H)$. The more columns we get to include in H (i.e., the larger we make n as a function of h) while satisfying this constraint, the better, because the rate of $\mathcal{C}$ will be at least $1 - h/n$ (try to see why).

What's the largest collection of pairwise distinct nonzero length-r binary vectors? That would just be the set of all nonzero length-r binary vectors, of which there are $2^r - 1$. If we focus on $r = 3$ and consider the matrix whose columns are all 7 nonzero length-3 binary vectors, we get the parity-check matrix of $\mathcal{C}_{\mathrm{Ham}}$ defined in Equation (1) (possibly up to a permutation of the columns, which does not matter)!

Given the above, we can define Hamming codes more generally as follows. For a block length $n = 2^r - 1$ for some integer $r \geq 1$, consider the $r \times n$ matrix H_r whose columns are all nonzero vectors in $\{0,1\}^r$. Then, the Hamming code $\mathcal{C}_{\mathrm{Ham},r}$ is defined as

$$\mathcal{C}_{\mathrm{Ham},r} = \ker(H_r).$$

An easy extension of the proof of Theorem 9 yields the following more general statement about the properties of $\mathcal{C}_{\mathrm{Ham},r}$.

Theorem 10 *Fix a block length $n = 2^r - 1$ for some integer $r \geq 1$ (and so $r = \log(n+1)$).*[5] *Then, $\mathcal{C}_{\mathrm{Ham},r}$ has block length n, size $2^{n-r} = \frac{2^n}{n+1}$, and minimum distance 3. In particular, $\mathcal{C}$ has redundancy $r = \log(n+1)$, rate $1 - \frac{\log(n+1)}{n}$, and corrects 1 error.*

Remarkably, by the Hamming bound (Theorem 6) we conclude that, for each $r \geq 1$, the $\mathcal{C}_{\mathrm{Ham},r}$ is the largest 1-error-correcting code of block length $n = 2^r - 1$.

A simple decoder for Hamming codes. The parity-check matrix perspective on Hamming codes also suggests a simple way to decode corrupted codewords.

Suppose that $\widetilde{c}$ is obtained by introducing 1 error in some codeword $c \in \mathcal{C}_{\mathrm{Ham},r}$. We may write

$$\widetilde{c} = c + e,$$

where e is a vector of Hamming weight $\mathrm{wgt}(e) = 1$. Let i denote the unique nonzero coordinate of e. Then,

$$H_r\widetilde{c} = H_r(c+e) = H_rc + H_re = H_re = (H_r)_{\cdot i},$$

where $(H_r)_{\cdot i}$ denotes the i-th column of H_r. Consequently, we can correct the error in $\widetilde{c}$ by first computing $H_r\widetilde{c}$ (this is usually called the *syndrome* of $\widetilde{c}$), finding the column of H that equals $H_r\widetilde{c}$, call its index i, and flipping the i-th coordinate of $\widetilde{c}$ to get back c.

4 More on binary linear codes

We will now develop our theory of binary linear codes a bit further. It is useful to keep in mind the analogy to linear subspaces of $\mathbb{R}^n$ that you saw in your linear algebra course. Recall that a binary linear code with block length n is a subset $\mathcal{C} \subseteq \{0,1\}^n$ that is closed under linear combinations, with addition of two vectors being coordinate-wise addition modulo 2 and multiplication of a vector by a scalar being coordinate-wise multiplication modulo 2. More precisely, for vectors $u, v \in \{0,1\}^n$ are vectors and a scalar $\alpha \in \{0,1\}$,

$$u + v = (u_1 + v_1 \pmod 2, \ldots, u_n + v_n \pmod 2),$$
$$\alpha u = (\alpha u_1 \pmod 2, \ldots, \alpha u_n \pmod 2).$$

Therefore, we may also see binary linear codes as linear subspaces of $\{0,1\}^n$ under these operations. The linear algebra you have seen in $\mathbb{R}^n$ carries over nicely to $\{0,1\}^n$. We will work this out partially in this section, and the rest will be left for you as homework.

[5] We denote by log the base-2 logarithm. The natural logarithm will be denoted by ln.

4.1 The generator matrix viewpoint

Recall that linear subspaces of $\mathbb{R}^n$ are spanned by a subset of linearly independent vectors (a *basis*), and the number of vectors in a basis is called the *dimension* of the subspace. An analogous property holds for binary linear codes, with respect to the notion of linear independence described in Definition 8. Before we state the theorem, we introduce one more concept.

Definition 9 (Span of a set of vectors) *The* span *of a set of vectors $\{v_1, \ldots, v_k\} \subseteq \{0,1\}^n$ is the set of all its linear combinations,*

$$\left\{ \sum_{i=1}^{k} \alpha_i v_i : \alpha_1, \ldots, \alpha_k \in \{0,1\} \right\}.$$

We record here some simple but useful observations. First, the span of a set of vectors S has size at most $2^{|S|}$. Furthermore, if S is a set of linearly independent vectors, then the span has size exactly $2^{|S|}$. We can now state and prove the relevant theorem.

Theorem 11 *Let $\mathcal{C} \subseteq \{0,1\}^n$ be a linear code. Then:*

1. *$|\mathcal{C}| = 2^k$ for some integer $k \geq 0$. We call k the* dimension *of $\mathcal{C}$.*
2. *There exist k linearly independent vectors $v_1, \ldots, v_k \in \{0,1\}^n$ such that*

$$\mathcal{C} = \left\{ \sum_{i=1}^{k} \alpha_i v_i : \alpha_1, \alpha_2, \ldots, \alpha_k \in \{0,1\} \right\}.$$

   In other words, $\mathcal{C}$ is the span of $\{v_1, \ldots, v_k\}$.

Proof: To see the first item, suppose that $2^k < |\mathcal{C}| < 2^{k+1}$ for some integer $k \geq 0$. We will show that $\mathcal{C}$ contains at least $k+1$ linearly independent vectors. Since the span (i.e., set of all linear combinations) of these vectors has size 2^{k+1} and is contained in $\mathcal{C}$ by linearity, we conclude that $|\mathcal{C}| \geq 2^{k+1}$, a contradiction.

We show the claim above by constructing a linearly independent set of vectors S vector-by-vector as follows:

1. Initially, S is empty. Choose any vector $v_1 \in \mathcal{C} \setminus \{0\}$ and add it to S. Such a vector v_1 is guaranteed to exist because $|\mathcal{C}| > 1$, and S is a linearly independent set at this stage.
2. For the t-th step, $2 \leq t \leq k+1$, suppose that we have already added $t-1$ vectors to S, and that S is still a linearly independent set. The span of S has size $2^{t-1} \leq 2^k < |\mathcal{C}|$. Therefore, there is at least one $v_t \in \mathcal{C}$ that is not in the span of S. Add one such v_t to S. Convince yourself that S remains a linearly independent set.

Besides proving that $|\mathcal{C}| = 2^k$ for some k, this procedure also shows that $\mathcal{C}$ contains a set of k linearly independent vectors $v_1, \ldots, v_k$. Since the span of these vectors has size 2^k and is contained in $\mathcal{C}$, the desired result follows. ∎

In the context of Theorem 11, consider the $n \times k$ matrix

$$G = \begin{pmatrix} | & | & \cdots & | \\ v_1 & v_2 & \cdots & v_k \\ | & | & \cdots & | \end{pmatrix}.$$

Then, we can write

$$\mathcal{C} = \{Gm : m \in \{0,1\}^k\},$$

and we call G a *generator matrix* of $\mathcal{C}$. For example, the length-7 Hamming code we saw has generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

In particular, the generator matrix perspective is useful because it shows that all linear codes have relatively efficient encoders. To encode a message $m \in \{0,1\}^k$, we can compute its corresponding encoding $c \in \mathcal{C}$ as $c = Gm$. If G has additional structure (e.g., is sparse), then encoding becomes even faster.

4.2 The parity-check matrix viewpoint

Above, we represented the length-7 Hamming code as the kernel of a 3×7 matrix H that we called its *parity-check matrix*. It turns out that all binary linear codes have an associated parity-check matrix.

Theorem 12 *Let $\mathcal{C} \subseteq \{0,1\}^n$ be a binary linear code of dimension k. Then, there is an $(n-k) \times n$ matrix H such that*

$$\mathcal{C} = \{x \in \{0,1\}^n : Hx = 0\} = \ker(H).$$

We call H a parity-check *matrix of $\mathcal{C}$.*

This is entirely analogous to the fact that a dimension-k linear subspace V of $\mathbb{R}^n$ has an orthogonal complement W of dimension $n-k$, with H^T being the basis of W. Furthermore, it is easy to obtain a parity-check matrix H from a generator matrix G and vice-versa (by solving the equation $HG^T = 0$). It is a good exercise to prove Theorem 12 yourself.

Recall that the parity-check matrix viewpoint is useful, in particular, for determining the minimum distance of $\mathcal{C}$ via Theorem 8.

References

[Coh17] Henry Cohn. A conceptual breakthrough in sphere packing. *Notices of the American Mathematical Society*, 64(02):102–115, February 2017.

[CS99] John Conway and Neil Sloane. *Sphere Packings, Lattices and Groups*. Springer New York, NY, 3rd edition, 1999.

[Ham50] Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[Sha48a] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[Sha48b] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, 1948.