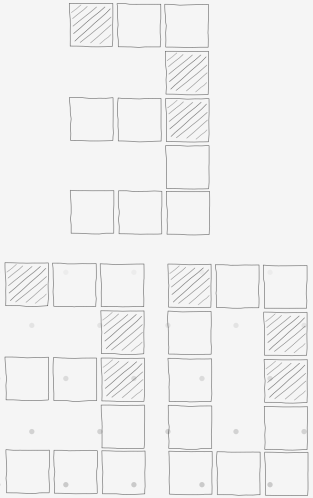
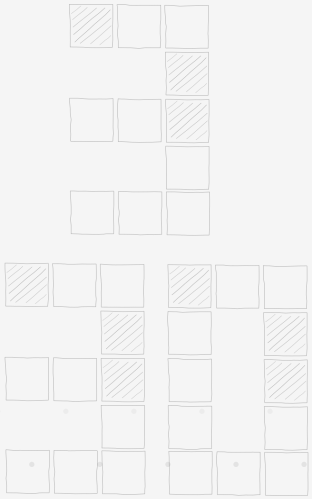


Information and Codes ▶ Error Correcting Codes II

Lecture 20

- Recall: Error-Correcting Codes Basics
- Linear Codes
- Parity Check Matrix
- Dual Codes



Information and Codes ▶ Error Correcting Codes II

Lecture 20

- Recall: Error-Correcting Codes Basics
- Linear Codes
- Parity Check Matrix
- Dual Codes

What is an Error-Correcting Code?

Goal: Transmit n-bit messages over a noisy channel or store them in corruptible memory

Idea: Add redundancy by encoding messages as longer m-bit strings ($m > n$)

Formally, a binary code consists of:

- **Encoding:** $\text{Enc} : \{0,1\}^n \to \{0,1\}^m$ (injective $\to$ different messages get different encodings)
- **Decoding:** $\text{Dec} : \{0,1\}^m \to \{0,1\}^n$ (recovers original message from possibly corrupted codeword)

Equivalently, think of a code through its codewords:

$$C = \{\text{Enc}(x) : x \in \{0,1\}^n\} \subseteq \{0,1\}^m$$

The code has **length** m, **dimension** n, and contains $|C| = 2^n$ codewords

Remark

Today we focus on **binary** codes ($\Sigma = \{0,1\}$), but codes over other alphabets exist (e.g., Reed-Solomon codes use larger finite fields)

Hamming Distance and Minimum Distance

Hamming Distance: Number of positions where two strings differ

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

Minimum Distance of Code C:

$$d(C) = \min_{\substack{c,c' \in C \\ c \neq c'}} d(c, c')$$

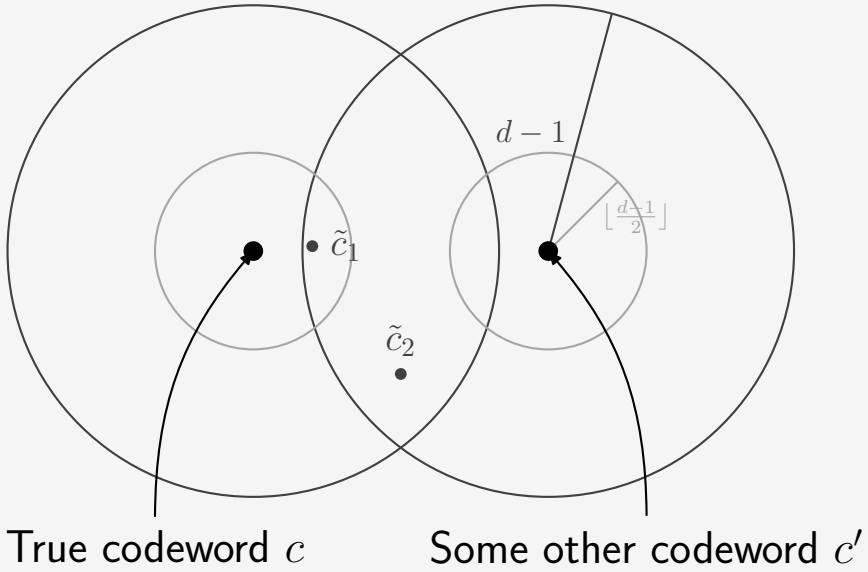
This measures how "spread out" the codewords are

Key Relationship: If $d(C) = d$, then:

- Can **detect** up to $d - 1$ errors
- Can **correct** up to $\lfloor (d-1)/2 \rfloor$ errors (via nearest-neighbor decoding)

Intuition: Codewords need to be far apart so corrupted versions don't overlap

Visualizing Distance and Error Correction



- **Inner circles** (radius $\lfloor (d-1)/2 \rfloor$): error-correction balls
- **Outer circles** (radius $d-1$): error-detection balls
- If $\leq \lfloor (d-1)/2 \rfloor$ bits flip, corrupted word $\tilde{c}$ stays in its inner ball
- Decoder finds unique codeword whose ball contains received word

Rate and the Redundancy Tradeoff

Rate: $R = n/m$ (information bits / total bits)

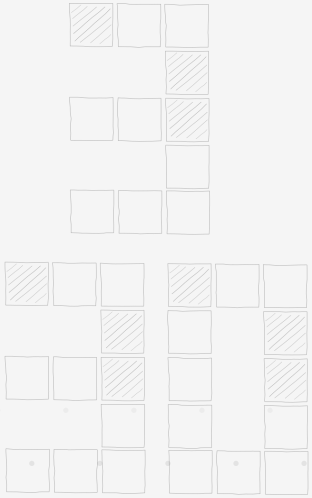
- Fraction of bits carrying actual information
- Example: $R = 1/2$ means half the bits are redundancy

Fundamental Tradeoff: High rate vs. high distance

- Want R close to 1 (low redundancy, efficient)
- Want d large (correct many errors, reliable)

Example: Repetition code $x \mapsto (x, x, x)$

- Has $d = 3$ (can correct 1 error)
- But $R = 1/3$ (not very efficient, wastes $2/3$ of bits on redundancy)



Information and Codes ▶ Error Correcting Codes II

Lecture 20

- Recall: Error-Correcting Codes Basics
- Linear Codes
- Parity Check Matrix
- Dual Codes

Linear Codes via Generator Matrices

Idea: Use linear algebra to create codes with structure

Definition: Given an $m \times k$ matrix G over $\{0,1\}$, define encoding:

$$\text{Enc}(x) = Gx \pmod 2$$

for messages $x \in \{0,1\}^k$

This gives codewords in $\{0,1\}^m$ with $|C| = 2^k$

Remark

Advantage: Encoding is efficient — just a matrix-vector multiplication!

Linear Algebra over $\mathbb{F}_2$

Field $\mathbb{F}_2 = \{0, 1\}$: Addition and multiplication mod 2

Addition (XOR):

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 1 = 0$$

Multiplication (AND):

$$0 \cdot 0 = 0, \quad 0 \cdot 1 = 0, \quad 1 \cdot 1 = 1$$

Vector space structure:

- Vectors, subspaces, basis, dimension work as usual
- Matrix operations: all arithmetic mod 2

Remark

Beyond $\mathbb{F}_2$: For any prime p, there exists a finite field $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ with addition and multiplication mod p. Codes can be built over any finite field (we won't cover this in class).

Sums of Codewords are Codewords

Lemma: Closure Under Addition

For a code $C = \{Gx : x \in \{0,1\}^k\}$ defined by a generator matrix G:
If $c_1, c_2 \in C$, then $c_1 \oplus c_2 \in C$

Proof

Since $c_1, c_2 \in C$, there exist messages $x_1, x_2 \in \{0,1\}^k$ with:

$$c_1 = Gx_1 \quad \text{and} \quad c_2 = Gx_2$$

Compute the sum:

$$\begin{aligned} c_1 \oplus c_2 &= Gx_1 \oplus Gx_2 \\ &= G(x_1 \oplus x_2) \quad \text{(by linearity of matrix multiplication)} \end{aligned}$$

Since $x_1 \oplus x_2 \in \{0,1\}^k$, we have $c_1 \oplus c_2 = G(x_1 \oplus x_2) \in C$

Generator Matrix and Basis

Generator matrix G: $m \times k$ matrix

$$G = \begin{bmatrix} g_1 & g_2 & \cdots & g_k \end{bmatrix}$$

Columns of G form a basis for C:

$$c = Gx = x_1 g_1 \oplus x_2 g_2 \oplus \cdots \oplus x_k g_k$$

Every codeword is a unique linear combination of the columns

In other words $C = \text{span}\{\text{cols}(G)\}$ is a subspace of $(\mathbb{F}_2)^m$

Remark

Dimension: $\dim(C) = k$ (rank of G), giving exactly 2^k codewords

Example: Hamming $[7, 4, 3]$ Code

Generator matrix:

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

Encoding: $\mathsf{Enc}(x) = Gx$ gives

$$\mathsf{Enc}(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4,\ x_2 \oplus x_3 \oplus x_4,\ x_1 \oplus x_3 \oplus x_4,\ x_1 \oplus x_2 \oplus x_4)$$

Example: $\mathsf{Enc}(1, 0, 1, 1) = (1, 0, 1, 1,\ 0 \oplus 1 \oplus 1,\ 1 \oplus 1 \oplus 1,\ 1 \oplus 0 \oplus 1) = (1, 0, 1, 1, 0, 1, 0)$

Distance in Linear Codes

Hamming weight: Number of 1s in a string

$$\mathsf{wt}(c) = |\{i : c_i = 1\}|$$

Lemma: Minimum Distance = Minimum Weight

For a linear code C:

$$d(C) = \min_{\substack{c \in C \\ c \neq \mathbf{0}}} \mathsf{wt}(c)$$

Proof

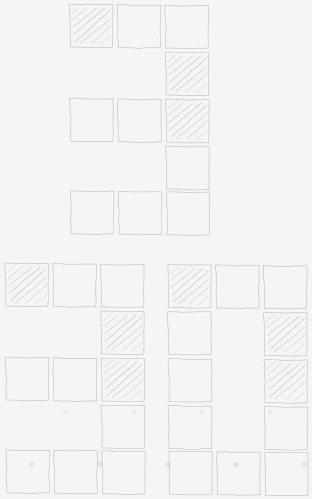
For any $c_1, c_2 \in C$ with $c_1 \neq c_2$:

$$d(c_1, c_2) = \mathsf{wt}(c_1 \oplus c_2)$$

Since C is linear: $c_1 \oplus c_2 \in C$ and $c_1 \oplus c_2 \neq \mathbf{0}$

Therefore:

$$d(C) = \min_{\substack{c_1, c_2 \in C \\ c_1 \neq c_2}} d(c_1, c_2) = \min_{\substack{c \in C \\ c \neq \mathbf{0}}} \mathsf{wt}(c)$$



Information and Codes ▶ Error Correcting Codes II

Lecture 20

- Recall: Error-Correcting Codes Basics
- Linear Codes
- Parity Check Matrix
- Dual Codes

Recall: Parity Checks

Idea: Add redundancy via linear constraints on the bits

Example: Simple parity check on 3 bits

- Constraint: $c_1 \oplus c_2 \oplus c_3 = 0$ (even parity)
- Valid codewords: $(0,0,0), (1,1,0), (1,0,1), (0,1,1)$
- Invalid: $(1,0,0), (0,1,0), (0,0,1), (1,1,1)$

Multiple constraints: Can enforce several parity checks simultaneously

- Each constraint is a linear equation over $\mathbb{F}_2$
- Codewords must satisfy *all* constraints

From Constraints to Matrices

Recall: A linear code C is a subspace of $(\mathbb{F}_2)^m$

Key insight: Any subspace can be described as the null space (kernel) of a matrix!

Null space of matrix H:

$$\{c \in (\mathbb{F}_2)^m : Hc = \mathbf{0}\}$$

For a k-dimensional subspace $C \subseteq (\mathbb{F}_2)^m$, there exists a $(m-k) \times m$ matrix H with:

$$C = \{c \in (\mathbb{F}_2)^m : Hc = \mathbf{0}\}$$

This H is called the **parity-check matrix** of C

Relationship Between G and H

Key property: $HG = 0$ (the zero matrix)

Equivalently: Rows of H are orthogonal to columns of G

Intuition:

- G **generates** the code (spans the subspace)
- H **defines constraints** codewords must satisfy
- Every codeword $c = Gx$ satisfies all parity checks: $Hc = HGx = 0$

Distance and Column Dependencies

Lemma: Minimum Distance from Parity-Check Matrix

If H is the parity-check matrix of a linear code C, then $d(C)$ equals the minimum number of columns of H that are linearly dependent.

Proof

Upper bound ($d(C) \leq$ min dependent set): If columns $h_{j_1}, \ldots, h_{j_t}$ are linearly dependent, then:

$$h_{j_1} \oplus h_{j_2} \oplus \cdots \oplus h_{j_t} = \mathbf{0}$$

The vector c with 1s in positions $j_1, \ldots, j_t$ satisfies $Hc = \mathbf{0}$, so $c \in C$ with $\text{wt}(c) = t$.

Lower bound ($d(C) \geq$ min dependent set): Let $c \in C$ be nonzero with $\text{wt}(c) = w$. Then c has 1s in positions $i_1, \ldots, i_w$.
Since $Hc = \mathbf{0}$:

$$h_{i_1} \oplus h_{i_2} \oplus \cdots \oplus h_{i_w} = \mathbf{0}$$

So columns $h_{i_1}, \ldots, h_{i_w}$ are linearly dependent.

Therefore: $d(C)$ equals the minimum size of a linearly dependent set of columns.

Example: Hamming [7,4] Parity-Check Matrix

Generator matrix:

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

Parity-check matrix:

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Verify: $HG = 0$ (exercise!)

Syndromes and Error Detection

Received word: $y = c + e$ where

- $c \in C$ is the codeword that was sent
- e is the error pattern (which bits flipped)

Syndrome:

$$\begin{aligned} s &= Hy \\ &= H(c+e) \\ &= Hc + He \\ &= \mathbf{0} + He = He \end{aligned}$$

Syndrome depends **only on the error**, not on the codeword!

Error detection: $s = \mathbf{0}$ iff $e \in C$ (undetectable) or $e = \mathbf{0}$ (no error)

Syndrome Decoding for Single Errors

For distance-3 codes (correct 1 error):

If single bit j is flipped: $e = e^{(j)}$ (all zeros except position j)

Syndrome:

$$s = He^{(j)} = j\text{-th column of } H$$

If all columns of H are distinct and nonzero, syndrome uniquely identifies error position!

Decoding algorithm:

1. Compute syndrome $s = Hy$
2. Find column of H matching s (this is position j)
3. Flip bit j to recover c

Worked Example: Syndrome Decoding

Use Hamming [7,4] code:

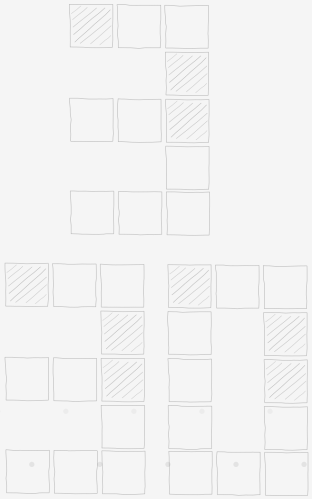
$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Suppose we send $c = (1, 0, 1, 1, 0, 1, 0)^T$
Error in position 3: receive $y = (1, 0, \mathbf{0}, 1, 0, 1, 0)^T$

Compute syndrome:

$$s = Hy = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \oplus 1 \oplus 0 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$



Information and Codes ▶ Error Correcting Codes II

Lecture 20

- Recall: Error-Correcting Codes Basics
- Linear Codes
- Parity Check Matrix
- Dual Codes

Dual Code Definition

Given a code C its **dual code** $C^\perp$ is given by:

$$C^\perp = \{y \in (\mathbb{F}_2)^m : \langle x, y \rangle = 0 \text{ for all } x \in C\}$$

where the inner product is over $\mathbb{F}_2$: $\langle x, y \rangle = x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_m y_m$

Lemma: $C^\perp$ is a Linear Code

If C is a linear code, then $C^\perp$ is also a linear code.

Proof

(1) Zero vector: For all $x \in C$:

$$\langle x, \mathbf{0} \rangle = x_1 \cdot 0 \oplus x_2 \cdot 0 \oplus \cdots \oplus x_m \cdot 0 = 0$$

(2) Closure: Let $y_1, y_2 \in C^\perp$. For any $x \in C$:

$$\langle x, y_1 \oplus y_2 \rangle = \langle x, y_1 \rangle \oplus \langle x, y_2 \rangle = 0 \oplus 0 = 0$$

Properties of Dual Codes

Property 1: If $C \subseteq (\mathbb{F}_2)^m$ has dimension k, then $C^\perp \subseteq (\mathbb{F}_2)^m$ has dimension $m - k$.

Property 2: $(C^\perp)^\perp = C$ (note that $m - (m - k) = k$)

Property 3: If H is a parity-check matrix for C, then H^T is a generator matrix for $C^\perp$.
Equivalently, if G is a generator matrix for C, then G^T is a parity-check matrix for $C^\perp$.

Proofs are left as exercises.

Geometric Interpretation

Think of $(\mathbb{F}_2)^m$ as m-dimensional vector space

- C is a k-dimensional subspace
- $C^\perp$ is the "orthogonal complement" of dimension $m - k$
- Together they have complementary dimensions (sum to m)

Remark

Note: Unlike over $\mathbb{R}$, we can have $C \cap C^\perp \neq \{\mathbf{0}\}$
In fact, it's possible that $C = C^\perp$ (called **self-dual** codes)

General Hamming Codes

Construction: For any $r \geq 2$, define the Hamming code via its parity-check matrix

Parity-Check Matrix H_r:
H_r is the $r \times (2^r - 1)$ matrix where column i is the binary representation of i

This matrix contains e_1 through e_r (binary representations of all powers of two from 1 to 2^{r-1}), so it has full row rank

Remark

Parameters of the Hamming Code:

- Length: $m = 2^r - 1$
- Dimension: $k = 2^r - 1 - r$ (by rank-nullity)
- Rate: $R = \frac{2^r - 1 - r}{2^r - 1} \approx 1$ (high rate!)

Distance of Hamming Codes

What is the minimum distance?

Remark

Recall: $d(C)$ equals the minimum number of linearly dependent columns of H_r

Analysis of column dependencies:

- Any single column is nonzero $\implies$ linearly independent
- Any two distinct columns are different $\implies$ linearly independent
- But three columns can be linearly dependent!

Example: Columns $1, 2, 3$ (binary: $001, 010, 011$) satisfy:

$$001 \oplus 010 = 011$$

Therefore: $d = 3$

The Dual of Hamming Codes

Starting point: Hamming code C with parity-check matrix H_r (size $r \times n$, where $n = 2^r - 1$).

Construction: We construct the dual code $C^\perp$ using Property 3.

- Since H_r checks parity for C, its transpose H_r^T **generates** $C^\perp$.

The Generator Matrix for $C^\perp$:

$$G_{dual} = H_r^T$$

This is an $n \times r$ matrix. The columns of H_r become the **rows** of the generator.

Parameters of the Dual Code:

- **Length:** $n = 2^r - 1$ (same as Hamming)
- **Dimension:** r (By Property 1: $2^r - 1 - (2^r - 1 - r) = r$)
- **Rate:** $R = \frac{r}{2^r - 1} \approx 0$

Observation: This code is highly redundant (low rate).

Hadamard Codes

Remark

This dual code is known as the **Hadamard Code** (or Simplex Code).

Encoding: A message $u \in \{0,1\}^r$ is encoded into a codeword $c \in \{0,1\}^n$:

$$c = \mathsf{Enc}(u) = H_r^T u$$

The Coordinate-Wise View (Intuition): Recall that the rows of H_r^T are exactly all the nonzero vectors $v \in \{0,1\}^r$.

We can index the positions of the codeword by these vectors v:

$$c = (c_v)_{v \in \{0,1\}^r \setminus \{0\}}$$

The bit at position v is simply the inner product:

$$c_v = \langle u, v \rangle$$

The codeword for u is the "truth table" of the function $f_u(v) = \langle u, v \rangle$ evaluated on all nonzero inputs.

Distance of Hadamard Codes

The Question: What is the Hamming weight of a codeword for a nonzero message u?

The bits of the codeword are the values $\langle u, v \rangle$ as v ranges over all nonzero inputs.

Intuition via Symmetry: Suppose $u = (1, 0, \ldots, 0)$. Then $\langle u, v \rangle$ is just the **first bit** of v.

- Exactly half of all binary strings of length r start with a 1.
- By symmetry, this holds for *any* nonzero u: the dot product $\langle u, v \rangle$ is 1 for exactly half of the possible vectors v.

The Calculation:

1. Total number of vectors $v \in \{0, 1\}^r$ is 2^r.
2. Exactly half (2^{r-1}) give a dot product of 1.
3. We exclude $v = \mathbf{0}$ (where the dot product is 0), so the count of 1s remains 2^{r-1}.

Result: $d(\mathsf{Had}_r) = 2^{r-1}$.

Hamming vs. Hadamard: A Duality

Property	Hamming Code	Hadamard Code
Length	$2^r - 1$	$2^r - 1$
Dimension	$2^r - 1 - r$	r
Min Distance	3	2^{r-1}
Rate	≈ 1	≈ 0
Errors Corrected	1	$2^{r-2} - 1$

The Rate-Distance Tradeoff:

- **Hamming:** High rate, small distance.
  (Efficient encoding, but corrects few errors)
- **Hadamard:** Low rate, large distance.
  (Highly redundant, but corrects many errors)