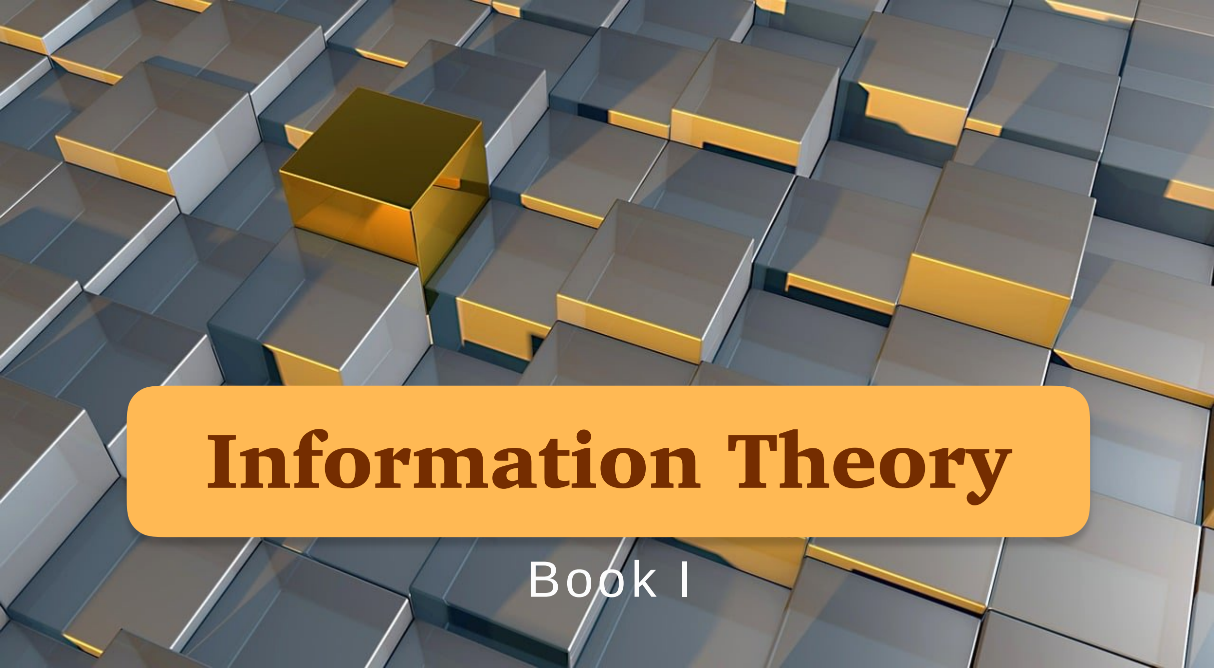
Information Theory

Book I



Information theory:

- Shannon information (statistical measurement of information content; classical information theory)
- Kolmogorov information (algorithmic information)
- Quantum information

Digital Photograph

10 MB bitmap $\longrightarrow$ 350 KB jpg $\xrightarrow{\text{lossy compression}}$ 80 KB

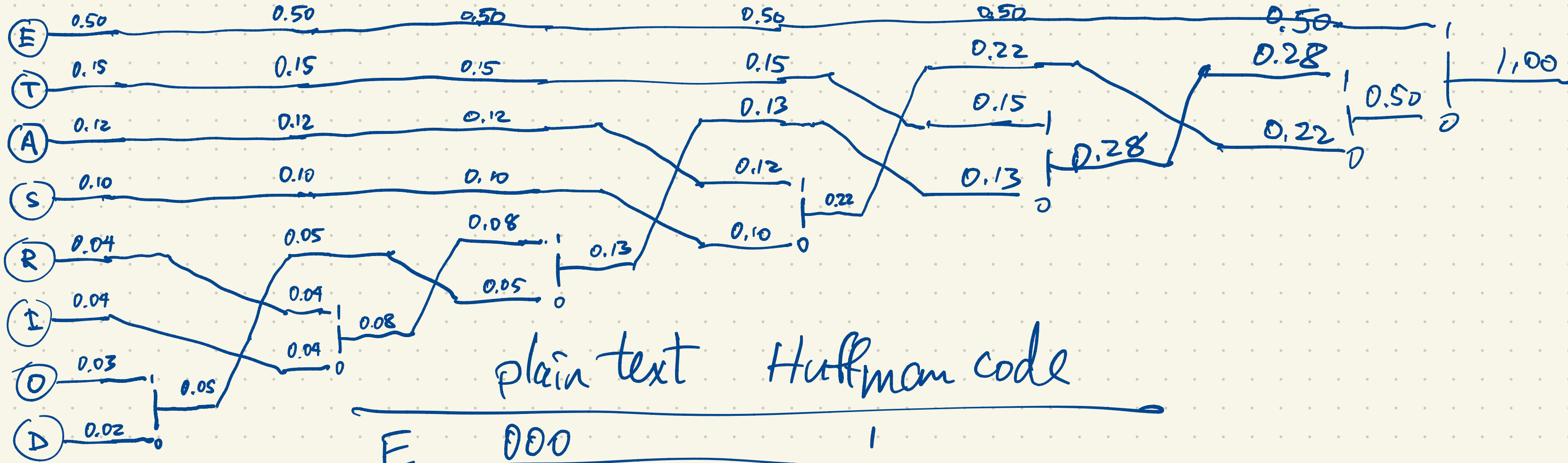
Text document

50 KB $\xrightarrow{\text{loss-less compression}}$ 2.3 KB $\longrightarrow$ 2.4 KB

a perfect copy of the original can be extracted/recovered.

Consider an information stream composed of E, T, A, S, R, I, O, ... D

(stream of independent letters) freq. 0.50, 0.15, ..., 0.02



plain text		Huffman code
E	000	1
T	001	011
A	010	001
S	011	000
R	100	01011
I	101	01010
O	110	01001
D	111	01000

Huffman encoding:
Encode STEER as $\underbrace{000}_{S}\underbrace{011}_{T}\underbrace{1}_{E}\underbrace{1}_{E}\underbrace{01011}_{R}$
Decoding

A string of n characters is represented as 3n bits (plain text) which the Huffman code compresses to 2.26n bits.
(75.3% of original)

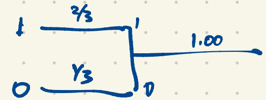
Example 2 of Huffman coding:

Stream of 0's and 1's

$\underbrace{\text{0's}}_{\frac{1}{3}}$ $\underbrace{\text{1's}}_{\frac{2}{3}}$

$X = \begin{cases} 1 & \text{with prob. } \frac{2}{3} \\ 0 & \cdots\cdots \ \frac{1}{3} \end{cases}$

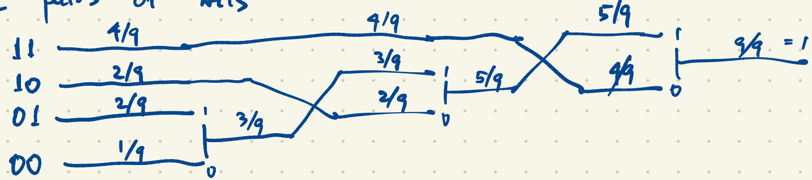
(binomial distribution)



Plain text	Huffman code
1	1
0	0

No compression

Take pairs of bits

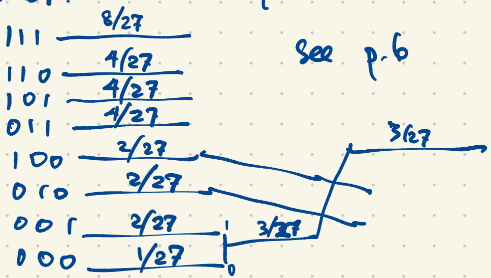


Plain text	Huffman code
11	0
10	10
01	111
00	110

On average, a plain text file of n bits encodes as $\frac{17}{18}n \approx 0.9444\,n$ bits.

Better: encode triples of bits

see p.6



Plain text	Huffman code
111	11
110	00
101	101
011	0111
100	100
010	0110
001	0101
000	0100

On average, n bits is encoded as $\frac{76}{27}n$ bits

$\approx 0.9383\,n$ bits.

What is the limit of the compression ratio (as the block size $\to \infty$)?

$0.9183\,n$ bits is the limit for compressing n bits from this stream

Shannon's first theorem showed that this stream has an entropy of

$H(X) = \frac{1}{3}\log\frac{1}{1/3} + \frac{2}{3}\log\frac{1}{2/3} \approx 0.9183.$

Example 1 Huffman code with blocksize 1 character gives n bits $\longrightarrow \frac{2.26}{3} n$ bits $\approx 0.753\, n$ bits

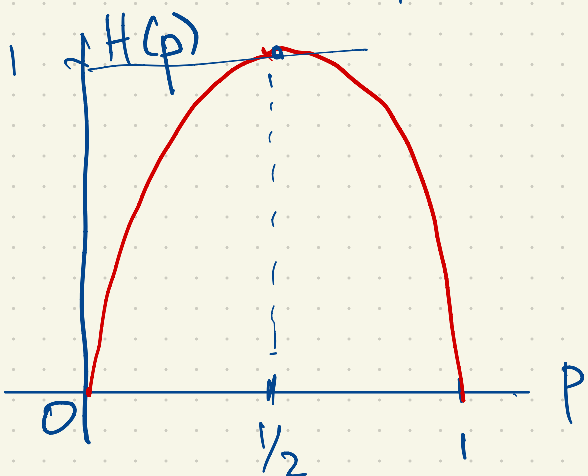
Entropy : $\sum_i p_i \log \frac{1}{p_i} \approx 1.55678$ bits per character

$p_i = 0.5, 0.15, \ldots, 0.12 \quad (i = 1, 2, \ldots, 8)$

Compare: plain text encoding of characters requires 3 bits.

Binary entropy function: A biased coin has heads with prob. p, tails ... $1-p$, $0 < p < 1$, with independent tosses

$H(\text{coin}) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$ = no. of bits (on average) to express the outcome of each coin flip.



Recall: If X is a random variable with outcomes $X = x_i$ $(1 \le i \le n)$ with prob. p_i $(\sum p_i = 1)$ then the binary entropy of X is $H_2(X) = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}$

= no. of bits on average required to express observed values of X.

When expressing information in base q, the q-ary entropy function $H_q(X) = \sum_{i=1}^{n} p_i \log_q \left(\frac{1}{p_i}\right) = \frac{1}{\log_2 q} H_2(X)$

Starting Friday, move to CR 144

Eg. A byte is 8 bits. $2^8 = 256$

If X can be encoded using N bits then it takes $\frac{N}{8}$ bytes.

If I buy a deck of cards its entropy is 0 in the sense that no information is required to express the order of the deck. After shuffling the deck, it takes 225.58 bits to express the order
← ignore jokers. $\log_2 52! \approx 225.58$
(about 68 decimals).

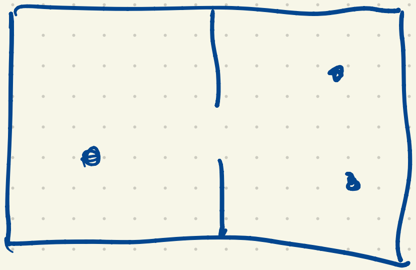
2nd Law of Thermodynamics
Watch the 7-8 minute video linked on course website

---

p. 49 Shannon's Source Coding Theorem (for channel without noise)

A channel is used to send a stream of symbols e.g. 0's and 1's reliably at a certain number C of bits per second. Information coming from a source X has finitely many outcomes with entropy $H(X) = H_2(X)$ bits per symbol eg. $x_1, \ldots, x_n$ or $A, B, C, D, \ldots$.
This information can be reliably send and received at a maximum rate $\frac{C}{H}\left\{\frac{\text{bits/sec}}{\text{bits/symbol}}\right\}$ symbols/sec.

Eg. X is a stream of characters $E, T, \ldots, D$ (first example) with prob. $0.50, 0.15, \ldots, 0.02$, $H(X) = 1.55$ bits/char.
If I transmit info. from this source using a noiseless channel with capacity 31 bits/sec. then
I can safely transmit less that $\frac{C}{H} = \frac{31 \text{ bits/sec}}{1.55 \text{ bits/char}} = 20$ char./sec.
We can get within any pos. ε of this optimal rate i.e. $20 - \varepsilon$.



Suppose X, Y are independent random variables each with finitely many possible values

X has value x_i with prob. $p_i \in (0,1)$ $(1 \le i \le m)$ $\sum p_i = 1$

Y has value y_j with prob. $q_j \in (0,1)$, $\sum q_j = 1$

$$H(X) = \sum_{i=1}^{m} p_i \log \frac{1}{p_i}, \quad H(Y) = \sum_{j=1}^{n} q_j \log \frac{1}{q_j}$$

The pair (X, Y) has value (x_i, y_j) with prob. $p_i q_j$

$$\sum_{\substack{1 \le i \le m \\ 1 \le j \le n}} p_i q_j = \sum p_i \cdot \sum q_j = 1 \cdot 1 = 1$$

$H(X,Y)$ joint entropy

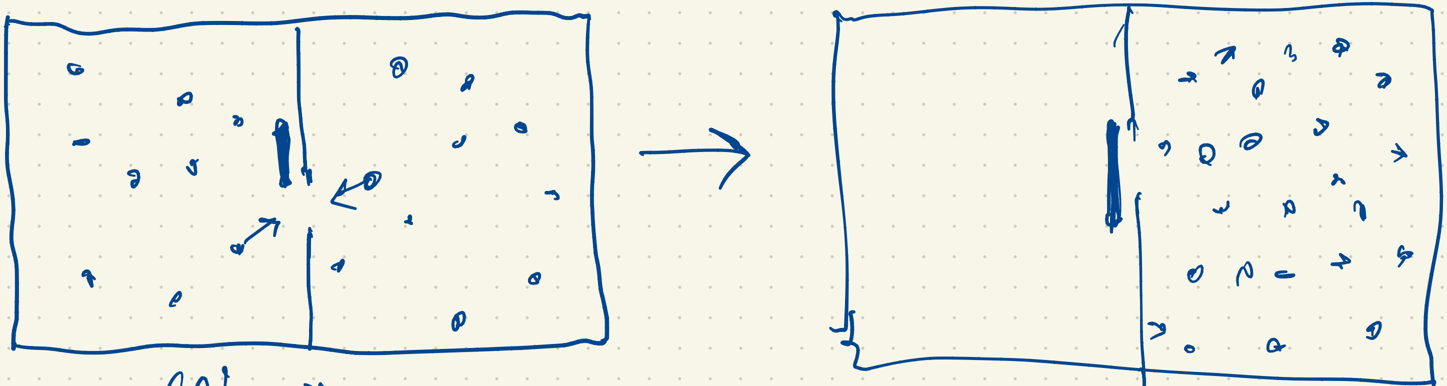
$$H(X,Y) = \sum_{i,j} p_i q_j \log\left(\frac{1}{p_i q_j}\right) = \sum_{i,j} p_i q_j \left(\log \frac{1}{p_i} + \log \frac{1}{q_j}\right)$$

$$= \sum_{i,j} p_i q_j \log \frac{1}{p_i} + \sum_{i,j} p_i q_j \log \frac{1}{q_j}$$

$$= \left(\sum_i p_i \log \frac{1}{p_i}\right) \underbrace{\sum_j q_j}_{1} + \underbrace{\left(\sum_i p_i\right)}_{1} \sum_j q_j \log \frac{1}{q_j} = H(X) + H(Y).$$

If X, Y are dependent

$$H(X,Y) \le H(X) + H(Y)$$



Maxwell's Demon

Computation requires some minimal expenditure of energy when initializing memory registers, and when reading memory registers, resulting in the creation of entropy.

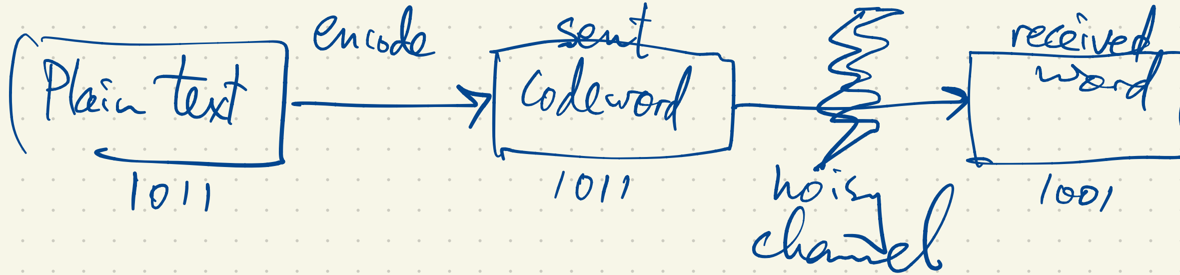
Sadi Carnot

Information is anything representable (usually without loss of information) as strings of letters over a given alphabet of q letters. Strings of letters are words. When $q=2$ we have 2 letters (usually 0, 1) called bits. A <u>code</u> is a scheme for translating words to words.

We are not doing cryptography.

In the theory of error-correcting codes ("coding theory") information is encoded before transmission so that the information can be protected from noise in the channel.

Msg. No.	Message Text	Scheme 1 ("As Is") Codeword
0	**0000**	0000
1	**0001**	0001
2	**0010**	0010
3	**0011**	0011
4	**0100**	0100
5	**0101**	0101
6	**0110**	0110
7	**0111**	0111
8	**1000**	1000
9	**1001**	1001
10	**1010**	1010
11	**1011**	1011
12	**1100**	1100
13	**1101**	1101
14	**1110**	1110
15	**1111**	1111



Msg. No.	Message Text	Scheme 1 ("As Is") Codeword	Scheme 2 (Parity Check) Codeword
0	**0000**	0000	00000
1	**0001**	0001	00011
2	**0010**	0010	00101
3	**0011**	0011	00110
4	**0100**	0100	01001
5	**0101**	0101	01010
6	**0110**	0110	01100
7	**0111**	0111	01111
8	**1000**	1000	10001
9	**1001**	1001	10010
10	**1010**	1010	10100
11	**1011**	1011	10111
12	**1100**	1100	11000
13	**1101**	1101	11011
14	**1110**	1110	11101
15	**1111**	1111	11110

plain text $\boxed{1011} \xrightarrow{\text{encode}}$ coded message $\boxed{10111} \longrightarrow$ received word $\boxed{10011}$

Scheme 2 (parity check code) is an example of a 1-error detecting code. It detects a single bit flip without being able to correct it.

Msg. No.	Message Text	Scheme 1 ("As Is") Codeword	Scheme 2 (Parity Check) Codeword	Scheme 3 (3-Repetition) Codeword
0	**0000**	0000	00000	000000000000
1	**0001**	0001	00011	000000000111
2	**0010**	0010	00101	000000111000
3	**0011**	0011	00110	000000111111
4	**0100**	0100	01001	000111000000
5	**0101**	0101	01010	000111000111
6	**0110**	0110	01100	000111111000
7	**0111**	0111	01111	000111111111
8	**1000**	1000	10001	111000000000
9	**1001**	1001	10010	111000000111
10	**1010**	1010	10100	111000111000
11	**1011**	1011	10111	111000111111
12	**1100**	1100	11000	111111000000
13	**1101**	1101	11011	111111000111
14	**1110**	1110	11101	111111111000
15	**1111**	1111	11110	111111111111

plain text 1011 $\xrightarrow{\text{encode}}$ codeword (sent) 111000111111 $\rightarrow$ noisy channel $\rightarrow$ received word 111000101111

error correction $\downarrow$ 111000111111 $\downarrow$ 1011

This 3-repetition code is a 1-error correcting code.

If at most one bit flip occurs during transmission, we can safely correct it.

This code has a $33\frac{1}{3}\%$ information rate

($\frac{1}{3}$ of the bits transmitted carry actual information; the other $\frac{2}{3}$ of the bits sent are used to provide redundancy for the purpose of error correction).

If one wants to send 4-bit messages and have one-error correcting ability, one can achieve much higher than 33% information rate.

You can achieve 57% information rate.

Table A: Four Schemes for Encoding of 4-bit Message Words

Msg. No.	Message Text	Scheme 1 ("As Is") Codeword	Scheme 2 (Parity Check) Codeword	Scheme 3 (3-Repetition) Codeword	Scheme 4 (Hamming) Codeword
0	**0000**	0000	00000	000000000000	**0000000**
1	**0001**	0001	00011	000000000111	**0001111**
2	**0010**	0010	00101	000000111000	**0010110**
3	**0011**	0011	00110	000000111111	**0011001**
4	**0100**	0100	01001	000111000000	**0100101**
5	**0101**	0101	01010	000111000111	**0101010**
6	**0110**	0110	01100	000111111000	**0110011**
7	**0111**	0111	01111	000111111111	**0111100**
8	**1000**	1000	10001	111000000000	**1000011**
9	**1001**	1001	10010	111000000111	**1001100**
10	**1010**	1010	10100	111000111000	**1010101**
11	**1011**	1011	10111	111000111111	**1011010**
12	**1100**	1100	11000	111111000000	**1100110**
13	**1101**	1101	11011	111111000111	**1101001**
14	**1110**	1110	11101	111111111000	**1110000**
15	**1111**	1111	11110	111111111111	**1111111**

If A is an alphabet of q symbols, a q-ary word is a string of n letters over A. There are q^n words. A code is a subset $\mathcal{C} \subseteq A^n$.

The (Hamming) distance between two words $w, w' \in \mathcal{C}$, denoted $d(w,w')$, is the number of positions in which they differ, eg.

$d(1011, 1110) = 2$.

The Hamming code listed here satisfies $d(w,w') \geqslant 3$ for all $w \neq w'$ in the code. 3 is the minimum distance of the code.

d is a metric:

$d(w,w') \geqslant 0$ for any two words w, w'. Equality iff $w = w'$.

$d(w',w) = d(w,w')$

$d(w,w') + d(w',w'') \geqslant d(w,w'')$ (triangle inequality)

If a code $\mathcal{C} \subseteq A^n$ has min. distance d then it is e-error correcting where $e = \lfloor \frac{d-1}{2} \rfloor$. In particular, in order to correct e errors, we want $d \geqslant 2e+1$.

If we send a word w and due to errors this is received as w' where $d(w,w') \le e$, then w is the unique codeword at distance $\le e$ from w' (assuming $\mathcal{C}$ has min. distance $d \ge 2e+1$).

If $w, w'' \in \mathcal{C}$ were both at distance $\le e$ from w' then $d(w,w'') \le e+e = 2e$

Big question: what is the maximum number $A_q(n,d)$ of codewords in a q-ary code $\mathcal{C} \subseteq A^n$ having a given minimum distance d?

eg. $A_2(7,3) = 16$. The existence of the Hamming code gives $A_2(7,3) \ge 16$.

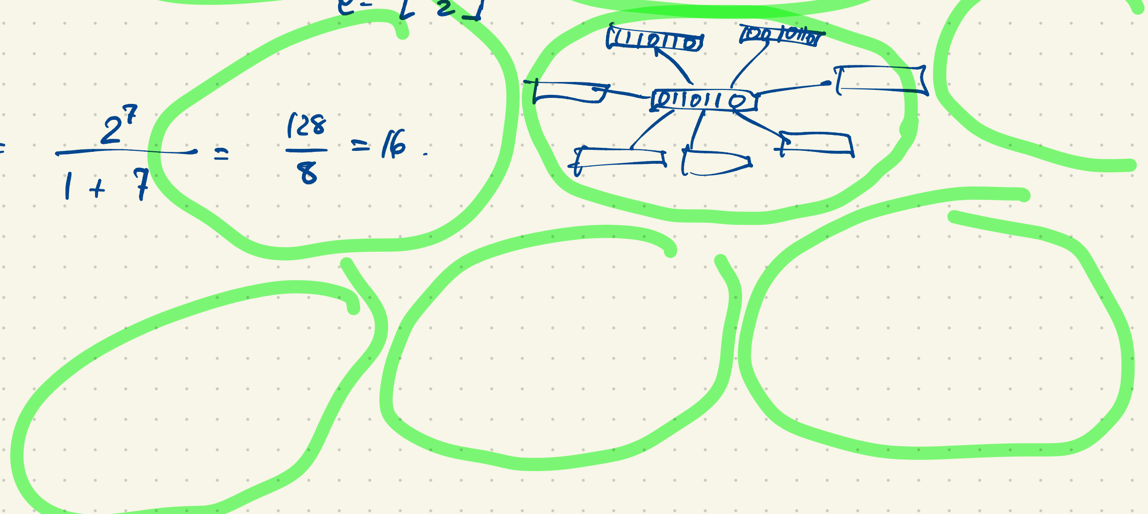
Hamming bound (an upper bound)

(Sphere-packing bound) $A_q(n,d) \le \dfrac{q^n}{\sum_{k=0}^{e} \binom{n}{k}(q-1)^k}$

$e = \lfloor \frac{d-1}{2} \rfloor$

eg. $A_2(7,3) \le \dfrac{2^7}{\sum_{k=0}^{1} \binom{7}{k}} = \dfrac{2^7}{1+7} = \dfrac{128}{8} = 16.$

$e = \lfloor \frac{3-1}{2} \rfloor = 1$



Codes achieving equality in the Hamming bound) are perfect codes.

The binary Hamming codes gives an infinite family of perfect codes

Connection between binary Hamming code and E_8 root lattice

$E_8 = E_8$ root lattice ← $\mathbb{Z}$-module i.e. additive abel. gp.

densest possible packing of uniform balls in $\mathbb{R}^8$

$= \{ v \in \mathbb{Z}^8 : v \bmod 2$ gives a codeword in $\hat{\mathcal{C}} \}$

$v = (v_1, \dots, v_8)$, $v_i \in \mathbb{Z}$

weight of $w = d(w, 0)$ ← 0^n

$\mathcal{C}$ = binary Hamming code $= \{0000000, 0001111, \dots, 1111111\}$, $|\mathcal{C}| = 16$

$\hat{\mathcal{C}}$ = extended ... ... $= \{00000000, 00011110, \dots, 11111111\}$, $|\hat{\mathcal{C}}| = 16$.

$\hat{\mathcal{C}}$ has 1 word of weight 0
14 words 4
1 word 8

Euclidean distance between $v \neq v'$ in E_8 is at least 2.
Equivalently: shortest nonzero vectors in E_8 have min length 2

$(0,0,0,\pm1,\pm1,\pm1,\pm1,0)$ $\quad 16 \cdot 14 = 224$ lattice vectors of length 2
$(0,0,\pm2,0,0,0,0,0)$ $\quad 2 \cdot 8 = 16$ ... ...

240 root vectors in E_8.

Unit balls in $\mathbb{R}^8$ centered at lattice vectors in E_8 lattice achieve the densest possible packing in $\mathbb{R}^8$.

Scheme 4 (Hamming) Codeword
0000000
0001111
0010110
0011001
0100101
0101010
0110011
0111100
1000011
1001100
1010101
1011010
1100110
1101001
1110000
1111111

The generator matrix of this Hamming code is

$$G = \left[\begin{array}{cccc|ccc} 1&0&0&0&0&1&1 \\ 0&1&0&0&1&0&1 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&1&1&1 \end{array}\right]$$

The encoding $\underbrace{x}_{1\times 4} \mapsto \underbrace{x}_{1\times 4}\underbrace{G}_{4\times 7}$ gives the Hamming codeword for each plaintext word x

eg. the plaintext for 11 is $x = 1011 \in F^4$, $F = \{0,1\} = \mathbb{Z}/2\mathbb{Z} = \mathbb{F}_2$ (arithmetic mod 2)

$$xG = [1\,0\,1\,1]\left[\begin{array}{cccc|ccc} 1&0&0&0&0&1&1 \\ 0&1&0&0&1&0&1 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&1&1&1 \end{array}\right] = [1\,0\,1\,1\,0\,1\,0]$$

Hamming encoding can be implemented very efficiently in real time, much faster than "look-up" in a table.

The (parity) check matrix for the Hamming code is

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

If a word $w \in F^7$ is received, we decode by first computing the error syndrome

$$\underbrace{H}_{3\times 7}\underbrace{w^T}_{7\times 1} = \begin{bmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{bmatrix}\begin{bmatrix} 1\\0\\1\\1\\0\\1\\0 \end{bmatrix} = \begin{bmatrix} 0\\0\\0 \end{bmatrix}$$

The Hamming code is the row space of G and it's the null space of H.

The Hamming code is linear over F (a vector space)

If we receive an erroneous word $w' = 1001010$ (having a bit flip in the 3rd coordinate) then its syndrome

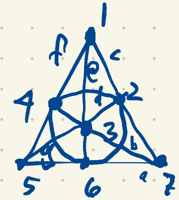
$$H(w')^T = \begin{bmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{bmatrix}\begin{bmatrix} 1\\0\\0\\1\\0\\1\\0 \end{bmatrix} = \begin{bmatrix} 0\\1\\1 \end{bmatrix}$$

3 in binary so $w = \boxed{1011}010$

the original message sent.

Scheme 4 (Hamming) Codeword
0000000
0001111
0010110
0011001
0100101
0101010
0110011
0111100
1000011
1001100
1010101
1011010
1100110
1101001
1110000
1111111

$$
\begin{array}{c}
 & \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\left[\begin{array}{ccccccc}
0&0&1&0&1&1&0\\
0&0&1&1&0&0&1\\
0&1&0&0&1&0&1\\
0&1&0&1&0&1&0\\
1&0&0&0&0&1&1\\
1&0&0&1&1&0&0\\
1&1&1&0&0&0&0
\end{array}\right]
\end{array}
$$



Proj. Plane of order 2

Fixing $k \geq 1$, let H be the $k \times n$ matrix ($n = 2^k - 1$) whose columns are all binary integers in $\{1, 2, \dots, n\}$ (written in binary). This gives a parity check matrix for a perfect 1-error correcting code.

eg. $k=2$ gives $H = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ gives the code $\{000, 111\}$, the 3-repetition code.