Error-correcting codes

Last modified: 2026-11-07 21:35:35 (PST)

An error-correcting code adds redundancy to a message so that the receiver can recover it even after some symbols are changed. This page covers the basic definitions, the Hamming and Reed–Solomon codes, and then how codes that can be decoded by reading only a few symbols lead to central results of computational complexity.

Codes and distance

Definition 0.1 (Block code)**.** Let Σ be a finite alphabet with $q \geq 2$ symbols, and let $k \leq n$ be positive integers. A **block code** with message length k and block length n is a one-to-one map $E : \Sigma^k \to \Sigma^n$. Its image $C = \{E(m) : m \in \Sigma^k\}$ is the set of *codewords*, and its **rate** is $R \stackrel{\text{def}}{=} k/n$, the number of message symbols carried per symbol sent.

Definition 0.2 (Hamming distance and minimum distance)**.** The **Hamming distance** $\Delta(x, y)$ between two strings $x, y \in \Sigma^n$ is the number of positions i with $x_i \neq y_i$. The **minimum distance** of a block code with codewords C is

$$d \stackrel{\text{def}}{=} \min \{\Delta(c, c') : c, c' \in C,\ c \neq c'\}.$$

Theorem 0.1 (Minimum distance sets how many errors a code handles)**.** *Let a block code have minimum distance d, and let y be a codeword c with some of its symbols changed.*

1. *If at most $d - 1$ symbols changed and at least one did, then y is not a codeword, so the receiver detects the error.*
2. *If at most $t \stackrel{def}{=} \lfloor (d-1)/2 \rfloor$ symbols changed, then c is the only codeword nearest to y in Hamming distance, so decoding y to its nearest codeword corrects the errors.*

Proof

Proof. For part 1, $0 < \Delta(c, y) \le d - 1 < d$, so $y \neq c$. If y were another codeword c', then $\Delta(c, c') = \Delta(c, y) < d$, which the minimum distance rules out.
For part 2, Hamming distance satisfies the triangle inequality, since a position where c and c' differ is a position where y differs from c or from c'. So for any codeword $c' \neq c$,

$$\Delta(c', y) \ge \Delta(c, c') - \Delta(c, y) \ge d - t \ge t + 1 > \Delta(c, y),$$

where $d - t \ge t + 1$ because $2t \le d - 1$. □

Example 0.1 (The repetition code corrects one error)**.** The binary repetition code sends each bit three times: $E(0) = 000$ and $E(1) = 111$. Its minimum distance is $d = 3$, so by Theorem 0.1 it corrects $t = 1$ error: a received 010 decodes to 000, the codeword it is nearest to. The price is the rate, $R = 1/3$. MacKay (2003, chap. 1) calls this code R_3, and the channel capacity page finds its error probability on a noisy channel.

Remark 0.1 (Random errors and worst-case errors). Source coding removes redundancy from a message; error-correcting codes add redundancy back, in a controlled way, so that the receiver can undo errors. Shannon (1948) modeled errors as random and found that a code can make the probability of a decoding error as small as wanted at every rate below the channel capacity, and at no rate above it. The codes on this page instead guard against any pattern of up to t errors, however they arise, which is the model complexity theory uses.

Linear codes

Definition 0.3 (Linear code)**.** Let the alphabet be a finite field $\mathbb{F}_q$, such as $\mathbb{F}_2 = \{0, 1\}$ with addition and multiplication modulo 2. A **linear code** is a block code whose set of codewords C is a k-dimensional subspace of $\mathbb{F}_q^n$. One way to describe it is by a **parity-check matrix**: an $(n - k) \times n$ matrix H of rank $n - k$ with $C = \{y \in \mathbb{F}_q^n : Hy = 0\}$, where y is treated as a column vector. A linear code with block length n, dimension k and minimum distance d is called an $[n, k, d]_q$ code.

Definition 0.4 (Weight)**.** The **weight** of a string $x \in \mathbb{F}_q^n$ is its number of nonzero symbols.

Remark 0.2 (The minimum distance of a linear code is its smallest nonzero weight). For codewords $c \neq c'$ of a linear code, $\Delta(c, c')$ is the weight of $c - c'$, which is itself a nonzero codeword. So the minimum distance is the smallest weight of a nonzero codeword.

Example 0.2 (The (7,4) Hamming code)**.** Let H be the 3×7 binary matrix whose jth column is the number j written in binary, most significant digit on top:

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

Columns 1, 2 and 4 of H are the three unit vectors, so H has rank 3. Taking H as the parity-check matrix therefore gives a linear code of dimension $k = 7 - 3 = 4$, with $2^4 = 16$ codewords. The weight argument of Remark 0.2 gives $d = 3$:

- No codeword has weight 1, because no column of H is zero.
- No codeword has weight 2, because no two columns of H are equal, so no two add to zero.
- 1110000 is a codeword of weight 3, because columns 1, 2 and 3 add to zero.

So this is a $[7, 4, 3]_2$ code. It corrects one error, like the repetition code, at rate 4/7 instead of 1/3.
Decoding is direct. If the codeword c arrives with an error in position j, the received word is $y = c + e_j$, where e_j has a 1 in position j only, and

$$Hy = Hc + He_j = 0 + (\text{column } j \text{ of } H),$$

which is j in binary. For example, if $c = 0101010$ arrives as $y = 0101110$, then $Hy = (1, 0, 1)$, which is 5 in binary, so the decoder flips position 5. MacKay (2003, sec. 1.2) builds the same code from parity checks, with its positions in a different order.

Reed–Solomon codes

Theorem 0.2 (Singleton bound)**.** *Every block code over an alphabet of q symbols, with message length k, block length n and minimum distance d, satisfies $d \leq n - k + 1$.*

Proof

Proof. Delete the last $d - 1$ symbols of every codeword. Two different codewords differ in at least d positions, so they still differ in one of the first $n - d + 1$ positions. So the q^k

codewords stay distinct as strings of length $n-d+1$, of which there are q^{n-d+1}. Hence $q^k \le q^{n-d+1}$, that is, $k \le n-d+1$. □

Definition 0.5 (Reed–Solomon code)**.** Let $\mathbb{F}_q$ be a finite field, let $k \le n \le q$, and fix n distinct points $a_1, \dots, a_n \in \mathbb{F}_q$. The **Reed–Solomon code** sends a message $m = (m_0, \dots, m_{k-1}) \in \mathbb{F}_q^k$ to the values of the polynomial $f_m(t) \stackrel{\text{def}}{=} m_0 + m_1 t + \cdots + m_{k-1} t^{k-1}$ at the n points:

$$E(m) \stackrel{\text{def}}{=} (f_m(a_1), \dots, f_m(a_n)).$$

Reed and Solomon (1960) introduced these codes.

Theorem 0.3 (Reed–Solomon codes meet the Singleton bound)**.** *The Reed–Solomon code is an $[n, k, n-k+1]_q$ linear code.*

Proof

Proof. The map $m \mapsto E(m)$ is linear, because $f_{m+m'} = f_m + f_{m'}$ and $f_{bm} = b f_m$ for every scalar $b \in \mathbb{F}_q$. If $m \neq 0$, then f_m is a nonzero polynomial of degree at most $k-1$, so it has at most $k-1$ roots, and $E(m)$ has at least $n-k+1 \ge 1$ nonzero entries. So E is one-to-one, the codewords form a k-dimensional subspace, and by Remark 0.2 the minimum distance is at least $n-k+1$. The Singleton bound says it is at most $n-k+1$. □

Example 0.3 (A small Reed–Solomon code)**.** Take $q = n = 5$, $k = 2$, and the points $0, 1, 2, 3, 4$ of $\mathbb{F}_5$, the integers modulo 5. The message $m = (1, 2)$ gives the line $f_m(t) = 1 + 2t$, and

$$E(m) = (1, 3, 0, 2, 4),$$

since $f_m(2) = 5 \equiv 0$, $f_m(3) = 7 \equiv 2$ and $f_m(4) = 9 \equiv 4$. By Theorem 0.3 the minimum distance is $5 - 2 + 1 = 4$, so the code corrects one error. In terms of lines: with one error, the true line passes through at least four of the five received values. Any other line meets the true line in at most one point, so it passes through at most two of the received values, and the nearest codeword is the true line.

From codes to complexity

Complexity theory uses codes that can be decoded by reading only a few symbols of the received word. The *Computational Complexity for Data Science* library covers the complexity classes

these results are about.

Definition 0.6 (Hadamard code)**.** The **Hadamard code** sends a message $x \in \mathbb{F}_2^k$, for $k \geq 1$, to the 2^k dot products

$$x \cdot a = x_1 a_1 + \cdots + x_k a_k \pmod 2,$$

one for each $a \in \mathbb{F}_2^k$, so its block length is $n = 2^k$. Also called the **Walsh–Hadamard code** (Arora and Barak 2009, chap. 11).

Theorem 0.4 (Minimum distance of the Hadamard code)**.** *The Hadamard code is a linear code with minimum distance* 2^{k-1}.

Proof

Proof. Let $x \neq 0$, and pick i with $x_i = 1$. Let e_i have a 1 in position i only. Then $x \cdot (a + e_i) = x \cdot a + 1$, so pairing each a with $a + e_i$ pairs a 0 with a 1, and exactly half of the 2^k entries of the codeword of x are 1. So every nonzero message has a codeword of weight $2^{k-1} \geq 1$. Each dot product $x \cdot a$ is linear in x, so the encoding map is linear, and only the zero message has the zero codeword, so the map is one-to-one and the code is linear. Then Remark 0.2 gives the minimum distance. □

Example 0.4 (The Hadamard code with $k = 2$)**.** List the positions a in the order $00, 01, 10, 11$. The Hadamard code sends the four messages to

- $x = 00 \mapsto 0000$,
- $x = 01 \mapsto 0101$, since $x \cdot a = a_2$,
- $x = 10 \mapsto 0011$, since $x \cdot a = a_1$,
- $x = 11 \mapsto 0110$, since $x \cdot a = a_1 + a_2 \pmod 2$.

Every nonzero codeword has weight 2, which is 2^{k-1}, as Theorem 0.4 says.

Theorem 0.5 (Two-query decoding of the Hadamard code)**.** *Let w be a binary string of length 2^k, with $w(a)$ its symbol at the position indexed by $a \in \mathbb{F}_2^k$, that differs from the Hadamard codeword of x in at most a fraction δ of its positions. Fix $i \in \{1, \ldots, k\}$, and let e_i have a 1 in position i only. Choose $a \in \mathbb{F}_2^k$ uniformly at random and output $w(a) + w(a + e_i) \pmod 2$. The output equals x_i with probability at least $1 - 2\delta$.*

Proof

Proof. Both a and $a + e_i$ are uniformly distributed on $\mathbb{F}_2^k$, so each of $w(a)$ and $w(a + e_i)$ is wrong with probability at most δ, and by the union bound both are correct with probability at least $1 - 2\delta$. When both are correct, the output is

$$x \cdot a + x \cdot (a + e_i) = x \cdot e_i = x_i,$$

because $x \cdot a + x \cdot a = 0$ modulo 2. □

Definition 0.7 (Locally decodable code)**.** A block code E is **locally decodable** with r queries and error fraction δ if there is a randomized decoder with the following property. For every message x, every word w that differs from $E(x)$ in at most a fraction δ of its positions, and every index i, the decoder, given i and access to w, reads at most r symbols of w and outputs x_i with probability at least $2/3$ over its random choices.

Remark 0.3 (Reading two symbols instead of the whole word). When $\delta \leq 1/6$, the success probability $1-2\delta$ in Theorem 0.5 is at least $2/3$, so the Hadamard code is locally decodable with 2 queries, however long the message is. When $\delta < 1/4$, the decoder is right with probability above $1/2$, and repeating it with fresh random choices and taking a majority vote drives its error probability as low as needed. Decoding to the nearest codeword, as in Theorem 0.1, instead reads the whole received word. The Hadamard code pays for local decoding with block length 2^k; polynomial codes in several variables, which extend Reed–Solomon codes, trade a few more queries for much shorter codewords (Arora and Barak 2009, chap. 19).

Remark 0.4 (Hardness amplification). A Boolean function $f : \{0,1\}^s \to \{0,1\}$ is determined by its *truth table*, a binary string of length 2^s, so it can be encoded like any message. Encode the truth table with a locally decodable code whose decoder is itself a small circuit, and read the codeword as the truth table of a new function g. If a small circuit computed g correctly on most inputs, then running a local decoder on that circuit's outputs would compute f correctly on every input, with high probability, using a circuit not much larger. So if no small circuit computes f on every input, then no small circuit computes g on most inputs. Arora and Barak (2009, chap. 19) develop this worst-case to average-case hardness amplification, and their chapter 20 uses it to build pseudorandom generators.

Remark 0.5 (Codes in the PCP theorem). The PCP theorem says that every statement in NP has a proof that a randomized verifier can check by reading a constant number of its bits and using only $O(\log n)$ random bits, where n is the length of the statement. If the statement is true, some proof makes the verifier accept with probability 1; if it is

false, every proof makes the verifier reject with probability at least 1/2. Arora and Barak (2009, chap. 11) prove a weaker version, in which the verifier uses polynomially many random bits, by writing the proof in the Hadamard code. The verifier then tests a few random positions of the encoded proof, in the spirit of Theorem 0.5, instead of reading all of it. They prove the full theorem in chapter 22.

References

Arora, Sanjeev, and Boaz Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press. https://doi.org/10.1017/CBO9780511804090.

MacKay, David J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Reed, I. S., and G. Solomon. 1960. "Polynomial Codes over Certain Finite Fields." *Journal of the Society for Industrial and Applied Mathematics* 8 (2): 300–304. https://doi.org/10.1137/0108018.

Shannon, C. E. 1948. "A Mathematical Theory of Communication." *Bell System Technical Journal* 27 (3): 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.