**Introduction to Coding Theory** **IST-UL, 2025/2026**

# Week 2: More on linear codes and bounds

*Lecturer: João Ribeiro*

**Recommended reading:** Essential Coding Theory, Sections 1.7, 1.8, 2.6, Chapter 3, Sections 4.1 and 4.2.

**Notation:** From now onwards, we will sometimes denote a code with block length $n$, size $M$, and minimum distance $d$ over an alphabet of size $q$ as an "$(n, M, d)_q$ code" for short. We will also sometimes denote a *binary linear code* with block length $n$, dimension $k$, and minimum distance $d$ as an "$[n, k, d]_2$ code". Note that every $[n, k, d]_2$ code is an $(n, 2^k, d)_2$ code, but the reverse is not true.

## Recap

In the previous lectures we introduced codes and saw some initial examples. In particular, we studied the Hamming code and saw how the "binary linear code" viewpoint is useful for generalizing code constructions, for proving properties of codes, and inspiring nice encoding and decoding algorithms. We also gave a simple "sphere packing bound" that shows the Hamming code is optimal among codes of distance 3 (i.e., codes which correct 1 error).

## 1 Dual codes

We already saw that we can take a dual perspective on an $[n, k, d]_2$ linear code $\mathcal{C}$ and see it as the kernel of an $(n-k) \times n$ parity-check matrix $H$ (whose rows are linearly independent). As in the real case, the rows of $H$ form a basis of the orthogonal subspace of $\mathcal{C}$. Since this orthogonal subspace is also a linear code, it makes sense to explore its properties.

The *dual code* $\mathcal{C}^\perp$ of $\mathcal{C}$ is the binary linear code whose generator matrix is $H^T$, where $H$ is the parity-check matrix of $\mathcal{C}$. The following properties are easy to prove.

**Theorem 1** *If $\mathcal{C}$ is an $[n, k, d]_2$ code with generator matrix $G$ and parity-check matrix $H$, then:*

- *$\mathcal{C}^\perp$ is a binary linear code with block length $n$ and dimension $n - k$;*
- *$\mathcal{C}^\perp$ has generator matrix $H^T$ and its parity-check matrix is $G^T$;*
- *$(\mathcal{C}^\perp)^\perp = \mathcal{C}$.*

The relationship between the minimum distances of $\mathcal{C}$ and $\mathcal{C}^\perp$ is not as clean. We will not discuss this now, but it is interesting to mention that the minimum distance of $\mathcal{C}^\perp$ is related to many important properties of $\mathcal{C}$, and there is nice mathematics behind it.

### 1.1 The simplex and Hadamard codes

Recall that the $r \times (2^r - 1)$ parity-check matrix $H_r$ of the Hamming code of block length $n = 2^r - 1$ was obtained by defining the columns to be all nonzero vectors of $\{0,1\}^r$. This is a curious matrix. What is the dual of the Hamming code?

By what we saw above, the dual of the Hamming code has generator matrix $H_r^T$, and so can be written as

$$\mathcal{C}_{\text{Simp}} = \{(\langle x, y \rangle)_{y \in \{0,1\}^r \setminus \{0\}} : x \in \{0,1\}^r\}.$$

This is sometimes called the *simplex code*. In other words, to encode a message $x$ under the simplex code, we compute its inner product with every nonzero $y \in \{0,1\}^r$.

Let's look at the properties of the simplex code. It is a binary linear code with block length $2^r - 1$ and dimension $r$. So its rate, $\frac{r}{2^r-1}$, is pretty bad... What is its minimum distance?

**Theorem 2** *The simplex code has minimum distance $2^{r-1}$. Therefore, it is a $[2^r - 1, r, 2^{r-1}]_2$ code. If we write $n = 2^r - 1$, then it is an $[n, \log(n+1), \frac{n+1}{2}]_2$ code.*

**Proof:** It suffices to check that the Hamming weight of every nonzero codeword is at least $2^{r-1}$.

Fix any nonzero $x \in \{0,1\}^r$ and let $i$ be a nonzero coordinate of $x$. Let $e_i$ be the vector which is 1 at coordinate $i$ and 0 elsewhere. Note that for every $y$ we have

$$\langle x, y + e_i \rangle = \langle x, y \rangle + \langle x, e_i \rangle = \langle x, y \rangle + 1 \pmod 2.$$

Therefore, the bits of the encoding of $x$ indexed by $y$ and $y + e_i$ are different. Since this holds for an arbitrary $y \in \{0,1\}^r$, we have $\langle x, y \rangle = 1$ for exactly half of the $y$'s in $\{0,1\}^r$. ∎

The simplex code has terrible rate but amazing minimum distance. It can correct a lot of errors.

The Hadamard code is a closely related cousin of the simplex code. It is obtained by adding the all-0s column vector to $H_r$, so that we can write the Hadamard code as

$$\mathcal{C}_{\text{Had}} = \{(\langle x, y \rangle)_{y \in \{0,1\}^r} : x \in \{0,1\}^r\}.$$

It is easy to see from the above that the Hadamard code is a $[2^r, r, 2^{r-1}]_2$ code. If we write $n = 2^r$, then it is an $[n, \log n, n/2]_2$ code.

The simplex and Hadamard codes may not seem very impressive right now. But it turns out they are extremely important codes, with many useful properties and applications in computer science. And, as we will see, it is not really their fault that their rate is very bad.

### 1.2 Self-dual codes

If one is used to thinking of linear algebra over $\mathbb{R}$, it is natural to conjecture that $\mathcal{C} \cap \mathcal{C}^\perp = \{0\}$ for any binary linear code $\mathcal{C}$. This is not true over $\{0,1\}$. In fact, there exist binary linear codes $\mathcal{C}$ such that $\mathcal{C} = \mathcal{C}^\perp$! These are called *self-dual codes*.

For example, the 2-repetition binary linear code with $4 \times 2$ generator matrix

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

is self-dual, since

$$\langle 1100, 1100 \rangle = 0, \quad \langle 0011, 0011 \rangle = 0, \quad \langle 1100, 0011 \rangle = 0.$$

It is a simple but useful exercise to show that an $[n, k, d]_2$ self-dual code must have dimension $k = n/2$ (and so, in particular, $n$ must be even).

## 2 What we can't do: The generalized Hamming bound

**The context.** A major direction in coding theory is to understand the optimal tradeoff between code rate and minimum distance, and how to get close to this tradeoff while simultaneously achieving efficient encoding and error-correction. We will dedicate a substantial part of this course to understanding this landscape in more generalit, and exploring interesting alternative coding problems that arise from this.

We already saw a simple argument for such a rate-distance tradeoff for the special case of minimum distance 3. We showed that the Hamming code is optimal among codes of minimum distance 3 (even non-linear ones!): If a code has minimum distance 3, then Hamming balls of radius 1 centered at any two codewords are disjoint. We can easily generalize this to any minimum distance $d$ and alphabet size $q$.

We denote the radius-$r$ Hamming ball in $\{0, \ldots, q-1\}^n$ centered at $x$ by $B_{n,r}(x)$. Let $\mathrm{Vol}_q(n, r)$ denote the size ("volume") of $B_{n,r}(0)$. Note that the size of a Hamming ball does not depend on its center. We have that

$$\mathrm{Vol}_q(n, r) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i,$$

generalizing what we already know for $q = 2$. Later we will discuss some useful estimates for $\mathrm{Vol}_q(n, r)$.

**Theorem 3 (Generalized Hamming/sphere packing bound)** *If $\mathcal{C}$ is an $(n, M, d)_q$ code, then*

$$M \leq \frac{q^n}{\mathrm{Vol}_q(n, \lfloor \frac{d-1}{2} \rfloor)}.$$

**Proof:** If $\mathcal{C}$ has minimum distance $d$, then Hamming balls of radius $r = \lfloor \frac{d-1}{2} \rfloor$ centered at any two distinct codewords $c, c' \in \mathcal{C}$ are disjoint. This means that

$$q^n \geq \left| \bigcup_{c \in \mathcal{C}} B_r(c) \right| = \sum_{c \in \mathcal{C}} |B_r(c)| = |\mathcal{C}| \cdot \mathrm{Vol}_q(n, r).$$

Rearranging yields the desired inequality. ∎

## 3 What we can do: The Gilbert-Varshamov bound

The Hamming bound tells us what we cannot achieve with codes. Can we also get a lower bound on the size of the largest code we can construct with a given minimum distance $d$?

The Gilbert-Varshamov bound gives such a lower bound. It is obtained by following a simple greedy algorithm.

**Theorem 4 (Gilbert-Varshamov bound)** *For any positive integers $n$, $q$, and $d \leq n$, there exists an $(n, M, d)_q$ code of size*

$$M \geq \frac{q^n}{\mathrm{Vol}_q(n, d-1)}.$$

**Proof:** We construct $\mathcal{C}$ via a greedy algorithm, starting with an empty $\mathcal{C}$ and adding one codeword at a time.

The algorithm maintains a set $S$ of potential codewords, initially set as $S = \{0, 1, \ldots, q-1\}^n$, and the code $\mathcal{C}$, initially set as $\mathcal{C} = \varnothing$. In the $i$-th iteration, the algorithm checks if $S$ is empty:

1. If it is empty, then the algorithm stops and outputs $\mathcal{C}$.
2. Otherwise, it adds a vector $v$ of $S$ to $\mathcal{C}$, and removes from $S$ all vectors at Hamming distance less than $d$ from $v$.

To see correctness, note that after every iteration the vectors left in $S$ are at distance larger than $d$ from all elements in $\mathcal{C}$, as otherwise they would have been removed from $S$ in an earlier iteration.

Each iteration of the algorithm increases the size of $\mathcal{C}$ by 1 and decreases the size of $S$ by at most $\mathrm{Vol}_q(n, d-1)$. Since the initial size of $S$ is $q^n$, the $i$-th iteration will execute Case 2 as long as

$$(i-1) \cdot \mathrm{Vol}_q(n, d-1) < q^n.$$

Therefore, Case 1 will only be executed once $i - 1 \geq \frac{q^n}{\mathrm{Vol}_q(n,d-1)}$, in which case $\mathcal{C}$ will have size $|\mathcal{C}| = i - 1 \geq \frac{q^n}{\mathrm{Vol}_q(n,d-1)}$. ∎

Two simple but important observations:

- The code constructed in the proof of the Gilbert-Varshamov bound is not necessarily linear, even for a binary alphabet!
- Intuitively, this is a very inefficient way of constructing a code, and it does not tell us how to efficiently encode messages and correct errors, nor how to compactly describe this code.

**Rate-distance tradeoffs so far, or: Hamming vs. Gilbert-Varshamov.** Summarizing, the size $M$ of the largest code with block length $n$, minimum distance $d$, and alphabet size $q$ satisfies

$$\frac{q^n}{\mathrm{Vol}_q(n, d-1)} \leq M \leq \frac{q^n}{\mathrm{Vol}_q(n, \lfloor \frac{d-1}{2} \rfloor)}.$$

We may also write this in terms of rate $R$ of the code instead of its size $M$ as

$$1 - \frac{\log_q \mathrm{Vol}_q(n, d-1)}{n} \leq R \leq 1 - \frac{\log_q \mathrm{Vol}_q(n, \lfloor \frac{d-1}{2} \rfloor)}{n}.$$

Can we do better? And can we find more efficient ways of constructing codes (ideally close to or better than the Gilbert-Varshamov bound), and of encoding messages and correcting errors?

# 4 Linear codes achieve the Gilbert-Varshamov bound

A first natural problem given our greedy unstructured approach for deriving the Gilbert-Varshamov (GV) bound is to see if we achieve (or get close to) this bound using a more structured type of codes. Ideally, we would like to achieve the GV bound with codes supporting efficient encoding and error-correction procedures. This will keep us busy for most of the course.

In this section we make some partial progress towards this. We will show that there exist binary linear codes that achieve the GV bound. In fact, our result will say something stronger: *most* binary linear codes are very close to the GV bound. Recall that linear codes have a compact description (a generator matrix or parity-check matrix) and support efficient encoding (by multiplying the message vector by the generator matrix).

**Theorem 5 (Linear codes achieve the GV bound)** *For any positive integers $n$ and $d \leq n$ there exists an $[n, k, d]_2$ linear code satisfying $2^k \geq \frac{2^n}{\mathrm{Vol}_2(n,d-1)}$. In other words, this code has dimension at least $n - \log \mathrm{Vol}_2(n, d-1)$, or, equivalently, rate at least $1 - \frac{\log \mathrm{Vol}_2(n,d-1)}{n}$.*

**Proof:** We prove this result via the *probabilistic method*. More precisely, we will prove the existence of the desired linear code by analyzing the properties of a randomly chosen linear code. The probabilistic method is an amazing mathematical technique with many surprising applications.[1]

Fix some $n$ and $k \leq n$. Consider randomly choosing an $n \times k$ generator matrix $G$ by independently sampling each entry of $G$ uniformly at random from $\{0, 1\}$. We will show that the probability that $G$ is the generator matrix for a code with minimum distance at least $d$ is positive when $k \leq n - \log \mathrm{Vol}_2(n, d-1)$. This implies that there *exists* such a binary linear code!

Let $x \in \{0,1\}^k$ be a nonzero vector, and note that $Gx$ is uniformly distributed over $\{0,1\}^n$. Therefore, the probability that the Hamming weight of $Gx$ is less than $d$ is

$$\Pr[w_H(Gx) \leq d-1] = \frac{\mathrm{Vol}_2(n, d-1)}{2^n}. \tag{1}$$

The event that the code with generator matrix $G$ has minimum distance at most $d-1$ is equivalent to the existence of some nonzero $x \in \{0,1\}^k$ such that $w_H(Gx) \leq d-1$. Therefore, we need to upper bound

$$\Pr[\exists x \in \{0,1\}^k, x \neq 0 : w_H(Gx) \leq d-1]. \tag{2}$$

[1] A separate short set of lecture notes covers other applications of the probabilistic method and a tiny bit of its history to help you get used to it.

We do this by combining Equation (1) with the simple but very useful *union bound*: $\Pr[E_1 \vee E_2 \vee \cdots \vee E_m] \leq \sum_{i=1}^{m} \Pr[E_i]$, valid for all choices of events $E_1, \ldots, E_m$. Applying the union bound to Equation (2) yields

$$\begin{aligned}\Pr[\exists x \in \{0,1\}^k, x \neq 0 : w_H(Gx) \leq d-1] &\leq \sum_{x \in \{0,1\}^k \setminus \{0\}} \Pr[w_H(Gx) \leq d-1] \\ &= \sum_{x \in \{0,1\}^k \setminus \{0\}} \frac{\mathrm{Vol}_2(n, d-1)}{2^n} \\ &= \frac{(2^k - 1)\,\mathrm{Vol}_2(n, d-1)}{2^n}.\end{aligned}$$

Now, we are guaranteed that this probability is strictly less than 1 if

$$\frac{(2^k - 1)\,\mathrm{Vol}_2(n, d-1)}{2^n} < 1.$$

This holds whenever $2^k \geq \frac{2^n}{\mathrm{Vol}_2(n, d-1)}$.

Finally, note that the argument above also shows that $G$ is the generator matrix of a code of dimension $k$. This is equivalent to all $k$ columns of $G$ being linearly independent, which follows from the fact $Gx \neq 0$ for all nonzero $x \in \{0,1\}^k$. ∎

You may have noticed that this proof is *non-constructive*. It does not tell us how we can "efficiently" construct such a binary linear code, let alone one such code with efficient error-correction procedures.

It also does not preclude the existence of linear codes performing *better* than the GV bound.

Breaking the GV bound with binary linear codes is still a major open problem in coding theory. However, we know that there exist codes better than the GV bound over larger alphabets (alphabet size $q = 49$). In some sense, these codes perform better than "random codes" – this is fascinating because in many places in discrete mathematics and theoretical computer science random objects attain essentially optimal parameters. If you are curious about this, search for "algebraic-geometry codes".

## 5 Asymptotic coding theory and notions of efficiency

We have talked about "efficient" constructions of codes and "efficient" encoding and decoding procedures. However, our discussions were very informal. To discuss these notions more formally, we need to take an asymptotic view on codes (with block length growing to infinity), and this means we will consider *families* of codes.

### 5.1 Families of codes

A family of codes $(\mathcal{C}_i)_{i \in \mathbb{N}}$ is a collection of codes where each $\mathcal{C}_i$ is an $(n_i, M_i, d_i)_{q_i}$ code for some parameters $n_i, M_i, d_i, q_i$ that depend on the index $i$. We consider only families of codes where the

block lengths $n_i$ is strictly increasing with $i$. Then, we define the (asymptotic) rate of this family as

$$R = \lim_{i\to\infty} \frac{\log M_i}{n_i \log q_i},$$

whenever this limit exists. We define the (asymptotic) relative distance of this family as

$$\delta = \lim_{i\to\infty} \frac{d_i}{n_i},$$

whenever this limit exists.

The limits above exist and are easy to compute for all families of codes we will see in this course, but they may not exist in general. Also, the relationship between $i$ and the various parameters will usually be simple. When $q_i = q$ for all $i$, we say that this is a family of *$q$-ary* codes.

For the remainder of this course we will mostly be dealing with families of codes. We will do so implicitly, and will often use the word "code" to refer to a family of codes when the parameterization of this family is clear from context.

Here are some examples of families of codes we have already seen implicitly.

**Hamming codes.** The Hamming codes we discussed form the following family of binary linear codes $(\mathcal{C}_r)_{r\in\mathbb{N}}$ parameterized by $r \in \mathbb{N}$. The code $\mathcal{C}_r$ is an $[n_r, k_r, d_r]_{q_r}$ code with $n_r = 2^r - 1$, $k_r = 2^r - r - 1$, $d_r = 3$, and $q_r = 2$. Therefore, the rate of this family is

$$R = \lim_{r\to\infty} \frac{2^r - r - 1}{2^r - 1} = 1,$$

while its relative distance is

$$\delta = \lim_{r\to\infty} \frac{3}{2^r - 1} = 0.$$

**Simplex codes.** Simplex codes are the dual of Hamming codes, and form a family of binary linear codes $(\mathcal{C}_r)_{r\in\mathbb{N}}$ with $n_r = 2^r - 1$, $k_r = r$, $d_r = 2^{r-1}$, and $q_r = 2$. Therefore, the rate of this family is

$$R = \lim_{r\to\infty} \frac{r}{2^r - 1} = 0,$$

while its relative distance is

$$\delta = \lim_{r\to\infty} \frac{2^{r-1}}{2^r - 1} = \frac{1}{2}.$$

**Asymptotically good codes.** Families of codes with positive relative distance and rate are said to be *asymptotically good*. Neither Hamming codes nor simplex codes are asymptotically good The existence of such families is not entirely obvious, but it is guaranteed by the Gilbert-Varshamov bound! The next natural challenge would then be to construct asymptotically good codes with efficient encoding and error-correction procedures.

### 5.2 Notions of efficiency

We will speak of efficiency in asymptotic terms. Consider a family of codes $(\mathcal{C}_i)_{i\in\mathbb{N}}$, each an $(n_i, M_i, d_i)_{q_i}$ code, with strictly increasing block lengths $n_i$.

Then, we say that an encoding or error-correction algorithm for this family of codes is *efficient* if it is deterministic and runs in time polynomial in the block length of the code.[2]

We also briefly discussed "efficiently constructing codes". By this we mean that there is an algorithm that given an index $i$ outputs a "description" of $\mathcal{C}_i$ in time polynomial in the block length $n_i$. For binary linear codes, such a description could be the $n_i \times k_i$ generator matrix of $\mathcal{C}_i$.

Note that our notion of "efficiency" does not readily correspond to "practical" encoding and error-correction algorithms. First, our notion is asymptotic – perhaps the algorithm requires $2^{1000}n$ steps, a runtime that is only non-trivial for large block lengths. Second, sometimes even algorithms running in $n^3$ or $n^2$ steps (with small leading constants) are already too inefficient in practice. Some reasons why we aim for the notion of efficiency above are that (i) it is already challenging to achieve it, (ii) it is usually a necessary requirement for practicality, and (iii) "practicality" is heavily application-dependent, and so cannot really be formalized. Designing codes with good concrete parameters and practical encoding and error-correction algorithms is a related but separate fascinating research direction in coding theory.

### 5.3 Asymptotic Hamming bound

Sometimes bounds on codes have cleaner statements in the asymptotic setting. Recall that the Hamming bound states that an $(n, M, d)_q$ code of rate $R$ must satisfy

$$R \leq 1 - \frac{\log_q \mathrm{Vol}_q(n, \lfloor \frac{d-1}{2} \rfloor)}{n}. \tag{3}$$

When we apply this statement to a family of codes and take the block length to infinity, what do we get? It is clear that to understand this we must understand the asymptotic behavior of $\mathrm{Vol}_q(n, r)$ as $n \to \infty$ and $r/n \to \delta$. This is given by the following theorem.

**Theorem 6** *For any integer $q \geq 2$ and any real number $0 \leq p \leq 1 - 1/q$ we have that*[3]

$$q^{h_q(p) - o(n)} \leq \mathrm{Vol}_q(n, pn) \leq q^{h_q(p)},$$

*where*[4]

$$h_q(p) = p \log_q(q-1) - p \log_q p - (1-p) \log_q (1-p)$$

[2] This is still a bit informal. What is an algorithm? For the readers who know a bit about the theory of computation, one possible formal definition for an encoding algorithm would be a Turing machine that takes as input a block length $n$ (in binary) and an input message $x$ (in binary) of the appropriate length, and outputs in time polynomial in $n$ the encoding of $x$ under the code from the family with block length $n$. A formal definition for an error-correction algorithm follows similarly.

[3] Here, $o(n)$ stands for something that goes to 0 as $n \to \infty$.

[4] For readers who know a bit of information theory, $h_2(p)$ is the Shannon entropy of a Bernoulli random variable with success probability $p$. There is a nice information-theoretic proof of this theorem that explains why this quantity appears in the asymptotic expression for the volume of a Hamming ball (see here).

*is the $q$-ary entropy function.*

Combining Equation (3) with Theorem 6, we get that every $q$-ary family of codes with rate $R$ and relative distance $\delta$ must satisfy

$$R \leq 1 - h_q(\delta/2).$$

This is the asymptotic version of the Hamming bound.

### 5.4 Asymptotic Gilbert-Varshamov bound

Likewise, we can establish an asymptotic version of the Gilbert-Varshamov bound. For any integer $q \geq 2$ and $\delta \leq 1 - 1/q$ there exists a family of $q$-ary codes with relative distance $\delta$ and rate

$$R \geq 1 - h_q(\delta).$$

Here is a comparison between the asymptotic GV and Hamming bounds for $q = 2$.

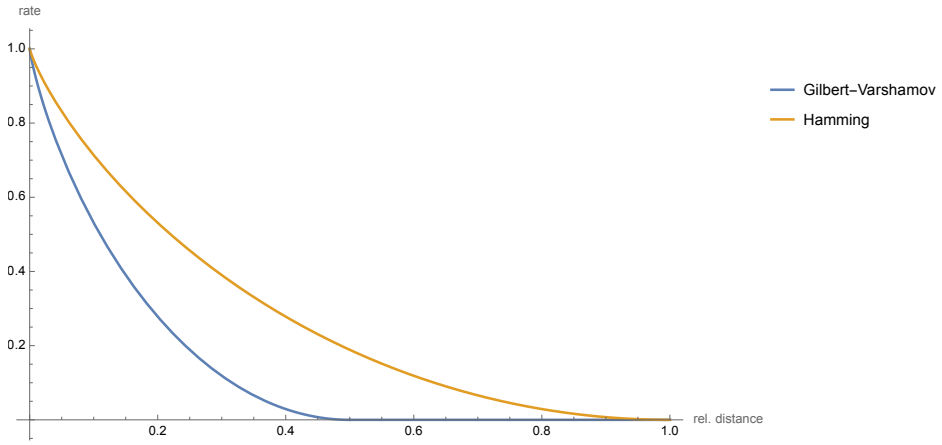


Figure 1: GV and Hamming bounds for binary codes.

We know that there are codes on the blue line, and that there are no codes beyond the orangle line. How about in between these lines?

Also, note that the rate guaranteed by the GV bound hits 0 when $\delta = 1 - 1/q$. Can we do better? For example, are there (families of) binary codes with relative distance $\delta \geq 1/2$ and positive rate $R > 0$?