Part IIB Lent 1996/YMS

COMMUNICATION THEORY

This handout does **not** *replace the lectures (and is not even mentioned in the list of the recommended literature): it is intended to help understand the material and sometimes to provide additional facts and alternative arguments. Students are strongly advised to take proper notes during the lectures and not rely entirely on the handout.*

The content of a lecture on the blackboard may differ from the one in the handout.

The definitions, statements (theorems, lemmas, corollaries and remarks), formulas and often examples are numbered by pairs of positive integers, the first indicating the lecture and the second the position of a given item in the text of the lecture. In some cases a group of formulas is numbered by triples (e.g., (1.6.1) *or* (3.2a) *). The symbol* □ *is used to indicate the end of the proof.*

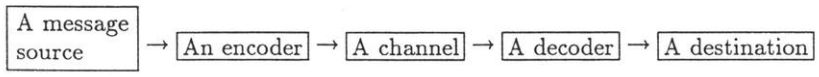
Chapter 1: Information sources and coding

In this chapter, symbol P refers to the probabilities (unconditional or conditional) of samples of sequences of random variables characterising sources of information. As a rule, these are sequences of independent and identically distributed random variables or Markov chains. *Viz.*, $\mathsf{P}(U_1 = u_1, \dots, U_n = u_n)$ is the joint probability that random variables U_1, ..., U_n take values u_1, ..., u_n and $\mathsf{P}(V_n = v \,|\, V_1 = u, W_1 = u')$ is the conditional probability that a random variable V_n takes value v, given that random variables V_1 and W_1 take values u and u', respectively. Likewise, E denotes the expectation with respect to P. On the other hand, in the context of Markov chains, P denotes the transition probability matrix, with entries $P(u, v)$. The r-step transition probabilities correspond to P^r and are denoted by $P^{(r)}(u, v)$.

The symbol p is used to denote various probabilities loosely.

Lecture 1: Basic concepts. The Kraft inequality

A typical scheme used in information transmission is as follows:



Example

1. A message source: a Cambridge college choir.
2. An encoder: a BBC recording unit. It translates the sound to a binary array and writes it to a CD track. The CD is then produced and put on the market.

3. A channel: a customer buying a CD in England and mailing it to Australia. The channel is subject to 'noise': possible damage (mechanical, electrical, chemical, etc.) incurred during transmission.
4. A decoder: a CD player in Australia.
5. A destination: an audience in Australia.
6. The goal: to ensure high-quality sound despite damage.

In fact, a CD can sustain damage done by a needle while making a neat hole in it, or by a tiny drop of acid (you are not encouraged to make such an experiment!)

In technical terms, typical goals of information transmission are:

(a) fast encoding of information,
(b) easy transmission of encoded messages,
(c) effective use of the channel available (i.e. maximum transfer of information per unit time),
(d) fast decoding,
(e) correcting errors (as many as possible) introduced by noise in the channel.

As usual, these goals contradict each other, and one has to find an optimal solution. This is what the course is about. However, do not expect perfect solutions: the theory aims to provide you with knowledge of the basic principles. A final decision is always up to the individual (or group) responsible.

A large part of the course will deal with *encoding* problems. The aims of encoding are:

(1) compressing data to reduce redundant information contained in a message,
(2) protecting text from unauthorised users,
(3) enabling errors to be corrected.

We start by studying *sources* and *encoders*. A source emits a a sequence of letters,

$$u_1 \, u_2 \, \dots \, u_n \, \dots \, , \tag{1.1}$$

where $u_j \in I$, $I(= I_m)$ is an m-element set $\{1, \dots, m\}$ (a source alphabet). In the case of literary English, $m = 26 + 7$, 26 letters plus 7 punctuation symbols: . , : ; – () . (Sometimes one adds ' ' and " ". A telegraph English corresponds to $m = 27$.

A common approach is to consider (1.1) as a *sample* from a random source, i.e. a sequence of random variables

$$U_1, \, U_2, \, \dots, \, U_n, \, \dots \tag{1.2}$$

and try to develop a theory for a reasonable class of such sequences.

Examples. 1.1. The simplest example of a random source is a sequence of independent, identically distributed (i.i.d.) random variables:

$$\mathsf{P}(U_1 = u_1, \, U_2 = u_2, \, \dots, \, U_k = u_k) = \prod_{j=1}^{k} p(u_j), \tag{1.3.1}$$

where $p(u) = \mathsf{P}(U_j = u)$, $u \in I$, is the marginal distribution of a single variable. A random source with i.i.d. symbols is often called a Bernoulli source.

A particular case where $p(u)$ does not depend on $u \in U$ (and hence equals $1/m$) corresponds to the equiprobable Bernoulli source.

1.2. The next example is a Markov source where the symbols form a Markov chain (M.c.):

$$\mathsf{P}(U_1 = u_1,\, U_2 = u_2,\, \dots,\, U_k = u_k) = p_1(u_1) \prod_{j=1}^{k-1} P(u_j, u_{j+1}), \qquad (1.3.2)$$

where $p_1(u) = \mathsf{P}(U_1 = u)$, $u \in I$, is the initial distribution and $P(u, u') = \mathsf{P}(U_{j+1} = u' | U_j = u)$, $u, u' \in I$, are the transition probabilities.

1.3. A 'degenerated' example of a Markov source is where a source emits repeated symbols. Here,

$$\begin{aligned} &\mathsf{P}(U_1 = U_2 = \,\dots\, U_k = u) = q(u), \quad u \in I, \\ &\mathsf{P}(U_k \neq U_{k'}) = 0, \quad 1 \leq k < k', \end{aligned} \qquad (1.3.3)$$

where $0 \leq q(u) \leq 1$ and $\sum\limits_{u \in I} q(u) = 1$. $q(u)$ is the probability of string $uu \,\dots\, u \,\dots$.

An (initial) piece of sequence (1.1)

$$u^{(n)} = u_1\, u_2\, \dots\, u_n$$

is called a (source) sample n-string, or n-word, (with digits from I) and is treated as a 'message'. Correspondingly, one considers a random n-string

$$U^{(n)} = (U_1, U_2, \dots, U_n).$$

An encoder (or coder) uses an alphabet $J (= J_a) = \{0, 1, \dots, a-1\}$; typically $a < m$ (or even $a \ll m$); in many cases $a = 2$ (a binary coder). A *code* (also *coding*, or *encoding*) is a map, f, that takes a symbol $u \in I$ into a finite string, $f(u) = (x_1 \,\dots\, x_s)$, with digits from J. In other words, f maps I into the set J^* of all possible strings:

$$f : I \to J^* = \bigcup_{s \geq 1} (J \times \cdots (s \text{ times}) \times J).$$

Strings $f(u)$ that are images, under f, of symbols $u \in I$ are called *codewords* (in code f). A code has (constant) length N if value s, the length of a codeword, equals N for all codewords. A message $u^{(n)} = u_1\, u_2\, \dots\, u_n$ is represented as a concatenation of codewords

$$f(u^{(n)}) = f(u_1)\, f(u_2)\, \dots\, f(u_n);$$

it is again a string from J^*.

Definition 1.1. A code is called *decipherable* if any string from J^* is the image of at most one message. A string x is a *prefix* in another string y if $y = xz$, i.e. y may be

represented as a result of a concatenation of x and z. A code is *prefix-free* if no codeword is a prefix in any other codeword (e.g. a code of constant length is prefix-free.)

A prefix-free code is decipherable, but not vice versa:

Example 1.4. A code with three source letters $1, 2, 3$ and the binary encoder alphabet $J = \{0, 1\}$ given by

$$f(1) = 0, \quad f(2) = 01, \quad f(3) = 011$$

is decipherable, but not prefix-free.

Theorem 1.1 (The Kraft inequality). *Given positive integers $s_1, \dots, s_m$, there exists a decipherable code $f\colon I \to J^*$, with codewords of lengths $s_1, \dots, s_m$ iff*

$$\sum_{i=1}^{m} a^{-s_i} \leq 1. \qquad (1.4)$$

Furthermore, under condition (1.4) there exists a prefix-free code with codewords of lengths $s_1, \dots, s_m$. □

Proof of Theorem 1.1. (I) Sufficiency. Let (1.4) hold. Your goal is to construct a prefix-free code with codewords of lengths $s_1, \dots, s_m$. Rewrite (1.4) as

$$\sum_{l=1}^{s} n_l a^{-l} \leq 1 \qquad (1.5)$$

where n_l is the number of codewords of length l and $s = \max s_i$. Rewrite (1.4) again:

$$n_s a^{-s} \leq 1 - \sum_{l=1}^{s-1} n_l a^{-l},$$

or, equivalently,

$$n_s \leq a^s - n_1 a^{s-1} - \dots - n_{s-1} a. \qquad (1.6.1)$$

Since $n_s \geq 0$, deduce that

$$n_{s-1} a \leq a^s - n_1 a^{s-1} - \dots - n_{s-2} a^2,$$

or

$$n_{s-1} \leq a^{s-1} - n_1 a^{s-2} - \dots - n_{s-2} a. \qquad (1.6.2)$$

Repeating this argument yields subsequently

$$n_{s-2} \leq a^{s-2} - n_1 a^{s-3} - \dots - n_{s-3} a, \qquad (1.6.3)$$

$$\dots \quad \dots \quad \dots$$

$$n_2 \leq a^2 - n_1 a, \qquad (1.6.s-1)$$

$$n_1 \leq a. \qquad (1.6.s)$$

You can perform the following construction. First choose n_1 words of length 1, using distinct symbols from J: it is possible in view of (1.6.s). This leaves $(a - n_1)$ symbols unused; you can form $(a - n_1)a$ words of length 2 by appending a symbol to each. Choose n_2 codewords from these: you can do so in view of (1.6.s-1). You still have $a^2 - n_1 a - n_2$ words unused: form n_3 codewords. Etc.

In the course of the construction, no new word contains a previous codeword as a prefix. Hence, the code constructed is prefix-free.

(II) Necessity. Suppose there exists a decipherable code in J^* with codeword lengths $s_1, \ldots, s_m$. Set again $s = \max s_i$ and observe that for any positive integer r

$$\left(a^{-s_1} + \ldots + a^{-s_m}\right)^r = \sum_{l=1}^{rs} b_l a^{-l}$$

where b_l is the number of ways r codewords can be put together to form a string of length l.

Because of decipherability, these strings must be distinct. Hence, you must have $b_l \leq a^l$, as a^l is the total number of l-strings. Then

$$\left(a^{-s_1} + \ldots + a^{-s_m}\right)^r \leq rs,$$

and

$$a^{-s_1} + \ldots + a^{-s_m} \leq r^{1/r} s^{1/r} = \exp\left[\frac{1}{r}(\log r + \log s)\right].$$

Since it is true for any r, you can take $r \to \infty$. The RHS goes to 1. □

Remarks. 1.1. *A given code obeying (1.4) is not necessarily decipherable.*

1.2. *Prefix-free codes suffice.*

One of the principal aims of the theory is to find a 'best' decipherable (or even prefix-free) code. We now take a probabilistic point of view and assume that symbol $u \in I$ is emitted by a source with probability $p(u)$:

$$\mathsf{P}(U_k = u) = p(u).$$

[At this point, there is no need to specify a joint probability of more than one subsequently emitted symbol.]

Given a code $f : I \mapsto J^*$, we encode a symbol $i \in I$ by a prescribed codeword $f(i) = x_1 \ldots x_{s_i}$ of length s_i. Thus the codeword becomes a random string from J^*; if the code is decipherable the probability of generating a given string, while encoding a symbol emitted, is precisely $p(i)$ if the string coincides with $f(i)$ and 0 if there is no $i \in I$ with this property. So, the length of a codeword becomes a *random variable*, S, with probability distribution

$$\mathsf{P}(S = s) = \sum_{\substack{1 \leq i \leq m: \\ s_i = s}} p(i).$$

We are looking tor a decipherable code that minimizes the expected word-length:

$$\mathsf{E}\, S = \sum_{s \geq 1} s P(S = s) = \sum_{i=1}^{m} s_i p(i).$$

The following problem therefore arises:

$$\begin{array}{lll} \text{minimize} & f(s_1, \ldots, s_m) = \mathsf{E}\, S & \\ \text{subject to} & \sum_i a^{-s_i} \leq 1 & \text{(Kraft)} \\ \text{and} & s_i \in \mathbf{Z}_+ & \text{(positive integers).} \end{array} \tag{1.7}$$

This is an integer optimization problem. If you drop the condition that $s_1, \ldots, s_m \in \mathbf{Z}_+$, replacing it with a 'relaxed' condition $s_i > 0$, $1 \leq i \leq m$, you could use the Lagrange sufficiency theorem. The Lagrangian is

$$L(s_1, \ldots, s_m, z; \lambda) = \sum_i s_i p(i) + \lambda(1 - \sum_i a^{-s_i} - z)$$

(here, $z \geq 0$ is a slack variable). Minimizing L in $s_1, \ldots, s_m$ and z yields

$$\lambda < 0, \quad z = 0, \quad \text{and} \quad \frac{\partial L}{\partial s_i} = p(i) + a^{-s_i} \lambda \ln a = 0,$$

whence

$$-\frac{p(i)}{\lambda \ln a} = a^{-s_i}, \quad \text{i.e. } s_i = -\log_a p(i) - \log_a(-\lambda \ln a),\ 1 \leq i \leq m.$$

(here and below, $\ln = \log_e$). Adjusting the constraint $\sum_i a^{-s_i} = 1$ $(z = 0!)$ gives

$$\sum_i p(i)/(-\lambda \ln a) = 1 \ \text{ i.e. } \ -\lambda \ln a = 1.$$

Hence,

$$s_i = -\log_a p(i), \quad 1 \leq i \leq m,$$

is an optimal solution, and

$$h_a = -\sum_i p(i) \log_a p(i) = \frac{h}{\log_2 a} \tag{1.8}$$

is the optimal value of the relaxed problem. Here

$$h(= h_2) = -\sum_i p(i) \log_2 p(i) \tag{1.9}$$

is the *binary entropy* of the source. Value h_a is a lower bound for the optimal value in the original problem:

$$\min \mathsf{E}S \geq h_a. \tag{1.10}$$

In future, we use the following convention:

$$\log = \log_2, \quad \text{and for any } b > 1, \ 0 \cdot \log_b 0 = 0 \cdot \log_b \infty = 0.$$

Lecture 2: Huffman encoding

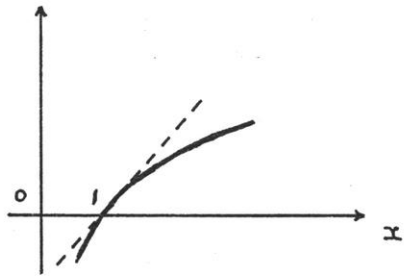
Theorem 2.1. (The Gibbs inequality). *Let $\{p(i)\}$ and $\{p'(i)\}$ be two probability distributions (on I). Then, for any $b > 1$,*

$$\sum_i p(i) \log_b \frac{p'(i)}{p(i)} \leq 0, \quad \text{i.e.} \ -\sum_i p(i) \log_b p(i) \leq -\sum_i p(i) \log_b p'(i), \tag{2.1}$$

and equality is attained iff $p(i) = p'(i)$, $1 \leq i \leq m$.

Proof of Theorem 2.1. The bound

$$\log_b x \leq \frac{x-1}{\log_e b}$$



holds for each $x > 0$, with equality iff $x = 1$. Denoting $I' = \{i : p(i) > 0\}$, we have

$$\sum_i p(i) \log_b \frac{p'(i)}{p(i)} = \sum_{i \in I'} p(i) \log_b \frac{p'(i)}{p(i)} \leq \frac{1}{\log_e b} \sum_{i \in I'} p(i) \left(\frac{p'(i)}{p(i)} - 1 \right)$$

$$= \frac{1}{\log_e b} \left(\sum_{i \in I'} p'(i) - \sum_{i \in I'} p(i) \right) = \frac{1}{\log_e b} \left(\sum_{i \in I'} p'(i) - 1 \right) \leq 0.$$

For equality we need: (a) $\sum_{i \in I'} p'(i) = 1$, i.e. $p'(i) = 0$ when $p(i) = 0$, and (b) $p'(i)/p(i) = 1$ for $i \in I'$. □

Theorem 2.2. (Shannon's noiseless coding theorem). *For a random source emitting symbols with probabilities $p(i) > 0$, the minimal expected codeword length, for a decipherable encoding in an alphabet $J_a = \{0, 1, \ldots, a-1\}$, obeys*

$$\frac{h}{\log a} \leq \min \mathsf{E}S < \frac{h}{\log a} + 1, \tag{2.2}$$

where h is the binary entropy of the source (see (1.9)).

Proof of Theorem 2.2. The LH inequality is established in (1.10). For the RH inequality, let s_i be a positive integer such that

$$a^{-s_i} \leq p_i < a^{-s_i+1}.$$

The non-strict bound here implies $\sum_i a^{-s_i} \leq \sum_i p_i = 1$, i.e. the Kraft inequality. Hence, there exists a decipherable code with codeword lengths $s_1, \ldots, s_m$. From the strict bound you get

$$s_i < -\frac{\log p(i)}{\log a} + 1,$$

and thus

$$\mathsf{E}S < -\frac{\sum_i p(i) \log p(i)}{\log a} + \sum_i p(i) = \frac{h}{\log a} + 1.$$

Example 2.1. Suppose $m = 2^k$ and the letters $i = 1, \ldots, m$ from the source alphabet I_m are equiprobable: $p(i) = 2^{-k}$. Then $h = k$, and hence you need, on average, at least k binary digits for decipherable encoding. Calling a 'bit' a unit of entropy, you can say that you need, on average, at least k bits to encode.

Shannon's NC theorem gives a base for a Shannon–Fano encoding procedure: you fix positive integer word-lengths $s_1, \ldots, s_m$ such that $a^{-s_i} \leq p(i) < a^{-s_i+1}$, or, equivalently,

$$-\log_a p(i) \leq s_i < -\log_a p(i) + 1, \quad \text{i.e.} \ s_i = \lceil -\log_a p(i) \rceil.$$

Then construct a prefix-free code, from the shortest s_i upwards, ensuring that the previous codewords are not prefixes. The Kraft inequality guarantees enough room.

A more elaborated procedure is due to Huffman. Huffman encoding leads to an optimal (i.e. minimal expected length) code f: $I_m \mapsto J_a^*$. Here, we discuss it for $a = 2$ (i.e. $J = \{0, 1\}$) only. The algorithm constructs a binary tree, as follows.

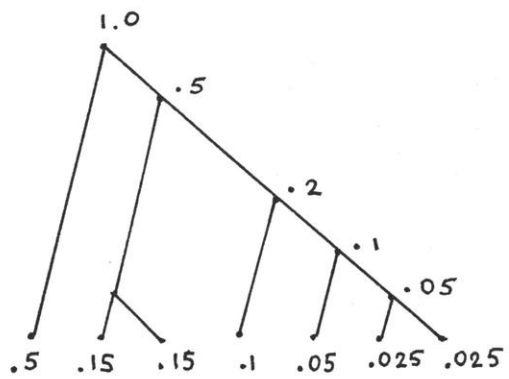
(i) You order the letters $i \in I$ so that $p(1) \geq p(2) \geq \ldots \geq p(m)$.
(ii) Assign symbol 0 to letter $m-1$ and 1 to letter m.
(iii) Construct a reduced alphabet $I_{m-1} = \{1, \ldots, m-2, (m-1, m)\}$, with probabilities

$$p(1), \ldots, p(m-2), p(m-1) + p(m).$$

Repeat steps (i) and (ii) with the reduced alphabet. Etc. You obtain a binary tree.

Example 2.2. $m = 7$.

i	p_i	$f(i)$	s_i
1	.5	0	1
2	.15	100	3
3	.15	101	3
4	.1	110	3
5	.05	1110	4
6	.025	11110	5
7	.025	11111	5



The number of branches you must pass through in order to reach a root i of the tree

equals s_i. The tree-structure, together with the identification of the roots as source letters, guarantees that encoding is prefix-free.

To prove the optimality of Huffman encoding, we need two lemmas.

Lemma 2.3. *Any optimal prefix-free code has the codeword lengths reverse-ordered versus probabilities:*

$$p(i) \geq p(i') \qquad \text{implies } s_i \leq s_{i'}. \tag{2.3}$$

Proof of Lemma 2.3. If not, you can form a new code, by swapping the codewords for i and j. This shortens the expected codeword length and preserves the prefix-free property. □

Lemma 2.4. *In any optimal prefix-free code there exists, among the codewords of maximum length, at least two agreeing in all but the last digit.* □

Proof of Lemma 2.4. If not, then either (i) there exists a single codeword of maximum length, or (ii) there exist two or more codewords of maximum length, and they all differ before the last digit. In both cases you can drop the last digit from all words of maximum length, without affecting the prefix-free property. □

Theorem 2.5. *Huffman encoding is optimal among the prefix-free codes.* □

Proof of Theorem 2.5. Proceed with induction in m. For $m = 2$, the Huffman code f_2 is $f_2(1) = 0$, $f_2(2) = 1$, or vice versa, and is optimal. Assume Huffman code f_{m-1} is optimal for I_{m-1}, whatever the probability distribution. Suppose further that Huffman code f_m is not optimal for I_m for some probability distribution. That is, there exists another prefix-free code, f_m^*, for I_m with shorter expected word-length:

$$\mathsf{E}S_m^* < \mathsf{E}S_m. \tag{2.4}$$

The probability distribution under consideration may be assumed, *wlog*, to obey

$$p(1) \geq \ldots \geq p(m).$$

By Lemmas 2.3 and 2.4, in both codes you can shuffle codewords so that the words corresponding to $m-1$ and m have maximum length and differ only in the last digit. This allows you to reduce both codes to I_{m-1}. Namely, in the Huffman code f_m you remove the final digit from $f_m(m)$ and $f_m(m-1)$, 'glueing' these codewords. This leads to Huffman encoding f_{m-1}, In f_m^* you do the same; this gives you a new prefix-free code f_{m-1}^*.

Observe that in Huffman code f_m the contribution to $\mathsf{E}S_m$ from $f_m(m-1)$ and $f_m(m)$ is $s_m(p(m-1)+p(m))$; after reduction it becomes $(s_m-1)(p(m-1)+p(m))$. That is, $\mathsf{E}S$ is reduced by $p(m-1)+p(m)$.

In code f_m^* the similar contribution is reduced from $s_m^*(p(m-1)+p(m))$ to $(s_m^*-1)(p(m-1)+p(m))$; the difference is again $p(m-1)+p(m)$. All other contributions to

$\mathsf{E}S_{m-1}$ and $\mathsf{E}S_{m-1}^*$ are the same as the corresponding contributions to $\mathsf{E}S_m$ and $\mathsf{E}S_m^*$, respectively.

Therefore, f_{m-1}^* is better than f_{m-1}: $\mathsf{E}S_{m-1}^* < \mathsf{E}S_{m-1}$, which contradicts the assumption. □

In view of Remark 1.2, we obtain

Corollary 2.6. *Huffman encoding is optimal among the decipherable codes.*

In what follows we consider, unless otherwise stated, the case of the binary codes ($a = 2$). A common modern practice is not to encode each letter $u \in I$ separately, but to divide a source message into 'segments' of a fixed length n and encode these as 'letters'. It obviously increases the nominal number of letters in the alphabet: the segments are from the Cartesian product $I^n = I \times \cdots$ (n times) $\times I$. But what matters is the binary entropy of the probability distribution of the segments in a typical message. Denote this entropy by $h^{(n)}$:

$$h^{(n)} = -\sum_{i_1,\ldots,i_n} \mathsf{P}(U_1 = i_1, \ldots, U_n = i_n) \log \mathsf{P}(U_1 = i_1, \ldots, U_n = i_n). \tag{2.5}$$

[Here you obviously need to know the joint distribution of the subsequently emitted source letters.] Denote by $S^{(n)}$ the random codeword length in a segmented code. The minimal expected word-length per source letter is defined by $e_n := \min \frac{1}{n} \mathsf{E}S^{(n)}$ and, by Shannon's NC theorem, it obeys

$$\frac{h^{(n)}}{n \log a} \leq e_n \leq \frac{h^{(n)}}{n \log a} + \frac{1}{n}. \tag{2.6}$$

You see that, for large n, $e_n \sim \dfrac{h^{(n)}}{n \log a}$.

Example 2.3. For a Bernoulli source (see Example 1.1), formula (2.5) yields

$$\begin{aligned} h^{(n)} &= -\sum_{i_1,\ldots,i_n} p(i_1)\cdots p(i_n) \log \Big(p(i_1)\cdots p(i_n)\Big) \\ &= -\sum_{j=1}^{n} \sum_{i_1,\ldots,i_n} p(i_1)\cdots p(i_n) \log p(i_j) = n\, h, \end{aligned} \tag{2.7}$$

and $e_n \sim \dfrac{h}{\log a}$. Thus, for n large, the minimal expected codeword length per source letter, in a segmented code, eventually attains the lower bound in (2.2), and hence is not greater than $\min \mathsf{E}S$, the minimal expected codeword length for letter-by-letter encodings. This phenomenon is much more striking in the case where the subsequent source letters are dependent: here we frequently have $h^{(n)} \ll n\, h$, that is, $e_n \ll \dfrac{h}{\log a}$. This is the gist of data-compression.

Hence, statistics of long strings is an important property of a source. Nominally, the strings $u^{(n)} = u_1 \dots u_n$ of length n 'fill' the Cartesian product $I \times \cdots (n \text{ times}) \times I$; the total number of such strings is m^n, and to encode them all we need $m^n = 2^{n \log\, m}$ distinct codewords. If the codewords have a fixed length (which guarantees the prefix-free property), this length is between $[n \log\, m]$ and $[n \log\, m] + 1$, and the rate of encoding, for large n, is $\sim \log\, m$ bits/source letter. But if some strings are rare, we can disregard them, reducing the number of codewords used (and consequently their length). This leads to the following definitions.

Definition 2.1. A source is said to be (reliably) *encodable at rate* $R > 0$ if, for any n you can find a set $A_n \in I \times \cdots (n \text{ times}) \times I$ such that

$$\sharp\, A_n \leq 2^{nR} \qquad \text{and } \lim_{n\to\infty} \mathsf{P}(U^{(n)} \in A_n) = 1. \tag{2.8}$$

In other words, you can encode messages at rate R with negligible error for long source strings.

Definition 2.2. The *information rate* H of a given source is the infimum of the reliable encoding rates:

$$H = \inf\,[R:\ R \quad \text{is reliable}]. \tag{2.9}$$

Theorem 2.7. *For a source with alphabet* I_m,

$$0 \leq H \leq \log\, m, \tag{2.10}$$

both bounds being attainable. □

Proof of Theorem 2.7. In fact, the LH inequality trivially follows from the definition. The LH equality is attained for a degenerate source (see Example 1.3); here A_n contains $\leq m$ constant strings, which is eventually beaten by 2^{nR} for any $R > 0$). On the other hand, $\sharp\, I^n = m^n = 2^{n \log\, m}$, hence the RH inequality. The RH equality is attained for a source with i.i.d. letters and $p(u) = 1/m$: in this case $P(A_n) = (1/m^n)\, \sharp\, A_n$, which goes to zero when $\sharp\, A_n \leq 2^{nR}$ and $R < \log\, m$.

Example. The telegraph English. As was noted, $m = 27 \simeq 2^{4.76}$, i.e. $H \leq 4.76$. In fact, $H << 4.76$; this enables: (i) data compression, (ii) error-correcting, (iii) code-breaking, (iv) crosswords. The precise value of H for the telegraph English (not mentioning the literary English) is not known: it is a challenging task to assess it accurately. [Those interested are recommended to read Section 6.3 of T.Cover and J.Thomas, *Elements of Information Theory*, Wiley, 1991.] Even more challenging is to compare different languages: which one is more appropriate for intercommunication? [This is certainly a task for future generations.]

We will return to the information rate of a source later and calculate the rates of Bernoulli and Markov sources. Two following lectures are devoted to properties of information and entropy.

Lecture 3: Entropy

Definition 3.1. Given an event A with probability $p(A)$, the *information* gained from the fact that A has occurred is defined as

$$i(A) = -\log\, p(A). \tag{3.1}$$

Let X be a random variable taking a finite number of distinct values $\{x_1, \dots, x_m\}$, with probabilities $p_i = p(x_i)$. The *entropy* $h(X)$ is defined as the expected amount of information gained from observing X:

$$h(X) = -\sum_{x_i} p(x_i) \log\, p(x_i) = -\sum_i p_i \log\, p_i. \tag{3.2}$$

[Here, and below, we assume that $0 \cdot \log\, 0 = 0$, so the sum may be reduced to those x_i's for which $p(x_i) > 0$.] It is clear that the entropy $h(X)$ depends on the probability distribution

$$h(X) = h(p_1, \dots, p_m),$$

but not on the values x_1, ..., x_m.

Given a pair of random variables, X, Y, with values x_i and y_j, the joint entropy $h(X,Y)$ is defined by

$$h(X,Y) = -\sum_{x_i, y_j} p_{X,Y}(x_i, y_j) \log\, p_{X,Y}(x_i, y_j), \tag{3.3}$$

where $p_{X,Y}(x_i, y_j) = p(X = x_i, Y = y_j)$ is the joint probability distribution. In other words, $h(X,Y)$ is the entropy of the random variable (or the random vector) (X,Y) with values (x_i, y_j).

Definition 3.1. The *conditional* entropy, $h(X|Y)$, of a random variable X, given random variable Y, is defined as the expected amount of information gained from observing X given that a value of Y is known:

$$h(X|Y) = -\sum_{x_i, y_j} p_{X,Y}(x_i, y_j) \log\, p_{X|Y}(x_i|y_j). \tag{3.4}$$

Here, $p_{X,Y}(i,j)$ is the joint probability $p(X = x_i, Y = y_j)$ and $p_{X|Y}(x_i|y_j)$ the conditional probability $p(X = x_i | Y = y_j)$. As follows from definitions (3.3) and (3.4),

$$h(X|Y) = h(X,Y) - h(Y). \tag{3.5}$$

Note that in general $h(X|Y) \neq h(Y|X)$.

Sometimes an alternative view is useful: $i(A)$ is an amount of information needed to specify event A, $h(X)$ is the expected amount of information needed to specify random variable X (i.e. the probabilities with which X takes its values), etc.

It is obvious from Definition 3.1 that for independent events A_1 and A_2,

$$i(A_1 \cap A_2) = i(A_1) + i(A_2), \tag{3.6}$$

and for event A with $p(A) = 1/2$,

$$i(A) = 1. \tag{3.7}$$

A justification of definition (3.1) comes from the fact that any function $i^*(A)$, which (i) depend on probability $p(A)$ (i.e., obeys $i^*(A) = i^*(A')$ if $p(A) = p(A')$), (ii) is continuous in $p(A)$, and (iii) satisfies (3.6) and (3.7), coincides with $i(A)$.

Straightforward properties of the entropy are given in Theorem 3.1 below:

Theorem 3.1. (a) *If random variable X takes $\leq m$ values, then*

$$0 \leq h(X) \leq \log\, m; \tag{3.8}$$

the LH equality occurs iff X takes a single value, and the RH equality iff X takes m values with equal probabilities.

(b) $$h(X,Y) \leq h(X) + h(Y), \tag{3.9}$$

with equality iff X and Y are independent.

Proof of Theorem 3.1. Use the Gibbs inequality (Theorem 2.1): (a) with $\{p(i)\}$ being the distribution of X and $p'(i) = 1/m$, $1 \leq i \leq m$; (b) with i being a pair (i_1, i_2) of values of X and Y, $p(i) = p_{X,Y}(i_1, i_2)$ being the joint distribution of X and Y and $p'(i) = p_X(i_1)p_Y(i_2)$, the product of their marginal distributions. Then: a) $h(X) = -\sum_i p(i) \log\, p(i) \leq \sum_i p(i) \log\, m = \log\, m$,

$$\text{b) } h(X,Y) = -\sum_{(i_1,i_2)} p_{X,Y}(i_1,i_2) \log\, p_{X,Y}(i_1,i_2) \leq -\sum_{(i_1,i_2)} p_{X,Y}(i_1,i_2)$$

$$\times \log\, (p_X(i_1)p_Y(i_2)) = -\sum_{i_1} p_X(i_1) \log\, p_X(i_1) - \sum_{i_2} p_Y(i_2) \log\, p_Y(i_2) = h(X) + h(Y);$$

we used here the fact that $\sum_{i_2} p_{X,Y}(i_1,i_2) = p_X(i_1)$, $\sum_{i_1} p_{X,Y}(i_1,i_2) = p_Y(i_2)$. □

Lemma 3.2. (The pooling inequality) *For any q_1, $q_2 \geq 0$, with $q_1 + q_2 > 0$,*

$$-(q_1+q_2)\log\,(q_1+q_2) \leq -q_1 \log\, q_1 - q_2 \log\, q_2 \leq -(q_1+q_2)\log\, \frac{q_1+q_2}{2}; \tag{3.10}$$

the LH equality occurs iff $q_1 q_2 = 0$ (i.e., one of the values q_1, q_2 vanishes), and the RH equality iff $q_1 = q_2$.

Proof of Lemma 3.2. (3.10) is equivalent to $0 \leq h\left(\frac{q_1}{q_1+q_2}, \frac{q_2}{q_1+q_2}\right) \leq \log\, 2\, (= 1)$. □

Lemma 3.2 means that when you 'glue' together values of a random variable you diminish the corresponding contribution to the entropy. On the other hand, if you 'redistribute' the probabilities making them equal you increase the contribution.

An immediate corollary of Lemma 3.2 is the following

Theorem 3.3. *Suppose that a random variable X is a function of random variable Y: $X = \phi(Y)$. Then*

$$h(X) \leq h(Y), \tag{3.11}$$

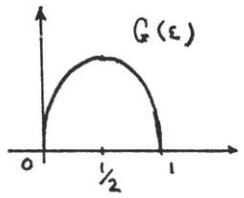
with equality iff ϕ is invertible.

Proof of Theorem 3.3. Indeed, if ϕ is invertible then the probability distributions of X and Y differ only in the order of probabilities, which does not change the entropy. If ϕ 'glues' some values y_j then you can repeatedly use the LH pooling inequality. □

Theorem 3.4. (The Fano inequality) *Suppose a random variable X takes $m > 1$ values, and one of them has probability $(1-\epsilon)$. Then*

$$h(X) \leq G(\epsilon) + \epsilon \log\, (m-1), \tag{3.12}$$

where



$$G(\epsilon) = -\epsilon \log\, \epsilon - (1-\epsilon)\log\,(1-\epsilon). \tag{3.13}$$

Proof of Theorem 3.4. Suppose that $p_1 = p(x_1) = 1-\epsilon$. Then

$$h(X) = h(p_1, \ldots, p_m) = -\sum_{i=1}^{m} p_i \log\, p_i = -p_1 \log\, p_1 - (1-p_1)\log\,(1-p_1)$$

$$+(1-p_1)\log\,(1-p_1) - \sum_{i=2}^{m} p_i \log\, p_i = h(p_1, 1-p_1) + (1-p_1)h\left(\frac{p_2}{1-p_1}, \ldots, \frac{p_m}{1-p_1}\right);$$

in the RHS the first term is $G(\epsilon)$ and the second does not exceed $\epsilon \log\,(m-1)$. □

The Fano inequality shows how the entropy $h(X)$ grows when X is 'near' a constant random variable.

Definition 3.2. Given a triple of random variables, X, Y and Z, we say that X and Y are conditionally independent, given Z, if

$$p(X = x, Y = y | Z = z) = p(X = x | Z = z) p(Y = y | Z = z) \tag{3.14}$$

for any z with $p(Z = z) > 0$ and all x and y.

For the conditional entropy you immediately obtain

Theorem 3.5. (a) $$0 \leq h(X|Y) \leq h(X); \tag{3.15}$$

the LH equality occurs iff X is a function of Y and the RH equality iff X and Y are independent,

(b) $$h(X|Y,Z) \leq h(X|Y) \leq h(X|\phi(Y)); \tag{3.16}$$

the LH equality occurs iff X and Z are conditionally independent given Y and the RH equality iff X and Y are conditionally independent given $\phi(Y)$.

Proof of Theorem 3.5. (a) The LH bound in (3.15) follows from the definition (see (3.4)). The RH bound follows from (3.5.) and (3.9). The LH equality in (3.15) is equivalent to $h(X,Y) = h(Y)$; the last equality occurs iff, with probability one, the map $(X,Y) \mapsto Y$ is invertible which means that X is a function of Y. The RH equality in (3.15) occurs iff $h(X,Y) = h(X) + h(Y)$, i.e. X and Y are independent.

(b) For the LH bound, use a formula analogous to (3.5):

$$h(X|Y,Z) = h(X,Z|Y) - h(Z|Y) \tag{3.17}$$

and an inequality analogous to (3.9):

$$h(X,Z|Y) \leq h(X|Y) + h(Z|Y), \tag{3.18}$$

with equality iff X and Z are conditionally independent given Y. For the RH bound, use (i) a formula that is a particular case of (3.17): $h(X|Y,\phi(Y)) = h(X,Y|\phi(Y)) - h(Y|\phi(Y))$, together with the remark that $h(X|Y,\phi(Y)) = h(X|Y)$, and (ii) an inequality which is a particular case of (3.18): $h(X,Y|\phi(Y)) \leq h(X|\phi(Y)) + h(Y|\phi(Y))$, with equality iff X and Y are conditionally independent given $\phi(Y)$. □

Theorem 3.6 below shows how the conditional entropy $h(X|Y)$ grows when X is 'nearly' a function of Y.

Theorem 3.6. (The generalized Fano inequality) *For a pair of random variables, X and Y, with values $x_1, ..., x_m$ and $y_1, ..., y_m$ if*

$$\sum_{j=1}^{m} p(X = x_j, Y = y_j) = 1 - \epsilon, \tag{3.19}$$

then

$$h(X|Y) \leq G(\epsilon) + \epsilon \log (m-1), \tag{3.20}$$

where $G(\epsilon)$ is defined in (3.13).

Proof of Theorem 3.6. Denoting $\epsilon_j = p(X \neq x_j | Y = y_j)$, you can write

$$\sum_j p_Y(y_j)\epsilon_j = \sum_j p(X \neq x_j, Y = y_j) = \epsilon.$$

By definition of the conditional entropy, the Fano inequality, and concavity of function $G(\epsilon)$,

$$h(X|Y) \leq \sum_j p_Y(y_j)\Big(G(\epsilon_j) + \epsilon_j \log (m-1)\Big)$$

$$\leqslant \sum_j p_Y(y_j) G(\epsilon_j) + \epsilon \log (m-1) \leqslant G(\epsilon) + \epsilon \log (m-1).$$

□

The most part of the properties listed are extended to the case of random vectors.

Theorem 3.7. *For a pair of random vectors, $X^{(n)} = (X_1, ..., X_n)$ and $Y^{(n)} = (Y_1, ..., Y_n)$,*

(a) $$h(X^{(n)}) = \sum_{i=1}^{n} h(X_i|X^{(i-1)}) \leq \sum_{i=1}^{n} h(X_i), \tag{3.21}$$

with equality iff $X_1, ..., X_n$ are independent, and

(b) $$h(X^{(n)}|Y^{(n)}) \leq \sum_{i=1}^{n} h(X_i|Y^{(n)}) \leq \sum_{i=1}^{n} h(X_i|Y_i), \tag{3.22}$$

with the LH equality iff $X_1, ..., X_n$ are conditionally independent, given $Y^{(n)}$, and the RH equality iff, for each $i = 1, ..., n$, X_i and $\{Y_r : 1 \leq r \leq n, r \neq i\}$ are conditionally independent, given Y_i.

The proof repeats the previously used arguments.

Definition 3.3. The *mutual entropy* between X and Y is defined as

$$i(X,Y) := \mathsf{E} \log \frac{p_{X,Y}(X,Y)}{p_X(X)p_Y(Y)} = \sum_{x,y} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} = h(X) + h(Y)$$

$$-h(X,Y) = \Big(= i(Y,X)\Big) = h(X) - h(X|Y) = h(Y) - h(Y|X). \tag{3.23}$$

In other words, $i(X,Y)$ measures the amount of information about X conveyed by Y (and vice versa). An immediate corollary of Theorems 3.1(b) and 3.5(b) is the following

Theorem 3.8. $0 \leq i(X,\phi(Y)) \leq i(X,Y)$; (3.24)

the LH equality occurs iff X and $\phi(Y)$ are independent, and the RH equality iff X and Y are conditionally independent, given $\phi(Y)$.

Note that X and Y in Definition 3.3 and Theorem 3.8 may be random vectors. In addition, for a pair of random vectors, $X^{(n)}$ and $Y^{(n)}$, you obtain, from Theorem 3.8,

Theorem 3.9. (a) $$\begin{aligned} i(X^{(n)},Y^{(n)}) &\geq h(X^{(n)}) - \sum_{i=1}^{n} h(X_i|Y^{(n)}) \\ &\geq h(X^{(n)}) - \sum_{i=1}^{n} h(X_i|Y_i), \end{aligned} \tag{3.25}$$

(b) *if $X_1, ..., X_n$ are independent,*

$$i(X^{(n)},Y^{(n)}) \geq \sum_{i=1}^{n} i(X_i, Y^{(n)}). \tag{3.26}$$

Observe that the RHS of (3.26) is always

$$\geq \sum_{i=1}^{n} i(X_i, Y_i). \tag{3.27}$$

Lecture 4: Shannon's First coding theorem

Definition 4.1.

$$D_n(R) := \max_{\substack{A \subset I^n : \\ \sharp\, A \le 2^{nR}}} \mathsf{P}(U^{(n)} \in A). \tag{4.1}$$

Here and below we set: $I^n := I \times \cdots (n \text{ times}) \times I$.

Lemma 4.1. *For any $\epsilon > 0$,*

$$\lim_{n\to\infty} D_n(H+\epsilon) = 1, \text{ and, if } H > 0,\ D_n(H-\epsilon) \not\to 1. \tag{4.2}$$

Proof of Lemma 4.1. $R := H + \epsilon$ is, by definition, a reliable encoding rate. Hence, there exists a sequence of sets $A_n \subset I^n$, with $\sharp\, A_n \le 2^{nR}$ and $\mathsf{P}(U^{(n)} \in A_n) \to 1$, as $n \to \infty$. Since $D_n(R) \ge \mathsf{P}(U^{(n)} \in A_n)$, then $D_n(R) \to 1$.

Now let be $H > 0$. $R := H - \epsilon$ is > 0 for ϵ small enough, but it is not a reliable rate. That is, there is no sequence A_n with the above properties. Take a set C_n where the maximum in (4.1) is attained. Then $\sharp\, C_n \le 2^{nR}$, but $\mathsf{P}(C_n) \not\to 1$. □

Given a string $u^{(n)} = u_1 \dots u_n$, consider its 'log-likelihood' value per source letter:

$$\xi_n(u^{(n)}) = -\frac{1}{n} \log_+ p_n(u^{(n)}), \tag{4.3a}$$

where $p_n(u^{(n)})$ is the probability assigned to string $u^{(n)}$. Here, and below, $\log_+ x = \log x$, if $x > 0$, and $= 0$, if $x = 0$. For a random string, $U^{(n)}$,

$$\xi_n(U^{(n)}) = -1/n\ \log_+ p_n(U^{(n)}) \tag{4.3b}$$

is a random variable.

Lemma 4.2. *For any R, $\epsilon > 0$,*

$$\mathsf{P}(\xi_n \le R) \le D_n(R) \le \mathsf{P}(\xi_n \le R+\epsilon) + 2^{-n\epsilon}. \tag{4.4}$$

□

Proof of Lemma 4.2. For brevity, omit upper index (n) in the notation $u^{(n)}$ and $U^{(n)}$. Set

$$B_n := \{u \in I^n : p_n(u) \ge 2^{-nR}\} = \{u \in I^n : -\log\, p_n(u) \le nR\} = \{u \in I^n : \xi_n(u) \le R\}.$$

Then $1 \ge \mathsf{P}(U \in B_n) = \sum_{u \in B_n} p(u) \ge 2^{-nR}\, \sharp\, B_n$, whence $\sharp\, B_n \le 2^{nR}$. Thus,

$$D_n(R) = \sup_{\substack{A_n \subseteq I^n : \\ \sharp\, A \le 2^{nR}}} \mathsf{P}(U \in A_n) \ge \mathsf{P}(U \in B_n) = \mathsf{P}(\xi_n \le R),$$

which proves the LH in (4.4).

On the other hand, there exists $C_n \subseteq I^n$ where the maximum in (4.1) is attained. For such a C_n,

$$\begin{aligned} D_n(R) &= \mathsf{P}(U \in C_n) = \mathsf{P}(U \in C_n,\ \xi_n \le R+\epsilon) + \mathsf{P}(U \in C_n,\ \xi_n > R+\epsilon) \\ &\le \mathsf{P}(\xi_n \le R+\epsilon) + \sum_{\substack{u \in C_n : \\ p_n(u) < 2^{-n(R+\epsilon)}}} p_n(u) \\ &< \mathsf{P}(\xi_n \le R+\epsilon) + 2^{-n(R+\epsilon)}\, \sharp\, C_n \le \mathsf{P}(\xi_n \le R+\epsilon) \\ &+ 2^{-n(R+\epsilon)}\, 2^{nR} = \mathsf{P}(\xi_n \le R+\epsilon) + 2^{-n\epsilon}. \end{aligned}$$

□

Definition 4.2. A sequence of random variables $\{\eta_n\}$ converges in probability to a constant r if for any $\epsilon > 0$,

$$\lim_{n\to\infty} \mathsf{P}\Big(|\eta_n - r| \ge \epsilon\Big) = 0 \tag{4.5}$$

Replacing, in this definition, r by a random variable η, you obtain a more general definition of convergence in probability to a random variable.

Convergence in probability is denoted in the sequel as $\eta_n \xrightarrow{\mathsf{P}} r$ (respectively, $\eta_n \xrightarrow{\mathsf{P}} \eta$).

Remark. It is precisely the convergence in probability (to an expectation value) that figures in the so-called law of large numbers (see below).

Theorem 4.3. (Shannon's First Coding Theorem) *If ξ_n converges in probability to a constant γ then $\gamma = H$, the information rate of a source.*

Proof of Theorem 4.3. Let $\xi_n \xrightarrow{\mathsf{P}} \gamma$. Since $\xi_n \ge 0$, $\gamma \ge 0$. By Lemma 4.2, for any $\epsilon > 0$,

$$\begin{aligned} D_n(\gamma+\epsilon) &\ge \mathsf{P}(\xi_n \le \gamma+\epsilon) \ge \mathsf{P}(\gamma-\epsilon \le \xi_n \le \gamma+\epsilon) \\ &= \mathsf{P}\Big(|\xi_n - \gamma| \le \epsilon\Big) = 1 - \mathsf{P}\Big(|\xi_n - \gamma| > \epsilon\Big) \to 1\ (n \to \infty). \end{aligned}$$

Hence, $H \le \gamma$. In particular, if $\gamma = 0$ then $H = 0$. If $\gamma > 0$, you have, again by Lemma 4.2,

$$D_n(\gamma-\epsilon) \le \mathsf{P}(\xi_n \le \gamma - \epsilon/2) + 2^{-n\epsilon/2} \le \mathsf{P}\Big(|\xi_n - \gamma| \ge \epsilon/2\Big) + 2^{-n\epsilon/2} \to 0.$$

By Lemma 4.1, $H \ge \gamma$. Hence, $H = \gamma$. □

Remarks. 1. Convergence $\xi_n \xrightarrow{\mathsf{P}} \gamma$ is equivalent to the following *asymptotic equipartition* property (AEP): for any $\epsilon > 0$,

$$\lim_{n\to\infty} \mathsf{P}\Big(2^{-n(H+\epsilon)} \le p_n(U^{(n)}) \le 2^{-n(H-\epsilon)}\Big) = 1. \tag{4.6}$$

In fact,

$$\begin{aligned}&\mathsf{P}\Big(2^{-n(H+\epsilon)} \le p_n(U^{(n)}) \le 2^{-n(H-\epsilon)}\Big)\\ &= \mathsf{P}\Big(H-\epsilon \le -\frac{1}{n}\log\, p_n(U^{(n)}) \le H+\epsilon\Big)\\ &= \mathsf{P}\big(|\xi_n - H| \le \epsilon\big) \;=\; 1-\mathsf{P}\big(|\xi_n - H| > \epsilon\big).\end{aligned}$$

In other words, for any $\epsilon > 0$ there exists $n_0 = n_0(\epsilon)$ such that, for any $n > n_0$, the set I^n decomposes into disjoint subsets, Π_n and T_n, with a) $\mathsf{P}\big(U^{(n)} \in \Pi_n\big) < \epsilon$, b) $2^{-n(H+\epsilon)} \le \mathsf{P}\big(U^{(n)} = u^{(n)}\big) \le 2^{-n(H-\epsilon)}$ for any $u^{(n)} \in T_n$.

Pictorially speaking, T_n is a set of 'typical' strings and Π_n the residual set. You may conclude that, for a source with AEP, it is worthwhile to encode the typical strings with codewords of the same length, and the rest anyhow. You will then have the effective encoding rate $H + o(1)$ bits/source-letter, though the source emits $\log m$ bits/source-letter.

2. Observe that

$$\mathsf{E}\xi_n = -\frac{1}{n}\sum_{u\in I^n} p_n(u)\log\, p_n(u) = \frac{1}{n}h^{(n)}, \tag{4.7}$$

(see (2.5)).

Consider now the simplest examples of sources. As before, start with a Bernoulli source.

Theorem 4.4. *For a Bernoulli source* $H = h$. □

Proof of Theorem 4.4. For an i.i.d. sequence $U_1, U_2, \ldots$, $p_n(u^{(n)}) = \prod_{i=1}^{n} p(u_i)$, $u^{(n)} = u_1 \ldots u_n$. Hence, $-\log p_n(u) = \sum_i -\log p(u_i)$. For a random string $U^{(n)} = (U_1, \ldots, U_n)$, $-\log\, p_n(U^{(n)}) = -\sum_{i=1}^{n} \log p(U_i)$, where random variables $-\log\, p(U_1)$, ..., $-\log p(U_n)$ are mutually independent and have the same distribution.

Denoting $\sigma_i = -\log p(U_i)$, $i = 1, 2, \ldots$, you can say that $\sigma_1, \sigma_2, \ldots$ form a sequence of i.i.d. random variables, and write $\xi_n = \frac{1}{n}\sum_{i=1}^{n}\sigma_i$. Observe that $\mathsf{E}\sigma_i = -\sum_j p(j)\log\, p(j)$ $= h$ and $\mathsf{E}\xi_n = \mathsf{E}\frac{1}{n}\sum_{i=1}^{n}\sigma_i = \frac{1}{n}\sum_{i=1}^{n}\mathsf{E}\sigma_i = \frac{1}{n}\sum_{i=1}^{n} h = h$; the last equality is in agreement with (4.7), since, for the Bernoulli source, $h^{(n)} = nh$ (see (2.7)), and hence $\mathsf{E}\xi_n = h$.

You immediately see that $\xi_n \xrightarrow{\mathsf{P}} h$ is a direct corollary of Theorem 4.5.

Theorem 4.5. (The Law of large number for i.i.d. random variables) *For any sequence of i.i.d. random variables* $\zeta_1, \zeta_2, \ldots$ *with* $\mathsf{E}\zeta_i = b$, *and for any* $\epsilon > 0$,

$$\lim_{n\to\infty} \mathsf{P}\Big(\Big|\frac{1}{n}\sum_{i=1}^{n}\zeta_i - b\Big| \ge \epsilon\Big) = 0. \tag{4.9}$$

The proof of Theorem 4.5 is based on a famous inequality:

Lemma 4.6. (The Chebyshov inequality) *For any random variable* η *and any* $\epsilon > 0$, $\mathsf{P}(\eta \ge \epsilon) \le \frac{1}{\epsilon^2}\mathsf{E}\eta^2$.

Proof of Lemma 4.6. $\mathsf{P}(\eta \ge \epsilon) = \mathsf{E}\mathbf{1}(\eta \ge \epsilon) \le \mathsf{E}(\eta/\epsilon)^2\mathbf{1}(\eta \ge \epsilon) \le (1/\epsilon)^2\mathsf{E}\eta^2$. □

Proof of Theorem 4.5. The Chebyshov inequality, applied to the LHS of (4.9), gives

$$\mathsf{P}\Big(\Big|\frac{1}{n}\sum_{i=1}^{n}\zeta_i - b\Big| \ge \epsilon\Big) \le \frac{1}{\epsilon^2}\mathsf{E}\left(\frac{1}{n}\sum_{i=1}^{n}\sigma_i - b\right)^2 = \frac{1}{\epsilon^2 n^2}\mathsf{E}\left(\sum_{i=1}^{n}(\zeta_i - b)\right)^2. \tag{4.10}$$

Now, the sum $\sum_{i=1}^{n}(\zeta_i - b)$ has the expectation value zero: $\mathsf{E}\sum_{i=1}^{n}(\zeta_i - b) = \sum_{i=1}^{n}\mathsf{E}(\zeta_i - b) =$ $\sum_{i=1}^{n}\mathsf{E}(b - b) = 0$. Hence, the quantity $\mathsf{E}\Big(\sum_{i=1}^{n}(\zeta_i - b)\Big)^2$ in the RHS of (4.10) is nothing but the variance $\operatorname{Var}\sum_{i=1}^{n}(\zeta_i - b)$. [Recall, the variance $\operatorname{Var}\eta = \mathsf{E}(\eta - \mathsf{E}\eta)^2 = \mathsf{E}\eta^2 -$ $(\mathsf{E}\eta)^2$, which gives $\operatorname{Var}\eta = \mathsf{E}\eta^2$ when $\mathsf{E}\eta = 0$.] But random variables $\zeta_1 - h$, $\zeta_2 - h$, ... are i.i.d. (they are obtained from i.i.d. random variables $\zeta_1, \zeta_2, \ldots$ by extracting a constant), hence $\operatorname{Var}\sum_{i=1}^{n}(\zeta_i - b) = \sum_{i=1}^{n}\operatorname{Var}(\zeta_i - b) = n\operatorname{Var}(\zeta_1 - b) = n\operatorname{Var}\zeta_1$; the last equality is due to the fact that adding a constant does not change the variance.

Thus, the RHS of (4.10) equals $\frac{n}{n^2\epsilon^2}\operatorname{Var}\zeta_1 = \frac{1}{n\epsilon^2}\operatorname{Var}\zeta_1$, which goes to 0 when $n \to \infty$. Theorem 4.5 is proved, and so is Theorem 4.4. □

Lectures 5 and 6: The entropy rate of a Markov source

We now consider a Markov source $U_1, U_2, \ldots$ with letters from alphabet $I = \{1, \ldots, m\}$ and assume that the transition probability matrix $\big(P(u,u')\big)$ (or rather its power) obeys

$$\min_{u,u'} P^{(r)}(u,u') = \rho > 0 \text{ for some } r \geq 1. \tag{5.1}$$

This condition means that the M.c. is irreducible and aperiodic. As one knows from the part IB Markov Chains course, the chain then has a unique invariant (equilibrium) distribution $w(1), \ldots, w(m)$:

$$0 \leq w(u) \leq 1, \ \sum_{u=1}^{m} w(u) = 1, \ \ w(v) = \sum_{u=1}^{m} w(u)P(u,v), \tag{5.2}$$

and the n-step transition probabilities $P^{(n)}(u,v)$ converge to $w(v)$ as well as as well as the probabilities $(p_1 P^{n-1})(v) = \mathsf{P}(U_n = v)$:

$$\lim_{n\to\infty} P^{(n)}(u,v) = \lim_{n\to\infty} \mathsf{P}(U_n = v) = \lim_{n\to\infty} \sum_{u} p_1(u)P^{(n)}(u,v) = w(v). \tag{5.3}$$

the last relation holds for any initial distribution $\{p_1(u),\ u \in I\}$. Moreover, the convergence in (5.3) is exponentially fast (see Theorem 5.1 below).

Pictorially speaking, the chain 'forgets' its initial state and 'rapidly' reaches a stationary regime described by $(w(1), \ldots, w(m))$.

Theorem 5.1. *Assume that condition (5.1) holds with $r = 1$. Then the M.c. $U_1, U_2, \ldots$ possesses a unique invariant distribution (5.2), and for any $u, v \in I$ and any initial distribution p_1 on I,*

$$|P^{(n)}(u,v) - w(v)| \leq (1-\rho)^n \text{ and } |\mathsf{P}(U^{(n)} = v) - w(v)| \leq (1-\rho)^{n-1}. \tag{5.4}$$

In the case of a general $r \geq 1$, one should replace, in the RHS of (5.4), $(1-\rho)^n$ by $(1-\rho)^{[n/r]}$ and $(1-\rho)^{n-1}$ by $(1-\rho)^{[(n-1)/r]}$.

The proof of Theorem 5.1 is not a part of the examinable material for the present course. However, for the sake of completeness it is given in the appendix.

The information rate H of a Markov source is given in the following Theorem 5.2.

Theorem 5.2. *For a Markov source, under condition (5.1),*

$$H = -\sum_{1\leq u,v\leq m} w(u)P(u,v)\log P(u,v) = \lim_{n\to\infty} h(U_{n+1}|U_n). \tag{5.5}$$

Proof of Theorem 5.2. We again use the Shannon FCT, aiming to check that $\xi_n \xrightarrow{\mathsf{P}} H$ where H is given by (5.5) and the random variable ξ_n is given by

$$\xi_n = -\frac{1}{n}\log p_n(U^{(n)}),$$

cf. (4.3b). In other words, we are going to prove the AEP for a Markov source under condition (5.1).

The Markov property means that, for any string $u^{(n)} = u_1 \ldots u_n$,

$$p_n(u^{(n)}) = p(u_1)P(u_1,u_2)\cdots P(u_{n-1},u_n), \tag{5.6a}$$

and

$$-\log p_n(u^{(n)}) = -\log w(u_1) - \log P(u_1,u_2) - \ldots - \log P(u_{n-1},u_n). \tag{5.6b}$$

For a random string, $U^{(n)}$,

$$-\log p_n(U^{(n)}) = -\log w(U_1) - \log P(U_1,U_2) - \ldots - \log P(U_{n-1},U_n), \tag{5.7}$$

As in the case of Bernoulli source, we denote

$$\sigma_1(U_1) := -\log p_1(U_1), \ \ \sigma_i(U_{i-1},U_i) := -\log P(U_{i-1},U_i), \ \ i = 2,3,\ldots, \tag{5.8}$$

and write

$$\xi_n = \frac{1}{n}\Big(\sigma_1 + \sum_{i=1}^{n-1}\sigma_{i+1}\Big); \tag{5.9}$$

Observe that the σ's have the expectation values

$$\mathsf{E}\sigma_1 = -\sum_{u} p_1(u)\log p_1(u) \tag{5.10a}$$

and

$$\begin{aligned}\mathsf{E}\sigma_{i+1} &= -\sum_{u,u'} \mathsf{P}(U_i = u, U_{i+1} = u')\log P(u,u') \\ &= -\sum_{u,u'} (p_1 P^{i-1})(u)P(u,u')\log P(u,u'), \ \ i \geq 1;\end{aligned} \tag{5.10b}$$

as follows from Theorem 5.1, $\lim_{i\to\infty} \mathsf{E}\sigma_i = H$. Hence, $\lim_{n\to\infty} \mathsf{E}\xi_n = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathsf{E}\sigma_i = H$;

this is a clear indication that convergence $\xi_n \xrightarrow{\mathsf{P}} H$ is again a Law of large numbers, for the sequence $\sigma_1, \sigma_2, \ldots$:

$$\lim_{n\to\infty} \mathsf{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}\sigma_i - H\right| \geq \epsilon\right) = 0. \tag{5.11}$$

However, the situation here is not as simple as in the case of a Bernoulli source. There are two difficulties to overcome: (i) $\mathsf{E}\sigma_i$ equals H only in the limit $i \to \infty$, (ii) $\sigma_1, \sigma_2, \ldots$ are no longer independent. Even worse, they do not form a M.c., or even a Markov chain of a higher order. [A sequence $\zeta_1, \zeta_2, \ldots$ is said to form a M.c. of order k, if, for any $n \geq 1$,

$$\mathsf{P}(U_{n+k+1} = u' | U_{n+k} = u_k, \ldots, U_{n+1} = u_1, \ldots)$$

$$= \mathsf{P}(U_{n+k+1} = u' | U_{n+k} = u_k, \ldots, U_{n+1} = u_1).$$

An obvious remark is that, in a M.c. of order k, the vectors $\bar{U}_n = (U_n, U_{n+1}, \ldots, U_{n+k-1})$, $n \geq 1$, form an ordinary M.c.] In a sense, the 'memory' in sequence $\sigma_1, \sigma_2, \ldots$ is infinitely long. However, it decays exponentially: the precise meaning of this is provided in Theorem 5.1.

Anyway, if you use again the Chebyshov inequality, you obtain

$$\mathsf{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}\sigma_i - H\right| \geq \epsilon\right) \leq \frac{1}{n^2\epsilon^2}\mathsf{E}\left(\sum_{i=1}^{n}(\sigma_i - H)\right)^2. \tag{5.12}$$

Theorem 5.2 immediately follows from Lemma 5.3 below:

Lemma 5.3. *The expectation value in the RHS of (5.8) satisfies the bound*

$$\mathsf{E}\left(\sum_{i=1}^{n}(\sigma_i - H)\right)^2 \leq C\,n, \tag{5.13}$$

where $C > 0$ is a constant that does not depend on n.

In fact, the RHS of (5.12) becomes $\leq \dfrac{C}{n\epsilon^2}$ and goes to zero as $n \to \infty$.

The proof of Lemma 5.3 is again given in the appendix.

Appendix: The proof of Theorem 5.1 and Lemma 5.3

Proof of Theorem 5.1 First note that the first bound $|P^{(n)}(u,v) - w(v)| \leq (1-\rho)^n$ in (5.3) implies all other assertions of the theorem. In fact, assume that this bound holds. Then the second one also holds, and $0 \leq w(u) \leq 1$ and $\sum_{u=1}^{m} w(u) = 1$. Furthermore,

$$w(v) = \lim_{n\to\infty} P^{(n)}(u,v) = \lim_{n\to\infty}\sum_{\widetilde{u}} P^{(n-1)}(u,\widetilde{u})P(\widetilde{u},v) = \sum_{\widetilde{u}} w(\widetilde{u})P(\widetilde{u},v), \tag{A.1}$$

which yields (5.2). If $w'(1), w'(2), \ldots, w'(m)$ is another invariant probability vector, i.e.,

$$0 \leq w'(u) \leq 1, \ \sum_{u=1}^{m} w'(u) = 1, \ \ w'(v) = \sum_{u} w'(u)P(u,v),$$

then $w'(v) = \sum_u w'(u)P^{(n)}(u,v)$ for any $n \geq 1$. The limit $n \to \infty$ gives then

$$w'(v) = \sum_{u} w'(u) \lim_{n\to\infty} P^{(n)}(u,v) = \sum_{u} w'(u)w(v) = w(v),$$

which means that the invariant probability vector is unique.

Hence, our task is to prove the first bound in (5.3). Denote

$$m_n(v) = \min_u P^{(n)}(u,v), \quad M_n(v) = \max_u P^{(n)}(u,v). \tag{A.2}$$

Then

$$m_{n+1}(v) = \min_u P^{(n+1)}(u,v) = \min_u \sum_{\widetilde{u}} P(u,\widetilde{u})P^{(n)}(\widetilde{u},v)$$
$$\geq \min_u P^{(n)}(u,v)\sum_{\widetilde{u}} P(u,\widetilde{u}) = m_n(v).$$

Similarly,

$$M_{n+1}(v) = \max_u P^{(n+1)}(u,v) = \max_u \sum_{\widetilde{u}} P(u,\widetilde{u})P^{(n)}(\widetilde{u},v)$$
$$\leq \max_u P^{(n)}(u,v)\sum_{\widetilde{u}} P(u,\widetilde{u}) = M_n(v).$$

Since $0 \leq m_n(v) \leq M_n(v) \leq 1$, both $m_n(v)$ and $M_n(v)$ have the limits

$$m(v) = \lim_{n\to\infty} m_n(v) \leq \lim_{n\to\infty} M_n(v) = M(v).$$

Furthermore,

$$M(v) - m(v) = \lim_{n\to\infty}(M_n(v) - m_n(v)) = \lim_{n\to\infty}\max_{u,u'}(P^{(n)}(u,v) - P^{(n)}(u',v)).$$

So, if we manage to prove that

$$\max_{u,u',v}(P^{(n)}(u,v) - P^{(n)}(u',v)) \leq (1-\delta)^n, \tag{A.3}$$

we will have $M(v) = m(v)$ for each v. Furthermore, denoting the common value $M(v) = m(v)$ by $w(v)$, we will have

$$|P^{(n)}(u,v) - w(v)| \leq M_n(v) - m_n(v) \leq (1-\delta)^n,$$

i.e. the first bound in (5.3).

To prove (A.3), consider a Markov chain on $I \times I$, with states (u_1, u_2), and transition probabilities

$$\begin{aligned}\mathbf{P}\Big((u_1,u_2),(v_1,v_2)\Big) &= P(u_1,v_1)P(u_2,v_2), \ \text{ if } \ u_1 \neq u_2,\\ &= P(u,v), \quad u_1 = u_2 = u \ \text{ and } \ v_1 = v_2 = v,\\ &= 0, \ \text{ if } \quad u_1 = u_2 \ \text{ and } \ v_1 \neq v_2,\end{aligned} \tag{A.4}$$

It is easy to check that $\mathbf{P}\Big((u_1,u_2),(v_1,v_2)\Big)$ is indeed a transition probability matrix (of size $m^2 \times m^2$): if $u_1 = u_2 = u$ then

$$\sum_{v_1,v_2}\mathbf{P}\Big((u_1,u_2),(v_1,v_2)\Big) = \sum_{v} P(u,v) = 1$$

whereas if $u_1 \neq u_2$ then

$$\sum_{v_1,v_2}\mathbf{P}\Big((u_1,u_2),(v_1,v_2)\Big) = \sum_{v_1} P(u_1,v_1)\sum_{v_2} P(u_2,v_2) = 1$$

(the inequalities $0 \le \mathbf{P}\Big((u_1,u_2),(v_1,v_2)\Big) \le 1$ follow directly from the definition (A.4)).

We call the chain on $I \times I$ a coupled M.c. and denote it by (V_n, W_n), $n \ge 1$. Its fundamental property is that both components V_n and W_n are M.c.'s with transition probabilities $\{P(u,v)\}$. More precisely, the components V_n and W_n move independently, with transition probabilities $\{P(u,v)\}$, until the first moment when they coincide. This moment is of course random: it is called the coupling time and denoted by τ. After time τ the components V_n and W_n 'stick' together and move synchronously, again with transition probabilities $\{P(u,v)\}$.

Suppose we start the coupled chain from a state (u,u'). Then

$$|P^{(n)}(u,v) - P^{(n)}(u',v)| = |\mathsf{P}(V_n = v|V_1 = u, W_1 = u') - \mathsf{P}(W_n = v|V_1 = u, W_1 = u')|$$

(because each component of (V_n, W_n) moves with transition probabilities $P(\,\cdot\,,\,\cdot\,)$)

$$= |\mathsf{P}(V_n = v, W_n \ne v|V_1 = u, W_1 = u') - \mathsf{P}(V_n \ne v, W_n = v|V_1 = u, W_1 = u')| \quad (A.5)$$

$$\le \mathsf{P}(V_n \ne W_n|V_1 = u, W_1 = u') = \mathsf{P}(\tau > n|V_1 = u, W_1 = u').$$

Now

$$\mathsf{P}(\tau = 1|V_1 = u, W_1 = u') \ge \sum_v P(u,v)P(u',v) \ge \rho \sum_v P(u',v) = \rho,$$

i.e,

$$\mathsf{P}(\tau > 1|V_1 = u, W_1 = u') \le 1 - \rho.$$

From the strong Markov property (of the coupled chain),

$$\mathsf{P}(\tau > n|V_1 = u, W_1 = u') \le (1-\rho)^n. \quad (A.6)$$

Bounds (A.6) and (A.5) together give (A.3). □

Proof of Lemma 5.3. Expand the square and use the additivity of the expectation:

$$\mathsf{E}\left(\sum_{i=1}^{n}(\sigma_i - H)\right)^2 = \sum_{1\le i\le n} \mathsf{E}(\sigma_i - H)^2 + 2 \sum_{1\le i<j\le n} \mathsf{E}(\sigma_i - H)(\sigma_j - H). \quad (A.6)$$

The first sum in (A.6) is OK: it contains n terms $\mathsf{E}(\sigma_i - H)^2$ each of which does not exceed a constant. Thus this sum is $\le C'n$ where C' is a constant. [From an argument that follows you will be able to deduce that C' may be taken to be $\big(H + |\log\rho|\big)^2$.] It is the second sum that causes problems: it contains $\dfrac{n(n-1)}{2}$ terms. We bound it as follows:

$$\left|\sum_{1\le i<j\le n} \mathsf{E}(\sigma_i - H)(\sigma_j - H)\right| \le \sum_{i=1}^{n}\left(\sum_{k=1}^{\infty}\Big|\mathsf{E}(\sigma_i - H)(\sigma_{i+k} - H)\Big|\right), \quad (A.7)$$

and reduce our problem to proving that

$$\sum_{k=1}^{\infty}\Big|\mathsf{E}(\sigma_i - H)(\sigma_{i+k} - H)\Big| \le \frac{(H + |\log\rho|)^2}{\rho}. \quad (A.8)$$

In turn, bound (A.8) follows from Lemma A.1:

Lemma A.1. *The following bound holds true:*

$$|\mathsf{E}(\sigma_i - H)(\sigma_{i+k} - H)| \le (H + |\log\rho|)^2(1-\rho)^{k-1}.$$

Proof of Lemma A.1. For definiteness, we assume that $i > 1$; the case $i = 1$ requires minor changes. Returning to the definition of random variables σ_i, $i > 1$, write

$$\mathsf{E}(\sigma_i - H)(\sigma_{i+k} - H) = \sum_{u,u'}\sum_{v,v'} \mathsf{P}(U_i = u, U_{i+1} = u'; U_{i+k} = v, U_{i+k+1} = v') \times (-\log\ P(u,u') - H)(-\log\ P(v,v') - H). \quad (A.9)$$

We want to compare this expression with

$$\sum_{u,u'}\sum_{v,v'} \Big(p_1 P^{i-1}\Big)(u)P(u,u')(\log\ P(u,u') - H)w(v)P(v,v')(\log\ P(v,v') - H). \quad (A.10)$$

Observe that (A.10) in fact vanishes because the sum $\sum_{v,v'}$ vanishes, by virtue of the definition of H (see (5.5)).

The difference between sums (A.9) and (A.10) comes from the fact that the probabilities

$$\mathsf{P}(U_i = u, U_{i+1} = u'; U_{i+k} = v, U_{i+k+1} = v')$$

$$= \Big(p_1 P^{i-1}\Big)(u)P(u,u')P^{(k-1)}(u',v)P(v,v')$$

and

$$\Big(p_1 P^{i-1}\Big)(u)P(u,u')w(v)P(v,v')$$

do not coincide. However, the difference of these probabilities in absolute value is

$$\le \big|P^{(k-1)}(u',v) - w(v)\big| \le (1-\rho)^{k-1}.$$

As $\big|-\log P(\,\cdot\,,\,\cdot\,) - H\big| \le H + |\log\rho|$, we obtain the result. □

This completes the proof of Lemma 5.3 and consequently of Theorem 5.2. □

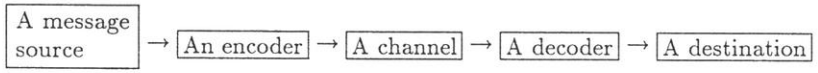
Chapter 2: Channels of information transmission

As in Chapter 1, symbols P and E refer to various probabilities of samples of sequences of random variables and the corresponding expectations. A typical example is a joint distribution of two sequences representing the input and output of an information channel. Likewise, $P(v|u)$ denotes a symbol-to-symbol channel transition probability (e.g., a probability of having a symbol v on the output of a channel, given that symbol u has been sent).

The symbol p is again used to denote various probabilities.

Lecture 7: Basic concepts.

This chapter of the course is related to *channels* of information transmission. Recall our main scheme:



So, a source emits a random text $U_1, U_2, \ldots$. Following the idea of segmentation (see the notes for Lecture 2), you encode a message $u^{(n)}$ by a binary codeword $x^{(N)}$, by using a code $f_n : I^n \to J^N$, $J = \{0,1\}$ (a relation between n and the codeword-length N is discussed below). Of course, the code f_n used is supposed to be known to the receiver. A typical feature of a channel is that it is subject to 'noise' which distorts the messages transmitted: a message at the output differs in general from the message at the input. Formally, a channel is characterized by a conditional distribution

$$\mathsf{P}_{\text{channel}}\left(\text{receive word } y^{(N)} \middle| \text{codeword } x^{(N)} \text{ sent}\right); \tag{7.1}$$

we again suppose that it is known to both sender and receiver. Speaking below of a channel, I shall refer to a conditional probability (or rather a family of conditional probabilities, depending on N), of form (7.1).

Nearly all concrete examples we will deal with are about the so-called *memoryless* binary channels (m.b.c.'s) where

$$\mathsf{P}_{\text{channel}}\left(y^{(N)}|x^{(N)}\right) = \prod_{i=1}^{N} P(y_i|x_i), \text{ if } y^{(N)} = y_1 \ldots y_N,\ x^{(N)} = x_1 \ldots x_N. \tag{7.2}$$

Here, $P(y|x)$, $x, y = 0, 1$, is a symbol-to-symbol channel probability (i.e. the conditional probability to have symbol y at the output of the channel given that symbol x has been sent). It is clear that $\{P(y|x)\}$ is a 2×2 transition probability matrix (often called the cannel matrix). In particular, if $P(1|0) = P(0|1) = p$, the channel is called symmetric (m.b.s.c.). The channel matrix then has the form

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \tag{7.3}$$

and p is called the row error probability (or the symbol-error probability).

You want to introduce a decoding rule $\widehat{f}_N : J^N \to I^n$ so that the overall probability of error

$$\begin{aligned} \epsilon &= \sum_{u^{(n)}} \mathsf{P}(\widehat{f}(y^{(N)}) \neq u^{(n)},\ u^{(n)} \text{ emitted}) \\ &= \sum_{u^{(n)}} \mathsf{P}_{\text{source}}(U^{(n)} = u^{(n)}) \mathsf{P}_{\text{channel}}(\widehat{f}_N(y^{(N)}) \neq u^{(n)} | f_n(u^{(n)}) \text{ sent}) \end{aligned} \tag{7.4}$$

is small. We will try (and in some cases succeed) to have quantity (7.4) tending to zero as $n \to \infty$.

The idea which is behind the construction is based on:

1) The fact that for a source with the AEP the number of distinct n-strings emitted is $2^{n(H+o(1))}$ where $H \leq \log m$ is the information rate of the source. Therefore, you have to encode not $m^n = 2^{n \log m}$ messages, but only $2^{n(H+o(1))}$ which may be considerably less. That is, your code f_n may be defined on a subset of I^n only, and you can choose the codeword-length $N = [nH] + 1$.

2) The fact that you may try even a bigger N: $N = [\overline{R}^{-1} nH] + 1$, where $\overline{R} \in (0,1)$ is a constant. In other words, you want to increase the length of the codewords used from $[nH] + 1$ to $[\overline{R}^{-1} nH] + 1$. That will allow you to introduce a redundancy in your code f_n, and you may hope to be able to use this fact for diminishing the overall error probability (7.4) (provided that you in addition have a 'good' decoding rule). It is of course desirable to minimize $\overline{R}^{-1}$, i.e. maximize $\overline{R}$: it will give you codes with optimal parameters. The question how big $\overline{R}$ is allowed to be depends of course on your channel.

A notational convention: As the codeword-length is a crucial parameter, I will write N instead of $\overline{R}^{-1} Hn$ and $\overline{R}N$ instead of Hn: the number of distinct strings emitted by the source becomes $2^{N(\overline{R}+o(1))}$. In future, the index $n \sim \dfrac{N\overline{R}}{H}$ will be omitted wherever possible (and replaced by N otherwise). It is convenient to consider a 'typical' set $\mathcal{U}_N$ of distinct strings emitted by the source, with $\sharp\, \mathcal{U}_N = 2^{N(\overline{R}+o(1))}$. Formally, $\mathcal{U}_N$ can include strings of different length; it is only the log-asymptotics of $\sharp\, \mathcal{U}_N$ that matters.

Definition 7.1. A value $\overline{R} \in (0,1)$ is called a *reliable transmission* rate (for a given channel) if, given that the source strings take *equiprobable* values from a set $\mathcal{U}_N$ with $\sharp\, \mathcal{U}_N = 2^{N(\overline{R}+o(1))}$, there exist an encoding rule $f_N : \mathcal{U}_N \to \mathcal{X}_N \subseteq \{0,1\}^N$ and a decoding rule $\widehat{f}_N : \{0,1\}^N \to \mathcal{U}_N$ with the error probability

$$\sum_{u \in \mathcal{U}_N} \frac{1}{\sharp\, \mathcal{U}_N} \mathsf{P}_{\text{channel}}\left(\widehat{f}_N(y^{(N)}) \neq u \middle| f_N(u) \text{ sent}\right) \tag{7.5}$$

tending to zero as $N \to \infty$.

That is, for each sequence $\mathcal{U}_N$ with $\lim\limits_{N\to\infty} \dfrac{1}{N} \log \sharp\, \mathcal{U}_N = \overline{R}$, there exists a sequence of encoding rules $f_N : \mathcal{U}_N \to \mathcal{X}_N$, $\mathcal{X}_N \subseteq \{0,1\}^N$, and a sequence of the decoding rules $\widehat{f}_N : \{0,1\}^N \to \mathcal{U}_N$ such that

$$\lim_{N\to\infty} \frac{1}{\sharp\, \mathcal{U}_N} \sum_{u \in \mathcal{U}_N} \sum_{y^{(N)}:\ \widehat{f}_N(y^{(N)}) \neq f_N(u)} \mathsf{P}_{\text{channel}}\left(y^{(N)} \middle| f_N(u)\right) = 0. \tag{7.6}$$

Definition 7.2. The *channel capacity* is defined as the supremum

$$C = \sup\,[\overline{R} \in (0,1) : \ \overline{R} \ \text{is a reliable transmission rate}\,]. \tag{7.7}$$

Remarks. 7.1. The reason for the equiprobable distribution on $\mathcal{U}_N$ is that it is the *worst* case. See Theorem 7.4 below.

7.2. If encoding rule f_N used is a one-to-one function then it suffices to treat the decoding rules as maps $\{0,1\}^N \to \mathcal{X}_N$ rather than $\{0,1\}^N \to \mathcal{U}_N$: if you guess correctly what codeword $x^{(N)}$ has been sent, you simply set $u = f_N^{-1}(x^{(N)})$.

7.3. To enable you to do Example Sheet 2, I give you an answer for an m.b.c.: the channel capacity is given by

$$C = \sup_{p_{X_n}} i(X_n, Y_n). \tag{7.8}$$

Here, $i(X_n, Y_n)$ is the mutual information between a single input and output letters X_n and Y_n (index n may be omitted *wlog*), with the joint distribution

$$\mathsf{P}(X = x, Y = y) = p_X(x)P(y|x), \quad x, y = 0, 1, \tag{7.9}$$

where $p_X(x) = \mathsf{P}(X = x)$. The supremum in (7.8) is over all possible distributions $p_X = (p_X(0), p_X(1))$. A useful formula is $i(X,Y) = h(Y) - h(Y|X)$ (see (6.12)). In fact, in the m.b.c.

$$h(Y|X) = -\sum_{x=0,1} p_X(x) \sum_{y=0,1} P(y|x) \log P(y|x). \tag{7.10}$$

For an m.b.s.c.,

$$\sum_{y=0,1} P(y|x) \ \log P(y|x) = -p \log\, p - (1-p) \log\,(1-p) = h(p, 1-p) \tag{7.11}$$

and hence $h(Y|X) = h(p, 1-p)$ does not depend on input distribution p_X. Thus, in this case

$$C = \sup_{p_X} h(Y) - h(p, 1-p). \tag{7.12}$$

But $\sup\limits_{p_X} h(Y)$ is equal to $\log\ 2 = 1$: it is attained at $p_X(0) = p_X(1) = 1/2$, because then

$$p_Y(0) = p_Y(1) = 1/2(p + 1 - p) = 1/2.$$

Therefore, for an m.b.s.c., with the row error probability p, $C = 1 - h(p, 1-p)$.

7.4. Suppose you have a source $U_1, U_2, \ldots$ with the AEP and information rate H. If you try to send a text emitted by the source through a channel of capacity C then you need to encode messages of length n by codewords of length $\dfrac{n(H+\epsilon_1)}{C+\epsilon_2}$ in order to have the overall error probability tending to zero as $n \to \infty$. Values $\epsilon_1 > 0$ and $\epsilon_2 > 0$ may be chosen arbitrary small. Hence, if $H/C < 1$, you are able to encode a text with a higher speed than it is produced: in this case you can use reliably your channel for transmitting information from your source. On the contrary, if $H/C > 1$, the text will

be produced with a higher speed than you can encode it and send reliably through a channel. In this case it is said that reliable transmission is impossible. For a Bernoulli or Markov source and an m.b.s.c., condition $H/C < 1$ is equivalent to $h(U) + h(p, 1-p) < 1$ or $h(U_2|U_1) + h(p, 1-p) < 1$, respectively.

Now a theorem about equidistribution over $\mathcal{U}$:

Theorem 7.5. *Fix a channel (i.e. a conditional probability* $\mathsf{P}_{\text{channel}}$ *in (7.1)). Fix a set* $\mathcal{U}$ *of the source strings and denote by* $\epsilon(\mathsf{P})$ *the overall error probability (7.4) for* $U^{(n)}$ *having a probability distribution* P $\left(= \mathsf{P}_{\text{source}}\right)$ *over* $\mathcal{U}$*, minimized over all encoding and decoding rules. Then*

$$\epsilon(\mathsf{P}) \le \epsilon(\mathsf{P}^0), \tag{7.13}$$

where P^0 *is the equidistribution over* $\mathcal{U}$.

Proof of Theorem 7.5. Fix an encoding and a decoding rules, f and $\widehat{f}$ and let a string $u \in \mathcal{U}$ have probability $\mathsf{P}(u)$. Set:

$$\beta(u) := \sum_{y:\widehat{f}(y) \neq f(u)} \mathsf{P}_{\text{channel}}(y|f(u)).$$

That is, $\beta(u)$ is the error-probability when u is emitted. The overall error probability equals

$$\epsilon(= \epsilon(\mathsf{P}, f, \widehat{f})) = \sum_{u \in \mathcal{U}} \mathsf{P}(u)\beta(u).$$

If you permute the allocation of codewords (i.e. encode u by $f(u')$ where $u' = \lambda(u)$ and λ is a permutation of degree $\sharp\,\mathcal{U}$), you get the overall error probability

$$\epsilon(\lambda) = \sum_{u \in \mathcal{U}} \mathsf{P}(u)\beta(\lambda(u)).$$

In the case $\mathsf{P}(u) = (\sharp\,\mathcal{U})^{-1}$ (equidistribution), $\epsilon(\lambda)$ does not depend on λ and equals

$$\bar{\epsilon} = \frac{1}{\sharp\,\mathcal{U}} \sum_{u \in \mathcal{U}} \beta(u) (= \epsilon(\mathsf{P}^0, f, \widehat{f})).$$

It is claimed that for each probability distribution $\{\mathsf{P}(u),\ u \in \mathcal{U}\}$ there exists Π such that $\epsilon(\lambda) \le \bar{\epsilon}$. In fact, take a *random* permutation, Λ, equidistributed among all $(\sharp\,\mathcal{U})!$ permutations of degree $\sharp\,\mathcal{U}$. Then

$$\min_{\lambda} \epsilon(\lambda) \le \mathsf{E}\epsilon(\Lambda) = \mathsf{E} \sum_{u \in \mathcal{U}} \mathsf{P}(u)\beta(\Lambda u)$$

$$= \sum_{u \in \mathcal{U}} \mathsf{P}(u)\mathsf{E}\beta(\Lambda u) = \sum_{u \in \mathcal{U}} \mathsf{P}(u) \frac{1}{\sharp\,\mathcal{U}} \sum_{\widetilde{u} \in \mathcal{U}} \beta(\widetilde{u}) = \bar{\epsilon}.$$

Hence, given any f and $\widehat{f}$, you can find new encoding and decoding rules with overall error probability $\le \epsilon(\mathsf{P}^0, f, \widehat{f})$. Minimizing over f and $\widehat{f}$ leads to (7.13). □

Lecture 8: Decoding rules

It is now time to discuss possible *decoding* rules. As was noted before, a decoding rule (or a decoder) is a map $\widehat{f}_N : \{0,1\}^N \to \mathcal{U}_N$ ($\{0,1\}^N \to \mathcal{X}_N$ in the case of a one-to-one encoding rule f_N, with a set of codewords $\mathcal{X}_N$). Equivalently, a decoding rule is given by fixing, for each codeword $x^{(N)}$, a set $A(x^{(N)}) \subset \{0,1\}^N$, so that $A(x_1^{(N)})$ and $A(x_2^{(N)})$ are disjoint for distinct codewords $x_1^{(N)}$ and $x_2^{(N)}$, and the union $\cup_{x^{(N)} \in \mathcal{X}_N} A(x^{(N)})$ gives the whole $\{0,1\}^N$. Given that $y^{(N)} \in A(x^{(N)})$, you set: $\widehat{f}_N(y^{(N)}) = x^{(N)}$.

Although in the definition of the channel capacity we assume that the source messages are equidistributed (which gives the worst case in the sense of Theorem 7.5), in reality of course the source does not always follow this assumption. We need to distinguish between two situations: (i) the receiver knows the probability distribution

$$p(u^{(N)}) = \mathsf{P}_{\text{source}}(U^{(N)} = u^{(N)}) \tag{8.1}$$

of the source strings (and hence the probability distribution $p_N(x^{(N)})$ of the codewords $x^{(N)} \in \mathcal{X}_N$), and (ii) he does not know $p_N(x^{(N)})$. Two natural decoding rules are, respectively,

(i) an *ideal observer* (i.o.) rule: you decode a received word $y^{(N)}$ by a codeword $x^{(N)\star}$ that maximizes the posterior probability

$$\mathsf{P}\left(x^{(N)} \text{ sent} \,\middle|\, y^{(N)} \text{ received}\right) = \frac{p_N(x^{(N)})\mathsf{P}_{\text{channel}}(y^{(N)}|x^{(N)})}{p_{Y^{(N)}}(y^{(N)})}, \tag{8.2}$$

where

$$p_{Y^{(N)}}(y^{(N)}) = \sum_{\widetilde{x^{(N)}} \in \mathcal{X}_N} p_N(\widetilde{x^{(N)}})\mathsf{P}_{\text{channel}}(y^{(N)}|\widetilde{x^{(N)}}),$$

and

(ii) a *maximum likelihood* (m.l.) rule: you decode a received word $y^{(N)}$ by a codeword $x_\star^{(N)}$ that maximizes the prior probability

$$\mathsf{P}_{\text{channel}}(y^{(N)}|x^{(N)}). \tag{8.3}$$

In Theorem 8.1 below we suppose that an encoding rule f is defined for all messages that occur with positive probability and is one-to-one. [The decoding rule is always assumed to be one-to-one.]

Theorem 8.1.

(a) *For any encoding rule, the i.o. decoder minimizes the overall error probability among all decoders.*

(b) *If the source message U is equiprobable on a set $\mathcal{U}$, and an encoding rule $f : \mathcal{U} \to \mathcal{X}$ then codeword $X = f(U)$ is equiprobable on $\mathcal{X}$, and the i.o. and m.l. decoders coincide.*

Proof of Theorem 8.1. (a) Note that, given a received word y, the i.o. obviously maximizes the joint probability

$$p(x)\mathsf{P}_{\text{channel}}(y|x)$$

(the denominator in (8.2) is fixed when word y is fixed). Suppose you use an encoding rule f and decoding rule $\widehat{f}$. You can write

The overall error probability (see (7.4)) =

$$\begin{aligned}
&\sum_u \mathsf{P}_{\text{source}}(U = u)\mathsf{P}_{\text{channel}}\left(\widehat{f}(y) \neq u \middle| f(u) \text{ sent}\right)\\
=&\sum_x p(x) \sum_{y:\widehat{f}(y)\neq x} \mathsf{P}_{\text{channel}}\left(y|x\right)\\
=&\sum_y \sum_{x:x\neq\widehat{f}(y)} p(x)\mathsf{P}_{\text{channel}}\left(y|x\right)\\
=&\sum_y \sum_{\text{all } x} p(x)\mathsf{P}_{\text{channel}}\left(y|x\right) - \sum_y p\left(\widehat{f}(y)\right)\mathsf{P}_{\text{channel}}\left(y|\widehat{f}(y)\right)\\
=&1 - \sum_y p\left(\widehat{f}(y)\right)\mathsf{P}_{\text{channel}}\left((y|\widehat{f}(y)\right)
\end{aligned}$$

It remains to note that each term in the sum $\sum_y p\left(\widehat{f}(y)\right)\mathsf{P}_{\text{channel}}\left(y|\widehat{f}(y)\right)$ is maximized when $\widehat{f}$ coincides with the i.o. rule. Hence, the whole sum is maximized, and the overall error probability minimized.

(b) The first statement is obvious, as, indeed is the second.

As we suppose, in the definition of the channel capacity, that the source messages are equidistributed, it is natural to use the m.l. decoder. We are going to do so in the rest of the course.

While using the m.l. decoder, an error can occur because either the decoder chooses a wrong codeword x or an encoding rule f used is non-injective. The probability of this is assessed in Lemma 8.2. For the sake of simplicity, we write $\mathbf{P}$ instead of $\mathsf{P}_{\text{channel}}$; symbol $\mathbf{P}$ is used mainly for the joint input/output distribution.

Lemma 8.2. *If the source messages are equidistributed over a set $\mathcal{U}$ then, while using the m.l. decoder and an encoding rule f, the overall error probability satisfies*

$$\epsilon(f) \leq \frac{1}{\sharp\,\mathcal{U}} \sum_{u\in\mathcal{U}} \sum_{u'\in\mathcal{U}:\, u'\neq u} \mathbf{P}\left(\mathbf{P}\left(Y|f(u')\right) \geq \mathbf{P}\left(Y|f(u)\right) \middle| U = u\right). \tag{8.4}$$

□

Proof of Lemma 8 If the source emits u and you use the m.l. decoder, you get

(a) an error when $\mathbf{P}(Y|f(u')) > \mathbf{P}(Y|f(u))$ for some $u' \neq u$,

(b) possibly an error when $\mathbf{P}(Y|f(u')) = \mathbf{P}(Y|f(u))$ for some $u' \neq u$ (this includes the case when $f(u) = f(u')$,

(c) no error when $\mathbf{P}(Y|f(u')) < \mathbf{P}(Y|f(u))$ for any $u' \neq u$.

Thus,

$$\mathsf{P}\left(\text{error}\,\middle|U = u\right) \leq \mathsf{P}\left(\mathbf{P}\left(Y|f(u')\right) \geq \mathbf{P}\left(Y|f(u)\right) \text{ for some } u' \neq u \middle| U = u\right)$$

$$\leq \sum_{u' \in \mathcal{U}:\, u' \neq u} \mathsf{P}\left(\mathbf{P}\left(Y|f(u')\right) \geq \mathbf{P}\left(Y|f(u)\right)\middle|U = u\right).$$

Multiplying by $\frac{1}{\sharp\, \mathcal{U}}$ and summing up over u yields the result. □

Remark 8.3. Bound (8.4) of course holds for any probability distribution $p(u) = \mathsf{P}_{\text{source}}$ $(U = u)$, provided you replace $\frac{1}{\sharp\, \mathcal{U}}$ by $p(u)$.

As was noted, we will use, together with deterministic encoding rules, *random* coding. A determistic encoding rule is a map $f\colon \mathcal{U} \to \{0,1\}^N$; if $\sharp\, \mathcal{U} = r$ then f is given as a collection of codewords

$$\left(f(u_1), \ldots, f(u_r)\right) \in \{0,1\}^N \times \ (r \text{ times}) \ \times \{0,1\}^N = \left(\{0,1\}^N\right)^r = \{0,1\}^{Nr}.$$

Here, u_1, ..., u_r are the source strings (not letters!) constituting set $\mathcal{U}$. If f is one-to-one, $f(u_i) \neq f(u_j)$ when $i \neq j$. A random encoding rule is a random element F of $\left(\{0,1\}^N\right)^r$ (i.e. a probability distribution on $\left(\{0,1\}^N\right)^r$):

$$\mathsf{P}(F = f), \quad f \in \left(\{0,1\}^N\right)^r.$$

Equivalently, F may be regarded as a collection of random codewords $F(u_i)$, $i = 1$, ..., u_r. A typical example is where codewords $F(u_1)$, $F(u_2)$, ..., $F(u_r)$ are independent and (random) symbols W_{i1}, ..., W_{iN} constituting word $F(u_i)$ are independent.

The reasons for considering random encoding rules are:

1^0 the existence of a 'good' deterministic code frequently follows from the existence of a good random code,

2^0 the calculations for random codes are usually more simple than for optimal deterministic codes, because a discrete optimization is replaced by an optimization over probability distributions.

A drawback of random coding is that it is not always one-to-one ($F(u)$ may coincide with $F(u')$ for $u \neq u'$). However, this occurs, for large N, with negligible probability.

Continuing with random coding, write the expected error-probability for a random encoding rule F:

$$E := \mathsf{E}\epsilon(F) = \sum_f \epsilon(f)\mathsf{P}(F = f). \tag{8.6}$$

Theorem 8.4.

(i) *There exists a deterministic encoding rule f with $\epsilon(f) \leq E$.*

(ii) $\mathsf{P}\left(\epsilon(F) < \frac{E}{1-\rho}\right) \geq \rho$ *for any* $\rho \in (0,1)$. □

Proof of Theorem 8.4. (i) is obvious. For (ii), use the Chebyshov inequality:

$$\mathsf{P}\left(\epsilon(F) \geq \frac{E}{1-\rho}\right) \leq \frac{1-\rho}{E}\mathsf{E}\epsilon(F) = 1 - \rho. \tag{8.7}$$

Lecture 9: Shannon's Second coding theorem

Definition 9.1. For random words $X^{(N)} = X_1, \dots, X_N$ and $Y^{(N)} = Y_1, \dots, Y_N$ define

$$C_N := \sup_{P_{X^{(N)}}} \frac{1}{N}\, i\left(X^{(N)}, Y^{(N)}\right). \tag{9.1}$$

[Recall that $i\left(X^{(N)}, Y^{(N)}\right)$ is the mutual entropy

$$i\left(X^{(N)}, Y^{(N)}\right) = h\left(X^{(N)}\right) - h\left(X^{(N)}\middle|Y^{(N)}\right) = h\left(Y^{(N)}\right) - h\left(Y^{(N)}\middle|X^{(N)}\right). \quad]$$

Theorem 9.2. (Shannon's Second Coding Theorem: converse part) *The channel capacity C obeys*

$$C \le \lim\sup_{N\to\infty} C_N. \tag{9.2}$$

Proof of Theorem 9.2. Consider a code $f(\,=f_N) : \mathcal{U}_N \to \mathcal{X}_N \subseteq \{0,1\}^N$, where $\sharp\mathcal{U}_N = 2^{N(\overline{R}+o(1))}$, $\overline{R} \in (0,1)$. We want to prove that for any decoding rule,

$$\epsilon(f) \ge 1 - \frac{C_N + o(1)}{\overline{R} + o(1)}. \tag{9.3}$$

The assertion of the theorem immediately follows from (9.3) and the definition of the channel capacity because

$$\lim\inf_{N\to\infty} \epsilon(f) \ge 1 - \frac{1}{\overline{R}} \lim\sup_{N\to\infty} C_N$$

which is > 0 when $\overline{R} > \limsup C_N$.

Wlog, assume that f is one-to-one (otherwise $\epsilon(f)$ is even bigger). Then a codeword $X^{(N)} = f(U)$ is equidistributed when string U is, and, if a decoding rule is $d : \{0,1\}^N \to \mathcal{X}$, you have, for N large enough,

$$\begin{aligned} NC_N &\ge i\left(X^{(N)}, Y^{(N)}\right) \ge i\left(X^{(N)}, d(Y^{(N)})\right) && \text{(see Theorems 6.5)} \\ &= h\left(X^{(N)}\right) - h\left(X^{(N)}\middle|d(Y^{(N)})\right) = \log r - h\left(X^{(N)}\middle|d(Y^{(N)})\right) \\ & && \text{(by equidistribution)} \\ &= \log r - \epsilon(f)\log(r-1) \\ & && \text{(by the generalized Fano inequality (see Theorems 6.3).} \end{aligned}$$

In fact, to prove the last bound, observe that the (random) codeword $X^{(N)} = f(U)$ takes r values $x_1^{(N)}, \dots, x_r^{(N)}$ from the codeword set $\mathcal{X}(= \mathcal{X}_N)$, and the error probability is

$$\epsilon(f) = \sum_{i=1}^{r} \mathsf{P}(X^{(N)} = x_i^{(N)}, d(Y^{(N)}) \ne x_i^{(N)})$$

So, you get from the generalized Fano inequality:

$$h\left(X^{(N)}\middle|d(Y^{(N)})\right) \le g(\epsilon) + \epsilon\log(r-1) \le 1 + \epsilon(f)\log(r-1).$$

Now, from the inequality $NC_N \ge \log r - \epsilon(f)\log(r-1)$ you have

$$NC_N \ge N(\overline{R} + o(1)) - \epsilon(f)\log\left(2^{N(\overline{R}+o(1))} - 1\right),$$

i.e.

$$\epsilon(f) \ge \frac{N(\overline{R}+o(1)) - NC_N}{\log\left(2^{N(\overline{R}+o(1))} - 1\right)} = 1 - \frac{C_N + o(1)}{\overline{R} + o(1)}.$$

□

In Theorem 9.3 below, $p(X^{(N)}, Y^{(N)})$ denotes the random variable that assigns, to random words $X^{(N)}$ and $Y^{(N)}$, the joint probability of having these words at the input and output of a channel, respectively. Similarly, $p_X(X^{(N)})$ and $p_Y(Y^{(N)})$ denote the random variables that give the marginal probabilities of words $X^{(N)}$ and $Y^{(N)}$, respectively.

Theorem 9.3. (The Shannon second coding theorem: direct part) *Suppose you can find a constant $c \in (0,1)$ such that for any $\overline{R} \in (0,c)$ and $N \ge 1$ there exists a random coding $F(u_1), \dots, F(u_r)$, where $r = 2^{N(\overline{R}+o(1))}$, with i.i.d. codewords $F(u_i) \in \{0,1\}^N$, such that the (random) input/output mutual information*

$$\eta_N := \frac{1}{N}\log\frac{p(X^{(N)}, Y^{(N)})}{p_X(X^{(N)})p_Y(Y^{(N)})} \xrightarrow{\mathsf{P}} c, \quad \text{as } N \to \infty. \tag{9.4}$$

Then the channel capacity $C \ge c$.

In other words, channel capacity C is no less than the supremum of the values c for which the convergence in probability in (9.4) holds for an appropriate random coding.

Corollary 9.4.

$$\boxed{\sup c \le C \le \limsup_{N\to\infty} C_N.} \tag{9.5}$$

So, if the LH and RH sides of (9.5) coincide then their common value gives the channel capacity.

The proof of Theorem 9.3 is rather long, and we perform it in the next lecture. We will now show how Shannon's SCT is used for calculating the capacity of a m.b.c.

Recall, for a m.b.c.,

$$\mathbf{P}\left(y^{(N)}\middle|x^{(N)}\right) = \prod_{i=1}^{N} P(y_i|x_i), \tag{9.6}$$

cf. (7.2).

Theorem 9.5. *For an m.b.c.,*

$$i\left(X^{(N)}, Y^{(N)}\right) \le \sum_{j=1}^{N} i(X_j, Y_j), \tag{9.7}$$

with equality if the input symbols $X_1, \dots, X_N$ are independent. □

Proof of Theorem 9.5. Since $\mathbf{P}\left(y^{(N)}|x^{(N)}\right) = \prod_{j=1}^{N} P(y_j|x_j)$, the conditional entropy

$$h\left(Y^{(N)}|X^{(N)}\right) = \sum_{j=1}^{N} h(Y_j|X_j),$$

and the mutual information

$$i\left(X^{(N)}, Y^{(N)}\right) = h\left(Y^{(N)}\right) - h\left(Y^{(N)}|X^{(N)}\right)$$

$$= h\left(Y^{(N)}\right) - \sum_{j=1}^{N} h(Y_j|X_j) \le \sum_{j=1}^{N} \left(h(Y_j) - h(Y_j|X_j)\right)$$

$$= \sum_{j=1}^{N} i(X_j, Y_j).$$

The equality holds iff $Y_1, \dots, Y_N$ are independent. But $Y_1, \dots, Y_N$ are independent if $X_1, \dots, X_N$ are. □

Remark 9.6. Cf. inequalities (6.15) and (6.16). Note the opposite inequalities in the bounds.

Theorem 9.7. *The capacity of an m.b.c. is*

$$C = \sup_{p_{X_1}} i(X_1, Y_1). \tag{9.8}$$

The supremum is over all possible distributions p_{X_1} of symbol X_1.

Remarks. 9.8. The pair (X_1, Y_1) may be replaced by any (X_j, Y_j), $j \ge 1$.

9.9. Recall, the joint distribution of X_1 and Y_1 is defined by $\mathbf{P}(X_1 = x, Y_1 = y) = p_{X_1}(x)P(y|x)$ where $P(y|x)$ is the channel matrix.

Proof of Theorem 9.7. By the definition of C_N (see the notes for Lecture 9),

$$NC_N \le \sup_{p_X} i(X^{(N)}, Y^{(N)})$$

$$\le \sum_{j=1}^{N} \sup_{p_{X_j}} i(X_j, Y_j) = N \sup_{p_{X_1}} i(X_1, Y_1)$$

So, by Shannon's SCT (converse part),

$$C \le \lim \sup_{N\to\infty} C_N \le \sup_{p_{X_1}} i(X_1, Y_1).$$

On the other hand, take a random coding F, with codewords

$$F(u_l) = V_{l1} \dots V_{lN}, \qquad l = 1, \dots, r,$$

containing i.i.d. symbols V_{lj} that are distributed according to $p_{\max}$, a probability distribution that maximizes $i(X_1, Y_1)$. [Such random coding is defined for *any* r, i.e. for any $\overline{R}$ (even $\overline{R} > 1$!).] For this random coding, the (random) mutual information

$$\eta_N = \frac{1}{N} \log \frac{p\left(X^{(N)}, Y^{(N)}\right)}{p_X\left(X^{(N)}\right) p_Y\left(Y^{(N)}\right)}$$

$$= \frac{1}{N} \sum_{j=1}^{N} \log \frac{p(X_j, Y_j)}{p_{\max}(X_j) p_Y(Y_j)} = \frac{1}{N} \sum_{j=1}^{N} \zeta_j,$$

where

$$\zeta_j := \log \frac{p(X_j, Y_j)}{p_{\max}(X_j) p_Y(Y_j)}.$$

The random variables ζ_j are i.i.d., with

$$\mathsf{E}\zeta_j = \mathsf{E} \log \frac{p(X_j, Y_j)}{p_{\max}(X_j) p_Y(Y_j)} = i_{p_{\max}}(X_1, Y_1).$$

By the Law of large numbers for i.i.d variables (see Theorem 3.5), for the random coding as suggested,

$$\eta_N \xrightarrow{\mathbf{P}} i_{p_{\max}}(X_1, Y_1) = \sup_{p_{X_1}} i(X_1, Y_1).$$

By Shannon's SCT (direct part),

$$C \ge \sup_{p_{X_1}} i(X_1, Y_1).$$

Thus,

$$C = \sup_{p_{X_1}} i(X_1, Y_1).$$

□

Remark 9.10. Although, as was noted, the construction holds for each r, that is, for each $\overline{R} \ge 0$, only $\overline{R} \le C$ are reliable.

Lecture 10: Shannon's Second coding theorem (cont.)

Formula (9.8) admits further simplification when the channel is symmetric, i.e.

$$P(1|0) = P(0|1) = p. \tag{10.1}$$

The channel matrix in this case is of the form

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix},$$

cf (7.3); value p is the row error probability.

Theorem 10.1. *For a m.b.s.c., with the row error probability p,*

$$C = 1 - h(p, 1-p) \tag{10.2}$$

(cf (7.12). The channel capacity is realized by a random coding with the i.i.d. symbols V_{lj} taking values 0 and 1 with probability 1/2.

Proof of Theorem 10.1. Formula (10.2) and the form of the maximizing probability distribution were established in Remark 7.3. □

We are now going to prove the direct part of Shannon's SCT (Theorem 9.3).

Proof of Theorem 9.3. The proof is based on two lemmas.

Lemma 10.2. *Let F be a random coding, independent on the message string U, such that the codewords $F(u_1), \dots, F(u_r)$ are i.i.d., with a probability distribution p_F:*

$$p_F(v) = \mathsf{P}(F(u) = v), \quad v(= v^{(N)}) \in \{0,1\}^N. \tag{10.3}$$

Here, u_j, $j = 1, ..., r$ are source strings, and $r = 2^{N(\overline{R}+o(1))}$. Define random codewords $V_1, \dots, V_r$ by

$$\begin{aligned} &\text{if } U = u_j \text{ then } V_i := F(u_i) \text{ for } i < j \text{ (if any)}, \\ &\qquad\qquad := F(u_{i+1}) \text{ for } i \geq j \text{ (if any)}, \\ &\qquad\qquad j = 1, \dots, r, \quad i = 1, \dots, r-1. \end{aligned} \tag{10.4}$$

Then U (the message string), $X = F(U)$ (the random codeword) and $V_1, \dots, V_{r-1}$ are independent words, and each of $X, V_1, \dots, V_{r-1}$ has distribution p_F. □

Proof of Lemma 10.2. You write, for a joint probability,

$$\begin{aligned} &\mathsf{P}(U = u_j,\ X = x,\ V_1 = v_1, \dots, V_{r-1} = v_{r-1}) \\ &= \mathsf{P}\left(U = u_j, \begin{pmatrix} F(u_1) \\ \vdots \\ F(u_{j-1}) \\ F(u_j) \\ F(u_{j+1}) \\ \vdots \\ F(u_r) \end{pmatrix} = \begin{pmatrix} v_1 \\ \vdots \\ v_{j-1} \\ x \\ v_j \\ \vdots \\ v_{r-1} \end{pmatrix} \right) \\ &= \mathsf{P}_{\text{source}}(U = u_j)\, p_F(x)\, p_F(v_1) \dots p_F(v_{r-1}). \end{aligned} \tag{10.5}$$

Now if random variable η_N is as in (9.4) then

$$\mathsf{E}\eta_N = \frac{1}{N}\, i\left(X^{(N)}, Y^{(N)}\right) \tag{10.6}$$

□

Lemma 10.3. *For random coding as in Lemma 10.1, for any $t > 0$,*

$$E = \mathsf{E}\epsilon(F) \leq \mathsf{P}(\eta_N \leq t) + r2^{-Nt}. \tag{10.7}$$

□

Proof of Lemma 10.4. For given words $x(= x^{(N)}), y(= y^{(N)}) \in \{0,1\}^N$, denote

$$S_y(x) := \left\{x' \in \{0,1\}^N : \mathbf{P}(y|x') \geq \mathbf{P}(y|x)\right\}.$$

[$\mathbf{P}$ stands for $\mathsf{P}_{\text{channel}}$.] That is, $S_y(x)$ includes all words the m.l. decoder may produce in the situation where x was sent and y received. Denote, for a given non-random encoding rule f and a source string u,

$$\delta(f,u,y) = \begin{cases} 1, & \text{if } f(u') \in S_y(f(u)) \text{ for some } u' \neq u, \\ 0, & \text{otherwise.} \end{cases}$$

Clearly,

$$\begin{aligned} \delta(f,u,y) &= 1 - \prod_{u':\, u' \neq u} \mathbf{1}_{\{f(u') \notin S_y(f(u))\}} \\ &= 1 - \prod_{u': u' \neq u} \left(1 - \mathbf{1}_{\{f(u') \in S_y(f(u))\}}\right) \end{aligned}$$

It is plain that, for any non-random encoding f,

$$\epsilon(f) \leq \mathsf{E}\delta(f,U,Y),$$

and for any random encoding F,

$$E = \mathsf{E}\epsilon(F) \leq \mathsf{E}\delta(F,U,Y).$$

Furthermore, for the random encoding as in Lemma 10.2,

$$\begin{aligned} \mathsf{E}\delta(F,U,Y) &\leq \mathsf{E}\left(1 - \prod_{i=1}^{r-1}\left(1 - \mathbf{1}_{\{V_i \in S_Y(X)\}}\right)\right) \\ &= \sum_x p_X(x) \sum_y \mathbf{P}(y|x) \\ &\quad \times \mathsf{E}\left(1 - \prod_{i=1}^{r-1}\left(1 - \mathbf{1}_{\{V_i \in S_Y(X)\}}\right) \middle| X = x,\ Y = y\right), \end{aligned}$$

which, owing to the independence in Lemma 10.2, is

$$= \sum_x p_X(x) \sum_y \mathbf{P}(y|x) \left(1 - \prod_{i=1}^{r-1} \mathsf{E}\left(1 - \mathbf{1}_{\{V_i \in S_y(x)\}}\right)\right).$$

Furthermore, due to the i.i.d. property (see again Lemma 10.1),

$$\prod_{i=1}^{r-1} \mathsf{E}\left(1 - \mathbf{1}_{\{V_i \in S_y(x)\}}\right) = (1 - Q_y(x))^{r-1},$$

where

$$Q_y(x) := \sum_{x' \in S_y(x)} p_X(x'),$$

and hence

$$E \le 1 - \mathsf{E}\,(1 - Q_Y(X))^{r-1}.$$

Denote by $\mathsf{T} = \mathsf{T}(y)$ the set of pairs of words x, y for which

$$\frac{1}{N} \log \frac{p(x,y)}{p_X(x) p_Y(y)} > t$$

and write two bounds:

$$1 - (1 - Q_y(x))^{r-1} = \sum_{j=0}^{r-2} (1 - Q_y(x))^j \, Q_y(x)$$

$$\text{(because of } \; 1 - B^n = \sum_{j=1}^{n-1} B^j (1 - B)) \tag{10.8a}$$

$$\le (r-1) Q_y(x), \quad \text{when } (x,y) \notin \mathsf{T},$$

and

$$1 - (1 - Q_y(x))^{r-1} \le 1, \quad \text{when} (x,y) \in \mathsf{T}. \tag{10.8b}$$

This yields

$$E \le \mathsf{P}\big((X,Y) \notin \mathsf{T}\big) + (r-1) \sum_{(x,y) \in \mathsf{T}} p_X(x) \mathbf{P}(y|x) Q_y(x). \tag{10.9}$$

Now observe that

$$\mathsf{P}\big((X,Y) \notin \mathsf{T}\big) = \mathsf{P}\big(\eta_N \le t\big). \tag{10.10}$$

Finally, for $(x,y) \in \mathsf{T}$ and $x' \in S_y(x)$,

$$\mathbf{P}(y|x') \ge \mathbf{P}(y|x) \ge p_Y(y) 2^{Nt}.$$

Multiplying by $\dfrac{p_X(x')}{p_Y(y)}$ gives

$$\mathsf{P}\left(X = x' \middle| Y = y\right) \ge p_X(x') 2^{Nt}$$

and summing over $x' \in S_y(x)$

$$1 \ge \mathsf{P}\left(S_Y(x) \middle| Y = y\right) \ge Q_y(x) 2^{Nt},$$

or

$$Q_y(x) \le 2^{-Nt}. \tag{10.11}$$

Substituting (10.10) and (10.11) into (10.9) yields the assertion of the lemma. □

We now can complete the proof of Theorem 9.3. In fact, take $\overline{R} = c - 2\epsilon$ and $t = c - \epsilon$. Then, as $r = 2^{N(\overline{R} + o(1))}$, we have

$$\begin{aligned} E = \mathsf{E}\epsilon(F) &\le \mathsf{P}(\eta_N \le c - \epsilon) + 2^{N(c - 2\epsilon + c + \epsilon + o(1))} \\ &= \mathsf{P}(\eta_N \le c - \epsilon) + 2^{-N\epsilon}. \end{aligned}$$

The RHS tends to zero as $n \to \infty$, because $\mathsf{P}(\eta_N \le c - \epsilon) \to 0$ owing to the condition $\eta_N \xrightarrow{\mathsf{P}} c$. Therefore, the random coding F gives the expected error probability that vanishes as $N \to \infty$.

By Theorem 8.4 (i), for any $N \ge 1$ there exists a deterministic encoding $f = f_N$ such that, for $\overline{R} = c - \epsilon$,

$$\lim_{N \to \infty} \epsilon(f) = 0.$$

Hence, $\overline{R}$ is a reliable transmission rate. This is true for any $\epsilon > 0$, thus $C \ge c$. □

Lecture 11: The channel capacity: concluding remarks

Theorems 9.7 and 10.1 may be extended to the case of a memoryless channel with an arbitrary (finite) output alphabet, say $\{0,\dots,t\}$. That is, at the input of the channel you now have a word $Y^{(N)} = Y_1 \dots Y_N$ where each Y_j takes a (random) value from $\{0,\dots,t\}$. For example, a channel may produce, in addition to the 0's and 1's, a 'splodge', $\star$. The memoryless property means, as before, that

$$\mathsf{P}_{\text{channel}}\left(y^{(N)}\middle|x^{(N)}\right) = \prod_{i=1}^{N} P(y_i|x_i), \tag{11.1}$$

and the symbol-to-symbol channel probabilities $P(y|x)$ now form a $2\times(t+1)$ stochastic matrix (the channel matrix). A memoryless channel is called symmetric if the rows of the channel matrix are permutations of each other and double symmetric if in addition the columns of the channel matrix are permutations of each other. The definitions of the reliable transmission rate and the channel capacity are carried through without change.

Theorem 11.1. *The capacity of a memoryless symmetric channel with an output alphabet* $\{0,\dots,t\}$ *is*

$$C \le \log(t+1) - h(p_0,\dots,p_t) \tag{11.2}$$

where $(p_0,\dots,p_t)$ *is a row of the channel matrix. The equality is realized in the case of a double-symmetric channel, and the maximizing random coding has i.i.d. symbols* V_i *taking values* 0 *and* 1 *with probability* 1/2.

Proof of Theorem 11.1. The whole proof (including Shannon's SCT) is carried out, and in particular

$$i(X_1,Y_1) = h(Y_1) - h(Y_1|X_1) \le \log(t+1) - h(Y_1|X_1).$$

But in the symmetric case

$$h(Y_1|X_1) = -\sum_{x,y}\mathsf{P}(X_1=x)P(y|x)\log\ P(y|x)$$

$$= -\sum_{x}\mathsf{P}(X_1=x)\sum_{k}p_k\log\ p_k = -h(p_0,\dots,p_t). \tag{11.3}$$

If, in addition, the columns of the channel matrix are permutations of each other, then $h(Y_1)$ attains $\log(t+1)$. Indeed, take a random coding as suggested. Then

$$\mathsf{P}(Y=y) = \sum_{x=0,1}\mathsf{P}(X_1=x)P(y|x) = \frac{1}{2}\sum_{x}P(y|x).$$

The sum $\sum_x P(y|x)$ is along a column of the channel matrix, and it does not depend on y. Hence, $\mathsf{P}(Y=y)$ does not depend on $y\in\{0,\dots,t\}$, which means equidistribution. □

Remarks 11.2. In the random coding F used in Lemmas 10.2 and 10.3 and Theorems 9.3, 9.7, 10.1 and 11.1, the expected error probability $E\to 0$ with $N\to\infty$. This guarantees not only the existence of a 'good' non-random coding for which the error probability vanishes as $N\to\infty$ (see Theorem 8.4 (i)), but also that 'almost' all codes are asymptotically good. In fact, by Theorem 8.4 (ii), with $\rho = 1-\sqrt{E}$,

$$\mathsf{P}\left(\epsilon(F) < \sqrt{E}\right) \ge 1-\sqrt{E}\to 1, \quad \text{as } N\to\infty.$$

However, this doles not help to *find* a good code: constructing a good code remains a challenging task in Information Theory. We will deal with this problem in the next chapter of the course.

11.3. In the case of an m.b.s.c. with the row error probability $p\in(0,1/2)$, the m.l. decoder looks for a codeword $x_*^{(N)}$ that has maximum number of digits coinciding with the received word $y^{(N)}$. In fact, if $y^{(N)}$ is received, the m.l. decoder compares the probabilities

$$\mathsf{P}\left(y^{(N)}\middle|x^{(N)}\right) = p^{d(x^{(N)},y^{(N)})}(1-p)^{N-d(x^{(N)},y^{(N)})}$$

$$= (1-p)^N\left(\frac{p}{1-p}\right)^{d(x^{(N)},y^{(N)})}$$

for different codewords $x^{(N)}$. Here,

$$d(x^{(N)},y^{(N)}) = \text{ the number of digits } i \text{ with } x_i\ne y_i. \tag{11.5}$$

As the first factor $(1-p)^N$ does not depend on $x^{(N)}$, the decoder seeks to maximize the second factor, that is to minimize $d(x^{(N)},y^{(N)})$ (since $0<\frac{p}{1-p}<1$.) Quantity $d(x^{(N)},y^{(N)})$ deserves a detailed discussion. We begin with the following observation.

Lemma 11.4. *$d(x^{(N)},y^{(N)})$ defines a distance on $\{0,1\}^N$. That is,*
(i) $0\le d(x^{(N)},y^{(N)})\le N$ *and* $d(x^{(N)},y^{(N)})=0$ *iff* $x^{(N)}=y^{(N)}$.
(ii) $d(x^{(N)},y^{(N)}) = d(y^{(N)},x^{(N)})$.
(iii) $d(x^{(N)},z^{(N)})\le d(x^{(N)},y^{(N)}) + d(y^{(N)},z^{(N)})$ (the triangle inequality).

Proof of Lemma 11.4. (i) and (ii) are obvious. (iii) is straightforward: any digit i with $z_i\ne x_i$ has either $y_i\ne x_i$ and then counted in $d(x^{(N)},y^{(N)})$ or $z_i\ne y_i$ and then counted in $d(y^{(N)},z^{(N)})$. □

Distance $d(x^{(N)},y^{(N)})$ is called the *Hamming distance*, and the space of binary words $\{0,1\}^N$ with distance $d(x^{(N)},y^{(N)})$ is called the *Hamming space of length N*. It contains 2^N elements.

Geometrically, the Hamming space may be identified with the collection of the vertices of a unit cube in N dimensions. The Hamming distance equals the lowest number of edges you have to pass from one vertex to another. It is a good practice to plot pictures for relatively low values of N:

$N = 2$ $N = 3$ $N = 4$ ...

As in any metric space, we can consider a ball of a given radius about a given word in the Hamming space. A ball of radius R about a word $x^{(N)}$ is given by

$$\mathrm{B}_R(x^{(N)}) = \{y^{(N)} : \ d(x^{(N)}, y^{(N)}) \leq R\}. \tag{11.6}$$

[An important (and hard) problem is to calculate the maximal number of disjoint balls of a given radius which can be packed in the Hamming space.]

Another important observation is that binary words admit an operation of addition modulo 2:

$$x^{(N)} + y^{(N)} = (x_1 + y_1) \mathrm{mod}\ 2 \ \ldots \ (x_N + y_N) \mathrm{mod}\ 2 \tag{11.7}$$

[Recall the rule of the binary arithmetic: $1 + 1 = 0$ mod 1.] This makes the Hamming space a commutative group, with the zero codeword $\mathbf{0} = 0 \ldots 0$ playing the role of the zero of the group. Each element of this group is opposite to itself: $x^{(N)} + x'^{(N)} = 0$ iff $x^{(N)} = x'^{(N)}$.

Henceforth, all operations over the binary words are understood in the sense of the binary arithmetic.

Lemma 11.4. *The Hamming distance on $\{0,1\}^N$ is invariant under the group translations:*

$$d(x^{(N)} + z^{(N)}, y^{(N)} + z^{(N)}) = d(x^{(N)}, y^{(N)}) \tag{11.8}$$

Proof of Lemma 11.4. If $z_i = 0$, the corresponding digits are in the same relation in words $x^{(N)} + z^{(N)}$ and $y^{(N)} + z^{(N)}$ as in words $x^{(N)}$ and $y^{(N)}$. But the same is true if $z_i = 1$. □

The next lecture opens a new chapter of the course, related to the theory of codes. We learned from the Shannon coding theorems that, under certain conditions, there exist asymptotically good codes that attain the limits imposed by the information rate of a source and the capacity of a channel. Moreover, we observed in Remark 11.2 that almost all codes are asymptotically good. However, in a practical situation, these facts are of a limited use: one wants to have a good code in an explicit form. Besides, as was discussed in Lecture 1, it is desirable to have a code that leads to fast encoding and decoding and maximizes the rate of the information transmission.

We will continue concentrating on the binary case where the symbols sent through a channel are 0 and 1. Moreover, we also assume that the source emits binary strings $u^{(n)} = u_1, \ \ldots, \ u_n, \ u_i = \begin{matrix} 0 \\ 1 \end{matrix}$. We have seen that, to obtain the overall error probability vanishing as $n \to \infty$, we have to encode words $u^{(n)}$ by longer codewords $x^{(N)}$ ($N \sim C^{-1}n$). Word $x^{(N)}$ is sent to the channel and is transformed into another word, $y^{(N)}$. It is convenient to represent the *error* occurred by the difference of the two words: $e^{(N)} = y^{(N)} - x^{(N)}$, or equivalently, write $y^{(N)} = x^{(N)} + e^{(N)}$, in the sense of (11.8).

Thus, the more digits 1 the error word $e^{(N)}$ has, the more symbols are distorted by the channel. The m.l. decoder then produces a 'guessed' codeword $x_\star^{(N)}$ that may or may not coincide with the $x^{(N)}$, and then reconstructs a string $u_\star^{(n)}$. In the case of an one-to-one encoding rule, the last procedure is (theoretically) straightforward: you simply invert the map $u^{(n)} \to x^{(N)}$. Also, it is clear that the order of the codewords does not play any important role, and so a code may be identified with the *set of codewords* $\mathcal{X}_N \subset \{0,1\}^N$.

Intuitively, a 'good' code is the one that allows the receiver to 'correct' the error $e^{(N)}$, at least in the case where word $e^{(N)}$ does not contain 'too many' digits 1.

As a convenient model of a channel we consider the m.b.s.c., with the row probability of the error $p < 1/2$. The m.l. decoder is then identified with a choice of a codeword $x_\star^{(N)}$ that leads to a word $e^{(N)}$ with a minimal number of the unit digits. In geometrical terms:

$$\begin{aligned} &x_\star^{(N)} \in \mathcal{X}_N \textbf{ is a codeword closest to } y^{(N)} \\ &\textbf{in the Hamming distance } d \textbf{ on } \{0,1\}^N. \end{aligned} \tag{11.9}$$

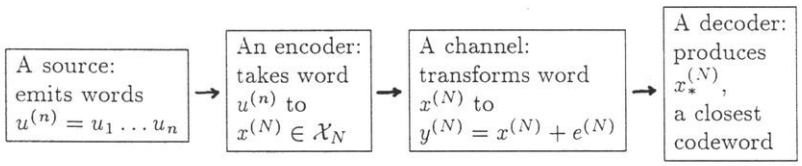
A drawback of this rule is that if you have more than one codeword at the same minimal distance from a received word you are 'stuck'. In this case you either choose a codeword arbitrarily (possibly, randomly), or, when you require a high quality of transmission, refuse to decode a received word and demand a re-transmission.

Chapter 3: Coding theory

Lecture 12: Bounds for codes

Recall, a code is identified below with a set of codewords $\mathcal{X}_N \subset \{0,1\}^N$.

Return to our basic scheme:



Definition 12.1. N is called the length of code $\mathcal{X}_N$, $r = \sharp\, \mathcal{X}_N$ the size and $\rho = \dfrac{\log r}{N}$ the transmission rate. A code $\mathcal{X}_N$ is said to be D-error *detecting* if making up to D changes in any codeword does not produce another codeword, and E-error *correcting* if making up to E changes in any codeword $x^{(N)}$ produces a word which is still (strictly) closer to $x^{(N)}$ than to any other codeword (that is, $x^{(N)}$ is correctly guessed from a distorted word). A code has minimal distance (or briefly distance) δ if

$$\delta = \min\,[d(x^{(N)}, x^{(N)}) :\ x^{(N)}, x'^{(N)} \in \mathcal{X}_N,\ x^{(N)} \neq x'^{(N)}]. \tag{12.1}$$

Examples of codes and ways in which they are produced from each other may be found in Questions 1,2 in Example Sheet 3.

Theorem 12.1 below is a simple exercise based on defintions:

Theorem 12.1. (a) *Code $\mathcal{X}_N$ is D-error detecting iff its distance $\delta \geq D+1$.*

(b) *Code $\mathcal{X}_N$ is E-error correcting iff the balls of radius E about the codewords are disjoint.*

Proof of Theorem 12.1. (a) is obvious. To prove (b), assume first that the balls of radius E are disjoint. Then, changing up to E digits in a codeword produces a word that is still in the corresponding ball, and hence is further apart from any other codeword. Conversely, let our code be E-error correcting. Then any word obtained by changing precisely E digits in a codeword does not fall in any ball of radius E but in the one about the original codeword. If you make less changes you again do not fall in any other ball, for if you do then, moving towards the second center will produce, soon or later, a word that is at distance E from the original codeword and at distance $< E$ from the second one, which is impossible. □

Clearly, if a code detects D errors and D is even then it corrects $E = D/2$ errors, and if D is odd than it corrects $(D-1)/2$ errors.

Observe that the 'volume' of the ball in the Hamming space is

$$\nu_N(R) = \mathrm{vol}\,(\mathbb{B}_R(z^{(N)})) = \sum_{i=0}^{R} \binom{N}{i}. \tag{12.2}$$

From Theorem 12.1 you see that you can't have the number of the codewords too high if you want to keep good a error-detecting and error-correcting ability. There are various bounds for parameters of codes. we begin with a bound discovered by Hamming in the late 40's:

Theorem 12.2. (The Hamming bound). *If a code $\mathcal{X}_N$ corrects E errors then its size $r = \sharp\, \mathcal{X}_N$ obeys*

$$r \leq \frac{2^N}{\nu_N(E)}. \tag{12.3}$$

Proof of Theorem 12.2. The E-balls about the codewords $x^{(N)} \in \mathcal{X}_N$ must be disjoint. Hence, the total number of points covered is $r\nu_N(E)$ and it should be $\leq 2^N$, the cardinality of the Hamming space $\{0,1\}^N$. □

You see that a 'good' code $\mathcal{X}_N$ correcting E errors must give a 'close-packing' of the Hamming space by balls of radius E. The problem of finding good codes becomes a *geometrical* problem.

If you manage to find a code $\mathcal{X}_N$ that gives you a 'true' close-packing partition, you have an additional advantage: your code not only corrects errors, but never leads to a refusal of decoding. More precisely:

Definition . An E-error correcting code $\mathcal{X}_N$ of size $\sharp\, \mathcal{X}_N = r$ is called *perfect* when the equality is achieved in the Hamming bound:

$$r = \frac{2^N}{\nu_N(E)}.$$

If a code $\mathcal{X}_N$ is perfect, every word $y^{(N)} \in \{0,1\}^N$ belongs to a (unique) ball $\mathbb{B}_E(x^{(N)})$. That is, you are always able to decode $y^{(N)}$ by a codeword: this leads to the correct answer if the number of errors is $\leq E$, and to a wrong answer if it is $> E$. But you never get 'stuck' in the case of decoding.

The problem of finding all perfect codes was solved $\sim$ 20 years ago. These codes exist only for

(a) $E = 1$: here $N = 2^l - 1$, $r = 2^{2^l - 1 - l}$; the corresponding codes are called the Hamming codes;

(b) $E = 3$: here $N = 23$, $r = 2^{12}$; the corresponding code is called the (binary) Golay code. [It is now use[illegible] together with some modifications) in the US space programme:

the quality of photographs encoded and transmitted fr Mars and Venus was so excellent that it did not require any improving procedure. In the former SU space vessels (and early American ones) there were other codes used (we may come across of them later): they produced lower quality photographs, and further manipulations were needed, based on statistics of the pictures.] If you consider non-binary codes, with three or more symbols, then there exists one more perfect code, for three symbols.

Let us continue with bounds on codes in the binary case.

Theorem 12.3. (The Gilbert–Varshamov bound). *There exists a code $\mathcal{X}_N$ with minimal distance δ such that*

$$r = \sharp\, \mathcal{X}_N \geq \frac{2^N}{\nu_N(\delta - 1)}\,. \qquad \square\,(12.4)$$

Proof of Theorem 12.3. Consider a code of maximal size among the codes of minimal distance δ (and length N). Then any word $y^{(N)} \in \{0,1\}^N$ must be distant $\leq \delta - 1$ from some codeword: otherwise you can add $y^{(N)}$ to the code without changing the minimal distance. Hence, the balls of radius $\delta - 1$ about the codewords cover the whole Hamming space $\{0,1\}^N$. That is, for our code,

$$r\nu_N(\delta - 1) \geq 2^N.$$

$\square$

As was mentioned before, there are several ways of producing a code from another code (or from a collection of codes). See Example Sheet 3. One particular procedure is 'truncation': in each codeword $x^{(N)}$ from an original code $\mathcal{X}_N$ you drop the last digit x_N. If code $\mathcal{X}_N$ had the minimal distance $\delta > 1$ then the new code, $\mathcal{X}^-_{N-1}$, has the minimal distance $\geq \delta - 1$ and the same size as $\mathcal{X}_N$. The truncation procedure leads to the following bound.

Theorem 12.4. (The Singleton bound). *Any code $\mathcal{X}_N$ with minimal distance δ has*

$$r = \sharp\, \mathcal{X}_N \leq 2^{N-\delta+1}. \qquad \square\,(12.5)$$

Proof of Theorem 12.4. You can repeat the above truncation procedure $\delta - 1$ times, still preserving the size of code $\mathcal{X}_N$. At the end you should 'fit' into the Hamming space $\{0,1\}^{N-\delta+1}$ that contains $2^{N-\delta+1}$ elements. $\square$

Corollary 12.5. *If $r^*(N,\delta)$ is the maximal size of a code $\mathcal{X}_N$ with minimal distance δ then*

$$\frac{2^N}{\nu_N(\delta-1)} \leq r^*(N,\delta) \leq \min\left[\frac{2^N}{\nu_N([\delta/2])}, 2^{N-\delta+1}\right]. \qquad \square\,(12.6)$$

To simp the notation, we systematically omit henceforth the subscript N and the superscript (N) in $\mathcal{X}_N$ and $x^{(N)}$, $y^{(N)}$, etc.

The upper bound in (12.6) becomes too rough when $\delta \sim N/2$. For example, in the case $N = 10$ and $\delta = 5$, it gives the bound $r \leq 18$, whereas in fact there is no code with $r \geq 13$ (see Example 45 from Example Sheet 3), but there exists a code with $r = 12$. The codewords are as follows:

0000000000
1111100000
1001011010
0100110110
1100001101
0011010101
0010011011
1110010011
1001100111
1010111100
0111001110
0101111001

The lower bound gives in this case 2 and is also far from being satisfactory.

In fact, in the case $\delta \sim N/2$ there are better upper bounds than (12.6), called the Plotkin bounds. See the Appendix to Lecture 13.

Appendix to Lecture 12

The bounds provided in Theorems A.1 and A.3 below are known as the Plotkin bounds.

Theorem A.1. *For any code $\mathcal{X}$ of length N and distance δ with $N < 2\delta$,*

$$r = \sharp\, \mathcal{X} \leq 2\left[\frac{\delta}{2\delta - N}\right]. \qquad \square(A.1)$$

Theorem A.2. *Let $r^*(N,\delta)$ denote the maximal size of a code of length N and distance δ. Then, for any N and l,* (a)

$$r^*(N, 2l-1) = r^*(N+1, 2l). \qquad (A.2)$$

and (b)

$$r^*(N-1, l) = \frac{1}{2}r^*(N,l). \qquad \square(A.3)$$

Theorem A.3. *In the notation of Theorem A.2, if l is even and $2l > N$ then*

$$r^*(N,l) \leq 2\left[\frac{l}{2l-N}\right] \qquad (A.4)$$

and

$$r^*(2l, l) \le 4l. \tag{A.5}$$

If l is odd and $2l+1 > N$ then

$$r^*(N, l) \le 2\left[\frac{l+1}{2l+1-N}\right] \tag{A.6}$$

and

$$r^*(2l+1, l) \le 4l+4. \qquad \square(A.7)$$

Proof of Theorem A.1. You have an obvious bound

$$r(r-1)\delta \le \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}} d(x, x') \tag{A.8}$$

On the other hand, write code $\mathcal{X}$ as an $r \times N$ matrix with rows as codewords. Suppose that column i of the matrix contains s_i zero's and $r - s_i$ one's. Then

$$\sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}} d(x, x') \le 2\sum_{i=1}^{N} s_i(r - s_i). \tag{A.9}$$

If r is even, the RHS of (A.9) is maximized when $s_i = r_i/2$ which, together with (A.8), yields

$$r(r-1)\delta \le \frac{1}{2}Nr^2, \tag{A.10}$$

or

$$r \le \frac{2\delta}{2\delta - N}.$$

As r is even, this implies

$$r \le 2\left[\frac{\delta}{2\delta - N}\right].$$

If r is odd, the RHS of (A.9) is $\le N(r^2-1)/2$ which, together with (A.8) yields

$$r \le \frac{N}{2\delta - N} = \frac{2\delta}{2\delta - N} - 1.$$

This implies in turn that

$$r \le \left[\frac{2\delta}{2\delta - N}\right] - 1 \le 2\left[\frac{\delta}{2\delta - N}\right],$$

because $[2x] \le 2[x] + 1$. $\square$

Proof of Theorem A.2. (a) Let $\mathcal{X}^*$ be a code of length N, distance $2l-1$ and size $r^*(N, 2l-1)$. Take its parity-check extention $\mathcal{X}^+$ (cf Example 2(b) fror ⁻xample Sheet 3). That is, add digit x_{N+1} to each codeword $x_1 \ldots x_N$ so that $\sum_{i=1}^{N+1} x_i = 0$. Then $\mathcal{X}^+$ is a code of length $N+1$, the same size $r^*(N, 2l-1)$ and distance $2l$. Therefore,

$$r^*(N, 2l-1) \le r^*(N+1, 2l).$$

Similarly, deleting the last digit leads to the inverse:

$$r^*(N, 2l-1) \ge r^*(N+1, 2l).$$

(b) Given a code of length N and distance l, divide the codewords into two classes: those ending with 0 and those ending with 1. One class must contain at least half of the codewords. Hence the result. $\square$

Proof of Theorem A.3. (A.4) follows from (A.1). (A.5) follows from (A.1) and (A.3):

$$r^*(4l, 2l) \le 2r^*(4l-1, 2l) \le 8l.$$

(A.6) follows from (A.2):

$$r^*(N, l) = r^*(N+1, l+1) \le 2\left[\frac{l+1}{2l+1-N}\right].$$

Finally, (A.7) follows from (A.2) and (A.5). $\square$

Lecture 13: Asymptotical bounds. Linear codes

It is interesting to write an asymptotical form of bounds (12.6). First, a technical lemma:

Lemma 13.1. *Let $\lambda \in (0, \frac{1}{2})$. Then*

$$\lim_{N\to\infty} \frac{1}{N} \log \nu_N([\lambda N]) = h(\lambda, 1-\lambda) \\ (= -\lambda \log \lambda - (1-\lambda)\log(1-\lambda) = G(\lambda) \quad \text{(see (6.2)))}. \tag{13.1}$$

Proof of Lemma 13.1. Use the formula

$$\nu_N(R) = \sum_{i=0}^{R} \binom{N}{i}, \quad R = [\lambda N].$$

The key remark is that the last term in the sum is the largest. In fact, consider the ratio of two successive terms:

$$\frac{\binom{N}{i+1}}{\binom{N}{i}} = \frac{N-i}{i+1} \le 1 \quad \text{for } 0 \le i \le R.$$

Hence,

$$\binom{N}{R} \le \nu_N(R) \le (R+1)\binom{N}{R}.$$

Now use Stirling's formula: $N! \sim N^{N+\frac{1}{2}} e^{-N}\sqrt{2\pi}$. Then

$$\log \binom{N}{R} = -(N-R)\, \log \frac{N-R}{N} - R \log \frac{R}{N} + 0(\log N)$$

and

$$\left(1-\frac{R}{N}\right)\log\left(1-\frac{R}{N}\right) - \frac{R}{N}\log\frac{R}{N} + \frac{1}{N}0(\log N) \le \frac{1}{N}\log \nu_N(R) \\ \le \frac{1}{N}\log(R+1) + \text{ the LHS}.$$

The limit $\frac{R}{N} \to \lambda$ yields the result.

We want to study the asymptotics of $r^*(N, [\lambda N])$, the maximal size of the code of length N and distance $[lN]$, as $N \to \infty$. That is, the asymptotics of the maximal size of a code that detects and corrects a number of errors which grows linearly with N, the length of a code. More precisely, we analyze

$$\lim_{N\to\infty} \frac{1}{N} \log r^*(N, [\lambda N]) = \alpha(\lambda), \quad 0 < \lambda < 1/2.$$

Theorem 13.2.

(a) $\alpha(\lambda) \le 1 - G(\lambda/2)$ *(Hamming)*, (13.2)

(b) $\alpha(\lambda) \le 1 - \lambda$ *(Singleton)*, (13.3)

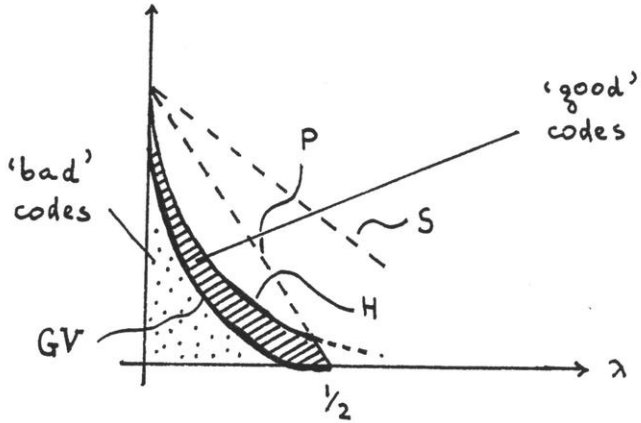
(c) $\alpha(\lambda) \ge 1 - G(\lambda)$ *(Gilbert–Varshamov)*. (13.4)

Proof of Theorem 13.2. By inspection.

By performing an asymptotical procedure similar to the above ones, for the Plotkin bound (see the Appendix), one obtains an asymptotical Plotkin bound:

$$\alpha(\lambda) \le 1 - 2\lambda \tag{13.5}$$

A figure below shows the asymptotical behaviour of the bounds established. 'Good' codes are the ones between the asymptotical Gilbert-Varshamov, Hamming and Plotkin bounds. *



Until 1973, there was no explicit construction known, which leads to codes achieving the asymptotical GV bound. [All known constructions led to codes below the the asymptotical GV curve.] The examples of codes that achieve the GV bound vere constructed by using algebraic-geometrical ideas.

The Gilbert–Varshamov bound itself is questionable. Until 1987 there was no better lower bound known (and in the case binary coding there is still no better lower bound known). However if the alphabet used contains $a \ge 49$ symbols, there exists a construction, again based on algebraic geometry, which produces better lower bound and gives examples codes that asymptotically exceed, as $N \to \infty$, the GV curve.

Practically all codes used in modern practice are *linear*:

Definition 13.1. A code $\mathcal{X}$ is called linear if, together with a pair of codewords, x and x', $\mathcal{X}$ contains the sum $x + x' (= x + x' \bmod 2)$, with digits $(x_i + x'_i) \bmod 2$.

* There are about a dozen of various upper bounds known, competing with each other; some of them are asymptotically insignificant (like the Singleton bound), although quite important in particular domains of parameters.

[In other words, $\mathcal{X}$ is a *linear subspace* of $\{0,1\}^N$, over the field $\{0,1\}$.**]

Linear codes are popular because they are easy to work with. For example, to identify a linear code $\mathcal{X}$ you only have to fix a 'basis' in the corresponding subspace. A *basis*, as usual, is a maximal linearly independent set of words (I will sometimes call them 'vectors'); a linear subspace is, as usual, generated by its basis, and all bases in a given subspace have the same number of words: this number is called the *dimension* or the *rank* of the subspace. A linear code $\mathcal{X}$ of length N and rank k is also called an (N,k) code. A linear code of length N, rank k and distance d is called an (N,K,d)-code.

Note that any linear code contains the zero codeword $\mathbf{0}$.

Lemma 13.3. *Any linear subspace of $\{0,1\}^N$ of rank k contains 2^k vectors.*

Proof of Lemma 13.3. A basis of the subspace contains k linearly independent vectors. The whole subspace is generated by the basis; hence it consists of the sums of basic vectors. There are precisely 2^k sums (the number of subsets of $\{1,\dots,k\}$ indicating the summands), and they all give different vectors.

Thus, any linear code $\mathcal{X}_N$ has $\sharp(\mathcal{X}_N) = 2^k$, where $k = \mathrm{rank}(\mathcal{X}_N)$. [That is, an (N,k) code may be used for encoding *all* possible source strings of length k.] But to identify set $\mathcal{X}_N$ you only need to indicate k linearly independent words. In other words, a linear code of rank k is chracterized by an $k \times N$ matrix of 0's and 1's, whose rows are linearly independent:

$$G = \begin{pmatrix} g_{11} & \dots & \dots & \dots & g_{1N} \\ g_{21} & \dots & \dots & \dots & g_{2N} \\ \vdots & & & & \vdots \\ g_{k1} & \dots & \dots & \dots & g_{kN} \end{pmatrix} \tag{13.6}$$

Namely, you take the rows $(g_{i1},\dots,g_{iN})$ as the basic vectors of a linear code. Matrix G is called a *generating matrix* or *generator* of a linear code. It is clear that two matrices may generate the same code: the relation between them is discussed in examples from Example Sheet 3.

Equivalently, a linear (N,k)-code $\mathcal{X}$ may be described as the kernel of a certain matrix H, again with the entries 0 and 1:

$$H = \begin{pmatrix} h_{11} & \dots & h_{1k} \\ h_{21} & \dots & h_{2k} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ h_{N1} & \dots & h_{Nk} \end{pmatrix} \tag{13.7}$$

$$\ker H = \{x^{(N)} - x_1 \dots x_N : x^{(N)} H = 0\} \tag{13.8}$$

** A *field* is a set endowed with operations of addition and multiplication so that natural properties of commutativity, associativity and distributivity ar lid. See below.

It is plain that the columns of H are vectors *orthogonal* to $\mathcal{X}$, in the sense of the 'usual' scalar product***

$$\langle x \cdot y \rangle = \sum_{i=1}^{N} x_i y_i. \tag{13.9}$$

Matrix H is called a *parity check* (or simply check) matrix of code $\mathcal{X}$. The parity check matrix is again not unique. If code $\mathcal{X}_N$ has rank k then H must contain $N-k$ linearly independent columns. For definiteness, we discard redundant columns and always think of H as an $N \times (N-k)$ matrix with linearly independent columns.

In many cases, the description of a code by a check matrix is more convenient than by a generator.

Example 13.1. (The Hamming (7,4) code). The code is determined by a 7×3 parity check matrix. The rows of the check matrix are all non-zero words of length 3. If you order these words lexicographicaly, you obtain

$$H^{\mathrm{lex}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}. \tag{13.10}$$

The corresponding generating matrix may be written as

$$G^{\mathrm{lex}} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{13.11}$$

In many cases it is convenient to write the check matrix of a linear (n,k) code in a *canonical* (or *standard*) form:

$$H^{\mathrm{can}} = \begin{pmatrix} I_{N-k} \\ H' \end{pmatrix}. \tag{13.12}$$

In the case of the Hamming (7,4) code it gives

$$H^{\mathrm{can}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{13.13}$$

*** Scalar product (13.9) (also called sometimes the dot–product) possesses all properties of the Euclidean scalar product in $\mathbf{R}^N$, but one: it is not *positive definite* (and therefore does not define a norm). That is, there are non-zero vectors $x \in \{0,1\}^N$ with $\langle x \cdot x \rangle = 0$. Luckily, do not need the positive definiteness.

with a generating matrix also in a canonical form:

$$G^{\rm can} = \left(G' \ I_k \right). \tag{13.14}$$

Namely,

$$G^{\rm can} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{13.15}$$

Formally, $G^{\rm lex}$ and $G^{\rm can}$ determine *different* codes. However, these codes are *equivalent*:

Definition 13.2. Two codes are called equivalent if they differ only in the order of digits.

In the sequel, unless otherwise stated, we do *not* distinguish between equivalent linear codes.

Definition 13.3. A *weight* $w(x)$ of a word x is the number of the non-zero digits in x:

$$w(x) = \{i : 1 \leq i \leq N,\ x_i \neq 0\}. \tag{13.16}$$

Remark 13.4. An advantage of writing G in a canonical form is that a source string $u^{(k)}$ is encoded as an N-vector $u^{(k)}G^{\rm can}$; according to (13.14), the last k digits form word $u^{(k)}$ (they are called information digits, whereas the first $N-k$ are used for the parity-check (and called parity-check digits). Pictorially, the parity-check digits carry the redundancy that allows the decoder to detect and correct errors.

Theorem 13.5.

(i) *The distance of a linear code equals the minimal weight of its codewords.*

(ii) *The distance of a linear code equals the minimal number of linearly dependent sets of the rows in the check matrix.*

Proof of Theorem 13.5.

(a) As code $\mathcal{X}$ is linear, the sum $x+y \in \mathcal{X}$ for each pair of codewords $x, y \in \mathcal{X}$. [**Note**: The zero word $0 = x + x$ is *always* a codeword in a linear code.] Owing to the invariance property of the Hamming distance (see Lemma 11.5), $d(x,y) = d(0, x+y) = w(x+y)$ for any pair of codewords. Hence, the minimal distance of $\mathcal{X}$ equals the minimal distance between 0 and the rest of the code, i.e., the minimal weight of a non-zero codeword from $\mathcal{X}$.

(b) Let $\mathcal{X}$ be a linear code with a parity-check matrix H and minimal distance δ. Then there exists a codeword $x \in \mathcal{X}$ with exactly δ non-zero digits. Since $xH = 0$, you conclude that there are δ rows of H which are linearly dependent (they correspond to non-zero digits in x). On the other hand, if $\exists(\delta-1)$ rows of H which are linearly dependent then their sum is zero. But that means that $\exists$ a word y, of weight $w(y) = \delta - 1$. such that $yH = 0$. Then y must belong to $\mathcal{X}$ which is impossible, since $\min[w(x) : x \in \mathcal{X}, x \neq 0] = \delta$. □

Lecture 14: Linear codes (cont.). The Hamming codes

Theorem 14.1. *The Hamming (7,4) code has minimal distance 3, i.e. it detects 2 errors and corrects 1.*

Proof of Theorem 14.1. Rows 1, 6, 7 of the parity check matrix $H^{\rm lex}$ are linearly dependent (in fact, to any pair of rows you can add their sum: this gives you a linearly dependent triple). No two rows are linearly dependent because they are distinct ($x+y=0$ means that $x = y$). □

Theorem 14.2. *The Hamming (7,4) code is a perfect 1-error correcting code.*

Proof of Theorem 14.2. The volume of the 1-ball $\nu_7(1)$ equals $1 + 7 = 8 = 2^3$, and the size of the code is 2^4. $2^{4+3} = 2^7$. □

You immediately conclude that the construction of the Hamming (7,4) code admits a straightforward generalization. Namely, consider a matrix whose elements are all possibel non-zero words of length l:

$$H = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ & & \vdots & & \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix} \quad \begin{matrix} \uparrow \\ \\ \\ \\ \\ 2^l - 1 \\ \downarrow \end{matrix}$$

$$\longleftarrow \quad l \quad \longrightarrow$$

The columns of this matrix are linearly independent, and hence it may be considered as a generating matrix of a linear code of length $N = 2^l - 1$ and rank $N - l = 2^l - 1 - l$. Any two rows of H are linearly independent but there exist linearly dependent triples of rows, e.g.

$$\begin{matrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \end{matrix}$$

and

$$1 \quad 1 \quad \dots \quad 0.$$

[As before, for any pair of rows $\exists$ a third row such that the whole triple is linearly dependent.] Hence, the code with the check matrix H has a minimal distance 3, i.e.(N)it detects 2 errors and corrects 1.

This code is called the Hamming $(2^l - 1, 2^l - 1 - l)$ code. It is a *perfect* 1-error correcting code: the volume of the 1-ball $\nu_{2^l-1}(1)$ equals $1 + 2^l - 1 = 2^l$, and size $\times$ volume $= 2^{2^l-1-l} \times 2^l = 2^{2^l-1} = 2^N$. The information rate is $\dfrac{2^l - l - 1}{2^l - 1} \to 1$ as $l \to \infty$. Thus,

Theorem 13.4. *The above construction defines a family of $(2^l - 1, 2^l - 1 - l)$ linear codes, with minimal distance 3, which are perfect 1-error correcting codes.*

Until the late 1950s, the Hamming codes were a unique family of codes existing in dimensions $N \to \infty$, with 'regular' properties. It was then discovered that these codes have a deep *albebraic* background. The development of the algebraic methods based on these observations is still a dominant theme in the modern coding theory.

We will be able to cover some basic concepts and facts from the algebraic coding theory. Before moving in this direction, we need to discuss the decoding procedure for a general linear code. As a linear code, $\mathcal{X}$ contains 2^k words where k is the rank of $\mathcal{X}$. As was noted before, it may be used for encoding source messages (strings) $u = u_1 \ldots u_k$ of length k. The source encoding $u \in \{0,1\}^k \mapsto \mathcal{X}$ becomes particularly simple when you use the generating and parity check matrices in the canonical (or standard) form

$$G^{\rm can} = \underbrace{\left(G' \right.}_{N-k} \left. I_k \right), \quad H^{\rm can} = \begin{pmatrix} I_{N-k} \\ H' \end{pmatrix} \Big\updownarrow k \ ;$$

cf. notes for Lecture 13.

Theorem 14.4. *For any linear code $\mathcal{X}$ there exists an equivalent code $\mathcal{X}'$ (obtained from $\mathcal{X}$ by permutating digits), with the generating matrix $G^{\rm can}$ and the check matrix $H^{\rm can}$ in standard form (13.1), and $G' = H'$.*

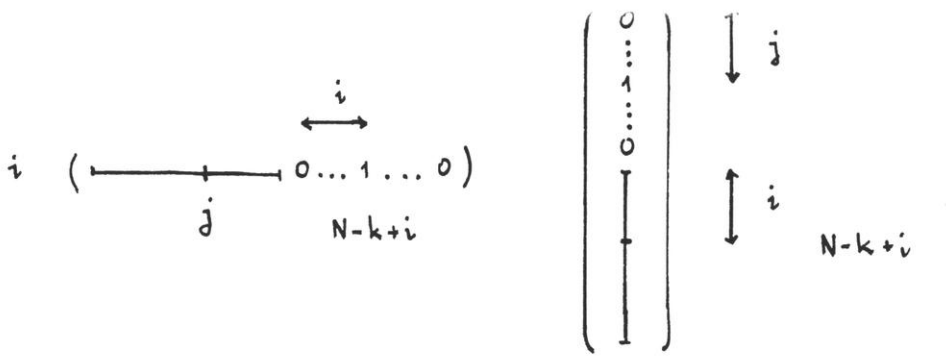
Proof of Theorem 14.4. Assume, *wlog*, that code $\mathcal{X}$ is non-trivial (that is, not reduced to the zero word 0 ... 0). Write a basis for $\mathcal{X}$ and take the corresponding generating matrix G. By performing row operations (where a pair of rows i and j is exchanged or row i is replaced by row i plus row j) we can change the basis, but do *not* change the code. Our matrix G contains a nonzero column, say l_1: perform row operations to make g_{1l_1} the only non-zero entry in this column. By permuting digits (columns), place column l_1 at poistion $N-k$. Drop row 1 and column $N-k$ (i.e., the old column l_1) and perform a similar procedure with the rest, ending up with the only non-zero entry g_{2l_2} in a column l_2. Place column l_2 at position $N-k+1$. Continue until you form an upper triangular $k \times k$ submatrix. Further operations may be reduced to this matrix only. If this matrix is a unit matrix, stop. If not, pick the first column with more than one non-zero entry. Add the corresponding rows from the bottom to 'kill' redundant non-zero entries. Repeat until you form a unit submatrix. You end up with a generating matrix in a standard form, and new code is equivalent to the original one.

To complete the proof, observe that matrices $G^{\rm can}$ and $H^{\rm can}$ figuring in (13.12), (13.14), with $G' = H'$, have k independent rows and $N-k$ independent columns, correspondingly. Besides, the $k \times (N-k)$ matrix $G^{\rm can} H^{\rm can}$ vanishes. In fact,

$$(G^{\rm can} H^{\rm can})_{ij} = \langle \text{ row } i \text{ of } G, \text{ column } j \text{ of } H \rangle$$

$$= g'_{ij} \cdot 1_j + 1_{N-k+i} h'_{ij} = g'_{ij} + g'_{ij} = 0.$$

See the diagram below.



Hence, $H^{\rm can}$ is a check matrix for $G^{\rm can}$. □

Returning to source encoding, suppose that generating matrix is in the canonical form $G^{\rm can}$. Then, given a string $u = u_1, \ldots, u_k$, you set $x = \sum_{i=1}^k u_i g_i^{\rm can}$, where $g_i^{\rm can}$ represents row i of $G^{\rm can}$. The last k digits in x give string u; they are called the information digits, whereas the first $N-k$ digits are used to ensure that $x \in \mathcal{X}$; they are called the parity check digits.

The idea of the *decoding* procedure for linear codes is also relatively simple. Recall that we agreed to decode a word $y = y_1 \ldots y_N$ by the closest codeword $x \in \mathcal{X}$.

Definition. Let $\mathcal{X}$ be a linear code of length N, and $u = u_1 \ldots u_N$ be a word from $\{0,1\}^N$. The *coset* of $\mathcal{X}$ determined by u is the set of all words of the form $u + x$ where $x \in \mathcal{X}$. We denote it by $u + \mathcal{X}$.

Theorem 14.5. *Let $\mathcal{X}$ be a linear code and u, v be words of length N. Then:*

(1) *If u is in the coset $v + \mathcal{X}$, then v is in the coset $u + \mathcal{X}$; in other words, each word in a coset determines this coset.*

(2) $u \in u + \mathcal{X}$.

(3) *u and v are in the same coset iff $u + v \in \mathcal{X}$.*

(4) *Every word of length N belongs to one and only one coset. That is, the cosets form a partition of the whole Hamming space $\{0,1\}^N$.*

(5) *All cosets contain the same number of words which equals $\sharp\, \mathcal{X}$. If the rang of $\mathcal{X}$ is k then there are 2^{N-k} different cosets, each containing 2^k words. Code $\mathcal{X}$ is itself a coset of any of the codewords.*

(6) *The coset determined by $u + v$ coincides with the set of elements of the form $x + y$, where $x \in u + \mathcal{X}$, $y \in \mathcal{X} + v$.*

Proof of Theorem 14.5. (An easy (and useful) exercise in linear algebra and set theory.)

Now the decoding rule for a linear code: you know the code $\mathcal{X}$ beforehand, hence you can calculate all cosets. Upon receiving a word y, you find its coset $y + \mathcal{X}$ and find a word $u \in y + \mathcal{X}$ of least weight. Such a word is called a *leader* of coset $y + \mathcal{X}$. A leader may not be unique: in that case you choose among the leaders arbitrarily (which of course

may lead to an error) or refuse to decode and demand a re-transmission. Suppose you have chosen a leader u. You then decode y by the word

$$x = y + u.$$

□

Theorem 14.6. *Word x is always a codeword that minimizes the distance between y and the words from $\mathcal{X}$.*

Proof of Theorem 14.6. As y and u are in the same coset, $y + u \in \mathcal{X}$ (see Theorem 14.5(3)). All other words from $\mathcal{X}$ are obtained as the sums $y + v$ where v runs over coset $y + \mathcal{X}$. Hence, for any $x \in \mathcal{X}$,

$$d(y, x) = w(y + x) \geq \min_{v \in y + \mathcal{X}} w(v) = w(u) = d(y, x).$$

□

The parity-check matrix provides a convenient description of the cosets $\mathcal{X}_N + u^{(N)}$.

Theorem 14.7. *Cosets $u + \mathcal{X}$ are in one-to-one correspondence with vectors of the form yH: two vectors, y and y' are in the same coset iff $yH = y'H$.*

In other words, cosets are identified with the rank (or range) space of the parity-check matrix.

Proof of Theorem 14.7. y and y' are in the same coset iff $y + y' \in \mathcal{X}$, i.e.

$$(y + y')H = yH + y'H = 0, \text{ i.e., } y \overset{H}{=} y'$$

□

In practice, the decoding rule is implemented as follows. Vectors of the form yH are called *syndromes*: for a linear (N, k) code there are 2^{N-k} syndromes. They are all listed in the syndrome 'table', and for each syndrome a leader of the corresponding coset is calculated. Upon receiving a word y, you calculate the syndrome yH and find, in the syndrome table, the corresponding leader u. Then set, as before,

$$x = y + u.$$

The procedure described is called *syndrome decoding*: although it is relatively simple, one has to write a rather long table of the leaders. Moreover, it is desirable to make the whole procedure of decoding algorithmically independent on a concrete choice of the code, i.e. of its generating matrix. This goal is achieved in the case of the Hamming codes:

Theorem 14.8. *For the Hamming code, for each syndrome the leader u is unique and contains not more than one non-zero digit. More precisely, if the syndrome $yH = s$ gives row i of the check matrix then the leader of the corresponding coset has the only non-zero digit i.*

Proof of Theorem 14.8. The leader minimizes the distance between the received word and the code. The Hamming code is perfect 1-error correcting. Hence, every word is either a codeword or within distance 1 of a unique codeword. Hence, the leader is unique and contains at most one non-zero digit. If the syndrome $yH = s$ occupies position i among the rows of the parity-check matrix then, for word $e_i = (0 \ \ldots \ 1 \ 0 \ \ldots \ 0)$ with the non-zero digit i,

$$(y + e_i)H = s + s = 0.$$

That is, $(y + e_i) \in \mathcal{X}$ and $e_i \in y + \mathcal{X}$. Obviously, e_i is the leader. □

Lecture 15: Cyclic codes

Our next goal is to study a special class of codes which contains the Hamming codes. This class is formed by the so-called *cyclic* codes. Before this class is introduced, we discuss some related 'polynomial' algebra.

We consider polynomials with coefficients 0 and 1:

$$a(X) = a_0 + a_1 X + \cdots + a_N X^N, \quad a_k = {}^{0}_{1}, \quad k = 0, \ldots, N.$$

These polynomials are added and multiplied in the usual fashion, except that $X^k + X^k = 0$. In the sequel we set $\deg 0 = 0$.

Examples. 1. $(1 + X + X^3 + X^4)(X + X^2 + X^3) = X + X^7$.

2. $1 + X^N = (1 + X)(1 + X + \cdots + X^{N-1})$.

3. $(1 + X)^{2^l} = 1 + X^{2^l}$ (A freshman's dream).

Theorem 15.1. (The division algorithm). *Let $f(X)$ and $h(X)$ be two polynomials with $h(X) \neq 0$. Then there exist unique polynomials $g(X)$ and $r(X)$ such that*

$$f(X) = g(X)h(X) + r(X) \quad \textit{with} \quad \deg r(X) < \deg h(X). \qquad \square$$

Polynomial $g(X)$ is called the quotient and $r(X)$ the remainder.

Proof of Theorem 15.1. If $\deg h(X) > \deg f(X)$ you simply set

$$f(X) = 0 \cdot h(X) + f(X).$$

If $\deg h(X) \leq \deg f(X)$, you can perform the 'standard' procedure of long division, with the rules of the binary addition and multiplication. $\square$

Example 4. The quotient $X + X^2 + X^6 + X^7 + X^8 \,/\, 1 + X + X^2 + X^4 = X^3 + X^4$, the remainder is $X + X^2 + X^3$.

Definition 15.1. Two polynomials, $f_1(X)$ and $f_2(X)$ are called equivalent modulo $h(X)$ if their remainders, after division by $h(X)$, coincide. That is,

$$f_i(X) = g_i(X)h(X) + r(X), \quad i = 1, 2$$

and $\deg r(X) < \deg h(X)$. In this case we write $f_1(X) = f_2(X) \bmod h(X)$.

Theorem 15.2. *Addition and multiplication of polynomials respect the equivalence. That is, if*

$$f_1(X) = f_2(X) \bmod h(X) \quad \textit{and} \quad p_1(X) = p_2(X) \bmod h(X)$$

then

$$f_1(X) + p_1(X) = f_2(X) + p_2(X) \bmod h(X)$$

and

$$f_1(X)p_1(X) = f_2(X)p_2(X) \bmod h(X).$$

Proof of Theorem 15.2. We have, for $i = 1, 2$,

$$f_i(X) = g_i(X)h(X) + r(X), \quad p_i(X) = q_i(X)h(X) + s(X),$$

with

$$\deg r(X),\ \deg s(X) < \deg h(X).$$

Hence

$$f_i(X) + p_i(X) = (g_i(X) + q_i(X))h(X) + (r(X) + s(X))$$

with

$$\deg (r(X) + s(X)) \leq \max[r(X), s(X)] < \deg h(X).$$

Thus

$$f_1(X) + p_1(X) = f_2(X) + p_2(X) \bmod h(X).$$

Furthermore, for $i = 1, 2$,

$$f_i(X)p_i(X) = \Big(g_i(X)q_i(X)h(X) + r(X)q_i(X) + s(X)g_i(X)\Big)h(X) + r(X)s(X).$$

Hence, the remainder for both polynomials $f_1(X)p_1(X)$ and $f_2(X)p_2(X)$ may come only from $r(X)s(X)$. Thus it is the same for both of them. $\square$

Return to linear codes. The first remark is that every linear code $\mathcal{X}_N$ corresponds to a set of polynomials, with coefficients ${}^{0}_{1}$, of degree $N - 1$ which is closed under addition mod 2:

$$\begin{aligned} a(X) = a_0 + \cdots + a_{N-1} \quad &\leftrightarrow \quad a^{(N)} = a_0 \ldots a_{N-1} \\ b(X) = b_0 + \cdots + b_{N-1} \quad &\leftrightarrow \quad b^{(N)} = b_0 \ldots b_{N-1} \\ a(X) + b(X) \quad &\leftrightarrow \quad a^{(N)} + b^{(N)} \bmod 2 \ = a_0 + b_0 \ldots a_{N-1} + b_{N-1}. \end{aligned}$$

[We changed the numeration of the digits in a word of lengths N: instead of numbers 1, ..., N we now use 0, ..., $N - 1$ which is more convenient. Also, instead of x, y, etc, we use for the words the notation a, b, etc.]

In what follows we systematically write $a(X) \in \mathcal{X}$ in the case word $a^{(N)} = a_0 \ldots a_{N-1}$, representing polynomial $a(X)$, belongs to $\mathcal{X}$.

Given a word $a = a_0 \ldots a_{N-1}$, we define the cyclic shift πa as a word $a_{N-1}a_0 \ldots a_{N-2}$.

Definition 15.2. A linear code $\mathcal{X}$ is called cyclic if the cyclic shift of each codeword is again a codeword.

A 'straightforward' way to form a cyclic code is as follows: take a word a, then its subsequent cyclic shifts πa, $\pi^2 a$ etc., and finally all sums of the vectors obtained. Such

a construction allows you to build a code from a single w. ., and eventually all the properties of the code may be inferred from the properties of word a.

It turns out that *every* cyclic code may be obtained in such a way: the corresponding word is called a generator of a cyclic code.

Lemma 15.3. *A code $\mathcal{X}$ is cyclic iff, for any vector a from a basis of $\mathcal{X}$, $\pi a \in \mathcal{X}$.*

Proof of Lemma 15.3. Each vector $u \in \mathcal{X}$ is a sum of the vector of the basis. But $\pi(u+v) = \pi u + \pi v$, hence the result. □

An important property of a cyclic shift is established in Lemma 15.4:

Lemma 15.4. *If word a corresponds to a polynomial $a(X)$ then word πa corresponds to $Xa(X) \bmod (1+X^N)$.*

Proof of Lemma 15.4. You can write

$$\begin{aligned} Xa(X) &= a_0X + a_1X^2 + \cdots + a_{N-2}X^{N-1} + a_{N-1}X^N \\ &= a_{N-1} + a_0X + a_1X^2 + \cdots + a_{N-2}X^{N-1} + a_{N-1}X^N + a_{N-1} \\ &= a_{N-1} + a_0X + a_1X^2 + \cdots + a_{N-2}X^{N-1} + a_{N-1}(X^N+1) \end{aligned}$$

which means that the polynomial

$$a_{N-1} + a_0X + \cdots + a_{N-2}X^{N-2}$$

corresponding to πa equals $Xa(X) \bmod (1+X^N)$. □

Continuing in the same way, you can argue that word x^2 corresponds to $X^2a(X) \bmod (1+X^N)$, etc. A corollary of these facts is the following theorem.

Theorem 15.5. *A cyclic code contains, with each pair of polynomials $a(X)$ and $b(X)$, the sum $a(X)+b(X)$ and any polynomial $v(X)a(X) \bmod (1+X^N)$.*

Proof of Theorem 15.5. The sum $a(X)+b(X) \in \mathcal{X}$ because of linearity. If $v(X) = v_0 + v_1X + \cdots v_{N-1}X^{N-1}$ then each polynomial $X^ka(X) \bmod (1+X^N)$ corresponds to $\pi^k a$ and hence belongs to $\mathcal{X}$. As

$$v(X)a(X) \bmod (1+X^N) = \sum_{i=0}^{N-1} v_i(X^ia(X) \bmod (1+X^N)),$$

the LHS belongs to $\mathcal{X}$. □

In other words, if you introduce the operation of multiplication of polynomials

$$a(X) \star b(X) = a(X)b(X) \bmod (1+X^N),$$

then the polynomials of degree $\leq N-1$, with the $\star$-multiplication and the usual addition, form a commutative *ring*, and cyclic codes are precisely the *ideals* of this ring.

Theorem 15.6. *Let $c(X) = \sum_{i=0}^{N-k} c_iX^i$ be a non-zero polynomial of minimum degree in a cyclic code $\mathcal{X}$. Then*

(i) *$c(X)$ is a unique polynomial of minimal degree,*

(ii) *code $\mathcal{X}$ has rank k,*

(iii) *the codewords corresponding to $c(X), \ldots, X^{k-1}c(X)$, form a basis in $\mathcal{X}$; they are cyclic shifts of word $c_0 \ldots c_{N-k}0 \ldots 0$,*

(iv) *$a(X) \in \mathcal{X}$ iff $a(X) = v(X)c(X)$ for some polynomial $v(x)$ of degree $< k$ (that is, $c(X)$ is a divisor of every polynomial from $\mathcal{X}$.)*

Proof of Theorem 15.6.

(i) Suppose $c'(X) = \sum_{i=0}^{N-k} c'_iX_i$ is another polynomial of minimal degree $N-k$ in $\mathcal{X}$. Then $c_{N-k} = c'_{N-k} = 1$, and hence $\deg(c'(X)+c(X)) < N-k$. But as $N-k$ is the minimal degree, $c'(X)+c(X)$ should equal zero. But this happens iff $c(X) = c'(X)$. Hence, $c(X)$ is unique.

(ii) follows from (iii).

(iii) Assume that property (iv) holds. Then each polynomial $a(X) \in \mathcal{X}$ has the form

$$c(X)v(X) = \sum_{i=1}^{r} v_iX^ic(X), \quad r < k.$$

Hence, each polynomial $a(X) \in \mathcal{X}$ is a linear combination of polynomials $c(X), Xc(X), \ldots, X^{k-1}c(X)$ (all of which belong to $\mathcal{X}$). On the other hand, polynomials $c(X), Xc(X), \ldots, X^{k-1}c(X)$ have distinct degrees and hence are linearly independent. Therefore words $c, \pi c, \ldots, \pi^{k-1}c$, corresponding to $c(X), Xc(X), \ldots, X^{k-1}c(X)$, form a basis in $\mathcal{X}$.

(iv) We know that each polynomial $a(X) \in \mathcal{X}$ has degree $> \deg c(X)$. By the Division Algorithm,

$$a(X) = v(X)c(X) + r(X).$$

Here, we must have

$$\deg v(X) < k \quad \text{and} \quad \deg r(X) < \deg c(X) = N-k.$$

But then $v(X)c(X)$ belongs to $\mathcal{X}$ owing to Theorem 15.5 (as $v(X)c(X)$ has degree $\leq N-1$, it coincides with $v(X)c(X) \bmod (1+X^N)$). Hence,

$$r(X) = a(X) + v(X)c(X) \in \mathcal{X}$$

by linearity. As $c(X)$ is a unique polynomial from $\mathcal{X}$ of minimum degree, $r(X) = 0$. □

Lecture 16: Cyclic codes (Cont.) The BCH codes

Corollary 16.1. *Every cyclic code is obtained from the codeword corresponding to a polynomial of minimum degree, by cyclic shifts and linear combinations.*

Definition 16.1. Polynomial $c(X)$ of the minimal degree in $\mathcal{X}$ is called a *generator* of a (cyclic) code $\mathcal{X}$.

Remark. There may be other polynomials that generate $\mathcal{X}$ in the sense of the above corollary. But the minimum degree polynomial is unique.

Theorem 16.2. *A polynomial $c(X)$ of degree $\leq N-1$ is the generator of a cyclic linear code of length N iff $c(X)$ divides $1+X^N$. That is,*

$$1+X^N = h(X)c(X)$$

for some polynomial $h(X)$ (of degree $N-1-\deg c(X)$).

Proof of Theorem 16.2. By the Division Algorithm,

$$1+X^N = h(X)c(X)+r(X), \quad \deg r(X) < c(X).$$

That is,

$$r(X) = h(X)c(X)+1+X^N, \quad \text{i.e.} \quad r(X) = h(X)c(X) \bmod (1+X^N).$$

By Theorem 15.5, $r(X)$ belongs to the cyclic code $\mathcal{X}$ generated by $c(X)$. But $c(X)$ must be the (unique) polynomial of minimum degree in $\mathcal{X}$. Hence, $r(X) = 0$ and $1+X^N = h(X)c(X)$. □

Corollary 16.3. *All cyclic codes of length N are in one-to-one correspondence with the divisors of polynomial $1+X^N$.*

Examples. 1. Polynomial $1+X^N$ has two 'standard' divisors:

$$1+X^N = (1+X)(1+X+\ldots+X^{N-1})$$

(recall, all operations are in the binary arithmetic). Polynomial $1+X$ generates the parity-check code $\{x = x_1 \ \ldots \ x_N\colon \sum_i x_i = 0\}$, whereas $1+X+\ldots+X^{N-1}$ the symbol-repetition code $\{\,00\ldots0,\ 11\ldots1\,\}$.

2. (The Hamming (7,4) code). Recall, the generating and check matrices are, in the lexicographic form:

$$G^{\text{lex}} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad \text{and} \quad H^{\text{lex}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & & 1 \end{pmatrix}.$$

If you re-order the digits $x_4x_7x_5x_3x_2x_6x_1$ (which leads to an equivalent code) then the rows of the generating matrix become subsequent cyclic shifts of each other:

$$G^{\text{cycl}} = \begin{pmatrix} \boxed{1\ 1\ 0\ 1} & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

and the cyclic shift of the last row is again in the code:

$$\pi(0\,0\,0\,1\,1\,0\,1) = (1\,0\,0\,0\,1\,1\,0) = (1\,1\,0\,1\,0\,0\,0)+(0\,1\,1\,0\,1\,0\,0)+(0\,0\,1\,1\,0\,1\,0)$$

By Lemma 15.2, the code is cyclic. By Theorem 15.6 (iii), the generating polynomial $c(X)$ corresponds to the framed part in matrix G^{cycl}:

$$\boxed{1\ 1\ 0\ 1} \sim c(X) = 1+X+X^3 = \text{ the generator.}$$

Theorem 16.4. (Not to be proved in the course.) *Any Hamming $(2^l-1, 2^l-1-l)$ code is cyclic.*

[The generator of the Hamming $(2^l-1, 2^l-1-l)$ code is a polynomial $c(X)$, of degree k, with a special property: $c(X)$ is a divisor of $1+X^{2^l-1}$, but *not* of $1+X^r$ for any $r < 2^l-1$. Such a polynomial exists for any l: it is called a primitive polynomial of degree l. $1+X+X^3$ is the primitive polynomial of degree 3.]

Cyclic codes admit encoding and decoding procedures in terms of the polynomials. It is convenient to consider a generating matrix, of a cyclic code $\mathcal{X}$, in a form similar to G^{cycl} for the Hamming (7,4) code (see above). That is, we want to find the basis in $\mathcal{X}$ which gives the following picture in the corresponding generating matrix:

$$G^{\text{cycl}} = \begin{pmatrix} \boxed{} & & & 0 \\ & \boxed{} & & \\ & & \boxed{} & \\ 0 & & \ddots & \boxed{} \end{pmatrix}$$

Such a basis is provided by Theorem 15.6 (iii): take the generator polynomial $c(X)$ and its multiples:

$$c(X), Xc(X), \ldots, X^{k-1}c(X), \ \deg c(X) = N-k.$$

Symbolically,

$$G^{\text{cycl}} = \begin{pmatrix} c(X) \\ Xc(X) \\ \vdots \\ x^{k-1}c(X) \end{pmatrix}.$$

Your code has rank k: you may use it for encoding words of length k as follows. Given a word $a = a_0 \ldots a_{k-1}$, you form the polynomial $a(X) = \sum_{i=0}^{k-1} a_i X^i$ and take the product $a(X)c(X)$. It belong[illegible] $\mathcal{X}$ (see Theorem 15.6) and hence defines a codeword. So all

you have to do is to store polynomial $c(X)$: the encoding will correspond to polynomial multiplication.

If encoding is given by multiplication, decoding must be related to division. Recall that we decode a received word by the closest codeword in the Hamming distance. Such a codeword is related to a leader of the corresponding coset: we have seen that the cosets are in one-to-one correspondence with the syndrome words of the form yH. See above.

In the case of a cyclic code, the syndromes are calculated straightforwardly. Recall that, if $c(X)$ is a generator polynomial of a cyclic code $\mathcal{X}$ and $\deg c(X) = N - k$, then the rank of $\mathcal{X}$ equals k, and there must be 2^{N-k} distinct cosets (see Theorem 9.5 (5)).

Theorem 16.5. *The cosets $y+\mathcal{X}$ are in one-to-one correspondence with the remainders $u(X) = y(X) \bmod c(X)$. In other words, two words y, y' belong to the same coset iff, in the Division Algorithm representation,*

$$y(X) = a(X)c(X) + u(X), \quad y'(X) = a'(X)c(X) + u'(X) \quad \text{and} \quad u(X) = u'(X).$$

Proof of Theorem 16.5. y and y' belong to the same coset iff $y + y' \in \mathcal{X}$ (see Theorem 14.5 (3)). This is equivalent to $u(X) + u'(X) = 0$, i.e. $u(X) = u'(X)$. □

Hence the cosets are labelled by the polynomials $u(X)$ of $\deg u(X) < \deg c(X) = N - k$: there are exactly 2^{N-k} such polynomials. To determine the coset $y + \mathcal{X}$ it is enough to compute the remainder $u(X) = y(X) \bmod (X)$.

Unfortunately, there is still a task to find a leader in each case: there is no simple algorithm for finding leaders, for a general cyclic code. Some particular classes of cyclic codes admit a relatively simple decoding: I will briefly comment on one of these classes, the so-called **Bose–Chaudhuri–Hocquengham** (BCH) codes.

To explain the idea behind the BCH construction let us return to the Hamming codes. The facts about these are summarized in Definition 16.2 and in Theorem 16.6 (cf. Theorems 14.2 and 14.8).

Definition 16.2. The Hamming $(2^\ell - 1, 2^\ell - 1 - \ell)$ code is a linear code whose parity-check matrix contains all non-zero words of length ℓ as the rows:

$$H_{\text{Ham}} = \begin{pmatrix} 0 & \dots & 0 & 1 \\ & \vdots & & \\ 1 & \dots & 1 & 1 \end{pmatrix} \quad \begin{matrix} \uparrow \\ 2^\ell - 1 \\ \downarrow \end{matrix}$$
$$\longleftarrow \ell \longrightarrow$$

Theorem 16.6. *The Hamming $(2^\ell - 1, 2^\ell - 1 - \ell)$ is a perfect 1-error correcting code of length $2^\ell - 1$. The procedure of decoding the Hamming $(2^\ell - 1, 2^\ell - 1 - \ell)$ code is as follows. Having a word $y = y_1 \dots y_N$, $N = 2^\ell - 1$, form the syndrome $s = yH_{\text{Ham}}$. This word has length ℓ: $s = s_1, \dots, s_\ell$. If $s = 0$, decode y by y. If $s \neq 0$ then s is among the rows of H_{Ham}. If this is row i, decode y by $x_* = y + e_i$, where word $e_i = (0 \dots 010 \dots 0)$ (1 in the i^{th} position, 0 otherwise).*

Let us repeat the proof of the last statement. In fact, x_* is a codeword:

$$x_*H = yH + e_iH = s + s = 0,$$

and $d(y, x_*) = 1$ which is of course the minimal non-zero distance.

So the Hamming code corrects a single error, but gives a wrong answer when the number of errors is two or more.

We now want to be able to correct more than one error (two to start with). And it is desirable to develop a procedure of decoding that is similar to Hamming's.

We can try the following idea. Take the rows of the parity-check matrix of length 2ℓ and write it in the form

$$\widetilde{H} = \begin{pmatrix} H_{\text{Ham}} & \prod H_{\text{Ham}} \end{pmatrix} \quad \begin{matrix} \uparrow \\ 2^\ell - 1 \\ \downarrow \end{matrix}$$
$$\begin{matrix} \longleftrightarrow & \longleftrightarrow \\ \ell & \ell \end{matrix}$$
$$\longleftarrow \quad 2\ell \quad \longrightarrow$$

where $\prod H_{\text{Ham}}$ is obtained by permutating the rows of H_{Ham} ($\prod$ is a permutation of degree $2^\ell - 1$):

$$\prod : \text{row } i \mapsto \text{row } \prod i.$$

The new matrix $\widetilde{H}$ must contain 2ℓ linearly independent columns: it then determines an $(2^\ell - 1, 2^\ell - 1 - 2\ell)$ linear code. The syndromes are now words of length 2ℓ (or pairs of words of length ℓ):

$$y\widetilde{H} = (s, s').$$

A syndrome (s, s') may or not be among the rows of H. Recall, we want the new code to be 2-error correcting, and the decoding procedure to be similar to the one for the Hamming codes.

Suppose two errors occur, i.e. y differs from a codeword x by two digits, say i and j. Then the syndrome is

$$y\widetilde{H} = (s_i + s_j, \quad s_{\prod i} + s_{\prod j})$$

where s_k is the word representing row k in H_{Ham}. Organize your permutation so that, knowing vector $(s_i + s_j, s_{\prod i} + s_{\prod j})$, you can always find i and j (or equivalently, s_i and s_j).

In other words, you should be able to solve the equations

$$s_i + s_j = z, \quad s_{\prod i} + s_{\prod j} = z' \tag{16.1}$$

for any pair (z, z') that may eventually occur as a syndrome under two errors.

A natural guess is to try a permutation $\prod$ that has some *algebraic* significance. For example,

$$s_{\prod i} = s_i * s_i = (s_i)^{*2} \quad \text{(a bad choice)}$$

or

$$s_{\prod i} = s_i * s_i * s_i = (s_i)^{*3} \quad \text{(a good choice)}$$

or, generally,

$$s_{\prod i} = s_i * s_i * \cdots * s_i \quad (q \text{ times})$$

where $*$ is an operation of multiplication of words (or the corresponding polynomials). We already met one such operation: the $\star$ multiplication, or multiplication modulo $1 + X^N$.

So, suppose you organize your check matrix as

$$\widetilde{H} = \begin{pmatrix} 0 & \dots & 0 & 1 & (0 & \dots & 0 & 1)^{*q} \\ & \vdots & & & & & & \\ 1 & \dots & 1 & 1 & (1 & \dots & 1 & 1)^{*q} \end{pmatrix} \quad \begin{matrix} \uparrow \\ 2^\ell - 1 \\ \downarrow \end{matrix}$$

$$\longleftarrow \quad 2\ell \quad \longrightarrow$$

You will then have to deal with equations of the type

$$\begin{matrix} s_i + s_j & = z \\ s_i^{*q} + s_j^{*q} & = z' \end{matrix} \quad \left(\text{similar to} \quad \begin{matrix} x + y = a \\ x^q + y^q = b \end{matrix} \right). \tag{16.2}$$

But for solving these equations you need not only multiplication but also *division*, as an

operation inverse to $*$. In other words, the set of words of length ℓ should be a (commutative) *field*.*

Definition 16.3. A commutative field is a commutative ring where each non-zero element has an inverse. In other words, a ring is a field if the multiplication generates a group.

So we need to study finite fields. Unfortunately, the above $*$ multiplication does not lead to a field. The reason is that polynomial $1 + x^N$ is always *reducilble*.

Definition 16.4. A polynomial $a(X) = a_0 + a_1 X + \cdots + a_{N-1} X^{N-1}$ is called *irreducible* if $a(X)$ cannot be written as a product of two polynomials, $b(X)$ and $b'(X)$, with $\min [\deg b(X), \deg b'(X)] \geq 1$.

Examples. Polynomials

$$1 + X + X^4, \quad 1 + X^3 + X^4 \quad \text{and} \quad 1 + X + X^2 + X^6 + X^8$$

are irreducible, whereas

$$1 + X^8, \quad 1 + X^4 + X^6 + X^7 + X^8 \quad \text{and} \quad 1 + X^2 + X^6 + X^8$$

are not. In fact, as was noted, polynomial $1 + X^N$ is always reducible:

$$1 + X^N = (1 + X)(1 + X + \ldots + X^{N-1}).$$

Theorem 16.7. *Let $g(X)$ be an irreducible polynomial of degree N. Then the multiplication mod $g(X)$ makes the set of the polynomials of degree $\leq N - 1$ (i.e., the Hamming space $\{0,1\}^N$) a commutative field. Conversely, if the multiplication mod $g(X)$ leads to a field then $g(X)$ is irreducible.*

A field obtained via the above construction is called a *polynomial* field. The multiplication in a polynomial field is denoted below by $*$. The zero polynomial and the unit

* In fact, the simplest consistent system of form (16.2) is

$$\begin{aligned} s + s' &= z, \\ s^{*3} + s'^{*3} &= z'; \end{aligned}$$

it is reduced to a single equation

$$z * s^{*2} - z^{*2} * s + z^{*3} - z' = 0,$$

and our problem becomes to decompose the polynomial $a * X^{*2} - a^{*2} * X + z^{*3} - z'$ into the product of polynomials of degree 1. It is well known that operation $*$ should lead to a field in order to gu ıtee a possibility of such decomposition.

polynomial are denoted, correspondingly, by $\mathbf{0}$ and $\mathbf{1}$: they are obviously the zero and the unity of the polynomial field.

Proof of Theorem 16.7. Among the properties to check the only non-trivial one is the existence of the inverse element. Take a non-zero polynomial $f(X)$, with $\deg f(X) \leq N-1$, and consider all polynomials of the form $f(X)h(X)$ (the usual multiplication) where $h(X)$ runs over the whole set of the polynomials of degree $\leq N-1$. These products must be distinct mod $g(X)$, because, if

$$f(X)h_1(X) = f(X)h_2(X) \bmod g(X),$$

then, for some polynomial $v(X)$ of degree $\leq N-2$,

$$f(X)(h_1(X) - h_2(X)) = v(X)g(X). \tag{16.3}$$

But equality (16.3) is impossible, for an irreducible polynomial $g(X)$ of degree N, unless $h_1(X) = h_2(X)$. So, for one and only one polynomial $h(X)$, you have

$$f(X)h(X) = \mathbf{1} \bmod g(X),$$

this $h(X)$ being the inverse $f(X)^{-*1}$.

On the other hand, if $g(X)$ is reducible, then $g(X) = b(X)b'(X)$ where both $b(X)$ and $b'(X)$ are non-zero and have degree $< N$. That is, $b(X)b'(X) = 0 \bmod g(X)$. If the multiplication mod g led to a field, both $b(X)$ and $b'(X)$ would have inverses, $b(X)^{-*1}$ and $b'(X)^{-*1}$. But then

$$b(X)^{-*1} * b(X) * b'(X) = b'(X) = \mathbf{0},$$

and similarly $b(X) = \mathbf{0}$. The contradiction obtained proves the theorem. □

A key role is played by the following.

Theorem 16.8. (a) *The plolynomial fields obtained by picking different irreducible polynomials of degree N are all isomorphic.*

(b) *The multiplicative group of any of these fields is isomorphic to the cyclic group $\mathbf{Z}_{2^N-1}$.*

Note that all fields figuring in (a) have 2^N elements, and they are identical as *linear spaces*. However, the field isomorphism is more intricate. The proof below shows that in fact assertions (a) and (b) are valid for all commutative fields containing 2^N elements. Such a field (which is unique up to isomorphism) is frequently denoted by $GF(2^N)$.

Proof of Theorem 16.8. What we really need in the future is assertion (b), so we prove it first. Assertion (a) is not used in the sequel (although it is quite illuminating), and its proof is based on some lemmas (which are of interest themselves).

So, let us show that the multiplicative group of any field with 2^N elements is isomorphic to $\mathbf{Z}_{2^N-1}$, the cyclic group of 2^N-1 elements. Take any element α of the field and

observe that $\alpha^{*i} = \underbrace{\alpha * \ldots * \alpha}_{i \text{ times}}$ (the multiplication in the field) takes at most $2^N - 1$ values (the number of elements in the field less one, because $\mathbf{0}$ is excluded). Hence there exist a positive integer r such that $\alpha^{*r} = \mathbf{1}$; we pick the smallest value of r and call it the order of α.

Choose α with the largest order r. Then the order of any other element α' divides r. In fact, let r' be the order of α'. Pick a prime factor p of r' and write

$$r' = p^{b'}\ell', \quad r = p^b \ell$$

where $b', b \geq 0$ and ℓ, ℓ are not divided by p. We want to show that $b \geq b'$. Indeed, element α^{*p^b} has order ℓ, $\alpha'^{*\ell'}$ has order p'^b and the product $\alpha^{*p^b} * \alpha'^{*\ell'}$ has order $\ell p^{b'}$. Hence, $b' \leq b$ or else r would not be maximal. This is true for any prime b, hence r' divides r.

Thus every element β in the field obeys $\beta^{*r} = \mathbf{1}$, i.e., polynomial $\mathbf{1} * X^{*r} - \mathbf{1}$ $(= \mathbf{1} * X^{*r} + \mathbf{1})$ ** is divisible by $\prod^*_{\beta \in \text{ field}} (X \pm \beta)$. It is easy to conclude that $r = 2^N - 1$, the number of non-zero elements of the field. Hence, $\mathbf{1}, \alpha, \ldots, \alpha^{2^N-1}$ exhaust the multiplicative groups of the field. □

We now pass to the proof of assertion (a) that all commutative fields with 2^N elements, and in particular all polynomial fields that are obtained via the above construction, for different irreducible polynomials of degree N, are isomorphic. For simplicity, we say 'a field', instead of 'a commutative field of 2^N elements'. We also say 'a polynomial' istead of 'a polynomial with coefficients from a field' (although, wherever convenient, we identify them with polynomials with coefficients 0 and 1). The definition of irreducibility is extended to these polynomials straightforwardly.

Definition. Any element α of the highest order $2^N - 1$ in a field is called a *primitive element*. A primitive element is not always unique, but it exists, as follows from Theorem 14.8.

Definition. Let β be a non-zero element of a field. The *minimal polynomial* of β is the lowest degree polynomial $m(X)$ such that $m(\beta) = \mathbf{0}$. In particular, if β is a primitive element of the field, the corresponding polynomial is called primitive.

Remark. Although, as we prove below, the polynomial fields obtained by taking different irreducible polynomials of degree N are isomorphic, they are *not* identical. That is, a given polynomial β is not mapped to itself under a natural isomorphism of two polynomial fields. This is related to the fact that, for the same β, different fields give different minimal polynomials.

Lemma 16.9. *Any minimal polynomial is irreducible and has degree $\leq N$. Any primitive polynomial has degree N. If $m(X)$ is a minimal polynomial of β and $f(X)$ is*

** Note that $\mathbf{1} * X^{*r} \pm \mathbf{1}$ is a polynomial whose coefficients are from a given field of 2^N elements (such a field may itself be a polynomial field). This is quite confusing, but fortunately many properties are valid if you forget about it and identify $\mathbf{1}X^{*r} \pm \mathbf{1}$ as $X^r \pm 1$, a polynomial with coefficients 0 and 1. Below we use such an identification every time when it works: it simplifies the notation and the arguments.

another polynomial with $f(\beta) = \mathbf{0}$, then $m(X)$ divides $f(X)$. In particular, any minimal polynomial divides $X^{2^N-1} + 1$.

Proof. If $m(X) = m^{(1)}(X) * m^{(2)}(X)$ with $\deg m^{(i)}(X) \geq 1$, then $m(\beta) = m^{(1)}(\beta) * m^{(2)}(\beta) = \mathbf{0}$, and hence at least one $m^i(\beta)$ is zero (this is a property of the field). But this contradicts the condition of minimality. Thus $m(X)$ is irreducible.

If $f(X)$ is another polynomial with $f(\beta) = \mathbf{0}$ then $\deg f(X) \geq \deg m(X)$. You can write

$$f(X) = m(X) * v(X) + r(X),\ \deg r(X) < \deg m(X).$$

But then

$$r(\beta) = f(\beta) + m(\beta) * v(\beta) = 0,$$

and you conclude that $r(X) = \mathbf{0}$. Finally, polynomial $X^{2^N-1} + 1$ has all elements β from the field as the roots, i.e., is divided by any minimal polynomial.

Let us now prove that $\deg m(X) \leq N$. Our polynomial field is a vector space of dimension N. Therefore, any $N+1$ elements, such as $\mathbf{1}, \beta, \ldots, \beta^{*N}$ are linearly dependent: $\sum_{i=0}^{N} a_i \beta^{*i} = 0$ where $a_i = \begin{matrix}1\\0\end{matrix}$ and not all $a_i = 0$. Thus $\Sigma a_i X^i$ is a polynomial of degree N having β as a root. Hence, $\deg m(X) \leq N$.

If β is primitive, then $\deg m(X) = N$; otherwise we could not obtain the whole 2^N elements of the field. □

We can now quickly complete the proof of assertion (a) of Theorem 14.8. Let $\mathcal{F}$ and $\mathcal{G}$ be two fields with 2^N elements. Let α be a primitive element in $\mathcal{F}$ with a minimal polynomial $m(X)$. Then $m(X)$ is irreducible and divides $X^{2^N} + 1 = X^{2^N} - 1$. Hence there exists an element $\beta \in \mathcal{G}$ for which $m(X)$ is minimal polynomial. Field $\mathcal{F}$ may be considered as the field of polynomials mod $m(X)$, and field $\mathcal{G}$ contains (and thus consists of) all polynomials of degree $\leq N-1$. Hence β is primitive in $\mathcal{G}$ and $\alpha \leftrightarrow \beta$ is an isomorphism. This completes the proof of Theorem 14.9. □

Examples. 1. Field $GF(2^2)$. There is a unique ireducible polynomial of degree 2: $X^2 + X + 1$. The corresponding field $GF(2^2)$ consists of polynomials $\mathbf{0}$ (the zero of the field), $\mathbf{1}$ (the unity of the field), X and $1 + X$. The corresponding words are $00 \sim \mathbf{0}$, $10 \sim \mathbf{1}$, $01 \sim X$ and $11 \sim 1 + X$. In this field, the multiplicative group is $\mathbf{Z}_3$. The multiplication rule is $X^2 = 1 + X$, $(1+X)^2 = X$, $X * (1+X) = \mathbf{1}$; both X and $(1+X)$ are primitive elements. [One frequently writes field $GF(2^2)$ as $\{0, 1, \alpha, \alpha^2 = 1 + \alpha\}$.] The minimal polynomials are: for $\mathbf{0}$ $m(X) = X$, for $\mathbf{1}$ $m(X) = X + 1$ and for X and $1 + X$ $m(X) = X^2 + X + 1$.

2. Field $GF(2^3)$. There are two irreducible polynomials of degree 3: $X^3 + X + 1$ and $X^3 + X^2 + 1$. The corresponding fields are isomorphic and have the multiplicative group

$\mathbf{Z}_7$. If you take polynomial $X^3 + X + 1$ then the field is described as follows

element of the polynomial field	word	element of $\mathbf{Z}_7$	minimal polynomial
$\mathbf{0}$	000	$--$	X
$\mathbf{1}$	100	1	$1 + X$
X	010	α	$X^3 + X + 1$
X^2	001	α^2	$X^3 + X + 1$
$1 + X$	110	α^3	$X^3 + X^2 + 1$
$X + X^2$	011	α^4	$X^3 + X + 1$
$1 + X + X^2$	111	α^5	$X^3 + X^2 + 1$
$1 + X^2$	101	α^6	$X^3 + X^2 + 1$

Now return to the BCH construction. Recall, we want to construct a 2-error correcting code with a parity check matrix

$$\widetilde{H} = \begin{pmatrix} 0 & \ldots & 0 & 1 & (0 & \ldots & 0 & 1)^{*q} \\ & \vdots & & & & & & \\ 1 & \ldots & 1 & 1 & (1 & \ldots & 1 & 1)^{*q} \end{pmatrix} \begin{matrix}\uparrow \\ 2^\ell - 1 \\ \downarrow\end{matrix}$$

$$\longleftarrow \quad 2\ell \quad \longrightarrow$$

where the multiplication is modulo an irreducible polynomial of degree ℓ. A good example is $\ell = 4$ and $q = 3$: here, $N = 15$, and the rank of a code with the check matrix $\widetilde{H}$ is expected to be $k = 15 - 8 = 7$ (if all 8 columns are linearly independent). Take the multiplication determined by an irreducible polynomial $1 + X + X^4$. Write the corresponding field $GF(2^4)$:

element of the polynomial field	word	element of $\mathbf{Z}_{15}$
$\mathbf{0}$	0000	$--$
$\mathbf{1}$	1000	1
X	0100	α
X^2	0010	α^2
X^3	0001	α^3
$1 + X$	1100	α^4
$X + X^2$	0110	α^5
$X^2 + X^3$	0011	α^6
$1 + X + X^3$	1101	α^7
$1 + X^2$	1010	α^8
$X + X^3$	0101	α^9
$1 + X + X^2$	1110	α^{10}
$X + X^2 + X^3$	0111	α^{11}
$1 + X + X^2 + X^3$	1111	α^{12}
$1 + X^2 + X^3$	1011	α^{13}
$1 + X^3$	1001	α^{14}

Then write matrix $\widetilde{H}$, with $q = 3$. In words:

$$\widetilde{H} = \begin{pmatrix} 1&0&0&0 & 1&0&0&0 \\ 0&1&0&0 & 0&0&0&1 \\ 0&0&1&0 & 0&0&1&1 \\ 0&0&0&1 & 0&1&0&1 \\ 1&1&0&0 & 1&1&1&1 \\ 0&1&1&0 & 1&0&0&0 \\ 0&0&1&1 & 0&0&0&1 \\ 1&1&0&1 & 0&0&1&1 \\ 1&0&1&0 & 0&1&0&1 \\ 0&1&0&1 & 1&1&1&1 \\ 1&1&1&0 & 1&0&0&0 \\ 0&1&1&1 & 0&0&0&1 \\ 1&1&1&1 & 0&0&1&1 \\ 1&0&1&1 & 0&1&0&1 \\ 1&0&0&1 & 1&1&1&1 \end{pmatrix}, \tag{16.4}$$

and in the 'field notation':

$$\widetilde{H} = \begin{pmatrix} \mathbf{1} & \mathbf{1} \\ \alpha & \alpha^3 \\ \alpha^2 & \alpha^6 \\ \alpha^3 & \alpha^9 \\ \alpha^4 & \alpha^{12} \\ \alpha^5 & \mathbf{1} \\ \alpha^6 & \alpha^3 \\ \alpha^7 & \alpha^6 \\ \alpha^8 & \alpha^9 \\ \alpha^9 & \alpha^{12} \\ \alpha^{10} & \mathbf{1} \\ \alpha^{11} & \alpha^3 \\ \alpha^{12} & \alpha^6 \\ \alpha^{13} & \alpha^9 \\ \alpha^{14} & \alpha^{12} \end{pmatrix}. \tag{16.4a}$$

Here, and below we omit $*$ while referring to the multiplication in the field. The left-hand 'half' of $\widetilde{H}$ is nothing but the corresponding Hamming $(15, 11)$ check matrix H_{Ham}. Note that $\mathbf{1}$ appears more than one time in the right-hand half of $\widetilde{H}$: this is because α^3 is not a primitive element of $GF(2^4)$.

Theorem 16.10. *A linear code with the check matrix $\widetilde{H}$ of form (16.4) or (16.4a) is a 2-error correcting $(15, 7)$ code.*

Proof. The rank 7 is due to the linear independence of the columns of $\widetilde{H}$. [It is a tedious although staightforward calculation; we omit it because of lack of time.] The key point is to check that the code corrects up to two errors. First suppose you received a word $y = y_1, \dots, y_{15}$ in which two errors occurred, in digits i and j that are unknown. You want to find these places. First, you calculate the syndrom $y\widetilde{H} = (z, z')$. Recall, z and

z' are words of length 4; the total length of the syndrome is 8. (Note that $z' \neq z^3$: if $z' = z^3$ there is precisely one error occurred.) You write a pair of equations

$$\begin{aligned} s + s' &= z, \\ s^3 + s'^3 &= z', \end{aligned} \tag{16.5}$$

where s and s' are words of length 4 (or equivalently their polynomials), and the multiplication is modulo $1 + X + X^4$. In the case of two errors it is guaranteed that there are at least two solutions to (16.2), one occupying position i and another position j among the rows of the left-hand (Hamming) half of matrix $\widetilde{H}$. To show that (16.4), cannot have other solutions, write

$$z' = s^3 + s'^3 = (s + s')(s^2 + ss' + s'^2) = z(z^2 + ss')$$

(because $z^2 = (s + s')^2 = s^2 + ss' + ss' + S'^2$) and deduce that

$$ss' = z'z^{-1} + z^2. \tag{16.6}$$

Now (16.6) and the first equation in (16.2) give that the solutions to (16.2) are precisely the roots of a quadratic equation

$$\mathbf{x}^2 + z\mathbf{x} + (z'z^{-1} + z^2) = 0. \tag{16.7}$$

(Again note that $z'z^{-1} + z^2 \neq 0$.) But the polynomial in the LHS of (16.7) cannot have more than two distinct roots (in principle it could have no root or a two coinciding roots, but it is excluded by the assumption that there are precisely two errors).

In the case where a single error occurred, you will have $z' = z^3$: in this case $s = z$ is the only root and you just find word z among the rows of the left-hand half of matrix $\widetilde{H}$. Q.E.D.

Thus, the decoding scheme for decoding, in the case of the above $(15, 7)$ code is as follows: upon receiving word y, form a syndrome $y\widetilde{H} = (z, z')$. Then

(i) If both z and z' are zero words, conclude that no error occurred and decode y by y itself.

(ii) If z is non-zero and $z^3 = z'$, conclude that a single error occurred and find the location of the error digit by identifying word z among the rows of the Hamming check matrix.

(iii) If z is non-zero and $z^3 \neq z'$, form the quadric (16.4), and if it has two distinct roots s and s', conclude that two errors occurred and locate the error digits by identifying words s and s' among the rows of the Hamming check matrix.

(iv) If z is non-zero and $z^3 \neq z'$ and quadric (16.7) has no roots, or if z is zero but z' is not, conclude that there are at least three errors occurred. The code is not perfect, so you have to perform additional (unreliable) procedures or refuse to decode the word received.

Observe that the case where z is non-zero, $z^3 \neq z'$ and quadric (16.7) has a single root is impossible: if (16.7) has a root, s say, then the quadric is divisible by $\mathbf{x} - s$:

$$\mathbf{x}^2 + z\mathbf{x} + (z'z^{-1} + z^2) = (\mathbf{x} - s)(\mathbf{x} - s'),$$

and either $s \neq s'$ in which case (16.7) has another root or $s = s'$ in which case $z^3 = z'$.

The decoding procedure outlined leads to a correct codeword in the cases where up to two errors occurred; it also allows to dectect, in some cases, that more than three errors occurred. However, in general, this procedure may lead to a wrong codeword when three or more errors occur.

Example. Suppose the syndrome is $(1001, 0100) = (\alpha^{14}, \alpha)$. Since $(\alpha^{14})^3 = \alpha^{12} \neq \alpha$, you may conclude that either case (iii) occurs or the first possibility in case (iv). You calculate $z'z^{-1} + z'^2 = \alpha^2 + \alpha^{13} = 1001 = \alpha^{14}$ and write the quadric (16.7):

$$\mathbf{x}^2 + \alpha^{14}\mathbf{x} + \alpha^{14} = (\mathbf{x} + \alpha^6)(\mathbf{x} + \alpha^8) = 0$$

whence $s = \alpha^6$, $s' = \alpha^8$. You locate the error digits as 6 and 8. [**Exercise:** Find a word y producing the above syndrome.]

On the other hand, if the syndrome is $(1101, 1100) = (\alpha^7, \alpha^4)$, equation (16.7) becomes

$$\mathbf{x}^2 + \alpha^7\mathbf{x} + \alpha^5 = 0.$$

By trying each element of the field, you can check that there is no solution to this equation. [There are general methods of finding the roots of a quadratic equation in a field $GF(2^\ell)$, but we do not have time to discuss them.] You conclude that three or more errors occurred.

Of course, nothing in the above construction depends on the length being 15: you can use any field $GF(2^\ell)$. The parity-check matrix is

$$\widetilde{H} = \begin{pmatrix} 1 & 1 \\ \alpha & \alpha^3 \\ \alpha^2 & \alpha^6 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ a^{2^\ell - 2} & \alpha^{3(2^\ell - 2)} \end{pmatrix}; \qquad (16.8)$$

to get it in words you should write a field table similar to the above examples. The last result to quote in this course is the following

Theorem 16.11. *Matrix $\widetilde{H}$ of form (16.5) is a parity-check matrix of a two-error correcting $(2^\ell - 1, 2^\ell - 1 - 2\ell)$ code. The code is equivalent to a cyclic code, with the generator $c(X) = m_\alpha(X)m_{\alpha^3}(X)$, where $m_\alpha(X)$ is the minimal polynomial of α (i.e., a primitive polynomial), and $m_{\alpha^3}(X)$ is the minimal polynomial of α^3.*

We do not have time to prove Theorem 16.10. However, the main idea is the same as in the above example of the (15, 7) code. The codes figuring in Theorem 16.2 are called two-error correcting BCH codes. The theory of BCH codes goes far beyond theses class of codes: one can construct, in a similar fashion, an $(2^\ell - 1, 2^\ell - 1 - 3\ell)$ code correcting three errors, etc. Moreover, the construction may be extended to any length N, not necessarily of the form $2^\ell - 1$.

Altogether, it gives a wide class of codes corecting any a priori given number of errors. the BCH codes were invented in the end of the 50's and there is still a good deal of activity around them. However, the BCH codes are asymptotically 'bad': for any sequence of BCH codes of length $N \to \infty$, either k/N or $\delta/N \to 0$. In other words, they lie at the bottom of the diagram (see the figure on p.3 of the notes for Lecture 12 and a figure below). To obtain codes that meet the Gilbert-Varshamov bound, one needs more powerful methods, based on Algebraic Geometry. Such codes were constructed in the early 70's (the Goppa codes, the Justesen codes). There is still an open problem to construct codes that lie *above* the GV curve (the GV curve gives a lower bound for the best code, but it does not forbid a code to be above it: the problem is to find such a code (or to prove that iit does not exist)). In 1983, there was a new class of codes invented, which are *above* the GV curve, but for the number of symbols in the code alphabet ≥ 49. For binary codes, the problem is waiting for solution.