MT341/441/5441 CHANNELS

MARK WILDON

These notes are intended to give the logical structure of the course; proofs and further examples and remarks will be given in lectures. Further installments will be issued as they are ready. All handouts and problem sheets will be put on Moodle.

These notes are based in part on notes written by Dr Alastair Kay. I gratefully acknowledge his advice. I would very much appreciate being told of any corrections or possible improvements.

You are warmly encouraged to ask questions in lectures, and to talk to me after lectures and in my office hours. I am also happy to answer questions about the lectures or problem sheets by email. My email address is `mark.wildon@rhul.ac.uk`.

Lectures: In 2019–20: Monday 2pm–4pm McCrea 0-04, Thursday 9am BLT2

Office hours in McCrea LGF 0-25: Tuesday 3.30pm, Wednesday 11am, Thursday 11.30am or by appointment.

Date: First Term 2019/20.

CHANNELS

We shall study how to compress messages and put then into binary form (source coding) and how to transmit them reliably on a noisy channel (channel coding). A fundamental unifying idea will be entropy: the amount of information in a message. We shall prove Shannon's two main theorems on source and channel coding and see many interesting and illuminating examples.

Outline. In the introduction we shall see the basic problems solved by source and channel coding and review binary notation and conditional probability.

(A) **Source coding**: entropy, prefix-free codes, Kraft inequality. Noiseless Coding Theorem. Huffman codes and their optimality.

(B) **Channel coding**: binary symmetric channel (BSC). Hamming balls and connection with entropy. Noisy Coding Theorem for BSC. Conditional entropy and mutual information. Channel capacity. Fano's Lemma and Data Processing Inequality. Noisy Coding Theorem in general.

(C) **Ergodic sources and Asymptotic Equipartition Property (AEP)**: AEP for memoryless sources. AEP for ergodic sources (stated only). Noiseless Coding Theorem for sources with AEP. Lempel–Ziv encoding.

Recommended Reading. All these books are in the library. If you find there are not enough copies, email me.

[1] *Codes and cryptography*, D. Welsh, Oxford University Press (1988), 001.5436 WEL.
Covers all of the course. Very clear and concise. Also useful for MT361/461/5461 Error Correcting codes and MT362/ 462/5462 Cipher Systems.

[2] *Elements of information theory*, Thomas M. Cover and Joy A. Thomas, Wiley (1991), 001.539 COV.
Covers the entire course and more.

[3] *Information theory: Inference and learning algorithms*, David J. C. Mackay, Cambridge University Press (2003), 001.539 MAC.
More advanced book on more recent directions in coding theory, with connections to machine learning and statistical inference.

Also you will find a link on Moodle to Dr. Alastair Kay's notes. These will give you a different view of the course material. Highly recommended.

Prerequisites. You need some probability theory, binary numbers and modular arithmetic. Since probability, especially expectation and conditional probability is a big part of this course, we shall spend some time reviewing it in lectures. Revision notes on probability are available on Moodle.

Problem sheets. There will be 8 marked problem sheets; the first is due in on **Thursday 10th October. 10% of your final mark for the course is given for making a reasonable attempt at the problem sheets.**

Moodle. All handouts, problem sheets and answers will be posted on Moodle. Once you are registered for the course you should find a link under 'My courses'. If not please go to

`moodle.royalholloway.ac.uk/course/view.php?id=374`.

This is the Moodle page for MT441 (the M.Sci. course) and MT5441 (the M.Sc. course) as well as MT341: everyone should have access. If you find you do not, email me at once.

Exercises in these notes. Exercises set in these notes are mostly simple tests that you are following the material. Some will be used for quizzes in lectures. Doing the others will help you to review your notes.

Optional questions and extras. Optional questions on problem sheets and any 'extras' in these notes are included for interest only, and to show you some mathematical ideas beyond the scope of this course. You should not worry if you find them difficult.

If you can do the compulsory questions on problem sheets, know the definitions and main results from lectures, and can prove the results whose proofs are marked as examinable in these notes, then you should do very well in the examination.

Question. What is information? What are source and channel coding and what problems do they solve?

In this section we shall give some informal answers to these questions. We then review probability theory and see its applications to coding.

Binary, bits and information.

Example 1.1. A friend has chosen a number x between 0 and 15. How many 'Yes'/'No' questions do you need to find x?

Exercise 1.2. Is there a questioning strategy that can guarantee to use three or fewer questions? Can you prove that your belief is correct?

One simple strategy uses binary. If

$$x = 2^{m-1}x_{m-1} + \cdots + 2x_1 + x_0.$$

then we say that x is $x_{m-1} \ldots x_1 x_0$ in binary, and write, for example, $13 = 01101 = 1101 = \ldots$. (You can write 1101_2 if you want to emphasise the base 2 of binary.) The *bi*nary digi*ts* 0 and 1 are called *bits*.

x	binary form	x	binary form
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

This table gives an easy four question strategy: just ask one question about each bit of x in turn. It also solves Exercise 1.2: clearly there is no way to learn four bits by asking three 'Yes'/'No' questions.

Exercise 1.3. Say that $x \in \mathbb{N}_0$ has *length ℓ in binary* if has a binary form $x_{\ell-1} \ldots x_1 x_0$ with $x_{\ell-1} = 1$. For instance $35 = 100011_2$ has length 6 and 0 has length 0.

(a) Let $m \in \mathbb{N}$. Which numbers x have length at most m?
(b) How many questions would you need if the game in Example 1.1 was changed so that $x \in \{0, 1, 2, \ldots, 99\}$?
(c) Let $\ell \in \mathbb{N}$. Which numbers x have length exactly ℓ?

We shall see in this course that the bit is the fundamental unit of information. To make this rigorous we need to bring in ideas of randomness.

For example, a number in $\{0, 1, 2, \ldots, 15\}$, chosen uniformly at random, has exactly 4 bits of information.

Source coding. The aim of source coding is to encode messages (English texts, large numbers, music, videos ...) into bits, compressing them as much as possible. For instance, the encoder for the messages $\{0, 1, \ldots, 15\}$ corresponding to the four question strategy is simply $0 \mapsto 0000$, $1 \mapsto 0001, \ldots, 15 \mapsto 1111$. By Exercise 1.2, no further compression is possible.

Exercise 1.4. Suppose that your friend's number is 0 with probability $\frac{1}{2}$, and each of $1, \ldots, 15$ with equal probability $\frac{1}{30}$. Suggest a good questioning strategy. How many questions does it use on average? What is the corresponding encoder?

The encoder $0 \mapsto 1, 1 \mapsto 0000, 2 \mapsto 00010, 3 \mapsto 00011, \ldots, 14 \mapsto 01110$, $15 \mapsto 01111$ has the shortest possible expected length of codewords for the probability distribution $\frac{1}{2}, \frac{1}{30}, \ldots, \frac{1}{30}$. We shall prove this in Part A, as a corollary of the optimality of Huffman codes. It gives an optimal strategy for the game in Exercise 1.4.

Example 1.5. The expected length for this encoder is

$$\frac{1}{2} \times 1 + \frac{1}{30} \times 4 + \frac{14}{30} \times 5 = \frac{15+4+70}{30} = \frac{89}{30} = 3 - \frac{1}{30}.$$

Why does this not contradict Exercise 1.2?

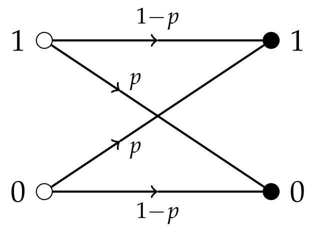
Exercise 1.6. Suppose that multiple numbers are encoded using this encoder by concatenating the codewords. You receive

$$1011\,1100\,0011\,1011\,0111.$$

(For readability this is split into blocks of size 4.) What numbers were you sent? Why can you be sure?

Exercise 1.7. Suppose that messages a, t, g, c have probabilities $\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}$. Using the 8 bit ASCII coding, the encoder is a $\mapsto$ 01000001, t $\mapsto$ 01010100, g $\mapsto$ 01000111, c $\mapsto$ 01000011. Since there are only four messages, it seems wasteful to use 8 bits. Suggest a more efficient binary code $u(\mathtt{a})$, $u(\mathtt{t}), u(\mathtt{g}), u(\mathtt{c})$.

Noisy channel coding. Suppose that Alice wants to send Bob a single 'Yes/No' message. She can only communicate by sending Bob the bits 0 and 1 through the *binary symmetric channel* with *cross-over probability* p, or $\mathrm{BSC}(p)$, that flips each bit with probability p, as shown overleaf.



When multiple bits are sent, each flips independently.

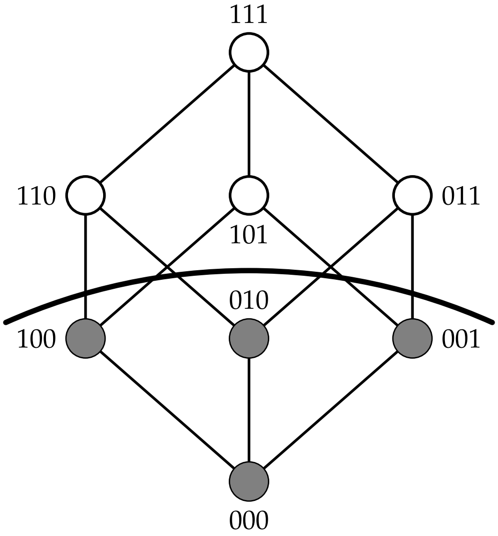
Exercise 1.8. Why may we, and Alice and Bob, assume that $p < \frac{1}{2}$?

If Alice encodes her 'Yes'/'No' message by a single bit, then with probability p, Bob will not receive the intended message.

Instead Alice and Bob decide to pad out the single bit with some redundant bits using the binary repetition code of length 3, with codewords 000 and 111. The agreed encoder is 'No' $\mapsto$ 000 and 'Yes' $\mapsto$ 111. Bob decodes by assuming that the majority bit in a received word is correct. So

- 000, 001, 010, 100 (grey dots) are decoded as 000 meaning 'No';
- 111, 110, 101, 011 (white dots) are decoded as 111 meaning 'Yes'.

All this is public knowledge: there is nothing secret about codes used for source encoding or for error correction.



Let X be Alice's sent codeword. Let Y be Bob's received word. Then

$$\mathbf{P}[Y = 000|X = 000] = (1-p)^3$$
$$\mathbf{P}[Y = 110|X = 000] = p^2(1-p).$$

The hardest thing here may be the notation for conditional probability. Informally $\mathbf{P}[A|B]$ is 'the probability of event A, given that event B has occurred'. This informal definition should work for this exercise.

Exercise 1.9. Find $\mathbf{P}[Y = 111 | X = 000]$ and

$$\mathbf{P}[Y \in \{111, 110, 101, 011\} | X = 000].$$

The second probability is $\mathbf{P}$[Bob decodes as 111|Alice sends 000].

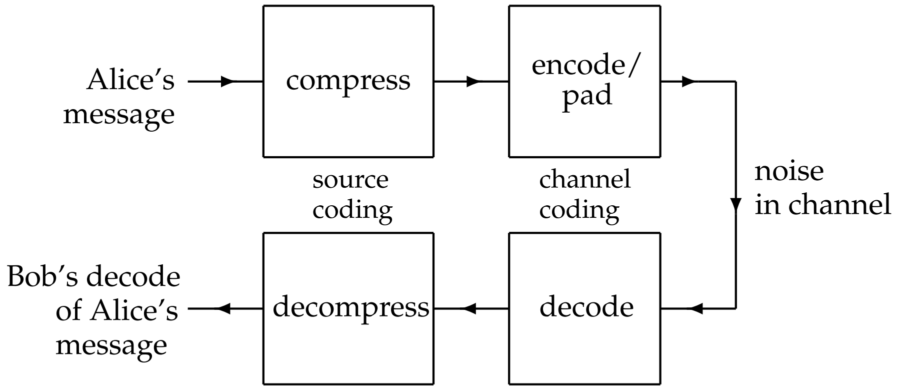
(a) Why is this equal to $\mathbf{P}$[Bob decodes wrongly]?

(b) Is this an improvement on Alice sending a single bit 0 or 1 to Bob?

(c) How does this probability change if instead Alice and Bob use the binary repetition code of length 5?

The aim of channel coding is to minimize the (general version) of the probability $\mathbf{P}$[Bob decodes wrongly]. In our toy model there are only two codewords, and repetition codes are optimal. In Part B we shall see the much richer theory when there are many messages and codewords.

The bigger picture. The diagram below shows how source coding and channel coding combine. Source coding *removes redundancy*. For example in Exercise 1.7 you replaced 8 bit ASCII with a much shorter code. Channel coding *adds redundancy*, in a controlled way to minimize the probability of a decoding error. Repetition codes are the simplest example.



Exercise 1.10. In Exercise 1.7 you solved the source coding problem for the messages A, T, G, C with probabilities $\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}$.

(a) Using the binary repetition code of length 3 as the channel code, what would you send through the BSC to communicate $CCTGC$?

(b) Give an example of a received binary word and how it is decoded.

(c) How is redundancy removed and added in this process?

Probability revision. Here is the setup for discrete probability.

Definition 1.11.

- A *probability measure* p on a finite set Ω assigns a real number p_ω to each $\omega \in \Omega$ so that $0 \leq p_\omega \leq 1$ for each ω and
$$\sum_{\omega \in \Omega} p_\omega = 1.$$

We say that p_ω is the *probability of* ω.

- A *probability space* is a finite set Ω equipped with a probability measure. The elements of Ω are called *outcomes*.
- An *event* is a subset of Ω.
- The *probability* of an event $A \subseteq \Omega$, denoted $\mathbf{P}[A]$, is the sum of the probability of the outcomes in A; that is

$$\mathbf{P}[A] = \sum_{\omega \in A} p_\omega.$$

It follows at once from this definition that $\mathbf{P}[\{\omega\}] = p_\omega$ for each $\omega \in \Omega$. We also have $\mathbf{P}[\varnothing] = 0$ and $\mathbf{P}[\Omega] = 1$.

Example 1.12.

(1) To model a throw of a single unbiased die, we take

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

and put $p_\omega = \frac{1}{6}$ for each outcome $\omega \in \Omega$. The event that we throw an even number is $A = \{2, 4, 6\}$ and as expected, $\mathbf{P}[A] = p_2 + p_4 + p_6 = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$.

(2) To model a throw of a pair of dice we could take

$$\Omega = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}$$

and give each element of Ω probability $\frac{1}{36}$. Alternatively, if we know we only care about the sum of the two dice, we could take $\Omega = \{2, 3, \ldots, 12\}$ with $p_2 = 1/36$, $p_3 = 2/36$, …, $p_6 = 5/36$, $p_7 = 6/36$, $p_8 = 5/36, \ldots, p_{12} = 1/36$. The former is natural and more flexible.

Definition 1.13. Let Ω be a probability space, and let $A, B \subseteq \Omega$ be events. If $\mathbf{P}[B] \neq 0$ then we define the *conditional probability of A given B* by

$$\mathbf{P}[A|B] = \frac{\mathbf{P}[A \cap B]}{\mathbf{P}[B]}.$$

The events A, B are said to be *independent* if $\mathbf{P}[A \cap B] = \mathbf{P}[A]\mathbf{P}[B]$.

Suppose that each element of Ω has equal probability p. Then

$$\mathbf{P}[A|B] = \frac{|A \cap B|p}{|B|p} = \frac{|A \cap B|}{|B|}$$

is the proportion of elements of B that also lie in A. This agrees with the intuitive idea that $\mathbf{P}[A|B]$ is the probability that, given B has occurred, then A has also occurred.

Exercise 1.14. Show that if A and B are events in a probability space such that $\mathbf{P}[B] \neq 0$, then $\mathbf{P}[A|B] = \mathbf{P}[A]$ if and only if A and B are independent.

Conditional probability can be quite subtle.

Exercise 1.15. Let $\Omega = \{HH, HT, TH, TT\}$ be the probability space for two flips of a fair coin, so each outcome has probability $\frac{1}{4}$. Let A be the event that both flips are heads, and let B be the event that at least one flip is a head. Write A and B as subsets of Ω and find $\mathbf{P}[A|B]$.

Example 1.16 (The Monty Hall Problem)**.** On a game show you are offered the choice of three doors. Behind one door is a car, and behind the other two are goats. You pick a door and then the host, *who knows where the car is*, opens another door to reveal a goat. You may then either open your original door, or change to the remaining unopened door. Assuming you want the car, should you change?

Random variables and expectation.

Definition 1.17. Let Ω be a probability space. A *random variable* on Ω is a function $X : \Omega \to \mathcal{R}$, for some set $\mathcal{R}$.

For instance in Example 1.12(2), define $X, Y : \Omega \to \mathbb{N}$ by $X\big((i,j)\big) = i$ and $Y\big((i,j)\big) = j$. Thus X is the first number rolled, Y is the second, and $X + Y$ is their sum. We have $\mathbf{P}[X + Y = 7] = \frac{6}{36} = \frac{1}{6}$.

Often $\mathcal{R}$ will be the set $\mathbb{R}$ of real numbers. But it will be useful in this course to allow other sets: for instance in Exercise 1.9, X and Y took values in $\{000, 001, 010, 100, 011, 101, 110, 111\}$.

Definition 1.18. Let Ω be a probability space with probability measure p. The *expectation* $\mathbf{E}[X]$ of a random variable $X : \Omega \to \mathbb{R}$ is defined to be

$$\mathbf{E}[X] = \sum_{\omega \in \Omega} X(\omega) p_w.$$

Intuitively, the expectation of X is the average value of X on elements of Ω, if we choose $\omega \in \Omega$ with probability p_ω. We have

$$\mathbf{E}[X] = \sum_{\omega \in \Omega} X(\omega) p_\omega = \sum_{x \in R} \sum_{\substack{\omega \\ X(\omega) = x}} x p_\omega = \sum_{x \in \mathbb{R}} x \mathbf{P}[X = x].$$

A critical property of expectation is that it is linear. Note that we *do not assume any independence* in this lemma. The proof is left as an exercise.

Lemma 1.19 (Linearity of expectation)**.** *Let Ω be a probability space. If $X_1, X_2, \ldots, X_k : \Omega \to \mathbb{R}$ are random variables then*

$$\mathbf{E}[a_1 X_1 + a_2 X_2 + \cdots + a_k X_k] = a_1 \mathbf{E}[X_1] + a_2 \mathbf{E}[X_2] + \cdots + a_k \mathbf{E}[X_k]$$

for any $a_1, a_2, \ldots, a_k \in \mathbb{R}$.

Variance and Chebyshev's inequality.

Definition 1.20. Let Ω be a probability space. The *variance* $\mathbf{Var}[X]$ of a random variable $X : \Omega \to \mathbb{R}$ is defined to be

$$\mathbf{Var}[X] = \mathbf{E}\big[(X - \mathbf{E}[X])^2\big].$$

The variance measures how much X can be expected to depart from its mean value $\mathbf{E}[X]$. So it is a measure of the 'spread' of X. This is made precise by Chebyshev's inequality.

Lemma 1.21. *If X is a random variable and $a > 0$ then*

$$\mathbf{P}\big[|X - \mathbf{E}X| \geq a\big] \leq \frac{\mathbf{Var}[X]}{a^2}.$$

We shall use Chebyshev's inequality later in the course. A proof is outlined on the first problem sheet.

Reliable communication. A subtle point, related to conditional probability is seen in this question.

Question 1.22. *Should we conclude from Exercise 1.9 that Alice and Bob can reliably communicate a 'Yes'/'No' message using a long enough repetition code?*

Probably the best answer is 'it depends'.

Exercise 1.23. Suppose that Alice's message is 'No' with probability $\frac{1}{100}$ and 'yes' with probability $\frac{99}{100}$ and that the cross-over probability in the BSC is $\frac{1}{10}$. Using the repetition code of length 3, show that

$$\mathbf{P}[\text{Alice's message is 'No'}|\text{Bob decodes as 'No'}] \approx 0.26$$

and find $\mathbf{P}[\text{Alice's message is 'Yes'}|\text{Bob decodes as 'No'}]$. What should Bob do when he receives 000?

On the first problem sheet we make an analogy with medical testing. Suppose that one in a hundred people have a rare disease. In its early stages there are no symptoms, but the disease can be identified by a test:

$$\begin{aligned}\mathbf{P}[\text{test positive}|\text{have disease}] &= 1 - p\\ \mathbf{P}[\text{test positive}|\text{do not have disease}] &= p\end{aligned}$$

It is correct to say the test works with probability $1 - p$; correspondingly an output bit from the BSC equals the input bit with probability $1 - p$. Continuing the analogy,

$$\begin{aligned}\mathbf{P}[\text{Bob decodes as 'No'}|\text{Alice's message is 'No'}] &\longleftrightarrow \mathbf{P}[\text{test positive}|\text{have disease}]\\ \mathbf{P}[\text{Alice's message is 'No'}|\text{Bob decodes as 'No'}] &\longleftrightarrow \mathbf{P}[\text{have disease}|\text{test positive}].\end{aligned}$$

If you find the answer to Question 1.22 surprising, you may well be confusing the top and bottom probabilities. Imagine you have just taken the medical test. Which of the two probabilities on the right do you care more about? Which of the probabilities on the left is more important? How small must p be for the test to be of use?

(A) Source coding

2. Prefix-free binary codes and Kraft's Inequality

> **Question.** What properties should a binary code have so that we can decode it easily? How do these restrict its codewords?

Prefix-free codes and trees. Recall that if $\mathcal{A}$ is a set then $\mathcal{A}^\ell$ is the set of all ℓ-tuples of elements of $\mathcal{A}$. For example $\mathbb{R}^3 = \{(x,y,z) : x,y,z \in \mathbb{R}\}$ is 3-dimensional space.

Definition 2.1. Let $\mathcal{A}$ be a set. A *word of length ℓ from $\mathcal{A}$* is an element of $\mathcal{A}^\ell$. We write $\mathcal{A}^\star$ for the set of all words from $\mathcal{A}$. We write $\ell(u)$ for the length of the word u. We write $\varnothing$ for the empty word.

Definition 2.2. A *binary word* is a word from $\{0,1\}$. A *binary code* is a non-empty finite set of binary words. The words in a code are called *codewords*.

For example $(1,1,0,1) \in \{0,1\}^4$ is a binary word of length 4. As already seen, we usually write binary words more simply: for example, the binary words of length 3 are $000, 001, 010, 011, 100, 101, 110, 111$; these form a binary code of size 8. We have $\ell(010) = 3$ and $\ell(\varnothing) = 0$.

Note that **there is nothing secret** about binary codes. For example, one important binary code is the ASCII code for letters in the Roman alphabet mentioned in Exercise 1.7. The entire world (or at least, their computers and mobile phones) needs to know that T is represented by 01010100.

Exercise 2.3. How many binary words are there of length n?

In Exercise 1.4 we created the binary code C with codewords 1, 0000, 00010, 00011, $\ldots$, 01111. In Exercise 1.6 we saw that when codewords from this code are concatenated to make a long binary word one can read the word left-to-right, and unambiguously decode it by stopping as soon as each codeword is seen. This worked because the code is prefix-free.

Definition 2.4. Let u and w be binary words, of lengths ℓ and m, respectively. We say that u is a *prefix* of w is $\ell \leq m$ and $w = u_1 \ldots u_\ell w_{\ell+1} \ldots w_m$. A binary code C is *prefix-free* if no codeword in C is a prefix of another codeword in C.

In Exercise 1.7 we saw the binary codes $C = \{0, 10, 110, 111\}$ and $C' = \{0, 01, 011, 111\}$. The first is prefix-free. The second is not, but there is still a unique way to decode each concatenation of codewords. Such codes are said to be *uniquely decipherable*.

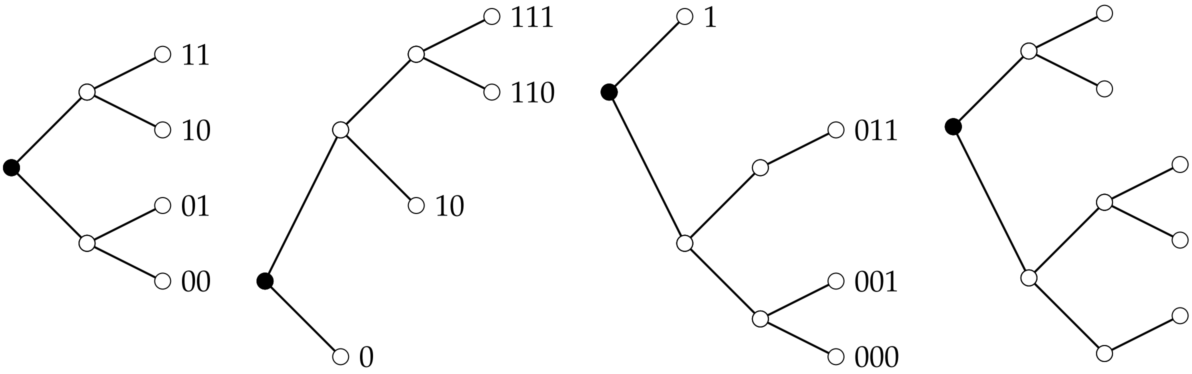
Exercise 2.5.

(a) Observe that C' is prefix-free when codewords are reversed. Why does this show that C' is uniquely decipherable?

(b) Find a binary word equal to concatenations of codewords from the code $\{0, 01, 100, 101\}$ in at least two different ways. Is this code uniquely decipherable? Is it useful for communication?

We shall concentrate on prefix-free codes, which are by far the most important in practice. For more on uniquely decipherable codes and their formal definition, see the 'extras' for this part.

It is useful to represent prefix-free binary codes by oriented rooted binary trees. The trees for $\{00, 01, 10, 11\}$, $\{0, 10, 110, 111\}$, $\{000, 001, 011, 1\}$ are below. (The rightmost tree is used in Exercise 2.6(b).) Since it is natural to read the codewords left to right, we grow the tree the same way, stepping up for 1 and down for 0.



Exercise 2.6.

(a) How could the third code $\{000, 001, 111, 1\}$ be made more efficient, while keeping it prefix-free? How can this improvement be seen from the tree?

(b) What is the code corresponding to the fourth tree above?

Exercise 2.7. Draw the tree corresponding to the prefix-free code in Exercise 1.4. What is the corresponding questioning strategy?

Kraft's Inequality.

Exercise 2.8. Which of the following are the lengths of the codewords in a prefix-free binary code: (i) $1, 3, 3, 3$; (ii) $1, 3, 3, 2$, (iii) $1, 2, 2, 3$; (iv) $3, 2, 2, 2$; (v) $1, 3, 3, 4, 4, 4, 5, 5$?

Proposition 2.9 (Kraft's Inequality). *Let $\ell_1, \ell_2, \ldots, \ell_s \in \mathbb{N}$. There is a prefix-free binary code whose codewords have lengths $\ell_1, \ell_2, \ldots, \ell_s$ if and only if*

$$2^{-\ell_1} + 2^{-\ell_2} + \cdots + 2^{-\ell_s} \leq 1.$$

For instance, in (ii) the sum is $2^{-1} + 2^{-3} + 2^{-3} + 2^{-2} = 1$, so the 'if' direction says that there is a prefix-free code with lengths $1, 3, 3, 2$. In (iii) the sum is $2^{-1} + 2^{-2} + 2^{-2} + 2^{-3} = \frac{9}{8}$, so the 'only if' direction says there is not a prefix-free code with lengths $1, 2, 2, 3$.

Proof of 'if' direction. Suppose that there are a_ℓ lengths equal to ℓ, for each $\ell \in \mathbb{N}$. Let m be the maximum length. The hypothesis is equivalent to

$$2^{-1}a_1 + 2^{-2}a_2 + \cdots + 2^{-(m-1)}a_{m-1} + 2^{-m}a_m \leq 1.$$

Multiplying through by 2^m and rearranging we get

$$a_m \leq 2^m - 2^{m-1}a_1 - 2^{m-2}a_2 - \cdots + 2a_{m-1}.$$

Hence $0 \leq 2^m - 2^{m-1}a_1 - 2^{m-2}a_2 - \cdots + 2a_{m-1}$. Dividing by 2 and rearranging we get

$$a_{m-1} \leq 2^{m-1} - 2^{m-2}a_1 - 2^{m-3}a_2 - \cdots - 2a_{m-2}.$$

Continuing in this way gives the further inequalities

$$\begin{aligned} a_{m-2} &\leq 2^{m-2} - 2^{m-3}a_1 - 2^{m-4}a_2 - \cdots - 2a_{m-3} \\ &\vdots \\ a_3 &\leq 2^3 - 2^2a_1 - 2a_2 \\ a_2 &\leq 2^2 - 2a_1 \\ a_1 &\leq 2. \end{aligned}$$

We now use each inequality, working from bottom to top. At each step except the last we use $(\star)$: a prefix $u_1 \ldots u_k$ of length k forbids the prefixes $u_1 \ldots u_k 0$, $u_1 \ldots u_k 1$ of length $k+1$.

(1) Pick a_1 codewords of length 1. This is possible because $a_1 \leq 2$. By $(\star)$ there are now $2a_1$ forbidden prefixes of length 2.

(2) Pick a_2 codewords of length 2, avoiding the $2a_1$ forbidden prefixes of length 2. This is possible because there are 2^2 binary words of length 2 and $a_2 \leq 2^2 - 2a_1$. By $(\star)$ there are now $2(2a_1 + a_2) = 2^2a_1 + 2a_2$ forbidden prefixes of length 3.

(3) Pick a_3 codewords of length 3, avoiding the $2^2a_1 + 2a_2$ forbidden prefixes of length 3. This is possible because there are 2^3 binary words of length 3 and $a_3 \leq 2^3 - 2^2a_1 - 2a_2$. By $(\star)$ there are now

$$2(2^2a_1 + 2a_2 + a_3) = 2^3a_1 + 2^2a_2 + a_3$$

forbidden prefixes of length 4.

$\vdots$

(m) Pick a_m codewords of length m, avoiding the

$$2^{m-1}a_1 + 2^{m-2}a_2 + \cdots + 2a_{m-1}$$

forbidden prefixes of length m. This is possible because there are 2^m binary words of length m and $a_m \leq 2^m - 2^{m-1}a_1 - \cdots - 2a_{m-1}$.

We now have a prefix-free code with the specified lengths. □

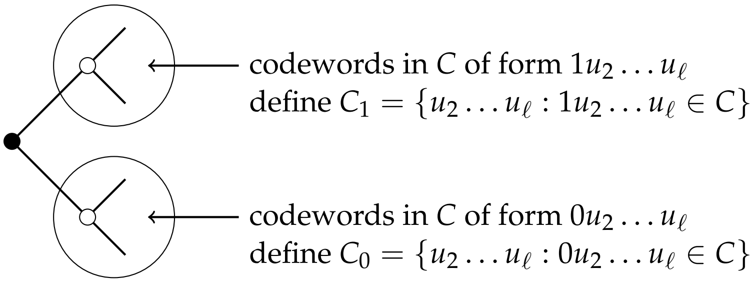
One of the most valuable things you can do when trying to understand a proof is to work through an example in parallel. This often shows one 'why things work'.

Example 2.10. We shall use the proof of the 'if' direction of Kraft's Inequality to construct a prefix-free binary code with codewords of lengths $1, 3, 3, 4, 4, 4, 5, 5$.

Proof of 'only if' direction. Let C be a prefix-free binary code. Let m be the maximum length of a codeword in C. To prove Kraft's inequality holds, we must show that $\sum_{u \in C} 2^{-\ell(u)} \leq 1$. We work by induction on m.

Base case: If $m = 1$ then C is either $\{0\}$, $\{1\}$ or $\{0, 1\}$ and Kraft's Inequality holds.

Inductive step: let $m \geq 2$. The oriented rooted binary tree for C looks like


codewords in C of form $1u_2 \ldots u_\ell$
define $C_1 = \{u_2 \ldots u_\ell : 1u_2 \ldots u_\ell \in C\}$
codewords in C of form $0u_2 \ldots u_\ell$
define $C_0 = \{u_2 \ldots u_\ell : 0u_2 \ldots u_\ell \in C\}$

Since C is prefix-free, so are C_0 and C_1 Each has maximum length at most $m - 1$. By induction, $\sum_{v \in C_0} 2^{-\ell(v)} \leq 1$ and $\sum_{v \in C_1} 2^{-\ell(v)} \leq 1$. Hence

$$\begin{aligned}
\sum_{u \in C} 2^{-\ell(u)} &= \sum_{v \in C_0} 2^{-(\ell(v)+1)} + \sum_{v \in C_1} 2^{-(\ell(v)+1)} && \text{(definition of } C_0, C_1\text{)} \\
&= \frac{1}{2}\Big(\sum_{v \in C_0} 2^{-\ell(v)} + \sum_{v \in C_1} 2^{-\ell(v)} \Big) && \text{(take out factor of } 1/2\text{)} \\
&\leq \frac{1}{2}(1 + 1) && \text{(inductive hypothesis)} \\
&= 1
\end{aligned}$$

as required. □

3. Sources, Entropy and Shannon Codes

Question. What is an efficient way to code messages when some are much more frequent than others?

Coding for sources. To answer this question we need some definitions.

Definition 3.1. An *alphabet* is a finite non-empty set of *symbols*. A *source* is a random process producing a sequence $U_1, U_2, \ldots$ of symbols from an alphabet. A source is *memoryless* if the U_t are independent and identically distributed.

Example 3.2.

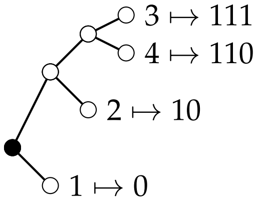
(1) A coin that lands heads with probability p, independently of previous flips, is a memoryless source producing symbols from the alphabet $\{\mathrm{H}, \mathrm{T}\}$. We have $\mathbf{P}[U_t = \mathrm{H}] = p$ for all t.

(2) A binary source emits $0, 0, 0, \ldots$ or $1, 1, 1, \ldots$ with equal probability $\frac{1}{2}$. Like the previous example with $p = \frac{1}{2}$, we have $\mathbf{P}[U_t = 0] = \mathbf{P}[U_t = 1] = \frac{1}{2}$ for all t. But since U_1 and U_2 are not independent, the source is not memoryless.

(3) A source produces random meaningful English messages in lower case with all punctuation except spaces deleted. The alphabet is the Roman alphabet together with space. After receiving '`the source is not memoryles`' you can easily guess the next character. Therefore . . .

Example 3.3. A memoryless source produces symbols from the alphabet $\{1, 2, 3, 4\}$ such that $\mathbf{P}[U_t = i] = p_i$ for each $i \in \{1, 2, 3, 4\}$, where $(p_1, p_2, p_3, p_4) = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$. We encode symbols using the prefix-free binary code $\{0, 10, 110, 111\}$ by

$$1 \mapsto 0,\ 2 \mapsto 10,\ 3 \mapsto 110,\ 4 \mapsto 111$$



The expected length of a codeword is then $\frac{1}{2}1 + \frac{1}{4}2 + \frac{1}{8}3 + \frac{1}{8}3 = \frac{7}{4}$.

Equivalently, suppose in the guessing game that your friend's number is $1, 2, 3, 4$ with the probabilities in Example 3.3. The prefix-free code used above corresponds to the decision tree in the margin. (The question can be 'do I step up for your number?' every time!) Minimizing the average number of questions is the same as minimizing the codeword length.

Exercise 3.4. For each of the following alphabets and probability measures find a binary encoder using a prefix-free code. Try to minimize the expected length of the codeword. [*Hint:* Kraft's Inequality tells you what lengths are possible; (iii) is related to Exercise 2.10(v).]

(i) $\{1, 2, 3, 4\}$: $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$;

(ii) $\{1, 2, 3\}$: $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$;

(iii) $\{1, 2, 3, 4, 5, 6, 7, 8\}$: $(\frac{1}{2}, \frac{1}{2^3}, \frac{1}{2^3}, \frac{1}{2^4}, \frac{1}{2^4}, \frac{1}{2^4}, \frac{1}{2^5}, \frac{1}{2^5})$;

(iv) $\{1, 2, 3, 4, 5\}$: $(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$.

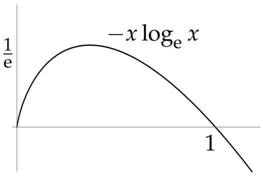
Remarkably there is a simple formula for the minimum possible expected length in the case when all the probabilities are of the form $1/2^c$. Here, before you turn the page, is your last chance to discover the definition of entropy for yourself.

Exercise 3.5. With the setup of Exercise 3.4, suppose that the alphabet is $\{1, \ldots, s\}$ and that $p_i = 1/2^{c_i}$ for each i. Show that there is a prefix-free binary code with codewords $u(1), \ldots, u(s)$ such that $u(i)$ has length c_i for each i. What is the expected codeword length? Express the expected codeword length just in terms of $p_1, \ldots, p_s$.

Entropy. The following definition is fundamental to the course.

Definition 3.6 (Entropy of probability measure)**.** Let p_x for $x \in \Omega$ be a probability measure on a set Ω. The *entropy* of p is

$$H(p) = -\sum_{\omega\in\Omega} p_\omega \log_2 p_\omega.$$



To deal with the case when $p_x = 0$, we use the convention that $0\log_2 0 = 0$. This is consistent with the graph of $x\log_e x$: see margin. In Exercise 3.5 you might have discovered the equivalent form $H(p) = \sum_{\omega\in\Omega} p_\omega \log_2 \frac{1}{p_\omega}$.

Exercise 3.7.

(a) Suppose that $p_i = \frac{1}{s}$ for $i \in \{1,\ldots,s\}$. What is $H(p)$?

(b) Show that in each case in Example 3.4, the expected length of the code is at least $H(p)$, and that equality holds for (i), (ii) and (iii).

Shannon codes. By Exercise 3.5, when all the probabilities in a probability measure p are powers of 2 there is a prefix-free binary code with expected length $h(p)$. In general, we cannot do quite so well, but using almost the same idea, we can still get a good code.

Recall that if $x \in \mathbb{R}$ then $\lceil x\rceil$ is the least natural number n such that $x \le n$. The function $x \mapsto \lceil x\rceil$ is called the *ceiling function*. For example

$$\lceil 3\tfrac{1}{4}\rceil = \lceil \pi\rceil = \lceil 4\rceil = 4.$$

Proposition 3.8 (Shannon Code)**.** *Let p be a probability measure on $\{1,\ldots,s\}$ such that $p_i > 0$ for each i.*

(i) *There is a prefix-free binary code with codewords $u(1),\ldots,u(s)$ such that $u(i)$ has length $\lceil\log_2 \frac{1}{p_i}\rceil$.*

(ii) *When $u(i)$ is used to encode i for each $i \in \{1,\ldots,s\}$, the expected codeword length is less than $1 + H(p)$.*

Proof. Let $\ell_i = \lceil\log_2 \frac{1}{p_i}\rceil$. By definition of the ceiling function

$$\log_2 \frac{1}{p_i} \le \ell_i. \tag{†}$$

By (†), we have $\frac{1}{p_i} \le 2^{\ell_i}$ and so $2^{-\ell_i} \le p_i$ for each i. Hence

$$\sum_{i=1}^{s} 2^{-\ell_i} \le \sum_{i=1}^{s} p_i = 1.$$

Therefore Kraft's Inequality is satisfied for the lengths $\ell_1,\ldots,\ell_s$ and (i) follows from the 'if' direction of Proposition 2.9. Again by (†), we have $\ell_i < 1 + \log_2 \frac{1}{p_i}$. Hence if $\bar{l}$ is the expected length then

$$\bar{\ell} = \sum_{i=1}^{s} p_i\ell_i < \sum_{i=1}^{s} p_i(1+\log_2\frac{1}{p_i}) = \sum_{i=1}^{s} p_i + \sum_{i=1}^{s} p_i \log_2 \frac{1}{p_i} = 1 + H(p)$$

as required for (ii). □

A binary code with lengths as in Proposition 3.8 is called a *Shannon code*. Note that if $p_i = 1/2^{c_i}$ then the corresponding length is c_i, as in Exercise 3.5.

Gibbs' Inequality. Motivated by Exercise 3.7, we now show that the expected length of a prefix-free binary code for a probability measure p is always at least $H(p)$. Thus Shannon Codes have expected length within 1 of the best possible. We need the following fundamental inequality.

Lemma 3.9 (Gibbs' Inequality). *Let p and q be probability measures on the set $\{1, \ldots, s\}$. Then*

$$-\sum_{i=1}^{s} p_i \log_2 p_i \le -\sum_{i=1}^{s} p_i \log_2 q_i$$

where the right-hand side is interpreted as $+\infty$ if $q_i = 0$ for some $p_i \neq 0$.

This proof is non-examinable, since the ideas come from analysis rather than coding theory, but is I hope of interest. The main idea is in the third paragraph: it is related to the theory of Lagrange multipliers.

Proof. We may assume without loss of generality that $p_i > 0$ for all i, since the ith summand on both sides is 0 when $p_i = 0$. If $s = 1$ then $p = q$ so equality holds. Suppose that $s \ge 2$.

Let $G(q) = -\sum_{i=1}^{s} p_i \log_2 q_i$. We consider how $G(q)$ changes as we vary q over the set $\{(q_1, \ldots, q_s) \in \mathbb{R}^s : q_i \ge 0, \sum_{i=1}^{s} q_i = 1\}$ of probability measures. Since the set of probability measures is closed and bounded, a minimum exists. Suppose that G takes its minimum value at $q^\star$. We have $0 < q_i^\star < 1$ for each i, since $-p_i \log_2 q_i \to \infty$ as $q_i \to 0$.

Let $1 \le j < k \le s$. On the line through $q^\star$ where we slightly increase q_j and slightly decrease q_k, the values of $G(q)$ are given by

$$\begin{aligned} g(t) &= G(q_1^\star, \ldots, q_j^\star + t, \ldots, q_k^\star - t, \ldots, q_s^\star) \\ &= p_j \log_2(q_j^\star + t) + p_k \log_2(q_k^\star - t) + \sum_{i \neq j,k} p_i \log_2 q_i^\star \end{aligned}$$

The function g is minimized when $t = 0$. Therefore, differentiating with respect to t, we get

$$g'(t) = \frac{p_j}{q_j^\star \log_e 2} - \frac{p_k}{q_k^\star \log_e 2}.$$

Here we used that the derivative of $\log_2 x = \frac{\log_e x}{\log_e 2}$ is $\frac{1}{x \log_e 2}$. Hence $p_j/q_j^\star = p_k/q_k^\star$ for all j and k. Since $p_i > 0$ for each i, there exists a constant C such that $q_i^\star = Cp_i$ for all i. Since $\sum_{i=1}^{s} q_i^\star = 1$, we have $C = 1$. Therefore the unique minimum is at $q^\star = p$ and

$$-\sum_{i=1}^{s} p_i \log_2 q_i \ge -\sum_{i=1}^{s} p_i \log_2 q_i^\star = -\sum_{i=1}^{s} p_i \log_2 p_i$$

as required. □

Corollary 3.10. *Suppose that a prefix-free binary code with codewords of lengths $\ell_1, \ldots, \ell_s$ is used to encode symbols from $\{1, \ldots, s\}$. If symbol i has probability p_i then the expected codeword length $\bar{\ell}$ is at least $H(p)$.*

As motivation for the following proof observe that

$$\bar{\ell} = \sum_{i=1}^{s} p_i \ell_i = -\sum_{i=1}^{s} p_i \log_2 2^{-\ell_i}.$$

If $\sum_{i=1}^{s} 2^{-\ell_i} = 1$ we could apply Gibbs' Inequality. What do we know about $\sum_{i=1}^{s} 2^{-\ell_i}$ in general?

Proof. Let $K = \sum_{i=1}^{s} 2^{-\ell_i}$. By Kraft's Inequality in the 'only if' direction, $K \leq 1$. Take $q_i = 2^{-\ell_i}/K$ in Gibbs' Inequality to get

$$-\sum_{i=1}^{s} p_i \log_2 \frac{2^{-\ell_i}}{K} \geq -\sum_{i=1}^{s} p_i \log_2 p_i = H(p).$$

The left-hand side above is

$$\sum_{i=1}^{s} p_i \ell_i - \sum_{i=1}^{s} p_i \log_2 \frac{1}{K} = \bar{\ell} + \sum_{i=1}^{s} p_i \log_2 K = \bar{\ell} + \log_2 K.$$

Since $K \leq 1$, we have $\log_2 K < 0$. Therefore combining the two displayed equations we get

$$\bar{\ell} \geq \bar{\ell} + \log_2 K \geq H(p)$$

as required. □

Summary. If symbols come from an alphabet with probability measure p then the expected length of a prefix-free binary code is at least $H(p)$. A Shannon code has expected length less than $H(p) + 1$.

4. Entropy and the Noiseless Coding Theorem

Definition 4.1 (Entropy of random variable)**.** Let X be a random variable taking values in a set $\mathcal{X}$. The *entropy* of X is

$$H(X) = -\sum_{x \in \mathcal{X}} \mathbf{P}[X = x] \log_2 \mathbf{P}[X = x].$$

Equivalently, $H(X) = H(p)$ where the probability measure p on $\mathcal{X}$ is defined by $p_x = \mathbf{P}[X = x]$ and $H(p)$ is as defined in Definition 3.6.

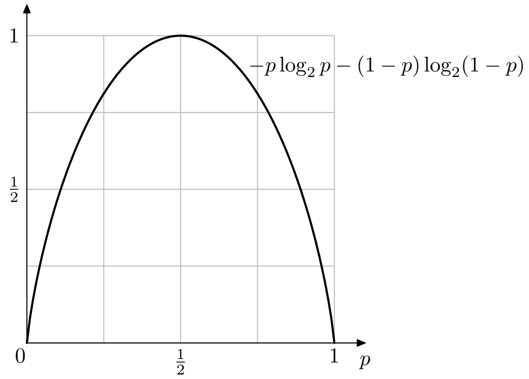
Since the bit is the unit of information, Exercise 3.4 and the results in Proposition 3.8 and Corollary 3.10 (summarized in the box above) suggest the following intuitive interpretation of entropy.

The entropy of a random variable X is the average number of bits needed to store X. This is the **amount of information in** X.

Exercise 4.2.

(1) Let X and Y be independent tosses of a fair coin. Then $H(X) = H(Y) = 1$ and $H((X,Y)) = 2$.

(2) Let U be a toss of a coin biased to land heads with probability p. Then $H(U) = -p \log p - (1-p)\log(1-p)$ as shown in the graph below.



Prove that, as suggested by the graph, the entropy is 0 when $p = 0$ and when $p = 1$ and is maximized at 1 when $p = \frac{1}{2}$. Is it intuitive that the graph is symmetric about $\frac{1}{2}$?

We define the *joint* entropy of random variables X and Y taking values in sets $\mathcal{X}$ and $\mathcal{Y}$ by

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \mathbf{P}[X=x, Y=y]\log_2 \mathbf{P}[X=x,Y=y].$$

Equivalently, $H(X,Y)$ is the entropy of the random variable (X,Y) taking values in $\mathcal{X}\times\mathcal{Y}$.

Exercise 4.3. Let X and Y be two independent flips of a coin biased to land heads with probability p. What is the joint distribution of X and Y? Express $H(X)$, $H(Y)$, $H(X,Y)$ and $H(X,X)$ in terms of $h = -p\log_2 p - (1-p)\log_2(1-p)$.

The proof of the following lemma is left to you on Problem Sheet 3. A special case was seen in the previous exercise.

Lemma 4.4. *If X and Y are independent random variables then $H(X,Y) = H(X) + H(Y)$.*

Example 4.5. A memoryless source produces symbols from the alphabet $\{\mathtt{a},\mathtt{b},\mathtt{c}\}$ so that $\mathbf{P}[U_t = \mathtt{a}] = \frac{1}{2}$, $\mathbf{P}[U_t = \mathtt{b}] = \frac{2}{5}$, $\mathbf{P}[U_t = \mathtt{c}] = \frac{1}{10}$ for all times t. We have

$$H(U_1) = \tfrac{1}{2}\log_2 2 + \tfrac{2}{5}\log_2 \tfrac{5}{2} + \tfrac{1}{10}\log_2 10 = \tfrac{1}{5} + \tfrac{1}{2}\log_2 5 \approx 1.361.$$

The Shannon code for the probability distribution $(\frac{1}{2}, \frac{2}{5}, \frac{1}{10})$ has codewords of lengths $\lceil \log_2 2\rceil = 1$, $\lceil \log_2 \frac{3}{2}\rceil = 2$, $\lceil \log_2 10\rceil = 4$. With one choice of codewords,

$$\mathtt{a} \mapsto 0, \mathtt{b} \mapsto 10, \mathtt{c} \mapsto 1111.$$

(We avoid shortening 1111 in order to follow the proof of Theorem 4.6.) The expected length is $\frac{1}{2}1 + \frac{2}{5}2 + \frac{1}{10}4 = \frac{17}{10}$. As expected by Proposition 3.8 (upper bound) and Corollary 3.10 (lower bound),

$$H(U_1) \leq \tfrac{17}{10} < H(U_1) + 1.$$

Since $1.7 - H(U_1) \approx 0.339$, for every symbol encoded, the Shannon code is worse by 0.339 bits compared to the entropy bound. Just for this example, say that 0.339 bits are *wasted* per symbol.

Suppose we now encode pairs of symbols. By a similar argument to Exercise 4.3, the probability distribution is as shown in the table below:

	`a`	`b`	`c`
`a`	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{20}$
`b`	$\frac{1}{5}$	$\frac{4}{25}$	$\frac{1}{25}$
`c`	$\frac{1}{20}$	$\frac{1}{25}$	$\frac{1}{100}$

For example the entry in row `a` and column `b` is the probability of `ab`, namely $\frac{1}{2} \times \frac{2}{5} = \frac{1}{5}$. The Shannon code has codewords of lengths $\lceil \log_2 4 \rceil$, $\lceil \log_2 5 \rceil$, $\lceil \log_2 20 \rceil$, $\lceil \log_2 5 \rceil$, $\lceil \log_2 \frac{25}{4} \rceil$, $\lceil \log_2 25 \rceil$, $\lceil \log_2 20 \rceil$, $\lceil \log_2 25 \rceil$, $\lceil \log_2 100 \rceil$, namely $2, 3, 5, 3, 3, 5, 5, 5, 7$. The expected length is

$$\tfrac{1}{4}2 + \tfrac{1}{5}3 + \tfrac{1}{20}5 + \tfrac{1}{5}3 + \tfrac{4}{25}3 + \tfrac{1}{25}5 + \tfrac{1}{20}5 + \tfrac{1}{25}5 + \tfrac{1}{100}7 = \tfrac{315}{100}.$$

By Lemma 4.4, the entropy of a pair of symbols is $2H(U_1) \approx 2.722$. Since $3.15 - 2H(U_1) \approx 0.428$, $0.428/2 = 0.214$ bits are wasted per symbol. This improves on 0.339 encoding single symbols.

The table below shows what happens when we encode symbols r at a time: $\bar{\ell}^{(r)}$ is the expected length of the Shannon code and the final column shows

$$\epsilon^{(r)} = \frac{\bar{\ell}^{(r)}}{r} - H(U_1),$$

the number of wasted bits per symbol.

r	$\bar{\ell}^{(r)}$	$rH(U_1)$	$\bar{\ell}^{(r)} - rH(U_1)$	$\epsilon^{(r)}$
1	1.700	1.361	0.339	0.339
2	3.150	2.722	0.428	0.214
3	4.475	4.083	0.392	0.131
4	5.800	5.444	0.356	0.089
5	7.156	6.805	0.351	0.070
6	8.528	8.1658	0.362	0.060
7	9.900	9.5267	0.373	0.053
8	11.268	10.889	0.380	0.048
9	12.634	12.249	0.386	0.043
10	14.000	13.610	0.390	0.039

For example, when $r = 10$, only 0.039 bits are wasted per symbol.

To state Shannon's Noiseless Coding Theorem concisely we need some more notation. Given a source $U_1, U_2, \ldots$ producing symbols from an alphabet $\mathcal{A}$, a binary code $C^{(r)}$ and an encoder $f^{(r)} : \mathcal{A}^r \to C^{(r)}$, let $\bar{f}^{(r)}$ be the expected length of a codeword encoding $(U_1, \ldots, U_r)$. In symbols

$$\bar{f}^{(r)} = \sum_{(u_1,\ldots,u_r)\in\mathcal{A}^r} \ell\big(f^{(r)}(u_1,\ldots,u_r)\big)\mathbf{P}\big[(U_1,\ldots,U_r) = (u_1,\ldots,u_r)\big].$$

When $r = 1$, we encode symbols one at a time and write f rather than $f^{(1)}$.

For instance in Example 4.5, we had $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and saw prefix-free encoders $f^{(r)}$ for the r-tuples of symbols from $\mathcal{A}$. When $r = 1$ we encoded symbols one at a time with $f(\mathtt{a}) = 0$, $f(\mathtt{b}) = 10$, $f(\mathtt{c}) = 1111$; the expected length was $\bar{f} = \bar{f}^{(1)} = \frac{1}{2}1 + \frac{2}{5}2 + \frac{1}{10}4 = 1.7$. The second column in the table shows the values of $\bar{f}^{(r)}$; for instance $\bar{f}^{(10)} = 14.000$.

Theorem 4.6 (Shannon's Noiseless Coding Theorem, Memoryless Case). *A memoryless source produces symbols $U_1, U_2, \ldots$ from an alphabet $\mathcal{A}$ such that each symbol has positive probability. Let $h = H(U_1)$.*

(i) *There exists a prefix-free binary code C and an injective encoder $f : \mathcal{A} \to C$ such that $\bar{f} < h + 1$.*
(ii) *For any prefix-free injective encoder g, we have $\bar{g} \geq h$.*

Proof. We may assume without loss of generality that $\mathcal{A} = \{1, \ldots, s\}$. Let $p_i = \mathbf{P}[U_1 = i]$ for each $i \in \mathcal{A}$.

(i) Define f so that $f(1), \ldots, f(s)$ is the prefix-free Shannon code given by Proposition 3.8(i) for p. (Note the hypothesis $p_i > 0$ for each i is satisfied.) By Proposition 3.8(ii), we have $\bar{f} < h + 1$.
(ii) Suppose that $g : \mathcal{A} \to C$ is a prefix-free injective encoder. Then $g(1), \ldots, g(s)$ is a prefix-free binary code so by Corollary 3.10 we have $\bar{g} \geq h$. □

Observe from the final column of the table in Example 4.5, the proportion of 'wasted' bits $\bar{f}^{(r)}/r - H(U_1)$ gets smaller with r. The asymptotic version of Shannon's theorem below says this proportion can be made arbitrarily small. For example, $\bar{f}^{(10)}/10 - H(U_1) \approx 0.039$, so $f^{(10)}$ is a suitable encoder in Theorem 4.7(i) when $\epsilon = 0.05$, but not when $\epsilon = 0.01$.

Theorem 4.7 (Shannon's Source Coding Theorem, Asymptotic Memoryless Case)**.** *A memoryless source produces symbols $U_1, U_2, \ldots$ from an alphabet $\mathcal{A}$ such that each symbol has positive probability. Let $h = H(U_1)$.*

(i) *For every $\epsilon > 0$ there exists $r \in \mathbb{N}$, a prefix-free binary code $C^{(r)}$ and an injective encoder $f^{(r)} : \mathcal{A}^r \to C^{(r)}$ such that*

$$\frac{\bar{f}^{(r)}}{r} < h + \epsilon.$$

(ii) *For any r and any prefix-free injective encoder $g^{(r)}$,*

$$\frac{\bar{g}^{(r)}}{r} \geq h.$$

Proof. For $r \in \mathbb{N}$, let $S_1 = (U_1, \ldots, U_r)$, $S_2 = (U_{r+1}, \ldots, U_{2r})$, and so on by the r-tuples of symbols produced by the source. Then $S_1, S_2, \ldots$ is a memoryless source with alphabet $\mathcal{A}^r$, producing each r-tuple in $\mathcal{A}^r$ with non-zero probability. By the generalization of Lemma 4.4 to r independent random variables we have

$$H(S_1) = H(U_1, \ldots, U_r) = H(U_1) + \cdots + H(U_r) = hr.$$

By Theorem 4.6(i) there is a prefix-free code $C^{(r)}$ and an injective encoder $f^{(r)} : \mathcal{A}^r \to C^{(r)}$ such that $\bar{f}^{(r)} \leq hr + 1$. Hence if we choose r so that $r > 1/\epsilon$ then

$$\frac{\bar{f}^{(r)}}{r} < h + \frac{1}{r} < h + \epsilon$$

as required for (i). For (ii), Theorem 4.6(ii) immediately implies that $\bar{g}^{(r)} \geq hr$, again as required. □

It is routine to extend Theorem 4.6 and Theorem 4.7 to the case when some symbols in $\mathcal{A}$ are never produced by the source. We illustrate this at the end of the following example.

Example 4.8. A memoryless source $U_1, U_2, \ldots$ produces symbols from the Roman alphabet $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ according to the the probability distribution of standard English:

$$p_{\mathtt{e}} = 0.127, p_{\mathtt{t}} = 0.091, p_{\mathtt{a}} = 0.082, \ldots, p_{\mathtt{z}} = 0.001.$$

We have $H(U_1) = H(p) = 4.195\ldots < 4.2$.

By Theorem 4.6(i), there is a prefix-free binary code C and an injective prefix-free encoder $f : \mathcal{A} \to C$ such that $\bar{f} < 5.2$. *Exercise:* why is using 5.2 bits per character not very impressive?

By Theorem 4.7(i), taking $\epsilon = \frac{1}{100}$, there exists $r \in \mathbb{N}$, a prefix-free binary code C and an injective prefix-free encoder $f^{(r)} : \mathcal{A}^r \to C$ such that

$\bar{f}^{(r)}/r < 4.21$. (In fact the proof shows that we may take $r = 100$.) Now about 4.21 bits per character are used. This improves on using 26 codewords of length 5.

If instead the source produces English words then it is not memoryless; instead, as seen in Example 3.2(3), one can often predict the next character from those before. We shall see how to encode such sources in Part C of the course. According to one estimate of Shannon, only 2.6 bits per character are needed.

Finally, suppose that the alphabet is extended by a new symbol $\heartsuit$, such that $\mathbf{P}[U_t = \heartsuit] = 0$ for all times t. Let $v = f^{(r)}(\mathtt{aa}\ldots\mathtt{a})$ be the encoding of 'a' repeated r times. We extend $f^{(r)}$ by setting $\widetilde{f}^{(r)}(u_1, \ldots, u_r) = v$ whenever some u_i is $\heartsuit$. The new encoder $\widetilde{f}^{(r)}$ is not injective, but decoding is still unambiguous because the codeword v is only seen when the message contains $\mathtt{aa}\ldots\mathtt{a} \in \mathcal{A}^r$.

5. Huffman codes

Returning to encoding symbol by symbol, we end by defining Huffman codes. We prove in Corollary 5.9 that they have the shortest possible expected length of prefix-free codes. Huffman codes are widely used because they are efficient to construct: they are used in JPEG image compression, MP3 audio compression and ZIP file compression.

The following definition is non-standard, but very useful.

Definition 5.1. A *Huffman set* for a probability measure $(p_1, \ldots, p_s)$ is a set of pairs (i, u) where, in each pair, $i \in \{1, \ldots, s\}$ and u is a binary word. The *weight* of a Huffman set is the sum of the p_i for those i in a pair in the set.

The input and output to each step of the Huffman algorithm is a collection of Huffman sets. We shall see that (i, v) appears in a Huffman set if and only if the codeword $v(i)$ encoding i **ends** with v. We say that v is a *suffix* of $v(i)$.

For example, $\{(1, \varnothing)\}$ and $\{(2, 001), (3, 101)\}$ are Huffman sets having weights p_1 and $p_2 + p_3$. From the second Huffman set we know that the codeword $v(3)$ encoding 3 has 101 as a suffix. For instance, $v(3)$ might be 101 or 0101 or 1101, and so on.

Algorithm 5.2 (Huffman)**.** The input is a probability measure $(p_1, \ldots, p_s)$ with $s \geq 2$.

- **Begin**. Take the s Huffman sets: $\{(1, \varnothing)\}, \ldots, \{(s, \varnothing)\}$.
- **Step**. Let X and Y be Huffman sets of the least two weights. Let

$$Z = \{(i, 0v) : (i, v) \in X\} \cup \{(j, 1w) : (j, w) \in Y\}.$$

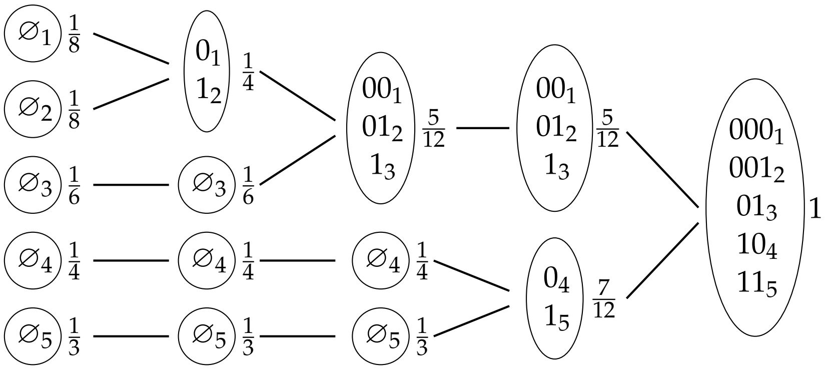
Replace X and Y with Z. **[Was misprinted as $0w$, change!]**

- **End**. End when there is only one Huffman set. Its pairs are $\big(1, v(1)\big), \ldots, \big(s, v(s)\big)$ where $v(i)$ is the codeword encoding i.

Example 5.3. Let $(p_1, p_2, p_3, p_4, p_5) = (\frac{1}{8}, \frac{1}{8}, \frac{1}{6}, \frac{1}{4}, \frac{1}{3})$. The table shows the full Huffman algorithm. Note that after step (1) there are two Huffman sets of the second least weight $\frac{1}{4}$. We chose $\big\{(1,0),(2,1)\big\}$ (rather than $\big\{(4,\varnothing)\big\}$); each of its suffixes 0 and 1 then had 0 prepended.

Begin	$\big\{(1,\varnothing)\big\},\big\{(2,\varnothing)\big\},\big\{(3,\varnothing)\big\},\big\{(4,\varnothing)\big\},\big\{(5,\varnothing)\big\}$
(1)	$\big\{(1,0),(2,1)\big\},\big\{(3,\varnothing)\big\},\big\{(4,\varnothing)\big\},\big\{(5,\varnothing)\big\}$
(2)	$\big\{(1,00),(2,01),(3,1)\big\},\big\{(4,\varnothing)\big\},\big\{(5,\varnothing)\big\}$
(3)	$\big\{(1,00),(2,01),(3,1)\big\},\big\{(4,0),(5,1)\big\}$
(4)	$\big\{(1,000),(2,001),(3,01),(4,10),(5,11)\big\}$
End	$1 \mapsto 000, 2 \mapsto 001, 3 \mapsto 01, 4 \mapsto 10, 5 \mapsto 11$

It is convenient to perform the algorithm by constructing an oriented rooted binary tree, starting with s leaves, and finishing at the root. We follow the same convention from §2 that we step up for 1 and down for 0 (and now horizontally for no change). Huffman sets are shown in ellipses with the weight to the right, denoting the pair $(2,\varnothing)$ by $\varnothing_2$ and the pair $(1,00)$ by 00_1, and so on, to save space.



$\varnothing_1$ $\frac{1}{8}$
$\varnothing_2$ $\frac{1}{8}$
$\varnothing_3$ $\frac{1}{6}$
$\varnothing_4$ $\frac{1}{4}$
$\varnothing_5$ $\frac{1}{3}$
0_1 1_2 $\frac{1}{4}$
$\varnothing_3$ $\frac{1}{6}$
$\varnothing_4$ $\frac{1}{4}$
$\varnothing_5$ $\frac{1}{3}$
00_1 01_2 1_3 $\frac{5}{12}$
$\varnothing_4$ $\frac{1}{4}$
$\varnothing_5$ $\frac{1}{3}$
00_1 01_2 1_3 $\frac{5}{12}$
0_4 1_5 $\frac{7}{12}$
000_1 001_2 01_3 10_4 11_5 1

Exercise 5.4. Use the tree method to construct the Huffman code for the probability distribution $(p_1, p_2, p_3, p_4, p_5) = (\frac{1}{8}, \frac{1}{8}, \frac{1}{6}, \frac{1}{4}, \frac{1}{3})$ in Example 5.3 choosing at step (2) the Huffman sets $\big\{(3,\varnothing)\big\}$ and $\big\{(4,\varnothing)\big\}$. (See the slides on Moodle for the tree and code.) Is there a more efficient code?

Question 5 on Sheet 3 gives a bigger example where choices in the Huffman algorithm lead to codes having different codeword lengths and different maximum length. By Corollary 5.9, these choices do not change the expected codeword length.

Lemma 5.5. *Huffman codes are prefix-free*

Proof. Consider the codewords $v(i)$ and $v(j)$ output by the algorithm.

- Find the step when there are pairs (i, v) and (j, w) entering the same Huffman set. By swapping i and j if necessary, we may suppose that in this step we prepend 0 to v and 1 to w.
- After this step, $(i, 0v)$ and $(j, 1w)$ are in the same Huffman set.
- In each subsequent step involving the Huffman set for i and j, we prepend the same bit to all suffixes in pairs in the set. Therefore $0v$ is extended to $v(i) = x0v$ and $1w$ is extended to $v(j) = x1w$, for some binary word x.

Comparing on $x0$ and $x1$ shows that $v(i)$ is not a prefix of $v(j)$ and $v(j)$ is not a prefix of $v(i)$. □

We end by showing that Huffman codes are best possible, in the following sense.

Definition 5.6. We say that the binary code $v(1), \ldots, v(s)$ is *optimal* for the probability measure $(p_1, \ldots, p_s)$ if it is prefix-free, and no other prefix-free code with s codewords has a smaller expected length.

We require the following lemma. As motivation, we note that the conclusion of this lemma holds for Huffman codes. For instance in Example 5.3, the two least probable symbols 1 and 2 got the longest codewords 000 and 001, differing only in their final position.

Part (a) should be intuitive: it simply says that in an optimal code less probable symbols get longer codewords. You are asked to give a careful proof in Question 1 on Problem Sheet 4.

Lemma 5.7. *In an optimal code for the probability measure $p_1 \leq p_2 \leq \ldots \leq p_s$ in which the codewords $v(1), v(2), \ldots, v(s)$ have lengths $\ell_1, \ell_2, \ldots, \ell_s$,*

(a) $\ell_1 \geq \ell_2 \geq \ldots \geq \ell_s$*;*

(b) $\ell_1 = \ell_2$ *and two of the codewords of this length differ only in their final positions.*

Proof of (b). By (a), ℓ_1 is the longest length. Let $\ell = \ell_1$. Thus $v(1) = v_1 \ldots v_{\ell-1} v_\ell$. Since the code is prefix-free, $v_1 \ldots v_{\ell-1}$ is not a codeword. We can reduce the expected length of the code by replacing $v(1)$ with $v_1 \ldots v_{\ell-1}$ *unless* $v_1 \ldots v_{\ell-1}$ is a prefix of another codeword. Since ℓ is the longest length, the only codeword that may have $v_1 \ldots v_{\ell-1}$ as a prefix is $v_1 \ldots v_{\ell-1} \overline{v}_\ell$, where $\overline{v}_\ell = 1 - v_\ell$ is the bit-flip of v_ℓ. Therefore $v_1 \ldots v_{\ell-1} \overline{v}_\ell$ is a codeword, differing from $v_1 \ldots v_{\ell-1} u_\ell$ only in its final position. □

Proposition 5.8. *Suppose that the probability measure $p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_s$* **[inequalities added in lecture]** *has an optimal code with expected length $\overline{L}$ and that the probability measure $p_1 + p_2, p_3, \ldots, p_s$ has an optimal prefix-free code with expected length $\overline{M}$. Then*

$$\overline{L} \geq \overline{M} + p_1 + p_2.$$

Proof. Let $v(1), v(2), v(3), \ldots, v(s)$ be an optimal code for $p_1, p_2, p_3, \ldots, p_s$. Let $\ell = \ell\big(v(1)\big)$. By Lemma 5.7(a), ℓ is the longest length of a codeword. By Lemma 5.7(b), reordering the codewords of length ℓ if necessary, we may assume that

$$\begin{aligned} v(1) &= x_1 \ldots x_{\ell-1}0 \\ v(2) &= x_1 \ldots x_{\ell-1}1 \end{aligned}$$

for some binary word $x_1 \ldots x_{\ell-1}$. Since the code is prefix-free, $x_1 \ldots x_{\ell-1}$ is not a codeword. Hence $x_1 \ldots x_{\ell-1}, v(3), \ldots, v(s)$ is a prefix-free code for $p_1 + p_2$, $p_3, \ldots p_s$ with codewords of length $\ell - 1$, $\ell_3, \ldots, \ell_s$. Its expected length is

$$\overline{L} - (p_1\ell - p_2\ell) + (p_1 + p_2)(\ell - 1) = \overline{L} - (p_1 + p_2).$$

Since an optimal prefix-free code for $p_1 + p_2, p_3, \ldots, p_s$ has expected length $\overline{M}$, we have $\overline{L} - (p_1 + p_2) \geq \overline{M}$ as required. □

Corollary 5.9. *Huffman codes are optimal.*

Proof. We work by induction on the size of the alphabet s. In the base case $s = 2$ and the Huffman algorithm gives the clearly optimal code 0, 1. By reordering symbols if necessary, we may assume that the probability measure is $(p_1, \ldots, p_s)$ where

$$p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_s$$

and that in Step 1 we form the Huffman set $\big\{(1,0), (2,1)\big\}$ of weight $p_1 + p_2$. The algorithm continues with Huffman sets of weights $p_1 + p_2, p_3 \leq \ldots \leq p_s$. As seen in the proof of Lemma 5.5, in each subsequent step involving the Huffman set for 1 and 2, we prepend the same bit to all suffixes in pairs in the set. Therefore if the algorithm terminates with $v(1), v(2), v(3), \ldots, v(s)$ then

$$\begin{aligned} v(1) &= x_1 \ldots x_{\ell-1}0 \\ v(2) &= x_1 \ldots x_{\ell-1}1 \end{aligned}$$

for some $x_1 \ldots x_{\ell-1}$. By making the same choices but starting at Step 2 with the the probability measure is $q = (p_1 + p_2, p_3, \ldots, p_s)$, we obtain the Huffman code with codewords $x_1 \ldots x_{\ell-1}$, $v(3), \ldots, v(s)$. The expected length of this code, for the probability measure q, is

$$\overline{m} = (p_1 + p_2)(\ell - 1) + p_3\ell\big(v(3)\big) + \cdots + p_s\ell\big(v(s)\big).$$

and the expected length of the original code for p is

$$\overline{\ell} = p_1\ell + p_2\ell + p_3\ell\big(v(3)\big) + \cdots + p_s\ell\big(v(s)\big).$$

By the same calculation as Proposition 5.8, $\overline{\ell} = \overline{m} + p_1 + p_2$.

Let $\overline{L}$ be the expected length of an optimal code for p. By induction, the Huffman code for q of expected length $\overline{m}$ is optimal. Proposition 5.8

implies that $\overline{L} \geq \overline{m} + p_1 + p_2$. We just saw that $\overline{m} + p_1 + p_2 = \overline{\ell}$. Therefore $\overline{L} \geq \overline{\ell}$ and the code for p is optimal. □

6. Extras for Part A

This section is non-examinable and included for interest only.

More on unique decipherability. Recall from Definition 2.1 that $\mathcal{A}^\star$ is the set of words (of any length) from the alphabet $\mathcal{A}$. Given a binary code C and an encoder $f : \mathcal{A} \to C$, define $f^\star : \mathcal{A}^\star \to \{0,1\}^\star$ by

$$f^\star(u_1, \ldots, u_s) = f(u_1) \ldots f(u_s)$$

where the right-hand side is the concatenation of the codewords $f(u_1)$, $\ldots$, $f(u_s) \in C$. Earlier in §2 we said that C was uniquely decipherable if there was a unique way to decode each such concatenation. This is consistent with the following standard definition.

Definition 6.1. An encoder $f : \mathcal{A} \to C$ is *uniquely decipherable* if $f^\star : \mathcal{A}^\star \to \{0,1\}^\star$ is injective. If such an encoder exists we say that C is *uniquely decipherable*.

Our concentration on prefix-free codes is justified by the following proposition (proof temporarily deferred) and corollary.

Proposition 6.2. *A uniquely decipherable binary code C has $\sum_{v \in C} 2^{-\ell(v)} \leq 1$.*

Corollary 6.3. *If C is a uniquely decipherable binary code then there is a prefix-free binary code with the same codeword lengths as C.*

Proof. Let C have codewords $v(1), \ldots, v(s)$ of lengths $\ell_1, \ldots, \ell_s$. By Proposition 6.2, we have $\sum_{i=1}^{s} 2^{-\ell_i} \leq 1$. Hence the lengths satisfy Kraft's Inequality. By the 'if' direction of Proposition 2.9, there is a prefix-free binary code with codewords of lengths $\ell_1, \ldots, \ell_s$. □

Thus any uniquely decipherable binary code can be replaced with a prefix-free binary code without changing the expected codeword length.

There is a very nice proof of Proposition 6.2 using generating functions. For more on generating functions see Wilf's book *generatingfunctionology* (the 2nd edition is free on his website), or do our course MT354/454/5454 Combinatorics. It will be convenient to denote by $[x^n]f(x)$ the coefficient of x^n in a polynomial or power series.

Proof of Proposition 6.2. Suppose that C has a_ℓ codewords of length ℓ for $\ell \in \mathbb{N}$. Let m be the maximum length of a codeword. Let

$$f(x) = a_1 x + a_2 x^2 + \cdots + a_m x^m,$$

chosen so that $[x^n]f(x) = a_n$ for all $n \in \mathbb{N}$. We must show that $f(\frac{1}{2}) \leq 1$. For $n \in \mathbb{N}$,

$$[x^n]f(x)^2 = \sum_{\substack{i,j \in \{1,\ldots,k\} \\ i+j=n}} a_i a_j$$

is the number of concatenated codewords vw of length n; the summand $a_i a_j$ counts those concatenations where v has length i and w has length j. More generally, the coefficient of x^n in $f(x)^t$ is the number of t-fold concatenated codewords of length n. Since there are 2^n binary words of length n, and C is uniquely decipherable, we have

$$[x^n]f(x) + [x^n]f(x)^2 + \cdots + [x^n]f(x)^n \leq 2^n$$

for all $n \in \mathbb{N}$. The left-hand side is the coefficient of x^n in $(1 - f(x))^{-1} = 1 + f(x) + f(x)^2 + \cdots$. Hence, comparing coefficients,

$$\frac{1}{1 - f(x)} \leq 1 + 2x + 4x^2 + \cdots + 2^n x^n + \cdots = \frac{1}{1 - 2x} \qquad (\ddagger)$$

for all $x < \frac{1}{2}$. Evaluating both sides at $\frac{1}{2} - \epsilon$ shows that $1 - f(\frac{1}{2} - \epsilon) > 0$ for all $\epsilon > 0$. Hence $f(\frac{1}{2} - \epsilon) < 1$ for all $\epsilon > 0$. Since f is continuous, it follows that $f(\frac{1}{2}) \leq 1$, as required. □

The following example should help you to see the main idea in the proof. It also shows a typical use of partial fractions to evaluate generating functions.

Example 6.4. The code $\{0, 10, 11\}$ is prefix-free and so is uniquely decipherable. Let $f(x) = x + 2x^2$ as in the proof and for $n \in \mathbb{N}_0$, let w_n be the number of concatenated codewords of total length n. For example, $w_3 = 5$ counts 0**10**, 0**11**, 10**0**, 11**0**, 0**0**0; it is the coefficient of x^5 in $f(x) + f(x)^2 + f(x)^3$. Correspondingly, $[x^3]f(x)^2 = [x^3](x^2 + 4x^3 + 4x^4)$ counts the 4 concatenations 0**10**, 0**11**, 10**0**, 11**0** and $[x^3]f(x)^3$ counts 0**0**0. By (‡) we have

$$\sum_{n=0}^{\infty} w_n x^n = \frac{1}{1 - f(x)} = \frac{1}{1 - x - 2x^2} = \frac{2}{3(1 - 2x)} + \frac{1}{3(1 + x)}.$$

Expanding each partial fraction as a geometric series we get

$$w_n = \tfrac{2}{3} 2^n + \tfrac{1}{3}(-1)^n$$

for $n \in \mathbb{N}_0$. In particular, $\lim_{n \to \infty} \frac{w_n}{2^n} = \frac{2}{3}$. Thus of the 2^n binary words of length n, about $\frac{2}{3}$ are concatenations of codewords.

The general result is given by the following exercise: it needs the idea from Theorem 2.4.3 in Wilf's book.

Exercise 6.5. Suppose that C is a uniquely decipherable binary code. Let $f(x)$ be as in the proof of Proposition 6.2.

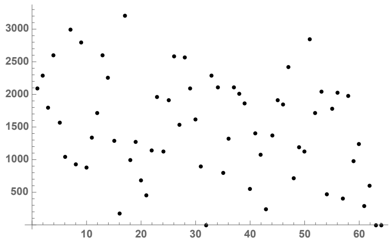
(i) Show that there exists a unique $\kappa \in \mathbb{R}$ with $1 \leq \kappa \leq 2$ such that $f(1/\kappa) = 1$.

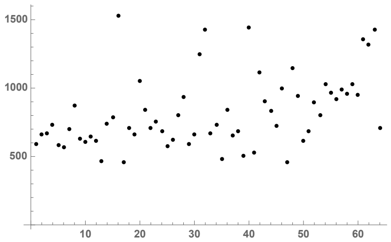
(ii) Let w_n be the number of concatenated codewords of length n. Show that $\lim_{n\to\infty} \frac{\log_2 w_n}{n} = \log_2 \kappa$.

In particular, in the ideal situation for source encoding where Kraft's Inequality is an equality, so $\sum_{v \in C} 2^{-\ell(v)} = 1$, we have $\kappa = 2$. The exercise then says that (as seen in Example 6.4) a positive proportion of all binary words of length n are concatenations of codewords, provided n is sufficiently large.

Thus after source encoding a long message, the sequence of bits should look random. For example, splitting the source encoded message into blocks of 8 bits, we can expect each of the 256 binary words of length 8 to appear roughly equally often. This justifies our assumption in Part B that each symbol (here the symbols correspond to 8 bit binary words) sent across the noisy channel is equally probable.

For example, the graphs below show the frequencies of the 64 binary words of length 6 in the encoding of Chapter 1 of *Persuasion*, first by the inefficient ASCII encoding, and then by the optimal Huffman code. (See the MATHEMATICA notebook `AustenHuffmanExample` on Moodle.)





(B) Channel coding

7. Channel capacity

> **Question.** How quickly can we communicate reliably through a noisy channel?

Noisy channels. In practice the communication channel might be the hard disk (or SSD) in your computer for communication through time, or the air carrying microwave radiation for communication by a mobile phone through space. We use the following mathematical abstraction.

Definition 7.1. Let $\mathcal{A}$ and $\mathcal{B}$ be alphabets. A *discrete memoryless channel* sends a symbol $\alpha \in \mathcal{A}$ to $\beta \in \mathcal{B}$ with a fixed probability $p_{\alpha\beta}$.

Here 'memoryless' is the property that each transmission through the channel is independent of those before.

Denote by X the input symbol and Y the output symbol. Thus

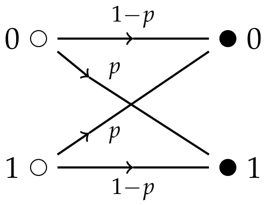
$$\mathbf{P}[Y = \beta | X = \alpha] = p_{\alpha\beta}$$

for all $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$. The matrix with rows labelled by $\mathcal{A}$, columns labelled by $\mathcal{B}$ and entries $p_{\alpha\beta}$ is called the *channel matrix*. Since $\sum_{\beta \in \mathcal{B}} p_{\alpha\beta} = \sum_{\beta \in \mathcal{B}} \mathbf{P}[Y = \beta | X = \alpha] = 1$, by the law of total probability, the channel matrices are stochastic: that is, every row has sum 1.

Example 7.2.

(1) In the introduction we saw the binary symmetric channel in which $\mathcal{A} = \mathcal{B} = \{0, 1\}$ and each bit flips independent with probability p. The channel matrix and the corresponding diagram are:

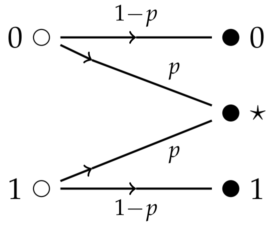
$$\begin{array}{c c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \end{array}$$



(2) In the *binary erasure channel* with *erasure probability* p, $\mathcal{A} = \{0, 1\}$ and $\mathcal{B} = \{0, \star, 1\}$. Each sent bit is received as sent with probability $1 - p$, and otherwise erased by the channel: we model this by supposing that after an erasure, the special symbol $\star$ is received. (Thus the receiver knows a bit was sent, but not what it was.)

The channel matrix and the corresponding diagram are:

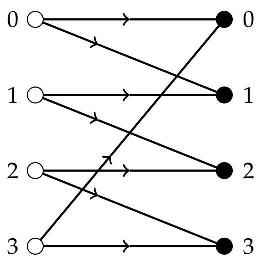
$$\begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & \star & 1 \end{array} \\ \begin{pmatrix} 1-p & p & 0 \\ 0 & p & 1-p \end{pmatrix} \end{array}$$



(3) The *lazy typist* channel with s symbols has input alphabet $\mathcal{A} = \{0, 1, \ldots, s-1\}$ and the same output alphabet. The transition probabilities are

$$\mathbf{P}[Y = x | X = x] = \mathbf{P}[Y = x + 1 \bmod s | X = i] = \tfrac{1}{2}.$$

When $s = 4$ these are shown by the diagram in the margin, in which all arrows have the same probability $\frac{1}{2}$.



Exercise 7.3. Take $s = 4$ in the lazy typist channel, so the input and output alphabets are $\{0, 1, 2, 3\}$.

(a) Write down the channel matrix.

(b) Find a way to encode the four messages `A`, `T`, `G`, `C` using 4 codewords of the same length so that the receiver can decode with zero probability of error.

(c) Specify the decoding rule as a function from words in the output symbols $\{0, 1, 2, 3\}$ to $\{$`A`, `T`, `G`, `C`$\}$.

(d) For each message sent, how many symbols are required? For a perfect typist, how many symbols are required per message?

You are asked to generalize this to the case $s = 2t$ on Problem Sheet 5 Question 3.

Conditional entropy and the Chaining Rule.

Definition 7.4. Let X and Y be discrete random variables taking values in sets $\mathcal{A}$ and $\mathcal{B}$, respectively. The *conditional entropy of X given that $Y = \beta$* is defined by

$$H(X|Y = \beta) = -\sum_{\alpha \in \mathcal{A}} \mathbf{P}[X = \alpha | Y = \beta] \log_2 \mathbf{P}[X = \alpha | Y = \beta].$$

The *conditional entropy of X given Y* is defined by

$$H(X|Y) = \sum_{\beta \in \mathcal{B}} \mathbf{P}[Y = \beta] H(K|Y = y).$$

Observe that $H(X|Y = \beta)$ is the entropy of the probability distribution on $\mathcal{A}$ in which $p_\alpha = \mathbf{P}[X = \alpha | Y = \beta]$ for each $\alpha \in \mathcal{A}$. The conditional entropy of X given Y is then the expected value of $H(X|Y = \beta)$ as β varies over $\mathcal{B}$.

If you have seen conditional expectation, do not be confused into thinking that conditional entropies are random variables: they are just numbers, like usual entropies.

Exercise 7.5. Fix $s \in \mathbb{N}$. Take the lazy typist channel with input and output alphabets $\mathcal{A} = \mathcal{B} = \{0, 1, \ldots, s-1\}$. As usual, let X be the input symbol and let Y be the output symbol.

(i) Suppose that X is uniformly distributed on $\{0, 1, \ldots, s-1\}$, so $\mathbf{P}[X = x] = \frac{1}{s}$ for each x. Find

$$H(X), H(Y), H(X|Y=0), H(X|Y), H(X,Y).$$

(ii) Now suppose that $s = 4$ and X is 0 with probability q and 1 with probability $1-q$. Find
(a) $\mathbf{P}[Y = 1|X = 0], H(Y|X = 0), H(Y|X), \mathbf{P}[Y = 1], H(Y)$;
(b) $\mathbf{P}[X = 0|Y = 1], H(X|Y = 1), H(X|Y)$.

The conditioning argument $\mathbf{P}[Y = \beta] = \sum_{\alpha \in \mathcal{A}} \mathbf{P}[Y = \beta|X = \alpha]\mathbf{P}[X = \alpha]$ is often useful.

The answers are on the slides available from Moodle. The probabilities $\mathbf{P}[Y = \beta|X = \alpha]$ can easily be read from the channel matrix. But a user of the channel cares much more about the probabilities $\mathbf{P}[X = \alpha|Y = \beta]$; we have seen how to compute these using conditional probability, or Bayes' Law. For an analogy with medical testing, see Question 7 on Problem Sheet 1.

Lemma 7.6 (Chaining Rule)**.** *Let X and Y be random variables. Then*

$$H(X|Y) + H(Y) = H(X,Y).$$

Intuitively, the chaining rule says that, after learning the $H(Y)$ bits of information in Y, the pair (X, Y) is determined by a further $H(X|Y)$ bits of information.

The two 'extreme' cases are important:

- If X and Y are independent then Y gives no information about X and $H(X|Y) = H(X)$: this follows from Lemma 4.4, that $H(X,Y) = H(X) + H(Y)$ when X and Y are independent.
- If X is determined by Y (more formally, X is a function of Y) then $H(X|Y) = 0$ and $H(X,Y) = H(Y)$.

In Question 4 on Sheet 4 you are asked to use Gibbs' Inequality to generalize the first case by showing that $H(X) \geq H(X|Y)$ with equality if and only if X and Y are independent.

If you are doing MT361/461/5461 Cipher Systems, then you will have already seen a proof of the Chaining Rule. Otherwise please do Question 5 on Sheet 4 which breaks it down into small steps.

In Example 7.5(ii) we saw that when $s = 4$ and $\mathbf{P}[X = 0] = q$, $\mathbf{P}[X = 1] = 1-q$ we have $H(X) = H(q, 1-q)$ and $H(Y|X) = 1$. Hence

$$H(X,Y) = H(Y|X) + H(X) = 1 + H(q, 1-q)$$

Since $H(Y) = 1 + \frac{1}{2}H(q, 1-q)$ and $H(X|Y) = \frac{1}{2}H(q, 1-q)$, applying the Chaining Rule the other way round gives

$$H(X,Y) = H(X|Y) + H(Y) = 1 + H(q, 1-q).$$

As expected we get the same answer. As expected by the remark before Lemma 7.6, the calculation using $H(Y|X)$ was easier.

Mutual information. Interpreting number of bits of information as a measure of uncertainty, we interpret $H(X|Y)$ as the *uncertainty that remains in X after learning Y*. Therefore $H(X) - H(X|Y)$ is the *uncertainty in X that is removed by learning Y*; that is $H(X) - H(X|Y)$ **is the number of bits of information that Y tells us about X**.

Definition 7.7. The *mutual information* of random variables X and Y is

$$I(X;Y) = H(X) - H(X|Y).$$

Since $H(X) \geq H(X|Y)$ with equality if and only if X and Y are independent, we have $I(X;Y) \geq 0$, with equality if and only if X and Y are independent.

Exercise 7.8. Since entropies are positive, $I(X;Y) \leq H(X)$. When does $I(X;Y) = H(X)$ hold?

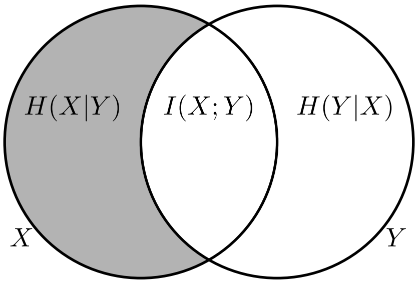
Example 7.9. Let X be the roll of a fair die and let Y be the answer to the question 'Did you roll 1 or 2?'. Then

$$H(X|Y) = \tfrac{1}{3}\log_2 2 + \tfrac{2}{3}\log_2 4 = \tfrac{5}{3}$$

and so $I(X;Y) = H(X) - H(X|Y) = \log_2 6 - \frac{5}{3}$. *Exercise:* compute $I(Y;X) = H(Y) - H(Y|X)$: is it intuitive that it is the same as $I(X;Y)$?

Exercise 7.10. By the Chaining Rule $H(X) - H(X|Y) = H(X) - \big(H(X,Y) - H(Y)\big) = H(X) + H(Y) - H(X,Y)$. Write down the analogous formula for $H(Y) - H(Y|X)$ and deduce that $I(X;Y) = I(Y;X)$.

Thus, as seen in Example 7.9 in a special case, Y is exactly as informative about X as X is about Y. This symmetry justifies representing conditional entropies and mutual information by Venn diagrams, such as the one below in which the shaded region represents $H(X|Y)$.



Example 7.11. Let X and Y be the input and output symbols in the lazy typist channel with s symbols.

(a) By Exercise 7.5(i), if all s input symbols have equal probability $\frac{1}{s}$ then $I(X;Y) = \log_2 s - 1$.

(b) Let s be even. Suppose that

$$p_\alpha = \begin{cases} \frac{2}{s} & \text{if } \alpha \text{ is even} \\ 0 & \text{if } \alpha \text{ is odd.} \end{cases}.$$

Then Y is uniformly distributed so $H(Y) = \log_2 s$ and $I(X;Y) = H(Y) - H(Y|X) = \log_2 s - 1$.

(c) Suppose that $s = 4$ and $p_0 = p_1 = \frac{1}{2}$. Then Y has probability distribution $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, 0)$ and $H(Y) = \frac{3}{2}$. We have $I(X;Y) = \frac{3}{2} - 1 = \frac{1}{2}$. The maximum value of $I(X;Y)$ is 1; by (b) the maximum is attained for $p = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ and $p = (\frac{1}{2}, 0, \frac{1}{2}, 0)$. *Exercise:* find another probability measure on X that maximizes $I(X;Y)$.

Here, and in general, it is easiest to compute $I(X;Y)$ using $I(X;Y) = H(Y) - H(Y|X)$. To motivate the following definition, think of $I(X;Y)$ as the amount of information a symbol $\beta \in \mathcal{B}$ output by the channel tells us about the input symbol $\alpha \in \mathcal{A}$.

Definition 7.12. Let $X \in \mathcal{A}$ be in the input symbol and let $Y \in \mathcal{B}$ be the output symbol to a channel. The *capacity* of the channel is $\max_q I(X;Y)$, where the maximum is taken over all probability measures q on $\mathcal{A}$.

After entropy, this is the fundamental definition in this course.

Example 7.13.

(a) Let $p \leq 1/2$. The capacity of the Binary Symmetric Channel with error probability p is $1 - H(p, 1-p)$.

(b) The capacity of the Binary Erasure Channel with erasure probability p is $1 - p$. You are asked to show this in Question 1 on Problem Sheet 5. *Exercise:* draw a graph comparing (a) and (b).

(c) We saw in Example 7.11 that for the lazy typist channel with $s = 4$, the maximum of $I(X;Y)$ is 1. Correspondingly, we saw that it is possible to transmit 1 bit of information through the channel with zero probability of incorrect decoding, using the codewords $\{0, 2\} \subseteq \{0, 1, 2, 3\}$. In general for s symbols, the same argument shows that the maximum is $\log_2 s - 1$, so this is the capacity of the channel, and $\log_2 s - 1$ bits can be sent with each use of the channel. (See Question 3 on Problem Sheet 5.)

Statement of Shannon's Noisy Coding Theorem.

Theorem 7.14. *Fix a discrete memoryless channel with input alphabet $\mathcal{A}$ and output alphabet $\mathcal{B}$ of capacity c.*

(a) *Let $\epsilon > 0$ be given. For every $r < c$ there exists $n \in \mathbb{N}$ and a code $C \subseteq \mathcal{A}^n$ such that $|C| \geq 2^{rn}$ and the error probability when C is used to send codewords through the channel is less than ϵ.*

(b) *If $r > c$ then, when n is large, it is impossible to find a code as in (a).*

We make the following remarks:

- In fact we will have $|C| \approx 2^{rn}$: the inequality is necessary only because 2^{rn} may not be an integer.
- For a codeword $u \in C$, we define the error probability for u to be the probability that when u is sent through the channel, and v is received, v is not decoded as u. The claim in (a) is that, by choosing the code *and decoding rule* suitably, we can make all these probabilities $< \epsilon$.

Example 7.15. Take the lazy typist channel on $\{0, 1, 2, 3\}$. The capacity of the channel is 1 by Example 7.11. In Example 7.3 we used the encoder

$$\mathtt{A} \mapsto 00, \mathtt{T} \mapsto 02, \mathtt{G} \mapsto 20, \mathtt{C} \mapsto 22.$$

and decoder $00, 01, 10, 11 \mapsto \mathtt{A}$, and so on. By generalizing the encoder to n-tuples of symbols it is not hard to prove that (a) in Shannon's Noisy Coding Theorem holds.

Example 7.15 is slightly unusual in that we can take $r = c$ and n does not need to be large. In general, as seen in the proof of Theorem 4.7 (Shannon's Source Coding Theorem), it is necessary to take n large.

8. Nearest neighbour decoding and Hamming distance

> **Question.** What decoding rule minimizes the probability of decoding error?

Hamming distance. In this section we answer the question above using ideas due to Hamming. These are developed more further in our course MT361/461/5461 Error-correcting Codes.

Definition 8.1. Let $\mathcal{A}$ be an alphabet. Let $u, v \in \mathcal{A}^n$ be words of length n. The *Hamming distance* between u and v, denoted $d(u, v)$, is the number of positions in which u and v are different.

In mathematical notation, $d(u, v) = \big|\{i \in \{1, 2, \ldots, n\} : u_i \neq v_i\}\big|$. We will usually abbreviate 'Hamming distance' to '*distance*'.

Example 8.2. Working with binary words of length 4 in $\{0, 1\}^4$, we have $d(0011, 1101) = 3$ because the words 0011 and 1101 differ in their first three positions, and are the same in their final position. Working with words over the alphabet $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$, we have $d(\mathtt{tale}, \mathtt{take}) = 1$ and $d(\mathtt{tale}, \mathtt{tilt}) = 2$.

Theorem 8.3. *Let $\mathcal{A}$ be an alphabet and let $u, v, w \in \mathcal{A}^n$.*

(a) *$d(u,v) = 0$ if and only if $u = v$;*
(b) *$d(u,v) = d(v,u)$;*
(c) *$d(u,w) \leq d(u,v) + d(v,w)$.*

Part (c) is called the *triangle inequality*. As an exercise, find all English words v such that

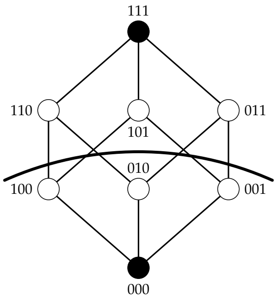
$$d(\texttt{warm}, v) = d(\texttt{cold}, v) = 2$$

and check that the triangle inequality holds when u, v, w are `warm`, `wall`, `cold`, respectively.

If you have seen metric spaces then you may have noticed that Theorem 8.3 says that $(\mathcal{A}^n, d)$ is a metric space.

Binary Symmetric Channel. In Exercise 1.9 Alice sent Bob a codeword $X \in \{000, 111\}$ across the Binary Symmetric Channel with error probability p, and Bob received $Y \in \{0,1\}^3$. We saw that $\mathbf{P}[Y = 111 | X = 000] = p^3$, $\mathbf{P}[Y = 110 | X = 000] = p^2(1-p)$, and so on.

In general, the power of p in $\mathbf{P}[Y = v | X = u]$ is the number of bits flipped by the channel. This is the Hamming distance $d(u,v)$. It is also the number of edges between u and v in the graph of $\{0,1\}^3$.



Lemma 8.4. *Suppose that $u \in \{0,1\}^n$ is sent through the* BSC(p). *The probability that $v \in \{0,1\}^n$ is received is $p^{d(u,v)}(1-p)^{n-d(u,v)}$.*

Theorem 8.5. *Suppose that we use a binary code C of length n to send messages through the* BSC(p) *with $p < 1/2$, and that each codeword in C is equally likely to be sent. Let X be the sent codeword and Y the received word. For each $u \in C$,*

$$\mathbf{P}[X = u | Y = v] = p^{d(u,v)}(1-p)^{n-d(u,v)}c(v).$$

where $c(v)$ does not depend on u. Hence $\mathbf{P}[X = u | Y = v]$ is maximized by choosing u to be the nearest codeword to v.

The assumption that every codeword is equally likely to be sent is vital to Theorem 8.5.

For instance suppose that, as in the introduction, Alice sends Bob a 'yes'/'no' using the repetition code 111, 000. We saw in Question 5 on Sheet 1 that if 000 has probability $\frac{1}{10}$ and 111 has probability $\frac{9}{10}$.

$$\frac{\mathbf{P}[X = 000 | Y = 000]}{\mathbf{P}[X = 111 | Y = 000]} = \frac{27}{77} \approx 0.351.$$

When 000 is received, it is almost three times as likely that 111 was sent as 000. Therefore, decoding by choosing u to maximize $\mathbf{P}[X = u | Y = v]$ Bob will always decode as 111. We conclude that this combination of code and channel is useless for communication.

The quantity $\mathbf{P}[X = u | Y = v]$ is the *likelihood* that we are correct in decoding a received word v by u. Here $Y = v$ is the event we observe, and $X = u$ is our inference; this is an example of *maximum likelihood decoding*. It is the right decoding strategy to use in Shannon's Noisy Coding Theorem.

We saw at the end of §1 that the binary string given by source coding a long text with a good code, such as a Huffman code, has each word in $\{0,1\}^m$ about the same number of times, provided m is not too large. Therefore if we encode the text by splitting this binary string into words of length m, the hypothesis for Theorem 8.5 holds. A code C with 2^m codewords is required.

Nearest neighbour decoding and Hamming balls.

Definition 8.6. Let $C \subseteq \mathcal{A}^n$ be a code. Suppose that a codeword is sent through the channel and we receive the word v. To decide v using *nearest neighbour decoding* look at all the codewords of C and pick the one that is nearest, in Hamming distance to v, choosing arbitrarily if there are several equally close.

Exercise 8.7. Take the code $C = \{00000, 11100, 00111, 11011\}$ from Question 4 on Problem Sheet 5. Show that the received words 00000, 01111 decode to 00000 and 00111 and that an arbitrary choice must be made to decode 01010.

Definition 8.8. Let $\mathcal{A}$ be an alphabet and let $u \in \mathcal{A}^n$. The *Hamming ball* of *radius* r about u is the set

$$B_r(u) = \{v \in \mathcal{A}^n : d(u,v) \leq r\}.$$

We saw in Exercise 8.7 that the Hamming balls of radius 1 about codewords are disjoint; therefore using this code, nearest neighbour decoding always finds the sent codeword when at most 1 error occurs. For example, the words in

$$B_1(11100) = \{11100, 01100, 10100, 11000, 11110, 11101\}$$

decode to 11100. See Question 4 on Sheet 5 for a related example and the formal definition of a 1-error correcting binary code.

Let $\mathbf{0}$ denote the all-zeros word $0 \ldots 0$. The table below shows the sizes of the Hamming balls of radius r about $0000 \in \mathbb{F}_2^4$. The second line shows the sizes of the shells: the words at distance exactly r.

r	0	1	2	3	4
$\lvert B_r(\mathbf{0})\rvert$	1	5	11	15	16
$\lvert B_r(\mathbf{0})\rvert - \lvert B_{r-1}(\mathbf{0})\rvert$	1	4	6	4	1

Lemma 8.9. *Let $n \in \mathbb{N}$.*

(a) $\big|\{v \in \{0,1\}^n : d(u,v) = s\}\big| = \binom{n}{s}$.

(b) $|B_r(\mathbf{0})| = \sum_{s=0}^{r} \binom{n}{s}$;

The size of a Hamming ball does not depend on its centre.

More on maximum likelihood decoding. Consider the Binary Erasure Channel. Since any received bit is correct, it is natural to modify nearest neighbour decoding with a binary code C so that a received word $v \in \{0, \star, 1\}^n$ is decoded as a closest codeword $u \in C$, *changing only the $\star$ bits in v.*

Example 8.10. As in Exercise 8.7 let $C = \{00000, 11100, 00111, 11011\}$. The received words $0000\star$, $001\star\star$, and $00\star\star\star$ decode as 00000, 00111 and either 00000 or 00111, depending on an arbitrary choice. We saw in Exercise 8.7 that the distance between every pair of codewords in this code is 3, therefore any two errors can be corrected.

Exercise 8.11. Show that this modification of nearest neighbour decoding implements maximum likelihood decoding. That is, the codeword $u \in C$ is chosen so that $\mathbf{P}[X = u | Y = v]$ is maximized.

9. Shannon's Noisy Coding Theorem for the BSC(p)

To prove Shannon's Noisy Coding Theorem for the Binary Symmetric Channel we need some good bounds on the sizes of Hamming balls. The size does not depend on the centre, so we choose the all-zeros word $\mathbf{0}$.

Proposition 9.1. *Let $n \in \mathbb{N}$ and let $0 \leq r \leq n/2$. Let $h = H(\frac{r}{n}, 1 - \frac{r}{n})$. Then*

$$\frac{1}{n+1} 2^{hn} \leq \binom{n}{r} \leq |B_r(\mathbf{0})| \leq 2^{hn}.$$

Proof. Set $p = r/n$. Let H be the number of heads when a coin biased to lands heads with probability p is flipped n times. Thus $H \sim \mathrm{Bin}(n, p)$ and $\mathbf{P}[H = s] = \binom{n}{s} p^s (1-p)^{n-s}$. (Some simulations are shown below.) Observe that $\mathbf{P}[H = s]$ is maximized when $s = \mathbf{E}[H] = pn = r$. This explains the appearance of the entropy function:

$$\begin{aligned} p^r(1-p)^{n-r} =& 2^{\log_2(p^r(1-p)^{n-r})} = 2^{r \log_2 p + (n-r) \log_2 (1-p)} \\ &= 2^{(p \log_2 p + (1-p)\log_2(1-p))n} = 2^{-H(p, 1-p)n} = 2^{-hn}. \end{aligned}$$

For the lower bound we argue that since H takes $n+1$ different values, with its maximum at r, we have $\mathbf{P}[H = r] \geq 1/(n+1)$. Hence

$$\binom{n}{r} p^r (1-p)^{n-r} \geq \frac{1}{n+1}$$

and so

$$\binom{n}{r} \leq \frac{1}{n+1} \frac{1}{p^r(1-p)^{n-r}} = \frac{2^{hn}}{n+1}.$$

For the upper bound, we use the inequalities seen in the proof of Theorem 8.5: $(1-p)^n > p(1-p)^{n-1} > \ldots > p^s(1-p)^{n-s} > p^{s+1}(1-$

$p)^{n-(s+1)} > \ldots$ to get

$$1 \geq \sum_{s=0}^{r} \binom{n}{s} p^s (1-p)^{n-s} \geq \sum_{s=0}^{r} \binom{n}{s} p^r (1-p)^{n-r}.$$

Hence, using Lemma 8.9,

$$|B_{\mathbf{0}}(r)| = \sum_{s=0}^{r} \binom{n}{s} \leq \frac{1}{p^r(1-p)^{n-r}} = 2^{hn}$$

as required. [In the lecture I got lost at this step, but it follows immediately from the previous line.] □

Another tool we need is linearity of expectation. This is stated formally in Lemma 1.19. For two real valued random variables X and Y it says $\mathbf{E}[X+Y] = \mathbf{E}[X] + \mathbf{E}[Y]$.

Exercise 9.2.

(a) Let X, Y be independent rolls of a fair die. Let $Z = X$. Find $\mathbf{E}[X]$, $\mathbf{E}[X+Y]$, $\mathbf{E}[X+Z]$, $\mathbf{E}[X+Y+Z]$. [*Hint:* the hard way to compute $\mathbf{E}[X+Y]$ is to use its probability distribution on $\{2,\ldots,12\}$, namely $(\frac{1}{36}, \frac{2}{36}, \ldots, \frac{6}{36}, \ldots, \frac{2}{36}, \frac{1}{36})$. The easy way is to use linearity of expectation.]

(b) Let F be the flip of a coin biased to lands heads with probability p and let

$$X = \begin{cases} 1 & \text{if } F = \text{heads} \\ 0 & \text{if } F = \text{tails.} \end{cases}$$

Then

$$\begin{aligned} \mathbf{E}[X] &= 1 \times \mathbf{P}[F = \text{heads}] + 0 \times \mathbf{P}[F = \text{tails}] \\ &= 1 \times p + 0 \times (1-p) \\ &= p. \end{aligned}$$

Thus the expectation of an 'indicator' random variable such as X is the probability of the event defining it. We use this to find $\mathbf{E}[g_i(v)]$ in (2) in the proof below.

(c) Suppose that 4 boys and 8 girls sit in a circle, choosing seats at random. On average, how many girls have a boy to their right? *Outline solution.* Number chairs from 0 to 11. Define

$$X_i = \begin{cases} 1 & \text{if chair } i \text{ has a girl and chair } i+1 \text{ (mod 12) has a boy} \\ 0 & \text{otherwise.} \end{cases}.$$

Show, using the idea in (b) that $\mathbf{E}[X_i] = \frac{8}{12} \times \frac{4}{11}$ and hence that the expected number of GB pairs is $12 \times \frac{8}{12} \times \frac{4}{11} = \frac{32}{11}$. Find the expected number of GG, BG and BB pairs similarly.

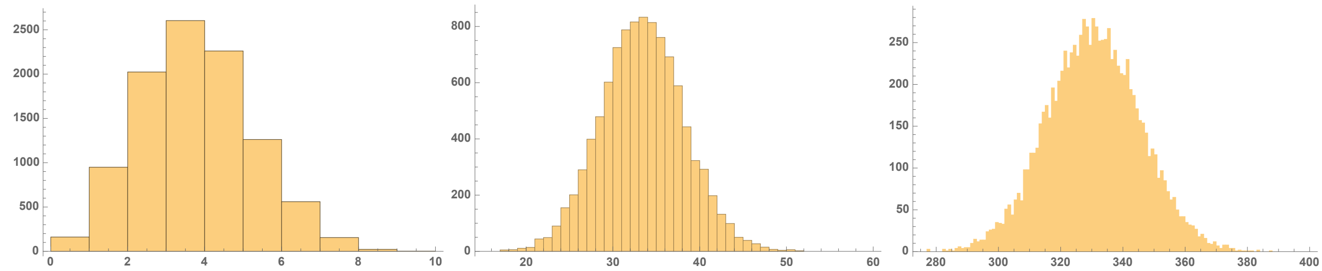
To reduce the technicalities in the proof, we simplify the Binary Symmetric Channel as follows.

Definition 9.3. Given $0 < p < 1/2$ and $n \in \mathbb{N}$ such that $pn \in \mathbb{N}$, the Toy $\mathrm{BSC}(p,n)$ is the channel with input and output alphabets $\{0,1\}^n$ such that when $u \in \{0,1\}^n$ is sent, exactly pn of the positions of u flip.

As motivation, observe that if F is the number of flips then $F \sim \mathrm{Bin}(n,p)$ and so $\mathbf{Var}\ F = np(1-p)$. By Chebychev's Inequality (see Problem Sheet 1, Question 5), we have

$$\mathbf{P}\big[|F-pn| > \epsilon n\big] \leq \frac{np(1-p)}{n^2\epsilon^2} = \frac{p(1-p)}{\epsilon^2}\frac{1}{n} \to 0 \text{ as } n \to \infty \qquad (\dagger)$$

for any $\epsilon > 0$. Therefore the number of flips is concentrated around pn, and the channel behaves like the Toy $\mathrm{BSC}(p,n)$. The histograms below show the number of flips for $p = \frac{1}{3}$ simulating by 10000 samples the number of flips when $n = 10$, 100 and 1000 bits are sent.



How small must ϵ be for the event $|F - \frac{1}{3}n| > \epsilon n$ never to occur in the simulation? When $n = 10$, every number occurred, so the least ϵ is $\frac{2}{3}$; when $n = 100$, every sample was within 20 of the mean, so we can take $\epsilon = \frac{20}{100} = 0.2$; when $n = 1000$, we can take $\epsilon = \frac{60}{1000} = 0.06$. This 'concentration of measure' is also predicted by the Weak Law of Large Numbers or the Central Limit Theorem.

Lemma 9.4. *Let c_n be the capacity of the Toy* $\mathrm{BSC}(p,n)$. *We have*

$$\frac{c_n}{n} \to 1 - H(p, 1-p)$$

as $n \to \infty$.

Proof. See Question 5 on Problem Sheet 6 and the model answers. □

Since n is part of the definition of the Toy BSC, the statement of Theorem 7.14 changes slightly.

Proposition 9.5. *Let $h = H(p, 1-p)$. Let $r < 1 - h$. Let $\epsilon > 0$ be given. Provided n is sufficiently large, there exists a binary code C of size $\geq 2^{rn}$ such that when C is used to communicate on the Toy* $\mathrm{BSC}(p,n)$ *using nearest neighbour decoding, the error probability is $< \epsilon$.*

Shannon's great insight was that choosing the code at random is likely to work. To go from this to a *particular* code that works, we need this basic lemma. In words it says 'not everyone can be strictly above average'. I hope this convinces you it's true: if not, see Problem Sheet 7. Applications of this lemma are sometimes called the 'First Moment Method'.

Lemma 9.6. *If $X : \Omega \to \mathbb{R}$ is a random variable then there is an outcome $\omega \in \Omega$ such that $X(\omega) \leq \mathbf{E}[X]$.* □

For technical reasons we must choose twice as many codewords as are eventually required. Set $M = 2\lceil 2^{rn} \rceil$ and let $U(1), \ldots, U(M) \in \{0,1\}^n$ be codewords, chosen independently and uniformly at random. We now have two probability spaces: to distinguish them we write

- $\mathbf{P}_{\text{ch}}$ for probabilities computed using the channel, such as $\mathbf{P}[Y = v | X = u]$, the probability that when u is sent, v is received;
- $\mathbf{P}_{\text{code}}$ and $\mathbf{E}_{\text{code}}$ for probabilities and expectations depending on the random choice of code; for instance, $P[U(1) = u] = \frac{1}{2^n}$ for each $u \in \{0,1\}^n$, since codewords are chosen uniformly at random.

We are now ready for the proof.

Proof of Proposition 9.5. Let P_i be the probability that, given $X = U(i)$ is sent, the received word Y is not decoded as $U(i)$ using nearest neighbour decoding.

Step 1: Upper bound on P_i. Conditioning on Y we have

$$P_i = \sum_{v \in \{0,1\}^n} \mathbf{P}[Y = v | X = U(i)] \begin{cases} 1 & \text{if } v \text{ is not decoded as } U(i) \\ 0 & \text{otherwise.} \end{cases}$$

By definition of the Toy BSC, $d(U(i), v) = pn$. If there are no codewords $U(j) \in B_{pn}(v)$ then v is decoded as $U(i)$. If $U(j) \in B_{pn}(v)$ then either $d(U(j), v) < pn$ and $U(j)$ is preferred to $U(i)$ in nearest neighbour decoding, or $d(U(j), v) = pn$ and, assuming the worst case, $U(j)$ may be chosen in preference to $U(i)$. Hence for each $v \in \{0,1\}^n$,

$$\begin{cases} 1 & \text{if } v \text{ is not decoded as } U(i) \\ 0 & \text{otherwise} \end{cases} \leq \big| \{ j : j \neq i, U(j) \in B_{pn}(v) \} \big|.$$

Set $g_i(v) = \big| \{ j : j \neq i, U(j) \in B_{pn}(v) \} \big|$. We have shown that

$$P_i \leq \sum_{v \in \{0,1\}^n} \mathbf{P}[Y = v | X = U(i)] g_i(v).$$

Step 2*: Expectations.* We now find the expectation of P_i *in the probability space of the random code* of P_i. Note that $g_i(v)$ depends only on the $U(j)$ for $j \neq i$, whereas $\mathbf{P}[Y = v|X = U(i)]$ depends only on $U(i)$. Therefore $\mathbf{P}[Y = v|X = U(i)]$ and $g_i(v)$ are independent random variables. Hence

$$\mathbf{E}_{\text{code}}\big[\mathbf{P}[Y = v|X = U(i)]g_i(v)\big] = \mathbf{E}_{\text{code}}\big[\mathbf{P}[Y = v|X = U(i)]\big]\mathbf{E}_{\text{code}}[g_i(v)].$$

Since $U(i)$ is uniformly distributed on $\{0,1\}^n$, so is the received word Y when $U(i)$ is sent. Hence $\mathbf{E}_{\text{code}}\big[\mathbf{P}[Y = v|X = U(i)]\big] = 1/2^n$. (For full details see Question 1 on Problem Sheet 7.) We may write

$$g_i(v) = \sum_{j \neq i} \begin{cases} 1 & \text{if } U(j) \in B_{pn}(v) \\ 0 & \text{otherwise.} \end{cases}$$

By the same argument as Exercise 9.2(b) and linearity of expectation we have

$$\mathbf{E}_{\text{code}}[g_i(v)] = \sum_{j \neq i} \mathbf{P}[U(j) \in B_{pn}(v)].$$

By Lemma 8.9, $|B_{pn}(v)| \leq 2^{hn}$. Since $U(j)$ is distributed uniformly at random on $\{0,1\}^n$, it follows that $\mathbf{P}[U(j) \in B_{pn}(v)] \leq 2^{hn}/2^n$. Since $|C| = M$,

$$\mathbf{E}[g_i(v)] \leq \frac{(M-1)2^{hp}}{2^n} \leq \frac{2\lceil 2^{rn}\rceil 2^{hn}}{2^n}.$$

Since $2\lceil 2^{rn}\rceil \leq 2(2^{rn}+1) = 2^{rn+1} + 2 \leq 2^{rn+2}$ **[details added]** (this bound is very crude) we have

$$\mathbf{E}[g_i(v)] \leq 2^{rn+2+hn-n} = 2^{(r+h-1+\frac{2}{n})n} = 2^{(-\eta+\frac{2}{n})n}$$

where $\eta = 1 - r - h$. [**Typo** $1 - r + h$ **corrected to** $1 - r - h$] By our choice of r, we have $\eta > 0$ and so $-\eta + \frac{2}{n} < -\eta/2$ for n sufficiently large and $\mathbf{E}[g_i(v)] \leq 2^{-\eta n/2}$. Therefore

$$\mathbf{E}_{\text{code}}\big[\mathbf{P}[Y = v|X = U(i)]\big]\mathbf{E}_{\text{code}}[g_i(v)] \leq 2^{-n}2^{-\eta n/2}.$$

Summing over all $v \in \{0,1\}^n$ we get $\mathbf{E}_{\text{code}}[P_i] \leq 2^{-\eta n/2}$.

Step 3*: We pick a code.* Let $P = \frac{1}{M}\sum_{i=1}^{M} P_i$. By linearity of expectation and the bound just proved, [**Typo:** m **in upper bound below corrected to** M]

$$\mathbf{E}_{\text{code}}[P] = \frac{1}{M}\sum_{i=1}^{M} \mathbf{E}[P_i] \leq \frac{1}{M}\sum_{i=1}^{M} 2^{-\eta n/2} = 2^{-\eta n/2}.$$

Hence there exists a *particular code* $C^\star$ such that, for this code, $\frac{1}{M}\sum_{i=1}^{m} P_i \leq 2^{-\eta n/2}$.

Step 4: We use $C^\star$ to find a suitable code. Recall that $M = 2\lceil 2^{rn}\rceil$ and so $M/2 \geq 2^{rn}$. We may relabel the codewords in $C^\star$ so that $P_1 \leq \ldots \leq P_{M/2} \leq P_{M/2+1} \cdots \leq P_M$. By our choice of $C^\star$,

$$M2^{-\eta n/2} \geq P_1 + \cdots + P_{M/2} + P_{M/2+1} + \cdots + P_M \geq \frac{M}{2} P_{M/2}.$$

Cancelling M, we get $P_{M/2} \leq 2 \times 2^{-\eta n/2}$ and so

$$P_1 \leq \ldots \leq P_{M/2} \leq 2 \times 2^{-\eta n/2}.$$

The right-hand side tends to 0 as $n \to \infty$. Therefore when n is sufficiently large, $P_1, \ldots, P_{M/2} < \epsilon$. The code $\{u(1), \ldots, u(M/2)\}$ therefore is as required. □

This proof will take a lot of thinking about.

- Question 1 on Problem Sheet 7 asks you to fill in the details in the argument at the start of Step 2. This should clarify the role played by the two different probability spaces.
- Question 2 then asks you to adapt the proof to the Toy Binary Erasure Channel, in which exactly pn bits are erased.

Extra: Shannon's Noisy Coding Theorem for the normal Binary Symmetric Channel. The proof above for the Toy BSC can be adapted, but there are several technical points that have to be addressed. See Question 6 on Problem Sheet 6 and the model answers for a detailed proof.

In outline: using (†) we may choose $n^\star$ so that if F is the number of errors in the channel when a binary word of length $n^\star$ is sent, then $\mathbf{P}\big[|F - pn^\star| > \epsilon n\big] < \epsilon/2$. Thus in the typical case the BSC behaves very like the Toy BSC. In the atypical case, we make the worst case assumption that nearest neighbour decoding *never* succeeds. This increases the upper bound for each P_i by $\epsilon/2$, but still by taking n sufficient large (with $n \geq n^\star$), each step in the proof can be adapted.

10. Converse in Shannon's Noisy Coding Theorem

The proof depends on two inequalities, both of interests in their own right.

- The *Data-Processing Inequality* states that if X, Y are random variables, taking values in sets $\mathcal{X}$ and $\mathcal{Y}$, and $d : \mathcal{Y} \to \mathcal{Z}$ is a function, then $I(X;Y) \geq I(X;d(Y))$. In words: processing Y by the function d cannot increase the amount of information Y has about X.
- *Fano's Inequality* states that if X and Y are random variables taking values in a set of size M, and $\mathbf{P}[X = Y] \geq 1 - \epsilon$ then $H(X|Y) \leq H(\epsilon, 1-\epsilon) + \epsilon \log_2(M-1)$.

We shall see that the Data-Processing Inequality easily reduces to the following lemma. For a characterisation of when equality holds, see the optional question on Problem Sheet 7.

Lemma 10.1. *Let X, Y and Z be random variables. Then*

$$H\big(X|(Y,Z)\big) \le H(X|Z).$$

Proof. Let X, Y, Z take values in $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, respectively. Fix $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$. Consider the two probability measures on $\mathcal{X}$ defined by $p_x = \mathbf{P}[X = x | Y = y, Z = z]$ and $q_x = \mathbf{P}[X = x|Z = z]$. By Gibbs' Inequality (Lemma 3.9) we have

$$H(X|Y = y, Z = z) = -\sum_{x\in\mathcal{X}} p_x \log_2 p_x \le -\sum_{x\in\mathcal{X}} p_x \log_2 q_x.$$

Multiplying by $\mathbf{P}[Y = y, Z = z]$ we get

$$\begin{aligned}&\mathbf{P}[Y = y, Z = z] H(X|Y = y, Z = z)\\ &\quad \le -\sum_{x\in\mathcal{X}} \mathbf{P}[Y = y, Z = z]\mathbf{P}[X = x|Y = y, Z = z] \log_2 \mathbf{P}[X = x|Z = z].\end{aligned}$$

We now let y and z vary and take the sum of both sides over all $y \in \mathcal{Y}$, $z \in \mathcal{Z}$. On the left-hand side we get, by Definition 7.4, the conditional entropy $H(X|Y,Z)$. Using $\mathbf{P}[Y = y, Z = z]\mathbf{P}[X = x|Y = y, Z = z] = \mathbf{P}[X = x, Y = y, Z = z]$ to simplify the right-hand side we get

$$\begin{aligned}H(X|Y,Z) &\le -\sum_{x\in\mathcal{X}}\sum_{z\in\mathcal{Z}}\Big(\sum_{y\in\mathcal{Y}} \mathbf{P}[X = x, Y = y, Z = z]\Big) \log_2 \mathbf{P}[X = x|Z = z]\\ &= -\sum_{x\in\mathcal{X}}\sum_{z\in\mathcal{Z}} \mathbf{P}[X = x, Z = z] \log_2 \mathbf{P}[X = x|Z = z]\\ &= \sum_{x\in\mathcal{X}}\sum_{z\in\mathcal{Z}} \mathbf{P}[Z = z]\mathbf{P}[X = x|Z = z] \log_2 \mathbf{P}[X = x|Z = z]\\ &= H(X|Z)\end{aligned}$$

as required. □

Lemma 10.2 (Data-Processing Inequality)**.** *If X, Y are random variables, taking values in sets $\mathcal{X}$ and $\mathcal{Y}$ respectively, and $d : \mathcal{Y} \to \mathcal{Z}$ is a function, then $I(X;Y) \ge I\big(X; d(Y)\big)$.*

Proof. Since Y determines $d(Y)$, we have $I(X;Y) = I(X;(Y,d(Y)))$. By Lemma 10.1 taking $Z = d(Y)$ we get

$$\begin{aligned}I(X;Y) = I\big(X;(Y,d(Y)\big) &= H(X) - H(X|(Y,d(Y))\big)\\ &\ge H(X) - H\big(X|d(Y)\big) = I\big(X;d(Y)\big)\end{aligned}$$

as required. □

To motivate the proof of Fano's Inequality consider the following example.

Example 10.3. Alice and Bob may go to the cinema, theatre or stay at home, each with equal probability. With probability $1 - p$ where we imagine p is small, their decisions X and Z agree. In the 'error' case they differ. A nice way to find the joint entropy (X, Z) is to condition on the event that $X \neq Z$. For this we introduce the 'indicator' random variable, as in Example 9.2(b),

$$F = \begin{cases} 1 & \text{if } X \neq Z \\ 0 & \text{if } X = Z. \end{cases}$$

Since F is determined by (X, Z) we have

$$H(X, Z) = H(X, Z, F) = H(X, Z|F) + H(F)$$

using the Chaining Rule (Lemma 7.6) for the second equality. Now $H(F) = H(p, 1 - p)$,

$$H(X, Z|F = 0) = H(X, X) = H(X) = \log_2 3$$

and $H(X, Z|F = 1) = \log_2 6 = 1 + \log_2 3$ since there are 6 equally likely pairs of destinations when $X \neq Z$. Therefore $H(X, Z|F) = (1 - p)\log_2 3 + p(1 + \log_2 3) = 1 - p$ and

$$H(X, Z) = p + \log_2 3 + H(p, 1 - p).$$

To prove Fano's Inequality we need that if X is a random variable taking m different values then $H(X) \leq \log_2 m$. By Question 6 on Problem Sheet 3, this follows easily from Gibbs' Inequality.

Lemma 10.4 (Fano's Inequality)**.** *Let X and Z be random variables taking values in a set of size M. Let $\epsilon < \frac{1}{2}$. If $\mathbf{P}[X = Z] \geq 1 - \epsilon$ then $H(X|Z) \leq H(\epsilon, 1 - \epsilon) + \epsilon \log_2(M - 1)$.*

Proof. We can suppose that X and Z take values in $\{1, \ldots, M\}$. Let F be the indicator random variable for the event $X \neq Z$, as defined in Example 10.3. Conditioning on (Z, F) we get

$$\begin{aligned} H\big(X|(Z, F)\big) = &\sum_{z=1}^{m} \mathbf{P}[Z = z, F = 0] H(X|Z = z, F = 0) \\ &+ \sum_{z=1}^{m} \mathbf{P}[Z = z, F = 1] H(X|Z = z, F = 1). \end{aligned}$$

Now $H(X|Z = z, F = 0) = H(Z|Z = z, F = 0) = 0$ and $H(X|Z = z, F = 1) \leq \log_2(M - 1)$ since X then takes at most $M - 1$ different values. Hence

$$\begin{aligned} H\big(X|(Z, F)\big) &\leq \log_2(M - 1) \sum_{z=1}^{m} \mathbf{P}[Z = z, F = 1] \\ &= \log_2(M - 1) \sum_{z=1}^{m} \mathbf{P}[Z = z, Z \neq X] \\ &= \log_2(M - 1)\mathbf{P}[X \neq Z] \end{aligned}$$

$$\le \epsilon \log_2(M-1).$$

By the Chaining Rule we have

$$H\big(X|(Z,F)\big) + H(Z,F) = H(X,Z,F) = H(X,Z) = H(X|Z) + H(Z).$$

Hence subtracting $H(Z)$ from each side, we have

$$\begin{aligned} H(X|Z) &= H\big(X|(Z,F)\big) + H(Z,F) - H(Z) \\ &= H\big(X|(Z,F)\big) + H(F|Z) \\ &\le H\big(X|(Z,F)\big) + H(F) \end{aligned}$$

where the final equality uses $H(F|Z) \le H(F)$, as seen on Question 4 of Problem Sheet 4. Since $\mathbf{P}[F=1] \le \epsilon$ and the entropy function $H(\epsilon, 1-\epsilon)$ is increasing for $\epsilon < \frac{1}{2}$ (see the graph on page 19) we get

$$H(X|Z) \le \epsilon \log_2(M-1) + H(\epsilon, 1-\epsilon)$$

as required. □

The final result we need to prove the converse in Shannon's Noisy Coding Theorem is Question 4 on Problem Sheet 7: when a memoryless channel of capacity c is used to send words of n symbols, its capacity is nc. This should be quite intuitive: since each symbol is sent independently, the amount of information about the input we can (at best) learn from each of the n received symbols is the original capacity c.

Proof of Theorem 7.14(b). Let $r > c$. Suppose, for a contradiction, that for all $\epsilon > 0$, and n sufficiently large, there is a code $C^{(n)} \subseteq \mathcal{A}^n$ with $|C| \ge 2^{rn}$ and a decoding rule $d : \mathcal{B}^n \to C^{(n)}$ such that when $C^{(n)}$ is used to communicate on the channel, the decoding error probability is $< \epsilon$.

Take $\epsilon = \frac{1}{2}(1 - c/r)$ and let $C^{(n)}$ be a code as above. Let X be the sent codeword, chosen uniformly at random from $C^{(n)}$, and let $Y \in \mathcal{B}^n$ be the received word. We have

$$\begin{aligned} nc &\ge I(X;Y) && \text{by Question 4 on Sheet 7} \\ &\ge I\big(X; d(Y)\big) && \text{by Data-Processing Inequality} \\ &= h(X) - h\big(X|d(Y)\big) \\ &\ge \log_2 M - \big(H(\epsilon, 1-\epsilon) + \epsilon \log_2(M-1)\big) && \text{by Fano's Inequality} \\ &\ge \log_2 M - 1 - \epsilon \log_2 M. \end{aligned}$$

Hence

$$\epsilon \ge \frac{\log_2 M - nc - 1}{\log_2 M} \ge 1 - \frac{nc-1}{nr} = 1 - \frac{c}{r} - \frac{1}{nr}.$$

Subtract $\epsilon = \frac{1}{2}(1 - c/r)$ from both sides to get $0 \ge \frac{1}{2}(1 - c/r) - \frac{1}{nr}$, or equivalently, $\frac{1}{nr} \ge \frac{1}{2}(1 - c/r)$. This is a contradiction for any n sufficiently large, since $\frac{1}{nr} \to 0$ as $n \to \infty$.

□

As a nice example of working with channel probabilities, we prove Shannon's Noisy Coding Theorem (b) directly for the lazy typist channel.

Example 10.5. Take the lazy typist channel on $2t$ symbol and use it send words of length n from $\{0,1,\ldots,2t-1\}^n$. (This is the n-extension of the channel, as in Question 4 on Sheet 7.) By Question 3 on Sheet 5, the capacity of the lazy typist channel is $\log_2 t$.

To make things more concrete, take $\epsilon = \frac{1}{10}$. Shannon's Noisy Coding Theorem (b) then says that if $r > \log_2 t$, when n is large, it is impossible to find a code $C \subseteq \{0,1,\ldots,2t-1\}^n$ such that $|C| \geq 2^{nr}$, and the error probability for each codeword is $< \frac{1}{10}$.

Suppose that C is such a code. For each $v \in \{0,1,\ldots,2t-1\}^n$ let

$$B(v) = \{u \in \{0,1,\ldots,2t-1\}^n : u_i = v_i \text{ or } u_i = v_i - 1 \bmod 2t \text{ for all } i\}.$$

This is the set of sent words that may be received as v. Let $M(v) = |B(v) \cap C|$. When v is received, the decoder must choose arbitrarily between the $M(v)$ equally likely codewords in $B(v)$. (Remember that each codeword is equally likely to be sent.) So the probability of decoding incorrectly is $1 - 1/M(v)$. Let P be the average probability of incorrect decoding; by assumption $P < \frac{1}{10}$. We have

$$P = \sum_{v \in \{0,1,\ldots,2s-1\}^n} \mathbf{P}[\text{decode wrongly} | Y = v]\mathbf{P}[Y = v]. \tag{†}$$

By the usual conditioning argument

$$\begin{aligned}\mathbf{P}[Y = v] &= \sum_{u \in C} \mathbf{P}[X = u]\mathbf{P}[Y = v | X = u] \\ &= \sum_{u \in C} \mathbf{P}[X = u] \begin{cases} \frac{1}{2^n} & \text{if } u \in B(v) \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{1}{|C|}\frac{M(v)}{2^n}.\end{aligned}$$

Let V be the set of words such that $M(v) \geq 1$. (By the previous equation, these are the words that may be received.) By (†),

$$P = \sum_{v \in V} \big(1 - \frac{1}{M(v)}\big) \frac{1}{|C|} \frac{M(v)}{2^n} = \frac{1}{|C|2^n} \sum_{v \in V} M(v) - \frac{|V|}{|C|2^n}.$$

Now $\sum_{v \in V} M(v) = 2^n|C|$, since each codeword $u \in C$ is counted 2^n times, once for each v it may be received as. Hence

$$P = 1 - \frac{|V|}{|C|2^n}.$$

Since $V \subseteq \{0,1,\ldots,2t-1\}^n$, we have $|V| \leq (2t)^n$. Therefore $\frac{1}{10} > P \geq 1 - (2t)^n/|C|2^n = 1 - t^n/|C|$. Hence $\frac{9}{10} < t^n/|C|$ and so $|C| < \frac{10}{9}t^n$. But by hypothesis $2^{rn} < |C|$. Taking logs we get $rn < \log_2 \frac{10}{9} + n \log_2 t$ and so

$$r < \frac{\log_2 \frac{10}{9}}{n} + \log_2 t.$$

For large n this contradicts our assumption that $r > \log_2 t$.

(C) Ergodic sources and the Asymptotic Equipartition Property (AEP)

11. Typical words from memoryless sources

Question. What is a typical word from a source?

In the final part of the course we make sense of this question and use the answer to give a new proof of Shannon's Source Coding Theorem and, in outline only, our first proof of the constructive part of Shannon's Noisy Coding Theorem in full generality. We end by considering some practical solutions to the problems of source and channel coding.

It is hard to give an example of a 'typical word'. It is a bit like asking 'what is a typical pine tree?': the trees that stand out are all, for one reason or another, atypical. The best answer is probably: 'go into a pine wood and choose one at random — then it's probably fairly typical'.

For memoryless sources (see Definition 3.1) there is a more precise, but still probabilistic answer. The (somewhat leading) questions in the following exercise should give it some motivation.

Exercise 11.1. Let $p < \frac{1}{2}$. A memoryless source emits 0 with probability $1 - p$ and 1 with probability p. Let $n \in \mathbb{N}$.

(i) What is the most common message of length n? Can one reasonably say it is typical?
(ii) How many 1s are there in a typical word of length r?
(iii) What is the probability of each word of length r with the average number of 0s and 1s? (Suppose that $pr \in \mathbb{N}$.)
(iv) How is this related to the entropy of the source?
(v) What does Shannon's Source Coding Theorem have to say about efficiently coding messages from this source?
(vi) In what sense are words with about pr 1s typical?

A good answer to (vi) is given by the Weak Law of Large Numbers. Despite its name, it is very powerful and useful! It can be proved using Chebyshev's Inequality: see Question 6(b) on Problem Sheet 1.

Proposition 11.2 (Weak Law of Large Numbers). *Let $X_1, \ldots, X_r$ be independent real-valued random variables each with expectation μ and variance σ^2. Then*

$$\mathbf{P}\left[\mu - \epsilon < \frac{X_1 + \cdots + X_r}{r} < \mu + \epsilon\right] \to 1 \quad \textit{as } r \to \infty.$$

For instance, if S_t is the symbol emitted by the source in Exercise 11.1 at time t then $\mathbf{P}[S_t = 1] = p$ and $\mathbf{P}[S_t = 0] = 1 - p$ for all t. We have $\mathbf{E}[S_t] = p$ and $\mathbf{Var}\ S_t = \mathbf{E}[X_t^2] - \mathbf{E}[S_t]^2 = p - p^2 = p(1-p)$ for all t. By the Weak Law of Large Numbers, the probability that a random word $S_1 \ldots S_n$ has significantly more than (or significantly less than) pn 1s tends to 0 as $n \to \infty$.

Exercise 11.1(iv) suggests that the random variable $\log_2 \mathbf{P}[S_1 \ldots S_n]$ is of interest, where $S_1 \ldots S_n$ is a random word of length n emitted by a source. Note that a probability appears 'inside' the random variable: this is not so unusual in information theory, but rarely seen in other fields using probability.

Example 11.3. Take the source from Example 11.1.

(a) The random variable $\log_2 \mathbf{P}[S_1]$ takes value $\log_2 p$ with probability p and $\log_2(1-p)$ with probability $1-p$. (Since $p < \frac{1}{2}$, these values are distinct.)

(b) The random variable $\log_2 \mathbf{P}[S_1 S_2 S_3]$ takes distinct values

$$\log_2(1-p)^3,\ \log_2 p(1-p)^2,\ \log_2(1-p)p^2,\ \log_2 p^3$$

with probabilities $(1-p)^3, 3p(1-p)^2, 3p(1-p)p^2, p^3$, respectively. *Exercise:* What is its expectation?

The next exercise generalizes Example 11.3.

Exercise 11.4. Let $S_1, S_2, \ldots$ be a memoryless source and let $h = H(S_1) = H(S_2) = \ldots$.

(a) Express $\mathbf{E}\big[\log_2 \mathbf{P}[S_1]\big]$ in terms of h.

(b) What is $\mathbf{E}\big[\log_2 \mathbf{P}[S_1 \ldots S_n]\big]$ in terms of h?

Lemma 11.5. *Let $S_1, S_2, \ldots$ be the output of a memoryless source producing symbols in an alphabet $\mathcal{S}$. Let $h = H(S_1)$ be the per-symbol entropy. Given $\epsilon > 0$, there exists $r \in \mathbb{N}$ and a subset $\mathcal{T}^{(r)}$ of $\mathcal{A}^r$ such that*

(i) $\mathbf{P}[S_1 \ldots S_r \in \mathcal{T}^{(r)}] > 1 - \epsilon$;

(ii) $2^{-r(h+\epsilon)} \le \mathbf{P}[s_1 \ldots s_r] \le 2^{-r(h-\epsilon)}$ *for all words* $s_1 \ldots s_r \in \mathcal{T}^{(n)}$.

Since each word $s_1 \ldots s_r$ in the 'typical set' $\mathcal{T}^{(r)} \subseteq \mathcal{S}^r$ from Lemma 11.5 has probability at least $2^{-r(h+\epsilon)}$, there are at most $2^{r(h+\epsilon)}$ such words. Therefore we can encode them by an injective encoder using $\lceil r(h+\epsilon) \rceil$ bits. (We need the ceiling only to make sure this is an integer.)

This is almost exactly what the asymptotic version of Shannon's Source Coding Theorem, Theorem 4.7(i), requires! You are asked to prove it using Lemma 11.5 on Problem Sheet 8.

12. The Asymptotic Equipartition Property

In this section we take the conclusion of Lemma 11.5 as our main *definition*.

Definition 12.1. Let $S_1, S_2, \ldots$ be the symbols in an alphabet $\mathcal{S}$ output by a source. We say the source satisfies the *Asymptotic Equipartition Property (AEP)* if there exists $h \geq 0$ such that for all $\epsilon > 0$ there exists $r \in \mathbb{N}$ and a subset $\mathcal{T}^{(r)}$ of $\mathcal{A}^r$ such that

(i) $\mathbf{P}[S_1 \ldots S_r \in \mathcal{T}^{(r)}] > 1 - \epsilon$;
(ii) $2^{-r(h+\epsilon)} \leq \mathbf{P}[S_1 \ldots S_r = s_1 \ldots s_r] \leq 2^{-r(h-\epsilon)}$ for all $s_1 \ldots s_r \in \mathcal{T}^{(n)}$.

By Lemma 11.5, a memoryless source $S_1, S_2, \ldots$ satisfies the AEP, with $h = H(S_1)$. In the remainder of this section we prove some corollaries of this result.

Shannon's Source Coding Theorem and the AEP. As a warm up, we prove a special case of the constructive part of Shannon's Source Coding Theorem, Theorem 4.7(i). See Problem Sheet 8 for the general version: this special case should be helpful.

Example 12.2. Let $S_1, S_2, \ldots$ be a memoryless source emitting the bits 0 and 1 each with probability $1 - p$ and p. Let

$$h = H(p, 1-p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

Then $H(S_1) = H(S_2) = \ldots = h$. Fix $\epsilon > 0$, to be chosen by the end of the proof. By the AEP, there exists a subset $\mathcal{T}$ of $\{0,1\}^n$ such that

$$\mathbf{P}[S_1 \ldots S_r \in \mathcal{T}] > 1 - \epsilon$$

and

$$2^{-n(h+\epsilon)} \leq \mathbf{P}[S_1 \ldots S_r = s_1 \ldots s_r] \leq 2^{-n(h-\epsilon)}$$

for all $s_1 \ldots s_r \in \mathcal{T}$. By the lower bound above,

$$\sum_{s_1 \ldots s_r \in \mathcal{T}} \mathbf{P}[S_1 \ldots S_r = s_1 \ldots s_r] \geq |\mathcal{T}| 2^{-n(h+\epsilon)}.$$

Therefore $|\mathcal{T}| \leq 2^{r(h+\epsilon)}$. We can therefore encode all the typical words from the source using the binary words of a fixed length $\geq r(h + \epsilon)$.

To turn this idea into a well-defined injective encoder $f^{(r)}$, let $m = 1 + \lceil r(h + \epsilon) \rceil$ and let $u(0), \ldots, u(M-1)$ be a list of all the words in $\mathcal{T}$. Note that $M \leq 2^{m-1}$, and so the binary form of each $j < M$ has at most $m - 1$ bits. We define $f^{(r)}$ as follows:

- if $s_1 \ldots s_r \in \mathcal{T}$, with $s_1 \ldots s_r = u(j)$, then

$$f^{(r)}(s_1 \ldots s_r) = 1 j_0 \ldots j_{m-2} \in \{0,1\}^m$$

where $j_0 \ldots j_{m-2}$ is the $(m-1)$-bit binary form of j.
- if $s_1 \ldots s_r \notin \mathcal{T}$, then

$$f^{(r)}(s_1 \ldots s_r) = 0 s_1 \ldots s_r \in \{0,1\}^{r+1}.$$

Exercise: check that the code $C = \{f^{(r)}(s_1 \ldots s_r) : s_1 \ldots s_r \in \{0,1\}^r\}$ is prefix-free.

The length of the codeword $f^{(r)}(s_1 \ldots s_r)$ depends only on whether or not $s_1 \ldots s_r$ is typical. We have

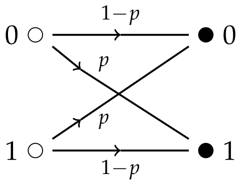
$$\begin{aligned}\overline{f}^{(r)} &= \mathbf{P}[S_1 \ldots S_r \in \mathcal{T}]m + \mathbf{P}[S_1 \ldots S_r \not\in \mathcal{T}](r+1) \\ &\leq m + \epsilon(r+1) \\ &\leq 2 + r(h+\epsilon) + \epsilon(r+1)\end{aligned}$$

and so

$$\frac{\overline{f}^{(r)}}{r} = h + 2\epsilon + \frac{2+\epsilon}{r}.$$

By choosing r sufficiently large and ϵ sufficiently small, we may make the right-hand side arbitrarily close to h, as required.

Shannon's Noisy Coding Theorem for the BSC(p) and the AEP. We now use the AEP to prove the constructive part of Shannon's Noisy Coding Theorem, Theorem 7.14(a), in the special case of the Binary Symmetric Channel with error probability p. The proof is quite demanding, so may be consider non-examinable.



Recall that the input and output alphabets are $\{0,1\}$ and that each sent bit flips, independently, with probability p, as shown by the diagram in the margin.

Let $h = H(p, 1-p)$. By Example 7.13(a), the capacity of the BSC(p) is $1-h$. Let $r < c$ be given and, as in the proof for the toy version of the channel seen in §9, let $M = 2\lceil 2^{rn} \rceil$, where n will be chosen by the end of the proof. As in the proof for the toy version, we choose a code $C = \{U(1), \ldots, U(M)\} \subseteq \{0,1\}^n$ by picking each codeword independently and uniformly at random from $\{0,1\}^n$.

Let $X \in \{0,1\}^n$ denote the sent codeword and $Y \in \{0,1\}^n$ denote the received word. The main idea is to apply the AEP to find the typical set for the random variable (X, Y). For this we need to know its entropy.

Lemma 12.3. *Let $X \in \{0,1\}^n$ be distributed uniformly and let $Y \in \{0,1\}^n$ be the received word when X is sent through the BSC(p). The pairs (X_i, Y_i) are independent random variables and $H(X_1, Y_1) = \ldots = H(X_n, Y_n) = 1 + h$.*

Proof. By the Chaining Rule (Lemma 7.6) we have $H(X_i, Y_i) = H(Y_i|X_i) + H(X_i)$. Since X_i is equally likely to be 0 and 1, and $Y_i = X_i$ with probability $1-p$, this is $H(p, 1-p) + H(\frac{1}{2}, \frac{1}{2}) = h + 1$, as required. □

Thus $(X_1, Y_1), (X_2, Y_2), \ldots$ is the sequence of symbols in $\{(0,0), (0,1), (1,0), (1,1)\}$ emitted by a memoryless source of entropy $1 + h$. By the

AEP, given $\epsilon > 0$, provided n is sufficiently large, there is a subset $\mathcal{T}$ of $\{0,1\}^{2n}$ such that if X and Y are as in the lemma, then

$$\mathbf{P}[(X,Y) \in \mathcal{T}] > 1 - \epsilon$$

and

$$2^{-n(1+h+\epsilon)} \leq \mathbf{P}[X = u, Y = v] \leq 2^{-n(1+h-\epsilon)}$$

for all $(u,v) \in \mathcal{T}$. We use $\mathcal{T}$ to define the following decoding rule:

- Suppose that $v \in \{0,1\}^n$ is received. If there exists a unique i such that $(U(i),v) \in \mathcal{T}$ then decode v as $U(i)$. Otherwise decode as $U(1)$.

Since $\mathcal{T}$ is a typical set, we expect that most of the time when we send $U(i)$, we receive a v such that $(U(i),v) \in \mathcal{T}$. Therefore decoding should succeed most of the time. To make this idea precise, we need two further properties of $\mathcal{T}$.

(a) Since $2^{-n(1+h+\epsilon)} \leq \mathbf{P}[X = u, Y = v]$ for all $(u,v) \in \mathcal{T}$, the same argument as Example 12.2 shows that $|\mathcal{T}| \leq 2^{n(1+h+\epsilon)}$.

(b) Suppose that $\widetilde{X}$ and $\widetilde{Y}$ are independently and uniformly distributed on $\{0,1\}^n$, so $\mathbf{P}[\widetilde{X} = u, \widetilde{Y} = v] = \frac{1}{2^n} \times \frac{1}{2^n} = \frac{1}{2^{2n}}$ for all $(u,v) \in \mathbb{F}_2^n$. Hence, by (a),

$$\begin{aligned}\mathbf{P}[(\widetilde{X},\widetilde{Y}) \in \mathcal{T}] &= \sum_{(u,v)\in\mathcal{T}} \mathbf{P}[\widetilde{X} = u, \widetilde{Y} = v] \\ &\leq 2^{-2n}2^{n(1+h+\epsilon)} = 2^{-n(1-h-\epsilon)}.\end{aligned}$$

Let p_i be the probability that when $U(i)$ is sent it is incorrectly decoded. Note this depends on both the choice of code *and* the behaviour of the channel. (Unlike the §9 proof, it is not possible to separate the two probability models entirely, and I should have written p_i the lecture, not P_i, which looks like a random variable.) We decode incorrectly only if $(U(i),Y) \notin \mathcal{T}$, or there is some other codeword $U(j)$ such that $(u(j),Y) \in \mathcal{T}$. Therefore

$$p_i \leq \mathbf{P}[(U(i),Y) \notin \mathcal{T}] + \sum_{j \neq i} \mathbf{P}[(U(j),Y) \in \mathcal{T}].$$

Since $U(i)$ is distributed uniformly at random, it is the same random variable as the X used in the AEP and so

$$\mathbf{P}[(U(i),Y) \notin \mathcal{T}] = \mathbf{P}[(X,Y) \notin \mathcal{T}] < \epsilon.$$

For the second summand note that Y is independent of $U(j)$, and that, since $U(i)$ is distributed uniformly on $\{0,1\}^n$, so is Y. (This is the BSC(p)-version of Question 1 on Sheet 7.) Therefore (X,Y) is the same random variable as the $(\widetilde{X},\widetilde{Y})$ in (b) above, and so

$$\mathbf{P}[(U(j),Y) \in \mathcal{T}] \leq 2^{-n(1-h-\epsilon)}$$

for each $j \neq i$. Since $M = 2\lceil 2^{rn} \rceil \leq 2\big(2^{rn} + 1\big) = 2^{rn+1} + 2 \leq 2^{rn+2}$,

$$\begin{aligned} p_i &< \epsilon + (M-1)2^{-n(1-h-\epsilon)} \\ &< \epsilon + 2^{rn+2-n(1-h-\epsilon)} \\ &= \epsilon + 2^{-n(1-h-r-\epsilon)+2}. \end{aligned}$$

By choosing n sufficiently large, we have $p_i < 2\epsilon$. Since the codewords $U(1), \ldots, U(M)$ all have the same uniform distribution, p_i does not depend on i, and is equal to the mean error probability $p = (p_1 + \cdots + p_M)/M$.

The remainder of the proof is as in the proof of Proposition 9.5: there is a particular code $C^\star$ of size M such that $p < 2\epsilon$. Choosing the best half of the codewords then gives a code of size $M/2 = \lceil 2^{rn} \rceil$ for which all the error probabilities are at most 2ϵ. Apart from the (unimportant) 2ϵ rather than ϵ, this is as required by Theorem 7.14(a).

Source coding with errors permitted. A memoryless source $S_1, S_2, \ldots$ emits the bits 0 and 1 each with equal probability $\frac{1}{2}$. Thus

$$\mathbf{P}[S_1S_2S_3 = 000] = \mathbf{P}[S_1S_2S_3 = 001] = \ldots = \mathbf{P}[S_1S_2S_3 = 111] = \tfrac{1}{8}.$$

The per-symbol entropy is $H(\frac{1}{2}, \frac{1}{2}) = 1$, so Shannon's Noiseless Coding Theorem (Theorem 4.7(b)) says that the average length of any injective encoder for words of length r is at least r.

In the remainder of this section we allow non-injective encoders. These lose information about the source; correspondingly, the source can be compressed beyond the bound in Shannon's Noiseless Coding Theorem.

In the proof of Proposition 12.5 outlined below, we need the inequality $1 - t \leq \mathrm{e}^{-t}$ and so $(1 - \frac{1}{M})^M \leq \mathrm{e}^{-1}$. In fact, when M is large, the two sides are very close.

Example 12.4. A lottery sells tickets numbered from $\{1, \ldots, T\}$. On the day of the draw, a random number is generated in this set: everyone whose ticket matches wins a prize. Let p_M be the probability that no-one wins when M people buy tickets. Then $p_M \leq \mathrm{e}^{-M/T}$. Moreover, $p_{\alpha T} \to \mathrm{e}^{-\alpha}$ as $T \to \infty$ for any $\alpha > 0$.

Proposition 12.5. *Let $D < \frac{1}{2}$ be given and let $h = H(D, 1-D)$. Let $\epsilon > 0$ be given. Provided n is sufficiently large, there is a binary code $C \subseteq \{0,1\}^n$ of size $\lceil 2^{n(1-h)} \rceil$ and an encoder $f : \{0,1\}^n \to C$ such that*

$$\mathbf{P}\big[d(f(S_1 \ldots S_n), S_1 \ldots S_n) \geq (D + \epsilon)\big] \leq \epsilon.$$

Above d denotes Hamming distance. Thus it is very likely that the codeword $f(S_1 \ldots S_n)$ chosen to encode $S_1 \ldots S_n$ differs from $S_1 \ldots S_n$ in at most $(D+\epsilon)n$ bits. Allowing a probability D of error on each bit allows us to compress n bits into $n(1-h)$ bits.

The result is included in this section because it can be proved using the AEP: see §10.5 of Cover and Thomas, *Elements of information theory*, [2] in the recommended reading. For interest only, a less technical proof is outlined below.

Outline proof. Let $h = H(D)$. Choose $M = \lceil 2^{(1-h)n} \rceil$ [**correction: was right in lecture**] codewords $U(1), \ldots, U(M)$ uniformly at random from $\{0,1\}^n$. Let $x \in \{0,1\}^n$. We try to encode x by a nearby codeword.

Define η by $h + \eta = H(D + \epsilon)$, so $\eta > 0$. By Lemma 9.1, the Hamming ball $B_{(D+\epsilon)n}(x)$ of radius $(D+\epsilon)n$ about x has at least $2^{(h+\eta)n}/(n+1)$ elements. Therefore the probability that a particular codeword $U(i)$ is within distance $(D+\epsilon)n$ [p **should be** D] of x is at least

$$2^{(h+\eta)n}/2^n(n+1).$$

The probability that no codeword is within distance $(D+\epsilon)n$ of x is therefore

$$\left(1 - \frac{1}{2^{(1-h-\eta)n}(n+1)}\right)^M.$$

Using the notation from the lottery, this probability is at most $e^{-M/T}$ where $T = 2^{(1-h-\eta)n}(n+1)$. Since $M \geq 2^{(1-h)n}$, we have

$$\frac{M}{T} \geq \frac{2^{(1-h)n}}{2^{(1-h-\eta)n}(n+1)} = \frac{2^{\eta n}}{n+1} \to \infty \quad \text{as } n \to \infty$$

and so $e^{-M/T} \to 0$ as $n \to \infty$.

Therefore, provided n is sufficiently large, our random code is very likely to have at least one codeword that can be used to encode x with at most $(D+\epsilon)n$ bits in error. An expectation argument, of the kind seen in §9, then shows that there is a particular code C satisfying the required conditions. □

Example 12.6. A source emits 120 bits per second, each equally likely to be 0 and 1. If a noiseless channel can only send 80 bits per second then we must compress by a factor of $\frac{2}{3}$. By Proposition 12.5, the least possible bit error probability D, must satisfy $1 - H(D, 1-D) = \frac{2}{3}$, or equivalently, $H(D, 1-D) = \frac{1}{3}$. Solving numerically we find that $D \approx 0.0615$. So this compression is theoretically possible provided a bit error probability of about 6.1% is acceptable. Each second 80 bits are sent and decoded to 120 bits; of these about 6.1% or 7.32 bits are likely to be wrong.

Compare this with the very naive encoder that simply forgets the final 40 bits each second. There is nothing better than to guess each forgotten bit. For each forgotten bit, there is a $\frac{1}{2}$ chance that the decoder makes the correct choice. So the bit error probability is $(80 \times 0 + 40 \times \frac{1}{2})/120 = \frac{1}{6}$, or about 16.7%. Of every 120 bits, about 20 are wrong.

13. General sources and Lempel–Ziv encoding

General sources. So far we have only considered memoryless sources. We end by briefly considering general sources.

Definition 13.1. The *entropy* of a source $S_1, S_2 \ldots$ is

$$\lim_{r\to\infty} \frac{H(S_1,\ldots,S_r)}{r}$$

when this limit exists.

Example 13.2.

(1) The memoryless binary source in Exercise 11.1 which emits 0 with probability $1-p$ and 1 with probability p has entropy $H(p,1-p)$.

(2) Consider the binary source for which $\mathbf{P}[S_1 = 0] = \mathbf{P}[S_1 = 1] = \frac{1}{2}$ and $S_t = S_1$ for all $t \in \mathbb{N}$. Thus the source emits either $000\ldots$ or $111\ldots$ with equal probability. Its entropy is 0.

(3) Consider the binary source which starts by flipping a coin biased to lands heads with probability p. If the coin lands heads, it emits $111\ldots$. Otherwise it behaves as the source in (1). You are asked to show on Problem Sheet 9 that the entropy of this source is $(1-p)H(p,1-p)$, and that it is not memoryless.

The following lemma generalizes (1) above. A proof is outlined on Problem Sheet 9.

Lemma 13.3. *The entropy of a memoryless source $S_1, S_2, \ldots$ exists and is $H(S_1)$.*

Definition 13.4. Let $S_1, S_2, \ldots$ be a source emitting symbols in an alphabet $\mathcal{A}$. The source is *stationary* if for all $\alpha_1, \ldots, \alpha_\ell \in \mathcal{A}$ and distinct times $t_1, \ldots, t_\ell$ we have

$$\mathbf{P}[S_{t_1} = \alpha_1, \ldots, S_{t_\ell} = \alpha_\ell] = \mathbf{P}[S_{t_1+r} = \alpha_1, \ldots, S_{t_\ell+r} = \alpha_\ell]$$

for all $r \in \mathbb{N}_0$.

For example, memoryless sources (see Definition 3.1) are stationary since the symbols are independent and identically distributed. Thus there is a probability measure p such that $\mathbf{P}[S_t = \alpha] = p_\alpha$ for all $\alpha \in \mathcal{A}$ and all $t \in \mathbb{N}$ and so the probabilities on either side are both $p_{\alpha_1} \ldots p_{\alpha_\ell}$.

The sources in Example 13.2(2) and (3) are also stationary. Example 13.7 gives a source that may not be stationary.

Definition 13.5. Fix a source $S_1, S_2, \ldots$ emitting symbols in an alphabet $\mathcal{A}$.

(i) The *frequency* of a word $\alpha_1 \ldots \alpha_\ell$, in the first r symbols, denoted $F_{\alpha_1\ldots\alpha_\ell}(r)$ is the number of times $t \in \{1, \ldots, r-\ell+1\}$ such that $S_t = \alpha_1, \ldots, S_{t+\ell-1} = \alpha_\ell$.

(ii) The source is *ergodic* if for all words $\alpha_1 \ldots \alpha_\ell$,

$$\lim_{r\to\infty} \frac{F_{\alpha_1\ldots\alpha_\ell}(r)}{r} = \mathbf{P}[S_1 = \alpha_1, \ldots, S_\ell = \alpha_\ell].$$

Note that in the left-hand side of (ii), $F_{\alpha_1\ldots\alpha_\ell}(r)$ is a random variable. The requirement is that, with probability 1, the limit exists and has the claimed value.

This is a stronger notion of convergence than the Weak Law of Large Numbers. Roughly put, ergodic sources are those for the long-term 'time' average (left-hand side) is certain to agree with the 'space' average (right-hand side).

The Strong Law of Large Numbers is equivalent to the statement that memoryless sources are ergodic. In particular, the source in Example 13.2(1) is ergodic.

Example 13.6. The source in Example 13.2(2) is not ergodic. The frequency $F_1(r)$ of the word 1 in the first r bits $S_1 \ldots S_r$ is either r or 0, with equal probability. Therefore

$$\frac{F_1(r)}{r} = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 0 & \text{with probability } \frac{1}{2}. \end{cases}$$

Hence the random variable $\lim_{r\to\infty} F_1(r)/r$ is equally likely to be 0 and 1. But the right-hand side in Definition 13.5(ii) is $\mathbf{P}[S_1 = 1] = \frac{1}{2}$ which is constant.

Example 13.7. A source $S_1, S_2, \ldots$ has alphabet $\{0, 1\}$. If $S_t = 0$ then $S_{t+1} = 0$ with probability $\frac{3}{4}$ and otherwise 1; if $S_t = 1$ then S_{t+1} is equally likely to be 0 and 1. These 'transition probabilities' can be recorded in a matrix

$$T = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

By diagonalizing T one finds that

$$T^t = \frac{1}{3}\begin{pmatrix} 2 + \frac{1}{4^s} & 1 - \frac{1}{4^s} \\ 2 - \frac{2}{4^s} & 1 + \frac{2}{4^s} \end{pmatrix} \to \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix} \quad \text{as } t \to \infty.$$

For instance, labelling rows and columns by $\{0, 1\}$, the top-left entry T^{t-1}_{00} is the probability that $S_t = 0$, given that $S_1 = 0$. By the limit above, when t is large, the source emits 0 about $\frac{2}{3}$ of the time.

You are asked to show on Problem Sheet 9 that, if the first symbol is 0 with probability $\frac{2}{3}$ (as in the limiting case) then the source is stationary. This is a special case of a general result about Markov sources with a unique stationary distribution.

Let h_t be the entropy of the first t bits emitted by the source. By the Chaining Rule (Lemma 7.6)

$$h_t = H(S_1, \ldots, S_{t-1}, S_t) = H(S_1, \ldots, S_{t-1}) + H(S_t | S_1, \ldots, S_{t-1}).$$

Since S_t depends on $S_1, \ldots, S_{t-1}$ only through S_{t-1}, this equation implies that $h_t = h_{t-1} + H(S_t | S_{t-1})$. We know that when t is large, S_{t-1} has, very nearly, the probability distribution $(\frac{2}{3}, \frac{1}{3})$. Therefore, by the usual formula for conditional entropy

$$\begin{aligned} H(S_t | S_{t-1}) &\approx \tfrac{2}{3} H(S_t | S_{t-1} = 0) + \tfrac{1}{3} H(S_t | S_{t-1} = 1) \\ &= \tfrac{2}{3} H(\tfrac{3}{4}, \tfrac{1}{4}) + \tfrac{1}{3} H(\tfrac{1}{2}, \tfrac{1}{2}) \\ &= \tfrac{4}{3} - \tfrac{1}{2} \log_2 3 + \tfrac{1}{3} \\ &= \tfrac{5}{3} - \tfrac{1}{2} \log_2 3 \end{aligned}$$

and so $h_t \approx h_{t-1} + (\frac{5}{3} - \frac{1}{3} \log_2 3)$ when t is large. It follows that the entropy of the source is

$$\lim_{t \to} \frac{h_t}{t} = \tfrac{5}{3} - \tfrac{1}{2} \log_2 3 \approx 0.874.$$

This should be compared with the entropy of the memoryless source with the same stationary distribution, namely $H(\frac{2}{3}, \frac{1}{3}) \approx 0.918$.

More generally, the output of an irreducible aperiodic Markov chain is stationary and ergodic, and so has a well-defined entropy.

Theorem 13.8. *A stationary ergodic source satisfies the Asymptotic Equipartition Property, as stated in Definition 12.1, taking h to be the entropy of the source.*

The proof of this theorem is beyond the scope of the course. You were asked to show in Question 2 on Sheet 8 that the AEP for memoryless sources implies Shannon's Noiseless Coding Theorem: it is routine to generalize this proof using Theorem 13.8 to prove Shannon's Noiseless Coding Theorem for a general stationary ergodic source.

Lempel–Ziv encoding. We end by seeing one practical way to encode a long binary word. Define a *dictionary* to be a function from binary words to $\mathbb{N}_0$. The words in the dictionary are the domain of the function and its *values* are its range. For instance $\varnothing \mapsto 0$, $1011 \mapsto 2$, $000 \mapsto 3$ is a dictionary with three words.

In Lempel–Ziv encoding we build a dictionary that efficiently represents the given binary word, and then encode the dictionary. We give an example and then specify the algorithm formally.

Example 13.9. Take

$$x = 10110\,10100\,010 = x_1 x_2 \ldots x_{13}.$$

We initialize the dictionary with the empty word $\varnothing$, to which we assign $0 \in \mathbb{N}_0$, so $\varnothing \mapsto 0$. (As a notational aid, values are written in red.) We then read the word from position 1.

At Step s, reading the word from position t, we take the longest subword $x_t \ldots x_{t+\ell-1}$ that is in the dictionary. (This could be the empty word when $\ell = 0$.) We then extend the dictionary by

$$x_t \ldots x_{t+\ell-1} x_{t+\ell} \mapsto s.$$

At the next step we continue from position $t + \ell + 1$. Doing this we split x as $x = 1\,0\,11\,01\,010\,00\,10$. The table below shows the longest subword in the dictionary and the word used to extend the dictionary for each step.

s	From	Subword	New word
1	1	$\varnothing$	$1 \mapsto 1$
2	2	$\varnothing$	$0 \mapsto 2$
3	3	1	$11 \mapsto 3$
4	5	0	$01 \mapsto 4$
5	7	01	$010 \mapsto 5$
6	10	0	$00 \mapsto 6$
7	12	1	$10 \mapsto 7$

The final dictionary determines the word: just concatenate its elements in value order. To represent the dictionary efficiently, note that each new word is obtained by appending a bit to a word already in the dictionary. For example at Step 5, 010 was obtained by appending 0 to 01, which had value 4. We may therefore replace 010 with $(4, 0)$. We then encode 4 in binary, as 100. In general, at Step s we append to a word with value in $\{0, 1, \ldots, s-1\}$, so we need $\lceil \log_2 s \rceil$ bits to distinguish all the values.

s	New Word	(Value, New Bit)	$\lceil \log_2 s \rceil$	(Binary Value, New Bit)	Encoding
1	$1 \mapsto 1$	$(0, 1)$	0	$(\varnothing, 1)$	1
2	$0 \mapsto 2$	$(0, 0)$	1	$(0, 0)$	00
3	$11 \mapsto 3$	$(1, 1)$	2	$(01, 1)$	011
4	$01 \mapsto 4$	$(2, 1)$	2	$(10, 1)$	101
5	$010 \mapsto 5$	$(4, 0)$	3	$(100, 0)$	1000
6	$00 \mapsto 6$	$(2, 0)$	3	$(010, 0)$	0100
7	$10 \mapsto 7$	$(1, 0)$	3	$(001, 0)$	0010

The final encoding is therefore 1000111011000010000010, or

$$1000111011000010000010$$

as it should be written using just the uncoloured binary alphabet. Note that without the colour coding, the convention that $\lceil \log_2 s \rceil$ bits are used for the value at step s is essential to decode unambiguously. **[Corrected 2 April 2020: the binary encoding for the final step was wrong in the final bit (which I then omitted to put in the final encoding).]**

Algorithm 13.10 (Lempel–Ziv)**.** The input is a word $x_1 \ldots x_r$ and the output is a binary word.

Initialize the dictionary: define $\varnothing \mapsto 0$ and set $t = 1$. Go to Step 1.

Step s: if $t > r$ then terminate.

- Read the word from position t. Choose ℓ maximal such that the dictionary contains $x_t \ldots x_{t+\ell-1}$: suppose that $x_t \ldots x_{t+\ell-1} \mapsto v$.
- Append the binary form of v of length $\lceil \log_2 s \rceil$ to the output word.
- If $t + \ell - 1 = r$ then terminate. Otherwise append $x_{t+\ell}$ and add $x_t \ldots x_{t+\ell} \mapsto s$ to the dictionary. Go to Step $s + 1$.

The only extra feature, not seen in Example 13.9 is that if x ends with a subword in the dictionary, we do not need to extend the dictionary, and can simply output the binary form of the value of the subword.

All this can be done using the MATHEMATICA notebook `LempelZiv.nb` on Moodle. The second column in the second table above is the dictionary output by `LempelZivWithDictionary`; the third and fifth columns can be calculated using `LempelZivPairs`, and `LempelZivBinaryPairs`. To get just the output of the Lempel–Ziv algorithm, use `LempelZiv`.

In any example you are likely to have the patience to do by hand, the Lempel–Ziv encoding will almost certainly be longer than the original word.

Example 13.11. Consider the word $w^{(s)}$ of length $2s$ formed by repeating 01. For example,

$$w^{(10)} = 0101010101\,0101010101$$

shown split into two blocks of size 10 for readability. Let ℓ_s be the length of the Lempel–Ziv encoding $w^{(s)}$. Even for this highly regular word, it is not until $s = 28$, so length 56, that the Lempel–Ziv encoding is shorter. The table below, found using the functions in the MATHEMATICA notebook `LempelZiv.nb`, shows the ratio $\ell_s/2s$.

s	1	2	5	10	20	27	28	29	30	40	50	60	120
$\ell_s/2s$	1.5	1.5	1.6	1.2	1.1	1	0.982	1.017	0.983	0.875	0.830	0.783	0.621

For a worked example of decoding, see Problem Sheet 9.

The Lempel–Ziv Algorithm was published in 1977. Later in 1991 it was proved that, for suitable sources, the algorithm achieves the bound in Shannon's Noiseless Coding Theorem, and so is optimal. Very, very roughly, one might expect this to be true because, by the AEP, a source of entropy h has a set of about 2^{hr} typical messages of length r, all of which enter the Lempel–Ziv dictionary; at this point in the algorithm the dictionary also has size about 2^{hr}, so $\lceil 2^{hr} \rceil + 1 \approx hr + 1$ bits are output for each subword of length r emitted by the source.